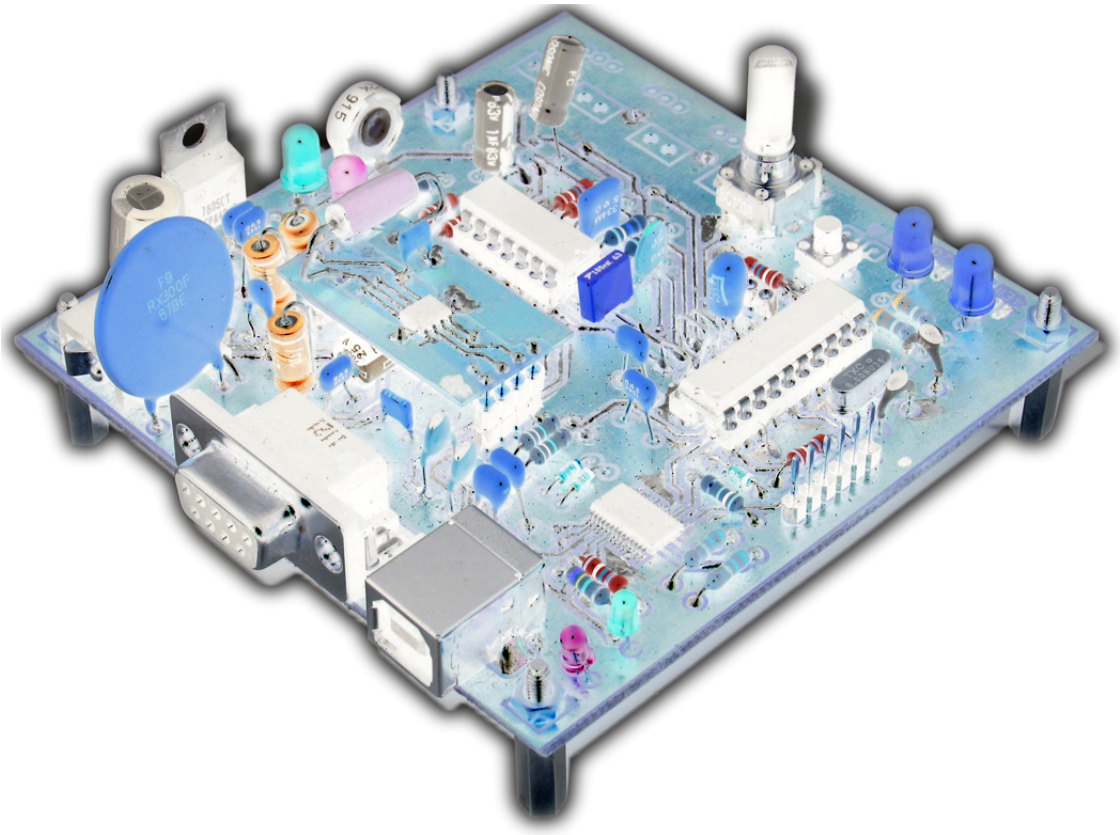
Projet Pluridisciplinaire
Solution Numérique



Xavier GALZIN, Stanislas BERTRAND, Romain DESILLE, Frédéric MESLIN

23 mai 2012

Table des matières

Introduction

Xavier

1 Communication

La partie communication du projet peut paraître secondaire quand on considère l'application dans sa globalité. Pour quelles raisons un luminaire aurait il besoin de communiquer avec un dispositif informatique ? L'intérêt semble limité ...

Dans les faits, cette fonctionnalité a été implémentée dans une optique d'assistance au développement. Elle s'est avérée utile pour mettre au point l'asservissement numérique et faciliter le réglage des coefficients de la fonction de transfert du correcteur. Elle permet aussi désormais de collecter diverses informations liées à l'automatique pour analyser la qualité (stabilité) de l'asservissement. On imagine que cette liaison série ne sera pas intégrée dans le produit final, sauf si les spécifications devaient évoluer vers un appareil nécessitant de la connectivité. On peut penser à l'intégration dans un réseau domotique via des technologies Zigbee ou même au travers le Wifi.

La communication a aussi un interêt pédagogique : nous avons fait le choix d'établir un protocole dédié sur une liaison de type série prise en charge par une connection USB. Nous avons préféré cette solution à celle présentée dans la partie MCSE du projet pour plusieures raisons qui vont être détaillées.

1.1 Choix de la liaison USB

Dans l'univers de la communication péri-informatique, la liaison USB est figure souveraine. La première version de cette spécification a vue le jour en 1996 dans l'objectif de remplacer progressivement les anciennes connectiques lentes et incompatibles entre elles.

Ce document normalise à la fois :

- un protocole complet de communication série maitre vers esclaves
- le matériel nécessaire pour faire transister l'information (cable, prises ...)
- toutes les caractéristiques électriques et mécaniques associées
- des classes de pilotes standards couvrant un nombre important d'applications génériques.

FIGURE 1: Logo USB 2.0

Notre application étant destinée à communiquer avec un ordinateur et l'USB fournissant une classe générique de communication série équivalente à un port COM virtuel, nous avons choisi d'opter pour cette solution.

1.2 Utilisation d'un contrôleur USB - Série

Afin de bénéficier de la connectique USB sans entrer dans les détails de la sous-couche de communication, nécessitant des cycles de développement lourds, deux solutions s'offraient à nous :

1. Choisir un micro-contrôleur disposant de fonctionnalités USB
2. Utiliser un contrôleur USB externe

La première solution est la plus polyvalente car elle permet d'implanter n'importe quelle classe de périphérique dans le programme du micro-contrôleur en se basant sur une bibliothèque USB fournie par le constructeur. Ceci nécessite une étude approfondie de la documentation associée et la rédaction de descripteurs USBs. Etant donné l'aspect auxiliaire de la communication, ce surcroit de développement a été écarté.

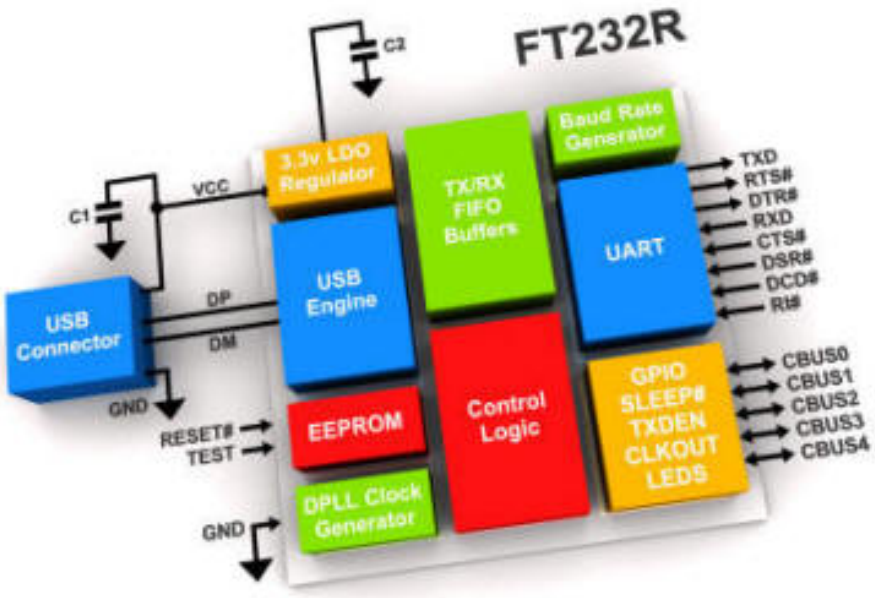


FIGURE 2: Diagramme interne du FT232

La seconde solution impliquant l'ajout d'un composant externe, au détriment du coût, a été retenue pour l'aisance de développement qu'elle apporte. Cette solution clef en main convertit une liaison série TTL en provenance du micro-contrôleur en liaison USB de manière presque transparente.

Le contrôleur circuit qui a été sélectionné est le FT232R de FTDI. Ce composant a été très simple à intégrer sur la carte du projet car il ne nécessite qu'une alimentation 5v et la liaison série pour fonctionner. De plus, il configure ses paramètres de transmission automatiquement en fonction de ceux communiqués par l'ordinateur à l'ouverture du port. Pour résumer ce composant est facile d'utilisation à défaut d'être peu couteux.

1.3 Elaboration d'un protocole

Dans le rapport de MCSE, nous avions présenté un protocole basique, exploitant un format de trames fixe de 3 octets consécutifs. Le premier contenant l'index de la commande et les deux suivants le corps de la donnée envoyée. Nous nous sommes rendus compte que ce protocole ne couvrait pas tous nos besoins et en particulier la possibilité de transmettre des messages d'erreurs ou de log.

Nous avons donc programmé un autre protocole, plus évolué et plus évolutif. Celui ci est capable de transmettre plusieurs types de trames à longueur variable et d'inclure des codes de gestion d'erreur de transmission (au besoin).

1.3.1 Configuration générale

Les paramètres généraux utilisés dans la transmission série sont les suivants :

Taille donnée 8 bits

Parité Aucune parité

Bits de stops 1 seul

Baudrate variable de 9600 à 115200

Le baudrate est variable car le micro-contrôleur dispose d'un module capable de déterminer automatiquement la vitesse de transmission de la liaison. La seule condition à respecter est l'émission du caractère ASCII "U" avant toute communication. Ce caractère correspond à un motif de niveaux

logiques hauts et bas alternés qui permettent au périphérique série du micro-contrôleur de calibrer son registre prescaler pour générer le bon baudrate.

1.3.2 Différentes trames

Nomenclature des symboles utilisés dans le protocole :

0 niveau bas logique

1 niveau haut logique

U niveau indéfini (non interprété par le logiciel)

X niveau à préciser selon l'utilisation

Les différentes trames disponibles sont les suivantes :

- **Trame de contrôle :**

typ	r/w	chr	ad4	ad3	ad2	ad1	ad0
1	X	X	ad4	ad3	ad2	ad1	ad0

typ : type de trame (0 : données — 1 : contrôle)

r/w : type de transfert (0 : lecture — 1 : écriture)

chr : type de données envoyées (0 : entiers 16 bits — 1 : chaine de caractères)

ad4 ... ad0 adresse de l'emplacement mémoire de la donnée (entière) envoyée (0 - 32)

La trame de contrôle est envoyée en premier et annonce une donnée de entière ou chaine de caractère. Elle est de type 0 pour la différencier des trames de données. Ceci permet de re-synchroniser la transmission en cas de perte d'un paquet de données. L'adresse de l'emplacement mémoire ne concerne pas les envois et réceptions de chaines.

En mode lecture (r/w = 0), le destinaire répond une trame de donnée entière ou plusieures trames de chaine directement après réception de la trame de contrôle. En mode écriture (r/w = 1), la trame de contrôle doit être immédiatement suivie d'une trame de donnée entière ou de plusieures trames de chaine.

- **Trame de donnée entière :**

MSB :	typ	b06	b05	b04	b03	b02	b01	b00
	0	u	u	u	d15	d14	d13	d12
MSB :	**typ**	**b06**	**b05**	**b04**	**b03**	**b02**	**b01**	**b00**
	0	u	u	u	d11	d10	d09	d08
LSB :	**typ**	**b06**	**b05**	**b04**	**b03**	**b02**	**b01**	**b00**
	0	u	u	u	d07	d06	d05	d04
LSB :	**typ**	**b06**	**b05**	**b04**	**b03**	**b02**	**b01**	**b00**
	0	u	u	u	d03	d02	d01	d00

typ : type de trame (0 : données — 1 : contrôle)

d15 ... d00 valeur de la donnée 16 bits

Cette trame contient une donnée entière 16 bits à charger dans l'emplacement mémoire précisé dans la précédente trame de contrôle. On remarque que les bits b06, b05 et b04 ne sont pas utilisés. Ils peuvent être mis à profit pour intégrer un code détecteur d'erreur. Cette amélioration est envisageable dans une future version du projet.

- **Trame de chaine de caractères :**

typ	r/w	log	ad4	ad3	ad2	ad1	ad0
0	ch6	ch5	ch4	ch3	ch2	ch1	ch0

typ : type de trame (0 : données — 1 : contrôle)

ch6 ... ch0 caractère table ASCII standard

Cette trame transmet caractère par caractère une chaine de texte. Pour distinguer les trames de données des trames de contrôle, on ne transmet que les 7 premiers bits de chaque caractère et on force le 8 ème à 0. Ce format autorise la représentation des caractères appartenant à la table ASCII standard mais pas à celle étendue. Pour terminer la transmission d'une chaine, il suffit d'envoyer une trame ne contenant que des zéros. Ceci équivaut à l'envoi du caractère de terminaison.

1.4 Implantation du protocole

Le présent protocole a été programmé dans la routine d'interruption de réception des données séries du micro-contrôleur ainsi que dans l'application interface. Il rend possible des liaisons bidirectionnelles où le micro-contrôleur comme l'application interface sont capables d'envoyer des trames de contrôle. En définitive, seul le sens application interface vers micro-contrôleur a été implémenté, l'autre sens ne présentant que peu d'intérêt.

2 Interface

Afin de faciliter la phase de débuggage et pour règler pratiquement les paramètres du correcteur numérique, nous avons développé une petite application interface en C#. Le programme consiste en une simple fenêtre présentant une section de connexion, une section de configuration et un espace de texte pour afficher les messages de logs.

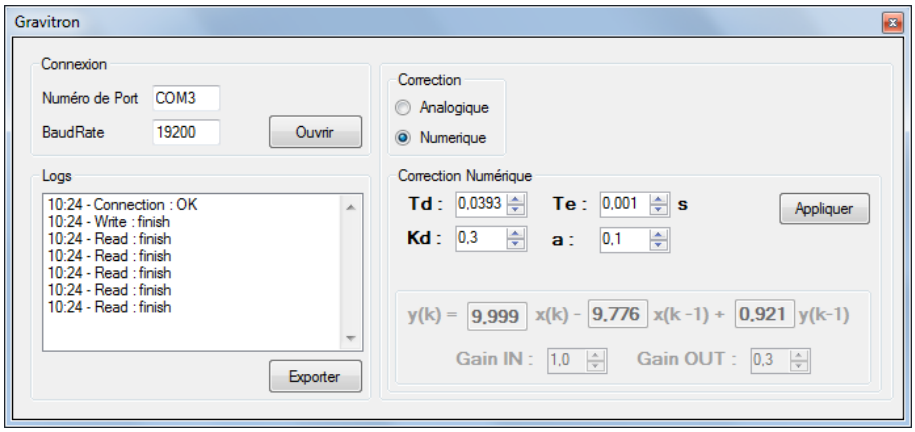


FIGURE 3: Interface de l'application de contrôle

Le panneau de connexion requiert de l'utilisateur le choix du port série et la vitesse de transfert. Une fois la connexion établie, les autres sections grisées deviennent accessibles.

La zone de messages logs liste chronologiquement les évènements de connexion et les messages textes envoyés par la maquette. Elle possède une fonctionnalité d'exportation permettant l'archivage des données dans un fichier texte.

3 Changement de Puissance

Xavier

4 Correction Numérique

Difference avec le rapport d'auto -¿ Foh vers Zoh -¿ Fréquence de travail

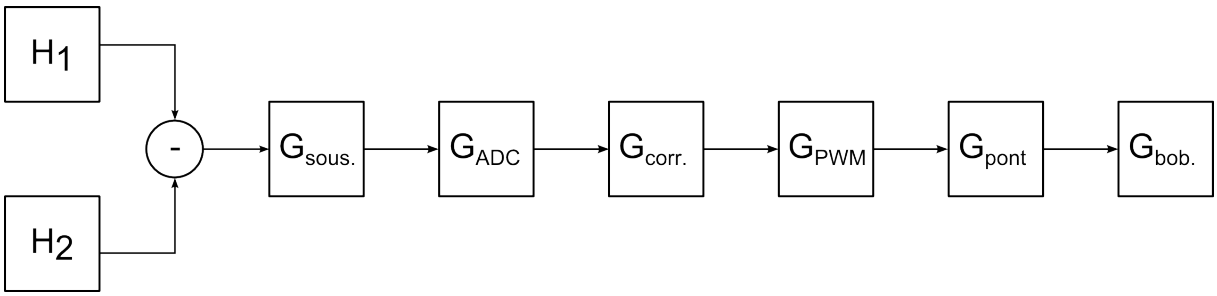


FIGURE 4: Chaine automatique des gains

Questions / causes probables : -¿ est-ce que le gain de l'avance de phase compte (analogique) ? -¿ mauvais correcteur -¿ mauvais modèle -¿ mauvais calcul -¿ saturation, écretage, arondis

Utilisation d'un outils : -¿ suivit des calculs via la série (diff, avance de phase, sortie)

dernière seance -¿ 1 gain du montage différentiel -¿ 2 validation avec matlab de la répartition du gain (av ph, num) -¿ 3 validation avec l'asservissement numerique du mercredi du solex -¿ 4 essaye d'amélioration de l'asservissement numérique

5 Répartition des gains

Xavier - Fred

5.1 Correcteur Analogique

5.2 Correcteur Numérique

Conclusion - Evolution

Xavier