Communication HF

OLIVIER COCHELIN (COCO)

7 septembre 2002

Table des matières

Table des figures

Chapitre 1

Introduction

Ce document présente le système de communication HF qui a été développé en 2002 dans le cadre du Club de Robotique de l'E.S.E.O. Ce système a pour objet principal de pouvoir transmettre des données de la balise principale vers le robot. Ces informations permettent au robot de connaître la position de l'adversaire. Cette position est obtenue à l'aide d'une triangulation laser détaillée dans un autre document.

On documente ici la solution qui a été retenue pour cette communication HF basée autour de modules RFM. On y présente en particulier le principe de fonctionnement du système. On insistera d'une part sur la conception de l'électronique de ces modules et d'autre part sur le protocole logiciel. Dans un dernier temps, on développera une critique sur cette communication HF en proposant des solutions aux problèmes rencontrés.

Chapitre 2

Architecture globale

Le but de cette communication HF était de transmettre des données de la balise principale vers le robot et éventuellement dans l'autre sens. Il s'est avéré au final que seule la première de ces directions était véritablement utile. L'autre qui pourrait être utilisée pour du debug ne l'a jamais été faute de temps et de fiabilité.

2.1 Constitution du système

Pour faciliter l'interfaçage de ces modules HF avec des microcontrôleurs, nous avons décidé de les connecter à chaque extrémité à une UART. La FIGURE 2.1 présente le principe de fonctionnement d'un tel module. Ceci présente plusieurs avantages :

- Il est ainsi facile de tester la communication avec un simple PC.
- Cette technique soulage également les microcontrôleurs en utilisant l'UART matérielle de ceux-ci, au lieu d'exécuter du code pour recevoir les bits un à un.
- On facilite ainsi le développement logiciel en manipulant directement des octets au lieu de manipuler des bits.

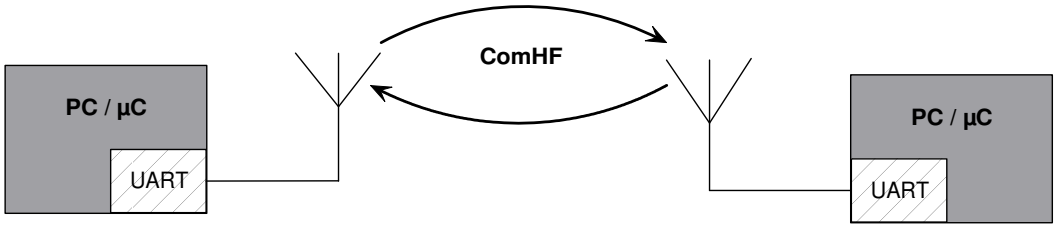

FIG. 2.1 – ***Principe d'une communication HF utilisant des UARTs matérielles***

2.2 Choix de la fréquence porteuse

L'utilisation d'une fréquence porteuse à 868 MHz n'est pas complètement un choix. En fait, il nous a été proposé de tester des modules HF fonctionnant à cette fréquence.

La bande 868 MHz a été récemment ouverte en Europe pour compléter la bande 433 MHz. Elle a donc pour avantage d'être encore peu occupée et donc un peu moins brouillée que la bande 433 MHz. En revanche, il est plus difficile au niveau de la réalisation de travailler à des fréquences avoisinant le gigahertz.

2.3 Bi-directionnalité

Une liaison série RS232 est par nature bi-directionnelle et généralement full-duplex. Les modules qui nous ont été proposés sont bi-directionnels (émetteurs-récepteurs dits transceiver, contraction de transmitter et de receiver) mais comme ils utilisent la même porteuse en émission qu'en réception, ils sont limités à du half-duplex faute de quoi ils se brouillent eux-mêmes.

Lorsqu'un module émet, il ne fait qu'émettre et pendant ce temps l'autre ne fait que recevoir et vice-versa. Il faudra donc que le protocole prenne en compte cette contrainte.

2.4 Module HF

Le module HF qui nous a été proposé et qui a été retenu est un module émetteur-récepteur hybride TR1001 fabriqué par RFM. Ce module fonctionnant à 868.35 MHz permet de réaliser de la modulation d'amplitude : OOK (On Off Keying) et ASK (Amplitude Shift Keying). Dans ce dernier mode, il permet d'atteindre un débit binaire de 115.2 kbps. Ce module qui s'alimente en 3 V dispose de fonctions de mise veille qui le rendent très économique.

Pour plus de renseignements, se reporter à la datasheet du composant.

Chapitre 3

Électronique

La FIGURE 3.1 présente le schéma électronique des modules HF. Dans cette partie, nous allons tenter d'éclaircir un certain nombre de points de ce schéma en insistant sur les choix qui ont été faits, leurs avantages, leurs inconvénients et les erreurs qui sont apparues lors des tests.

Là aussi, il est fortement conseillé de se reporter aux documentations du fabriquant RFM pour plus de renseignements.

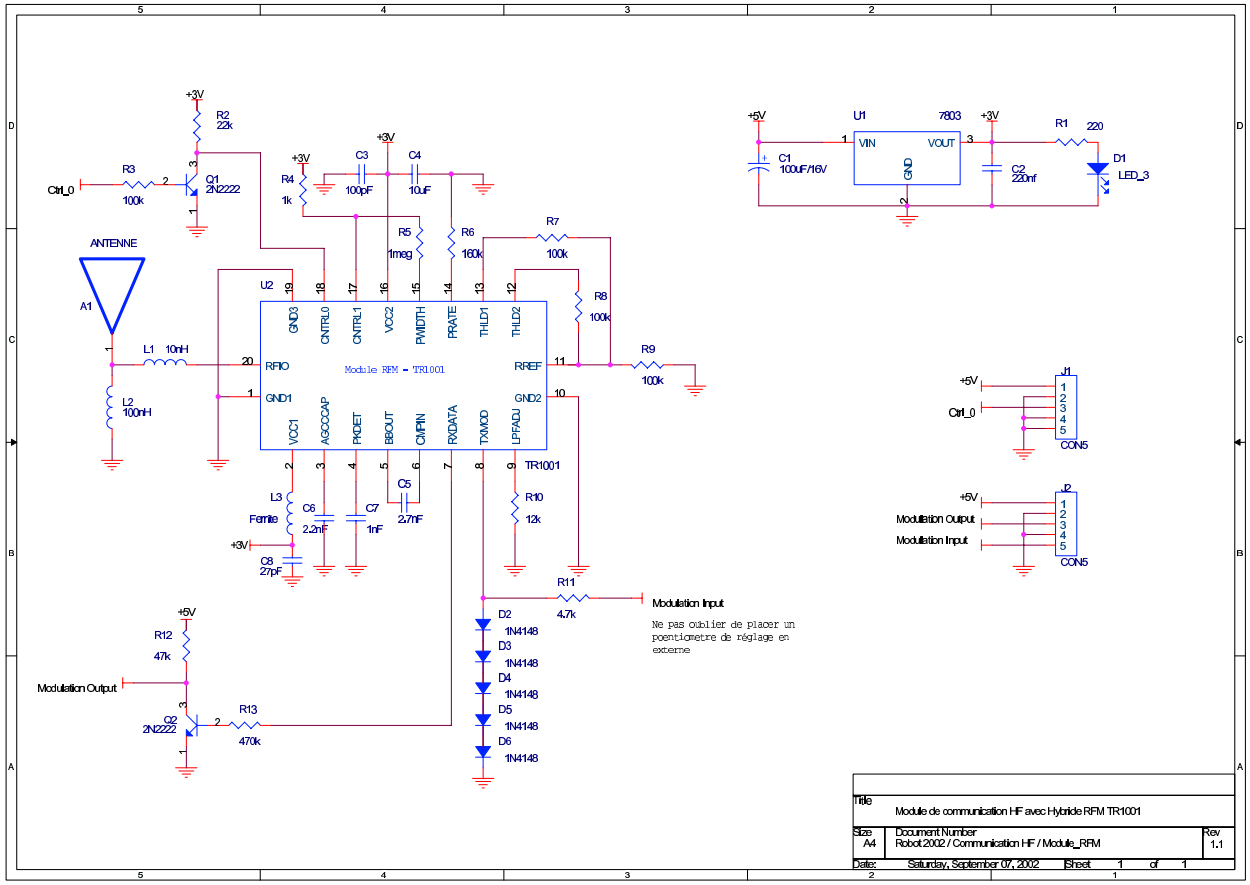

FIG. 3.1 – ***Schématique des modules HF***

3.1 Débit binaire et Modulation

Le bug :

A l'origine, nous avons conçu le module pour l'exploiter au maximum de ses possibilités, c'est à dire en utilisant une modulation ASK 4 avec un débit binaire de 115 200 bps. Ceci signifie que le module émet (ou reçoit) des symboles codés sur quatre niveau. Chaque symbole est alors composé de 2 bits et la durée de chaque symbole est de $2/115200 = 17.36\mu s$. Par exemple, 00 est codé par une amplitude A_1, 01 par une amplitude A_2, 10 par une amplitude A_3 et 11 par une amplitude A_4.

Nous pensions alors qu'il suffisait de fournir au module un train binaire à la vitesse de 115 200 bps et que le module se chargeait alors de regrouper les bits deux par deux pour composer les symboles et ensuite les émettre. En fait, il n'en est rien.

Les faits :

Lorsqu'on utilise la modulation OOK, les bits sont envoyés sur l'entrée $TXMOD$ du module émetteur et un seuillage permet au module récepteur de déterminer si le bit qu'il vient de recevoir est un '0' ou un '1' logique qu'il ressort ensuite sur la pin $RXDATA$.

Par contre, lorsqu'on utilise pleinement la modulation ASK, il faut coder soi-même les symboles. En effet, dans ce mode il faut regrouper les bits 2 par 2 pour composer soi-même les 4 symboles qui seront codés par 4 amplitudes différentes. Ces quatre symboles sont transmis en appliquant un niveau analogique correspondant sur l'entrée $TXMOD$ du module émetteur. Au niveau du récepteur, un signal analogique est délivré sur la pin $BBOUT$. La sortie $RXDATA$ est en fait un seuillage binaire de la sortie $BBOUT$. On ne peut donc pas récupérer le signal sur cette sortie puisque le symbole émis est codé sur 4 niveaux et celui récupéré sur $RXDATA$ n'en compte que 2. Donc, pour récupérer ce symbole, il faut soi-même acquérir le signal analogique au niveau de $BBOUT$ et réaliser un traitement approprié (un seuillage 4 niveaux par exemple). Pour réaliser une communication avec un débit binaire de 115 200 bps en utilisant cette modulation ASK 4, il faut envoyer les symboles (composés chacun de 2 bits) à la vitesse de 57 600 bauds.

La solution :

Notre intention n'a bien sûr jamais été de réaliser une modulation ASK comme celle-ci. Comme il est très difficile de revenir en arrière compte tenu du soudage du module, nous avons choisi de nous adapter et de faire avec ce dont on disposait.

Au final, nous avons utilisé les modules tel quel mais sans vraiment utiliser la modulation ASK. En effet, nous avons appliqué directement sur l'entrée $TXMOD$ le train binaire provenant de n'importe quelle UART, avec un débit de 57 600 bps. le module émetteur réalise une modulation ASK de cette entrée, qui ne compte en réalité que 2 symboles sur 4 (00 et 11). Au niveau du module récepteur, on récupère le train sur la sortie $RXDATA$ qui n'a en fait qu'à faire un seuillage entre deux symboles.

En fin de compte, nous avons une communication HF avec un débit de 57 600 bits par secondes utilisant une modulation ASK 4 limitée à 2 symboles ; ce qui revient à peu près à une modulation OOK.

3.2 L'alimentation

Comme nous l'avons vu, les modules RFM TR1001 doivent être alimentés en 3 V. Hors, nous ne disposons sur toutes les autres cartes électroniques que d'alimentations 5 V obtenues à partir des batteries de 7.2 V.

Les modules HF sont donc munis d'un régulateur 3 V. Pour fonctionner correctement, les modules RFM - et plus généralement tous les circuits radiofréquences et hyperfréquences - doivent être alimentés par plusieurs pins V_{CC1} et V_{CC2} et correctement découplés. Pour ce faire, on place sur la ligne d'alimentation et au plus près du module des capacités de découplage de différentes valeurs : $C_4 = 10\mu F$, $C_3 = 100pF$ et

$C_8 = 27pF$. Ces capacités ont pour rôle de répondre aux appels de courants du module et ainsi de stabiliser l'alimentation. L'autre élément important est la self de choc L_3. Cette ferrite évite aux parasites hautes fréquences générés par le module de remonter vers l'alimentation.

3.3 La masse

En relation directe avec l'alimentation, la masse est un élément fondamental pour le bon fonctionnement d'un circuit RF. Là encore, trois pins y sont dédiées sur le module RFM GND_1, GND_2 et GND_3. Sur le PCB, il faut que la liaison entre toutes ces masses soit la moins inductive possible. L'idéal pour cela est un plan de masse qui doit être le moins "découpé" possible. Dans le cas d'un circuit double faces, les deux plans de masse doivent être reliés entre eux en plusieurs endroits.

3.4 L'antenne

L'entrée/sortie $RFIO$ du module est prévue pour accueillir une antenne de 50 Ω. Toutefois, l'antenne que l'on réalise n'a pas nécessairement une impédance de 50 Ω. Pour pallier à ce problème, on dispose des inductances variables $L_1 = 10nH$ et $L_2 = 100nH$ qui permettent d'adapter des antennes ayant une impédance comprise entre 35 et 72 Ω. Cependant, les essais de réglage effectués sur ces deux inductances n'ont pas montré de résultats probants.

L'inductance L_2 tient lieu à la fois de self d'accord mais aussi de protection électrostatique. En effet, lorsque l'antenne se charge électrostatiquement, cette inductance écoule les charges vers la masse en protégeant ainsi le module.

Pour l'antenne elle-même, nous avons choisi de réaliser une antenne spirale sur le PCB lui-même. La longueur de cette antenne est de $\lambda/4$ et doit absolument être associée avec le plan de masse correspondant. Ce plan de masse joue le rôle de "miroir" (comme la carrosserie d'une voiture), de telle manière que l'antenne $\lambda/4$ avec son plan de masse soient équivalents à une antenne fouet de longueur $\lambda/2$. Cette antenne a pour avantages d'être assez sélective autour de la fréquence porteuse, d'être simple à réaliser et d'être peu encombrante. En revanche, son diagramme de rayonnement montre qu'elle n'émet pas tout à fait la même puissance dans toutes des directions et qu'elle est assez facilement désaccordée par les éléments environnant. Pour plus de renseignements sur la fabrication des antennes, on peut se reporter à un document RFM comparant différents types d'antennes et indiquant comment les réaliser.

La pratique a montrer que l'antenne avait une grande influence sur la qualité de la transmission. Toutefois, pour des distances de 20 à 50 cm, on est quasiment certaine d'avoir une transmission correcte ce qui peut-être intéressant lorsque l'on fait des tests. La présence d'objets métalliques massifs (marteau, pinces ...) à proximité immédiate des antennes joue sur celles-ci en bien ou en mal selon les cas.

3.5 Le contrôle du module

Les pins $CNTRL_0$ et $CNTRL_1$ permettent de sélectionner le mode de fonctionnement du module. Lorsque ces 2 broches sont à l'état bas, le module est en veille. Cette configuration n'est pas possible ici car elle ne présentait pas d'intérêt dans notre cas.

Pour utiliser la modulation ASK, $CNTRL_1$ doit être tiré à V_{cc} et à la masse pour une modulation OOK. Il faut alors placer $CNTRL_0$ à l'état bas pour passer en émission.

Pour placer le module en réception, il suffit de placer les deux broches $CNTRL_0$ et $CNTRL_1$ à l'état haut. La modulation est totalement sans influence pour la réception.

On peut noter qu'il a été placé un circuit d'adaptation pour la pin $CNTRL_0$. Ce circuit permet de faire l'adaptation 5 V / 3 V de manière à protéger le module des surtensions mais il réalise en même temps une inversion du signal. Vu de l'extérieur, il suffit donc d'appliquer un niveau bas sur $Ctrl\_0$ pour placer le module en réception et d'appliquer un niveau haut pour le placer en émission.

3.6 Émission et réception de données

Émission :

L'émission des données se fait en rentrant le train de bits sur l'entrée $TXMOD$. En fait, il faut remarquer que la puissance d'émission du module est contrôlée par le courant que l'on délivre sur cette entrée en ne dépassant toutefois pas la puissance limite.

Il n'y a donc pas besoin ici de circuit d'adaptation 5 V / 3 V puisque, seul le courant importe vraiment. La résistance $R11$ sert donc à protéger le module dans le cas où on applique des signaux d'amplitude 5 V sur l'entrée $Modulation\ Input$. Il est toutefois recommandé de placer un potentiomètre en amont de cette entrée pour contrôler la puissance d'émission du module.

Les 5 diodes 1N4148 servent à prévenir les surtensions sur l'entrée $TXMOD$. En effet, si la tension devient supérieure à $5 \times 0.6V = 3V$, les diodes deviennent passantes et écoulent le courant vers la masse en préservant ainsi le module.

Réception :

Les données reçues par le module ressortent sur la broche $RXDATA$. Mais là encore, la sortie est en 3 V et il faut donc un circuit d'adaptation pour faire la conversion 3 V / 5 V au prix d'une inversion du signal.

Il faut noter que l'impédance de sortie de la pin $RXDATA$ est extrêmement élevée et est donc très sensible aux parasites. Le montage d'adaptation situé juste en sortie sert donc également à immuniser le signal en l'amplifiant son courant.

On peut remarquer que le signal subit une inversion à la réception et nulle part ailleurs. Le signal qui sort d'un module récepteur est donc l'inverse du signal qui rentre sur le module émetteur. Pour remédier à ce petit problème, il y a plusieurs solutions.

- Inverser le signal en entrée de l'émetteur à l'aide d'un transistor ou d'une porte logique.
- Inverser une deuxième fois le signal en sortie du module récepteur. C'est ce que nous avons fait en diminuant encore ainsi la résistance de sortie de l'ensemble.
- Complémenter les signaux en soft, soit à l'émission soit à la réception. Cette solution n'est pas envisageable dans le cas d'une utilisation avec une UART matérielle car on ne peut pas complémenter les bits de start et de stop que génèrent celle-ci.

3.7 Le reste du circuit

Le reste du circuit est tiré d'une application proposée par le constructeur dans la datasheet du module RFM TR1001. Il est donc recommandé de se référer à cette datasheet pour mieux comprendre le fonctionnement interne de ce module et le rôle des composants externes.

3.8 La réalisation

La réalisation du circuit en elle-même ne pose pas vraiment de problèmes particuliers. Il faut cependant veiller à un certain nombre de choses.

Comme pour toute réalisation RF, surtout à ces fréquences, il faut apporter un soin particulier au routage du circuit. On doit en particulier garder les pistes les plus courtes possibles. L'utilisation d'un plan de masse améliore la CEM et est ici indispensable pour l'antenne. L'utilisation de composants CMS est recommandée pour réduire l'encombrement et raccourcir les pistes. Un exemple de routage et d'implantation des composants est présenté à la FIGURE 3.2. Le PCB final n'est pas tout à fait celui-ci mais il en est très proche.

L'inconvénient majeur de ce module HF est son soudage. En effet, les pattes du composants se trouvent complètement sous celui-ci. Il a donc fallu le souder par capillarité mais ce soudage n'est pas aisé et les faux contacts ne sont pas faciles à détecter. De plus, le moindre choc peut provoquer le déssoudage d'une des

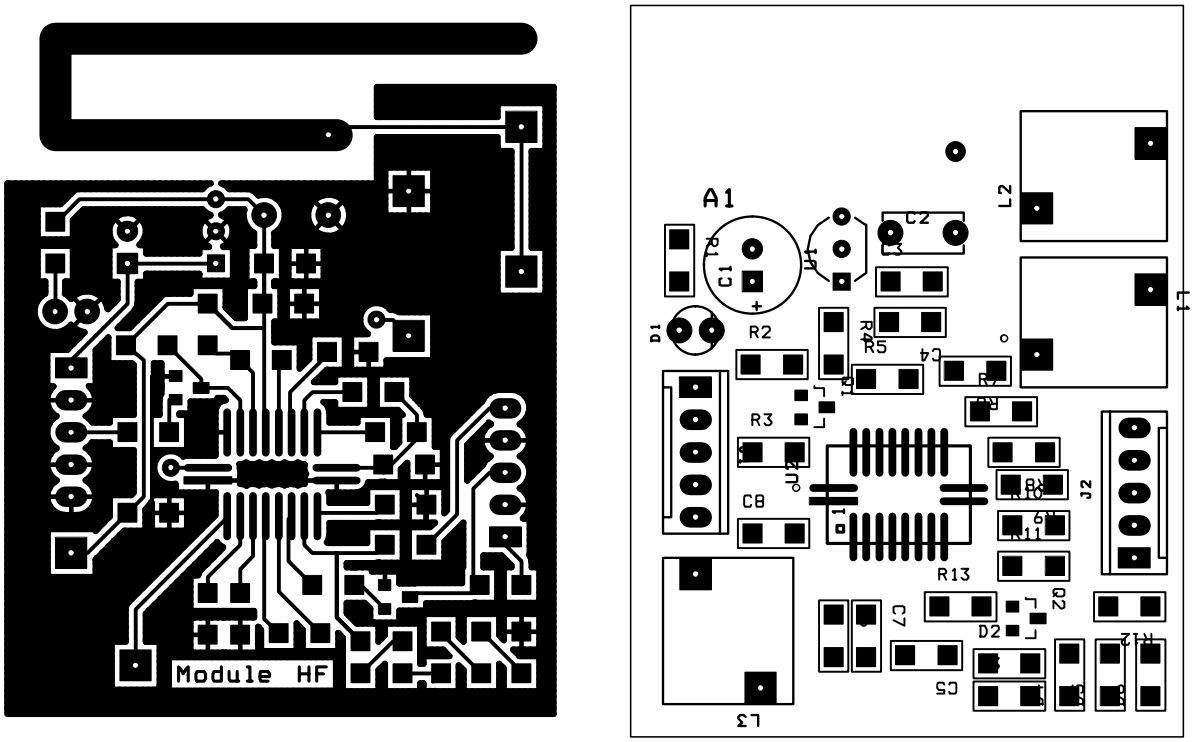

FIG. 3.2 – ***Routage et implantations des composants des modules HF***

pattes. Une autre technique pour le soudage de ces modules consiste à réaliser une empreinte inversée sur le PCB, de percer un trou de la taille du module dans la PCB de telle manière que les pattes du module viennent se positionner juste à coté des pistes constituant l'empreinte. Les pattes et les pistes sont alors complètement visibles et n'y a plus qu'à réaliser un pont de soudure entre les deux.

3.9 Les tests

Pour tester les modules réalisés, nous avons appliqué la démarche suivante. Il ne faut pas oublié que les premiers tests ont été réalisé avec un seul module.

1. La première chose a testé a bien sûr été les alimentations 3,3 V qui se sont révélées correctes. Il a également fallu s'assurer que le toutes les entrées du module recevait bien les niveaux nécessaires à son bon fonctionnement.
2. Le second point à vérifier est la capacité du module à émettre une porteuse et éventuellement des données. Il suffit pour cela de placer le module en émission et de lui envoyer un train de bits (ici, un simple signal carré suffit). Ensuite, pour s'assurer que le module émet quelque chose, nous avons utilisé un analyseur de spectre couplé à une antenne, tous deux disponibles dans le labo Maxwell. En centrant l'analyseur de spectre sur la fréquence centrale des modules, on voit apparaître un pic pour la porteuse et de chaque côté une bande correspondant à la modulation. Lors des tout premiers tests, un problème de soudage du module empêchait le module d'émettre une porteuse.
3. Une fois passé ce test, nous avons réalisé un second module testé de la même manière. Pour valider une communication entre 2 modules, il nous a fallu développer une interface adaptée. Celle-ci est constitué de deux petites cartes qui se connecte au port série d'un PC et qui utilisent un MAX232 pour convertir les niveaux logiques. La FIGURE 3.3 présente le principe de fonctionnement de l'interface matérielle utilisée pour ces tests. L'autre partie de l'interface est constitué par un programme de test ap-

pelé **ComHF**. Ce programme permet d'émettre ou de recevoir des messages via le port série. Il affiche en outre des statistiques concernant la réception ce qui permet d'évaluer la qualité d'une transmission.

4. C'est ce petit programme qui à l'origine n'utilisait aucun protocole qui nous a permis de développer et de valider le protocole présenté dans la suite de ce document. C'est ainsi que petit à petit sont venus s'ajouter des contrôles et des améliorations au protocole pour obtenir une communication à peu près fiable et qui surtout ne va jamais se bloquer suite à une erreur quelconque. C'est ce programme qui a ensuite été porté sur les PICs pour la communication finale entre la balise principale et le robot.

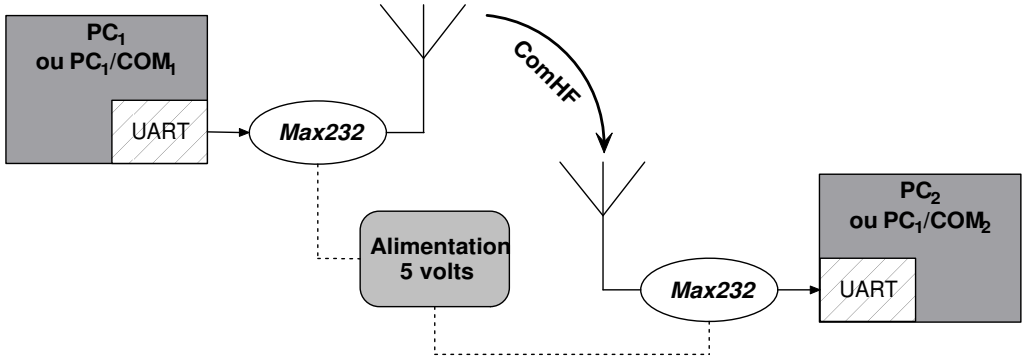

FIG. 3.3 – ***Principe de fonctionnement de l'interface matérielle utilisée pour les tests***

Chapitre 4

Protocole

Maintenant que ces modules sont matériellement fonctionnels, voyons plutôt comment les utiliser.

4.1 DC balance

On comprend aisément que le récepteur, qui dispose d'un contrôle actif de gain et d'un seuillage dynamique, doit pour fonctionner correctement recevoir des données "équilibrées". En effet, il faut que les données transmises contiennent en moyenne chaque symbole en nombre égal ; c'est ce qu'on appelle le "DC Balance".

De plus, il est fortement recommandé que le module ne reçoivent jamais plus de 4 symboles successifs identiques. En effet, s'il reçoit un nombre important de zéros successifs, le récepteurs va augmenter grandement son contrôle actif de gain ce qui rendra les données reçues totalement erronées en interprétant le bruit comme un symbole.

Le protocole que l'on a conçu respecte donc ces deux conditions. Pour cela, nous avons utilisé un codage spécifique qui correspond à la conversion d'un octet en un mot de 12 bits. En fait, il s'agit plus exactement d'une conversion de chaque quartet en un sextuplet.

4.2 Conversion Quartet $\longrightarrow$ Sextuplet

Ainsi, chaque fois que l'on désire transmettre un octet de données, cet octet est séparé en un quartet de poids faible et un quartet de poids fort. Chaque quartet est ensuite converti en un mot de 6 bits comportant autant de 1 que 0 et ayant au plus 3 bits successifs identiques. Puis, chacun de ces sextuplets est encapsulé dans une "trame" RS232 pour être émis, en commençant par la "trame" correspondant au quartet de poids fort. Devant le sextuplet, on ajoute '10' s'il correspond au quartet de poids fort et '01' sinon. On peut ainsi vérifier à la réception que l'on a pas manqué un quartet

Table de conversion :

Quartet	Sextuplet
$00 = 0x0$	$11 = 0x0B = 0b001011$
$01 = 0x1$	$13 = 0x0D = 0b001101$
$02 = 0x2$	$14 = 0x0E = 0b001110$
$03 = 0x3$	$19 = 0x13 = 0b010011$
$04 = 0x4$	$22 = 0x16 = 0b010110$
$05 = 0x5$	$25 = 0x19 = 0b011001$
$06 = 0x6$	$26 = 0x1A = 0b011010$
$07 = 0x7$	$28 = 0x1C = 0b011100$
$08 = 0x8$	$35 = 0x23 = 0b100011$
$09 = 0x9$	$37 = 0x25 = 0b100101$
$10 = 0xA$	$38 = 0x26 = 0b100110$
$11 = 0xB$	$41 = 0x29 = 0b101001$
$12 = 0xC$	$44 = 0x2C = 0b101100$
$13 = 0xD$	$49 = 0x31 = 0b110001$
$14 = 0xE$	$50 = 0x32 = 0b110010$
$15 = 0xF$	$52 = 0x34 = 0b110100$

Si l'on veut transmettre l'octet $67 = 0x43$, le quartet $0x4$ va devenir le sextuplet $22 = 0x16 = 0b010110$ et le quartet 3 va devenir le sextuplet $19 = 0x13 = 0b010011$. Comme $0x4$ est le quartet de poids fort, on va envoyer en premier $0b10010110 = 150 = 0x96$ puis le quartet de poids faible $0x3$ sous la forme : $0b01010011 = 83 = 0x53$. Ces deux octets sont encapsulées dans des "trames" RS232 avec un bit de start à 0 et un bit de stop à 1 en commençant par le bit de poids faible.

On constate que les trames ainsi construites remplissent complètement les deux conditions évoquées plus haut au prix bien sûr d'une division par deux du débit. A la réception, il n'y a plus qu'à recomposer le message en appliquant la démarche inverse.

Les fichiers sources correspondant à ce codage pour la balise principale sont : **HF_PIC.C** et **HF_PIC.H**.

4.3 Trames de synchronisation

Pour que la transmission soit correcte, il faut que l'UART de l'émetteur et celle du récepteur restent synchronisées. Hors, avec une communication HF, ceci ne peut être garantit en permanence puisque l'on peut mal interpréter les bits de start et de stop et ainsi désynchroniser les UARTs.

Pour cela, nous avons mis en place deux trames de synchronisation. Elles sont composées des deux octets suivants $0x55 = 0b01010101$ et $0xAA = 0b10101010$, ce qui donne une fois encapsulés dans des trames RS232 $0b0010101011$ et $0b0101010101$. Ces trames sont elles aussi équilibrées mais elles présentent une autre propriété intéressante.

En effet, si on les insère plusieurs fois l'une après l'autre, la position des bits est telle qu'une UART désynchronisée finira toujours en moins de 8 trames de ce type par se resynchroniser.

Ainsi, à chaque début de communication, on émet 10 trames de ce type pour synchroniser les UARTs et régler le contrôle actif de gain du récepteur. De même, tant qu'il n'y a pas de données à émettre, on émet ces trames. Et, après chaque paquet de données émis, ces trames de synchronisation sont également transmises pour marquer la fin du message.

4.4 Les paquets de données (Haut niveau)

Typiquement, les paquets de données transmis avaient la forme suivante :

– Entête (1 octet) sert à identifier de quel type de message il s'agit.

– Données (Longueur variable fonction de l'entête connue de l'émetteur et du récepteur)
– Checksum (1 octet) pour contrôler la validité des données reçues.

Le checksum est tout simplement calculé sur des *unsigned char* en additionnant tous les octets de données + l'entête sans tenir compte des débordements. Le calcul de celui-ci à l'émission et à la réception permet de s'assurer que les données n'ont pas été corrompues dans la communication. S'il s'avère qu'il y a eu une erreur de transmission (Nombre d'octets reçus incorrect, checksum faux ...), on jette le paquet et on attend un nouveau message suivant une trame de synchronisation.

Les fichiers sources correspondant à la création de ces paquets de données pour la balise principale sont : **HF_PIC.C** et **HF_PIC.H**.

4.5 L'envoi de données

Lorsque la balise principale souhaite envoyer un paquet de données (en général quelques octets), elle place dans une structure appelée `PAQUET` le type de message dont il s'agit et tous les octets de données l'accompagnant. Elle appelle ensuite routine *envoi()* qui calcule le checksum et transforme chaque octet en quartet puis sextuplet ... et les place dans un boite d'envoi `tx_box`. La routine *envoi()* est bloquante tant que l'on a pas émis au moins une fois le paquet précédent de manière à éviter l'écrasement d'informations.

Lorsque l'UART est prête à émettre (son tampon d'un octet est vide), une interruption est levée qui appelle la routine *transmit()*. Cette routine bas niveau vérifie s'il y a quelque chose à émettre et si oui l'émet sinon, elle émet une trame de synchronisation. De même, c'est cette routine qui gère l'envoi des trames de synchronisation après chaque paquet.

En outre, chaque paquet de données est retransmis autant de fois qu'il est possible de le faire avant qu'il n'y ait d'autres données à envoyer. Ceci est fait pour améliorer la probabilité de bonne transmission des données qui reste incertaine.

Les fichiers sources correspondant à l'envoi des données pour la balise principale sont : **HF_PIC.C** et **BALISE_FIXE.C**.

4.6 La réception de données

Dès que l'UART reçoit un octet du module HF, une interruption est levée et la routine bas niveau *receive()* est appelée. Cette routine vérifie qu'il n'y a eu ni erreur de tramage ni débordement du tampon et regarde si l'octet reçu est un octet de synchronisation ou non. Si oui, elle l'ignore. Si non, elle vérifie que les bits indiquant le poids du quartet correspondent à l'octet que l'on est en train de recevoir. Dès qu'une erreur de ce type est détectée, on repart de zéro. Cette routine décode également les octets reçus en vérifiant que les codes sont corrects ; elle en extrait les sextuplets, retrouve les quartets correspondants et finalement recompose les octets de données originaux. Pour finir, cette routine *receive()* place les octets de données reçus dans une boite de réception `rx_box`.

Dès que cette routine bas niveau reçoit de nouveau une séquence de synchronisation, elle lève un drapeau `rx_frame_received` qui déclenchera l'exécution de la routine *reception()*. Cette routine teste de quel type de message il s'agit, vérifie que le nombre d'octets reçus correspond à ce type de message et recalcule le checksum. Si l'un de ces tests donne un mauvais résultat, le paquet est ignoré. Suivant le type de message reçu, les actions correspondantes sont exécutées.

Les fichiers sources correspondant à la réception des données sont disponibles pour la balise principale bien qu'inutilisés : **HF_PIC.C** et **BALISE_FIXE.C**.

Chapitre 5

Résultats et Perspectives

Dans cette partie, nous allons développer une critique de cette communication HF afin d'étudier des solutions et en s'appuyant sur les résultats obtenus cette année.

5.1 Les résultats

Une fois les modules assemblés et couplés au protocole de communication, il nous était possible de transmettre des données de la balise principale au robot. Toutefois, la portée du système reste très réduite et pour assurer un fonctionnement correct sur l'aire de jeu, le module récepteur à changer plusieurs fois d'emplacement sur le robot. En effet, ces modules semblent être assez sensible à leur environnement et à leur orientation par rapport à l'émetteur. Ceci est sans doute en partie dû aux antennes qui ne sont certainement pas optimales (longueur, formes, accord ...) rendant ainsi la réception difficile.

De plus, la puissance d'émission de ces modules est très faible rendant là encore la transmission sur des distances importantes délicate. En outre, il est probable que le choix (involontaire) d'une modulation ASK boiteuse à 57 600 bps rende le système encore plus incertain. Une modulation OOK à 19 200 kps donnerait probablement de meilleurs résultats mais là aussi sans aucune certitude.

Dans son état actuel, le système est suffisant pour faire ce qu'il fait c'est à dire transmettre toujours dans le même sens des données permettant la localisation de l'adversaire. En effet, si l'on perd certaines de ces données pendant un moment, ce n'est pas catastrophique. En revanche, la communication apparaît trop peu fiable pour assurer un bon transfert de données bidirectionnel. De fait, même en répétant le message plusieurs fois, on n'est pas certain que celui-ci sera reçu. On pourrait bien sûr mettre en place un protocole avec acquittement des messages permettant de s'assurer qu'ils ont été reçus mais dans certaines conditions, la communication semble vraiment trop peu fiable pour ce type de transfert.

Bien sûr, il s'agit d'une communication HF et les aléas qui en découle seront toujours présents, c'est pourquoi un protocole de vérification comme celui qui a été mis en place sera toujours nécessaire. Néanmoins, il est très certainement possible d'améliorer le dispositif actuel pour le rendre plus fiable et ainsi permettre d'avoir une liaison assurant l'envoi d'ordres en HF et éventuellement du debug.

5.2 Les perspectives

L'un des problèmes majeurs du système qui le rend peu fiable est sa sensibilité. Pour pallier à ce problème, il y a à priori deux points susceptibles d'améliorer les résultats :

– La modification de l'antenne pour qu'elle soit parfaitement adaptée, accordée sur la fréquence porteuse et avec une bonne omnidirectionnalité. Cependant, il faut pour cela reconcevoir les modules en réalisant des simulations sous ADS et en prévoyant des points de mesures (fiches BNC) pour pouvoir régler l'antenne avec des appareils comme l'analyseur de réseaux vectoriel ou l'analyseur de spectre.

Une telle tâche n'est pas aisée, demande beaucoup de temps et augmentera le volume occupé par les modules sans garantie de résultats. Dans ce cas, il faudrait bien sûr revoir la modulation utilisée.

– L'autre voie est l'augmentation de la puissance d'émission des modules. Pour cela, on peut soit placer un amplificateur entre la sortie du module et l'antenne, soit réaliser soi-même un nouvel émetteur et n'utiliser les modules RFM que pour la réception. Pour le premier cas, RFM propose quelques schémas et il existe de nombreux schémas d'amplificateurs à 868 MHz mais là encore, il faut totalement reconstruire les modules. Dans le deuxième cas, on peut réaliser un émetteur relativement puissant en réalisant simplement un oscillateur à 868 MHz stabilisé par un quartz et dont l'oscillation est contrôlée par le train binaire. Quand on émet un 1, l'oscillateur est activé et la porteuse est émise. Quand on émet un 0, l'oscillateur est arrêté et la porteuse n'est pas émise. Il s'agit typiquement d'une modulation OOK. On peut alors utiliser les modules RFM en réception car ils intègrent à ce niveau de nombreuses fonctionnalités difficiles à mettre en oeuvre. Il existe là aussi des schémas de ce type d'émetteurs et on peut se reporter aux cours de HF dispensé en I_1 et I_2 à l'E.S.E.O. Cette solution présente l'avantage de ne pas nécessiter la reconception des modules existants.

On peut aussi éventuellement se reporter sur d'autres modules assez semblables, un peu plus intégrés mais présentant souvent des débits inférieurs : Radiométrix, Aurel ... Néammoins, la puissance de ces modules reste toujours très modeste et il est probable que l'on soit confrontés aux mêmes problèmes.

Une autre solution pourrait consister à s'orienter vers des solutions HF beaucoup plus évoluées (Blue-Tooth ou IEEE 802.11) mais leur mise en oeuvre et leur encombrement est une tout autre histoire et ceci ne me semble pas être une bonne solution dans l'immédiat.