Rapport projet intégré

Antoine Aupée – Joachim Draps – Nathan Dwek

9 mai 2015

Table des matières

1 Introduction **1**
1.1 Analyse du cahier des charges 1

2 Communication **2**
2.1 Codage de l'ordre 2
2.2 Notre chaîne d'acquisition 2

3 Transmission des ordres entre les deux microcontrôleurs **3**
3.1 Première analyse 3
3.2 Configuration des modules UART 4

Table des figures **a**

Table des codes source **b**

Chapitre 1

Introduction

Le but de ce projet intégré est de doter un robot d'un système de contrôle à distance. La base mécanique du robot est fournie, et celui-ci est déjà muni première carte à microcontrôleur complète. Cette dernière permet d'une part de s'interfacer avec les moteurs et les encodeurs à partir du premier microcontrôleur, et d'autre part d'accéder à certaines pattes de celui-ci ainsi que des alimentations. Les autres blocs nécessaires doivent être conçus et implémentés par les étudiants et le code des microcontrôleurs doit être produit.

Ce rapport décrit la démarche adoptée, ainsi que les solutions techniques adoptées pour mener à bien ce projet. Dans un premier temps le cahier des charges doit être analysé pour diviser l'objectif final en différent sous-problème bien définis.

1.1 Analyse du cahier des charges

Le robot doit

Chapitre 2

Communication

Pour ce projet, notre robot doit être capable de répondre à trois ordres différents : avance tout droit, tourne à droite, tourne à gauche. Le signal audio que nous lui envoyons doit donc contenir cet ordre ainsi qu'une quantité associée, le nombre de centimètres à parcourir ou le nombre de degré qu'il doit tourner. Dans ce chapitre, nous allons développer le chemin que parcourt l'information entre le moment où elle est envoyée sous forme sonore par l'émetteur qui nous a été fourni et la réception par le microcontrôleur.

2.1 Codage de l'ordre

2.2 Notre chaîne d'acquisition

Chapitre 3

Transmission des ordres entre les deux microcontrôleurs

Ce chapitre couvre la transmission des trames démodulées par le premier microcontrôleur vers le deuxième microcontrôleur. Dans la section suivante, le format des entrées et sorties de ce bloc vont être détaillées.

3.1 Première analyse

Puisqu'on désire limiter le nombre d'opérations effectuées par le microcontrôleur effectuant le traitement du signal audio, les trames de 10 bits renvoyées par la fonction `fskDetector` sont envoyées telles quelles au deuxième microcontrôleur. L'entrée du bloc transmission est donc une trame de 10 bits. L'uart, implémenté en hardware des deux côtés est utilisé pour effectuer la transmission. Celui-ci utilise des trames de 8 ou 9 bits, et il faudra donc 2 trames d'uart pour transmettre une trame de FSK. L'émetteur et le récepteur sera donc logiquement des machines à état séquentielles. Plutôt que de simplement reconstituer la trame de 10 bits originale, on choisit que le récepteur renvoie directement d'une part les 2 bits de commande et d'autre part les 8 bits d'arguments contenus dans une transmission.

Les parties récepteur et émetteur vont être abordées en parallèle dans la suite, puisqu'elles sont fortement liées. Tout d'abord, la configuration des modules UART est examinée[1], ensuite, l'émetteur et le récepteur vont être construits.

[1] La plupart des paramètres doivent forcément être les mêmes des deux côtés de la transmission pour que celle-ci soit possible.

3.2 Configuration des modules UART

Pour que la communication soit possible, les deux UART doivent être en accord sur quatre paramètres : le Baud Rate, la polarité des ports `TX` et `RX`, le format de trame, et enfin le protocole d'envoi de trame. Ces paramètres sont fixés dans le code source 3.1, qui est donc commun aux deux microcontrôleurs.

```
/*Extrait de initUart*/
void initUart(void){
    //Config Générale
    U1MODEbits.IREN = 0;//IRDA off.
    U1MODEbits.UEN = 0b00;//Seuls les ports U1TX et U1RX sont utilisés.
                          //=>il ne faut pas config l'hardware flow-control
    U1MODEbits.LPBACK = 0;//0 :inter uC. 1 : test uC vers lui même.
    U1MODEbits.ABAUD = 0;//Auto Baud off.
    U1MODEbits.BRGH = 1;//16coups de clock par bit envoyé
    //Plus robuste (3 samples par bit) et de toute façon on a un petit baudrate.
    U1BRG = BRGVAL;//Fixe le baud rate par la longueur du timer lié
    //BRGVAL = f_uC/(4×f_Baud) − 1
    U1MODEbits.PDSEL = 0b01;//8bit data, bit de parité (paire)
    U1MODEbits.STSEL = 0;//1 stop bit.

    //Polarité
    U1STAbits.UTXINV = 1;
    U1MODEbits.URXINV = 1;//tout actif à l'état haut

    //Routage des ports TX et RX vers les pattes utilisées : propre à chaque μC

    //Start uart et ses composants
    U1MODEbits.UARTEN = 1;//Active l'uart 1
    U1STAbits.UTXEN = 1;//UART prend le controle des ports
}
```

Code source 3.1 : Configuration commune aux deux UART

Le Baud Rate est fixé à 9600. Cette valeur rend la transmission d'une instruction (deux trames) pratiquement instantanée par rapport aux autres constantes de temps en présence ($f_{regul} = 100\,\text{Hz}$, $f_{symbol} = 10\,\text{Hz}$), tout en étant suffisamment basse pour permettre d'utiliser l'horloge de l'UART en *16X speed mode*, ce qui diminue la probabilité d'erreur car chaque bit est alors échantillonné trois fois au lieu d'une.

Les trames sont choisies longues de 8 bits avec un bit de parité et terminées par 1 stop bit. Passer à 9 bit de longueur de trame, mais sans bit de parité, ne présente pas d'intérêt puisqu'il faut toujours envoyer deux trames pour transmettre une instruction complète (10 bits). La détection rudimentaire d'erreur fournie par le bit de parité est donc légèrement préférable. La polarité n'a elle aucune importance, du moment qu'elle est identique de part

et d'autre d'une ligne TX(μC_1)-RX(μC_2).

Vu les très faibles contraintes sur l'UART, le risque d'erreur et la probabilité que les FIFO de réception et transmission se remplissent est pratiquement nul, on peut donc se contenter du protocole le plus simple avec seulement deux fils RX et TX et pas de *flow control*.

Table des figures

Table des codes source

3.1 Configuration commune aux deux UART 4