TP2 PWM et RS232
Rapport de laboratoire

Ecole supérieure

Électronique
MINF
SLO2

TP2 PWM et RS232

Réalisé par :

Nicolas Besson
Vitor Coelho

A l'attention de :

Professeur M. Castoldi
Professeur M. Bovey

Dates :

Début du laboratoire : 08.01.2025
Fin du laboratoire : 30.01.2025

Table des matières

1. Introduction

Dans ce projet, nous reprenons les bases du TP1 pour ajouter une liaison RS232. Cette liaison permet de contrôler la vitesse d'un moteur DC et l'angle d'un servomoteur à partir des données reçues via un câble série D-SUB. Une application C# fournie sera utilisée pour tester et simuler la communication UART entre microcontrôleurs et/ou avec un PC.

Le projet inclut le développement de fonctions dans MPLABX pour traiter ces messages, ainsi que la réalisation d'un structogramme de la fonction GetMessage. Des tests et mesures seront réalisés pour analyser la performance de la communication série et valider le système.

2. Mesure

2.1. Schéma de mesures

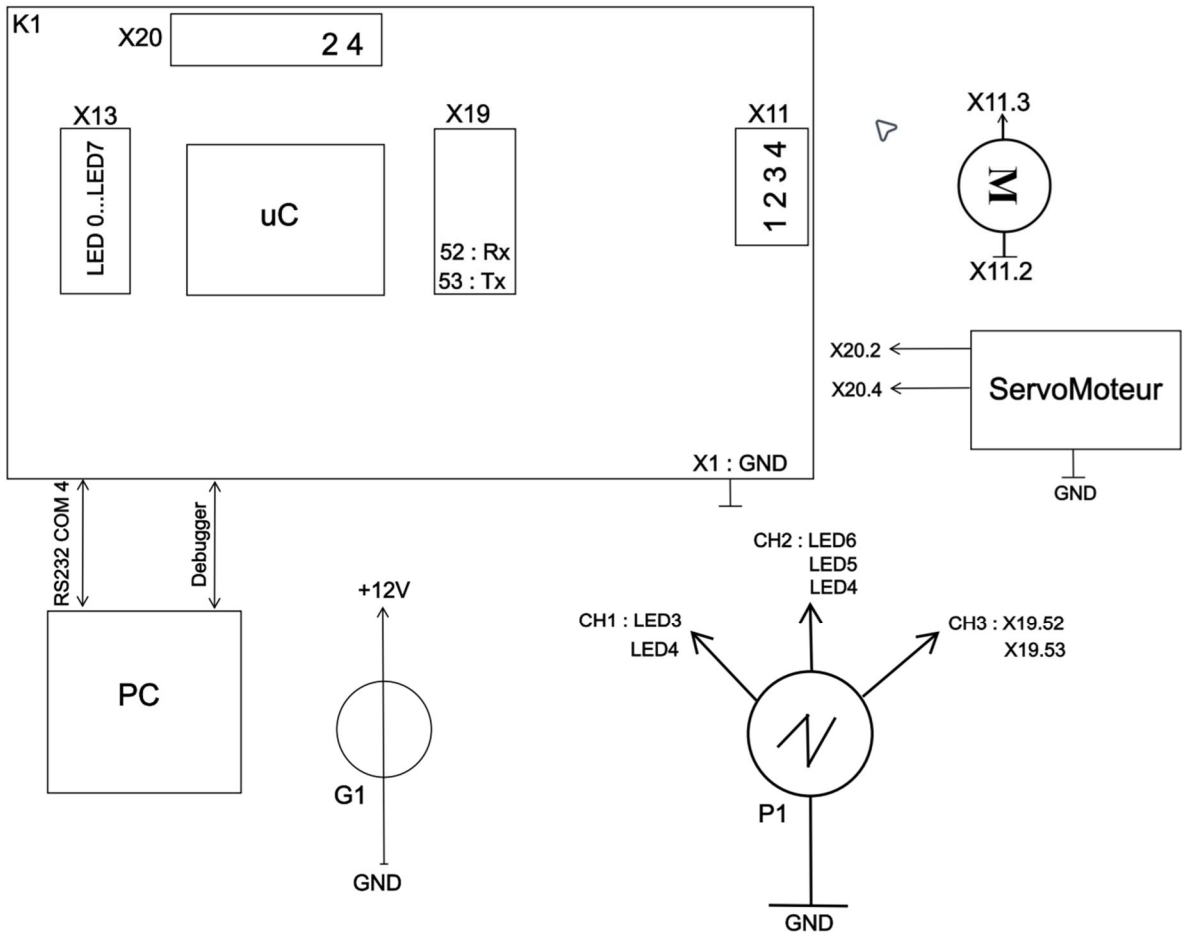

Figure 1 Schéma de mesures

2.2. Protocole de mesure

2.2.1. Mesure N°1

Branchement des sondes :

- Connecter la sonde du canal n°1 sur la LED n°3.
- Connecter la sonde du canal n°2 sur la LED n°4.
- Connecter la sonde du canal n°3 sur la pin n°52 (RX).

Paramètres oscilloscope :

- Trigger sur le flanc descendant du canal n°3.
- Mettre en place les paramètres du décodage UART (Rx) sur le canal n°3.

Paramètres application C# :

- Lancer l'application en se connectant avec le port série du PC (COM4 dans notre cas).
- Envoyer les trames en continue.

2.2.2. Mesure N°2

Branchement des sondes :

- Connecter la sonde du canal n°1 sur la LED n°3.
- Connecter la sonde du canal n°2 sur la LED n°5.
- Connecter la sonde du canal n°3 sur la pin n°53 (TX) du PIC32.

Paramètres oscilloscope :

- Trigger sur le flanc descendant du canal n°3.
- Paramétrer le décodage UART (Tx) sur le canal n°3.

Paramètres application C# :

- Lancer l'application en se connectant avec le port série du PC (COM4 dans notre cas).
- Envoyer les trames une par une.

2.2.3. Mesure N°3

Branchement des sondes :

- Connecter la sonde du canal n°1 sur la LED n°4.
- Connecter la sonde du canal n°2 sur la LED n°6
- Connecter la sonde du canal n°3 sur la pin n°52 (RX) du PIC32.

Paramètres oscilloscope :

- Trigger sur le flanc descendant du canal n°3.
- Paramétrer le décodage UART (Rx) sur le canal n°3.

Paramètres application C# :

- Lancer l'application en se connectant avec le port série du PC (COM4 dans notre cas).
- Envoyer les trames une par une.

3. Résultats

3.1. Mesure n°1

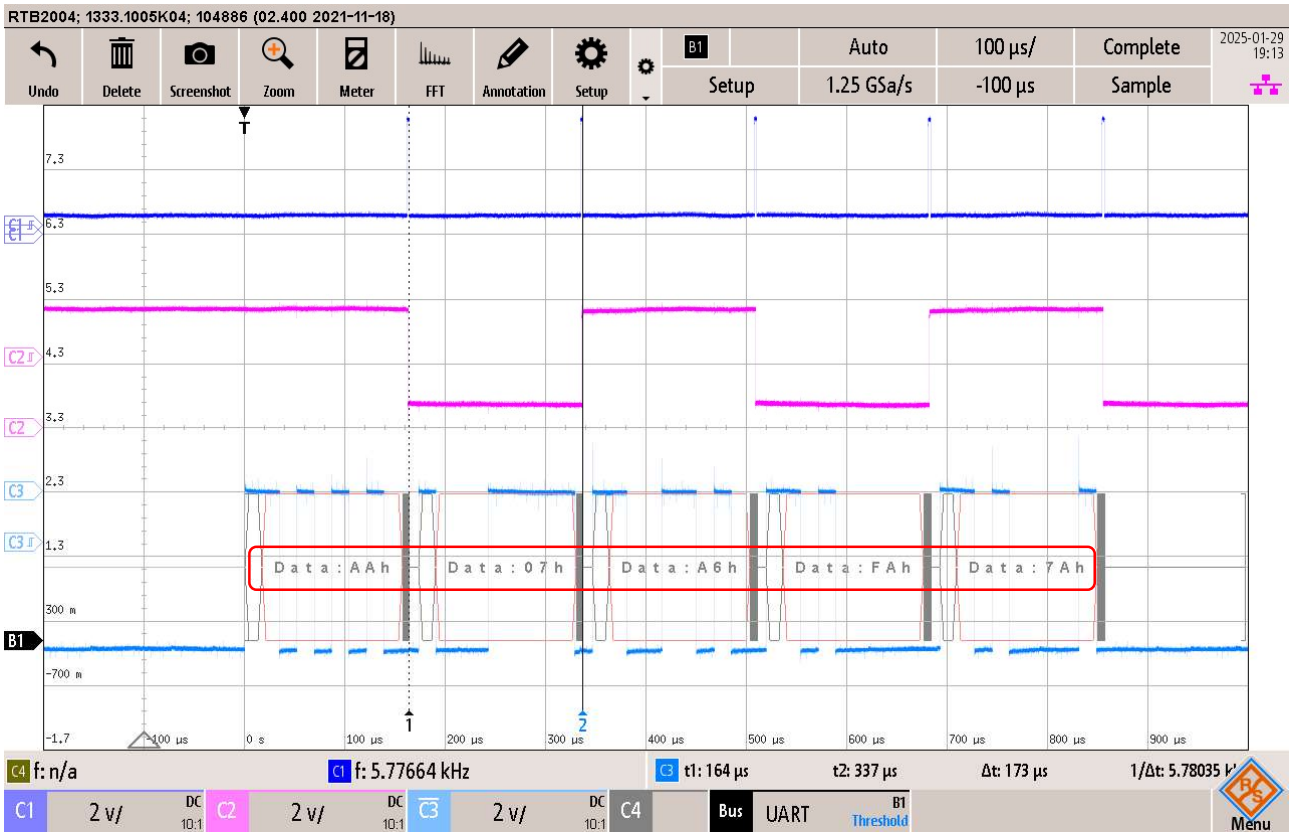

Figure 2 Interruption de réception et signal Rx

3.1.1. Analyse

D'après cette mesure, il est possible d'observer le message reçu par la carte depuis l'ordinateur (voir encadré en rouge). Nous constatons que chaque information du message commence légèrement après l'interruption de l'UART (CH1). Ce décalage est normal, car le message doit d'abord être transféré de la FIFO hardware vers la FIFO logicielle avant d'être traité. La LED n°3 (CH1) indique l'interruption de l'UART. Le temps entre deux interruptions a été mesuré à 173 [µs]. Sur le canal n°2, le changement d'état de la LED n°4 est correct, celui-ci se produit à chaque interruption de l'UART.

Ce comportement correspond au code programmé dans l'interruption de l'UART. Voici ci-dessous en figure 3, notre message envoyé depuis l'application. Nous constatons que nous arrivons à piloter notre carte PIC32 à partir de l'application. Nous avons également remarqué que notre affichage LCD se mets à jour en fonction de la trame envoyé.

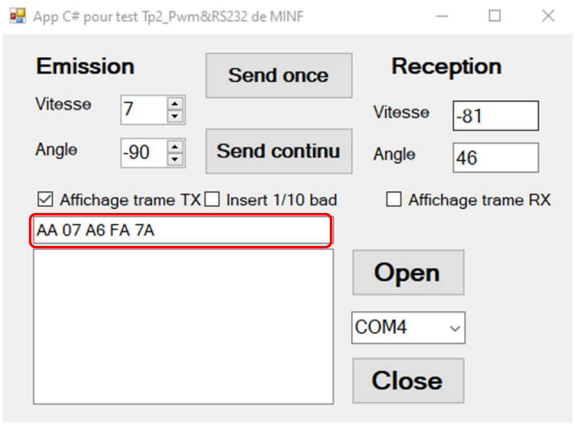

Figure 3 Observations depuis l'application

3.2. Mesure n°2

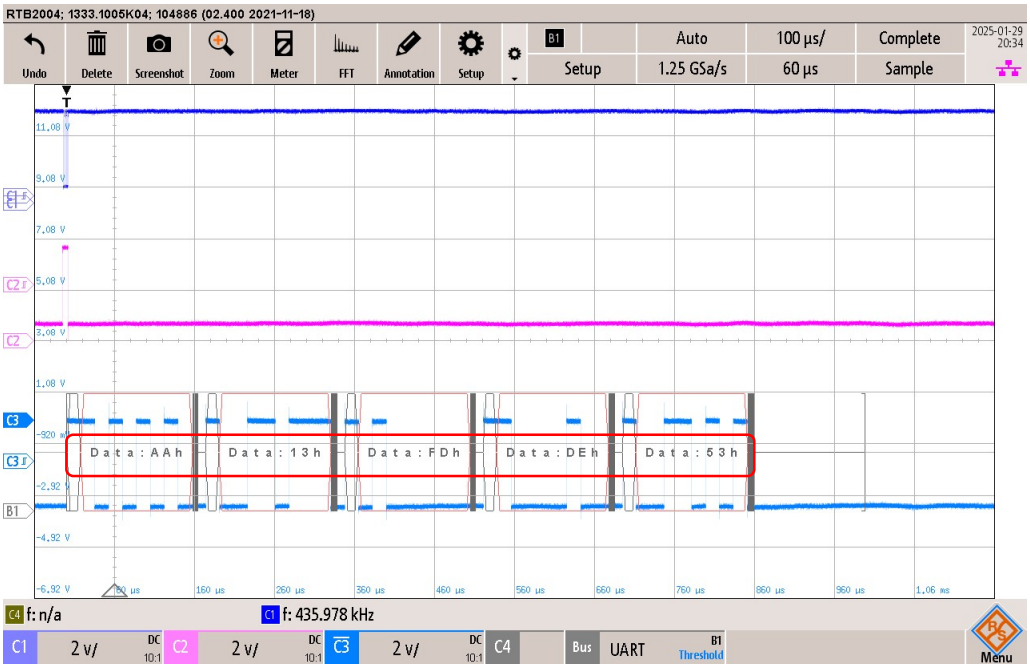

Figure 4 Interruption d'émission et signal Tx

3.2.1. Analyse

Sur la figure n°4, nous pouvons observer la trame que nous avons envoyé depuis la carte PIC32 au PC (encadré en rouge) la trame envoyée depuis la carte vers le PC correspond à la trame reçue dans l'application. L'interruption de l'UART (CH1) se déclenche avant l'envoi du message (CH3). Ce comportement est attendu, car avant l'envoi, il est nécessaire de vérifier la disponibilité de l'espace dans la FIFO ainsi que la capacité du PC à recevoir le message. Une fois ces conditions remplies, l'envoi est autorisé. Nous constatons que la LED n°5 change d'état à la fin de chaque interruption de l'UART, conformément au programme implémenté dans la carte. Pour finir, nous validons la réception correcte du message par le PC sur l'application. Voici le résultat obtenu :

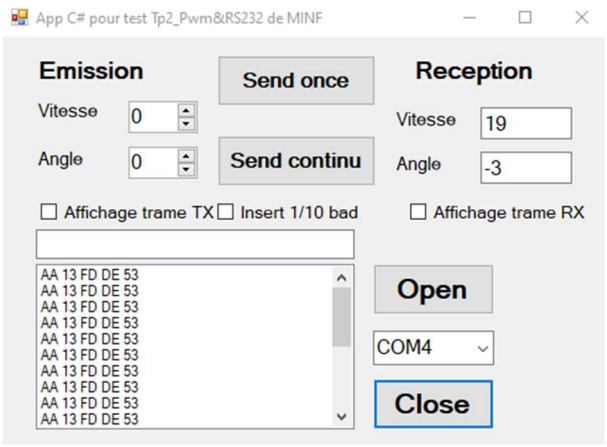

Figure 5 Observations depuis l'application

Nous constatons que nous retrouvons notre signal envoyé depuis notre carte tel quel.

3.3. Mesure n°3

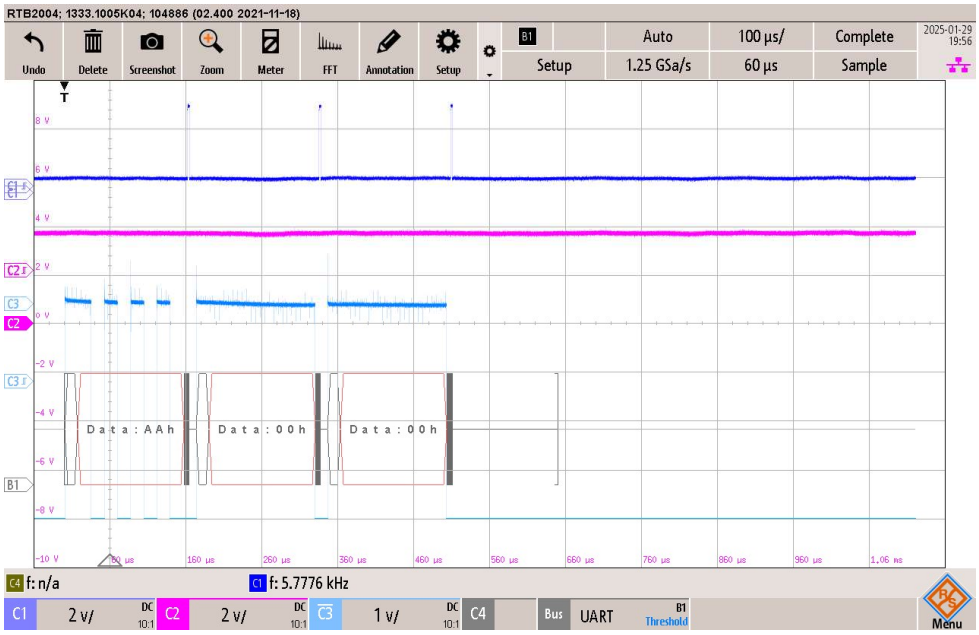

Figure 6 Message érroné

Sur cet oscillogramme le message est erroné afin de confirmer l'absence du CRC. Pour mieux visualiser cette suppression, le même message que dans la mesure n°1 a été envoyé depuis l'application. Nous constatons que la LED n°4 changes d'état (toggle) à chaque information du message. Comme expliqué précédemment, ce comportement est conforme aux attentes de ce TP. Afin de voir le réel impact d'un mauvais CRC, nous avons diminuer l'échelle de temps et regarder le comportement des LEDs indicatrices. Voici la figure 7 :

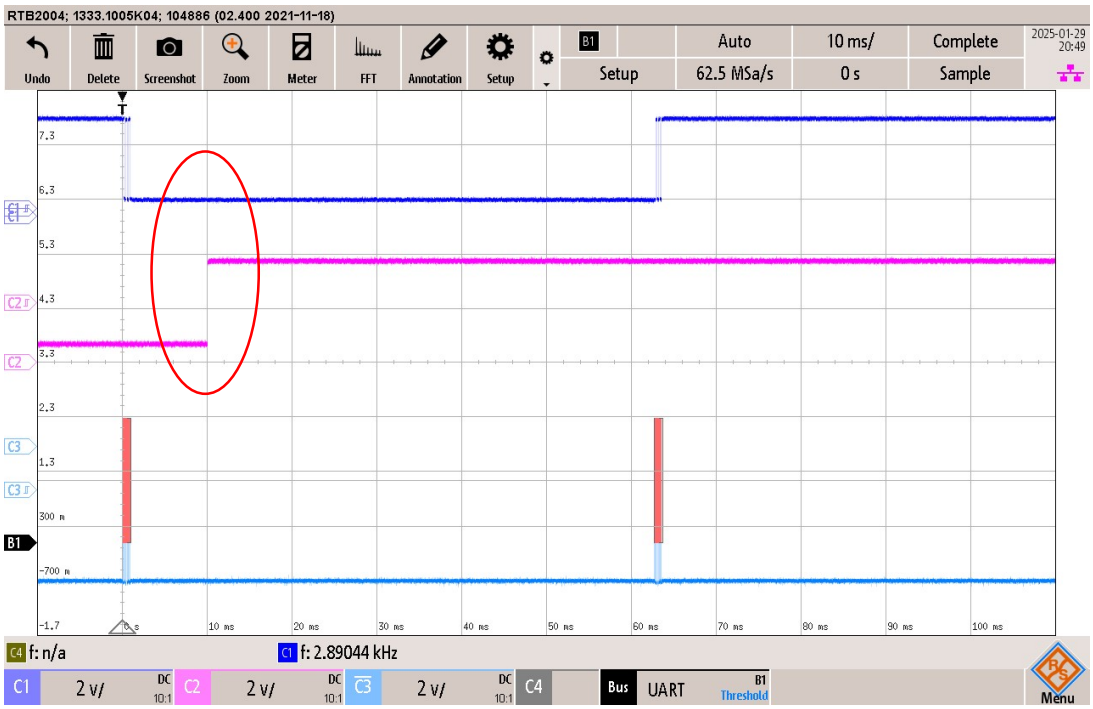

Figure 7 LED n°6

Vous pouvez remarquer que la LED n°6 ne change pas d'état immédiatement après la détection d'un CRC erroné. Cela vient du fait que le programme interprète le message comme contenant trois données, ce qui le fait considérer comme invalide. Lors de l'interruption suivante, le décalage du message reçu entraîne une incohérence dans les données, provoquant ainsi un CRC incorrect. C'est à ce moment-là que la LED n°6 toggle. Lorsque le CRC est invalide, l'affichage sur le LCD bascule alors en mode « Local Settings ».

4. Conclusion

Dans cette étude, les exigences du cahier des charges ont été respectées en s'appuyant sur les configurations du TP1, notamment celles liées aux timers et aux sorties OC. Une modification a été apportée en supprimant le Timer 4 via le configurateur graphique de MPLABX. Pour la communication UART, l'USART1 a été configuré selon les instructions fournies. Certaines parties du code ont ensuite été ajustées afin de répondre aux spécifications du TP.

Les tests effectués ont été organisés en trois protocoles distincts, chacun visant à analyser différents aspects de la transmission et du traitement des données. Ces essais ont permis de vérifier que le programme est capable d'envoyer et de recevoir des messages. Le programme répond aux exigences du cahier des charges et cette étude a permis de mettre en pratique la communication UART dans un environnement concret.

5. Annexes

5.1. Liste de matériel

Designation	Description	N° de serie
P1	Oscilloscope Rohde&Schwarz	ES.SLO2.05.01.11
G1	Alimentation USB via PC 5V	SLO-R110-M312
-	Debugger Microchip	R110-11
K1	Kit PIC32	ES.SLO2.00.05.41
M1	Moteur DC	-
M2	ServoMoteur	-

Tableau 1 Matériel utilisé