Polytech Sorbonne Université

Microcontroleur 2
COMMUNICATION SPI AVEC UN ACCELOROMETRE ET VISUALISATION DES DONNEES
Rapport

TP3 : SPI avec ADXL345

Élève(s) :
Daniel FERREIRA LARA

Enseignant(s) :
Arouna Darga
Francis Bras
Yamine Sellami

18 novembre 2024

Table des matières

1 Introduction

Ce rapport présente les travaux réalisés dans le cadre du TP 3 de Microcontrôleur 2. L'objectif principal est de développer des compétences avancées en programmation bas niveau et de créer des bibliothèques génériques pour les microcontrôleurs de la famille STM32F103. Ces bibliothèques sont conçues pour être réutilisables dans de futurs projets.

Le document est structuré de manière à mettre en évidence les objectifs, la méthodologie employée et les résultats obtenus, en privilégiant une approche technique et précise.

Tout est documenté sur : Repo Github.

2 Analyse Fonctionnelle

En première analyse, les broches SPI1 ont été sélectionnées, avec GPIOA 4 comme puce de sélection et, bien que nous ne les utilisions pas, deux broches possibles pour les interruptions

Broche ADXL345	Fonction	Correspondance STM32
CS	Chip Select (SPI Enable)	PA4
SCL/SCLK	Horloge SPI	PA5
SDA/SDI/SDIO	Données SPI (Entrée/Sortie)	PA7
SDO/ALT_ADDRESS	Données SPI (Sortie)	PA6
INT1	Interruption 1	GPIO avec inter. ext. (Ex : PA1)
INT2	Interruption 2	GPIO avec inter. ext. (Ex : PA0)
VS	Alimentation	3.3V
GND	Masse	GND

Table 1 – Entrees/Sorties

Après avoir lu la documentation de la fiche technique du capteur ADXL345, il a été possible de définir le tableau ci-dessous, qui est essentiel pour la communication entre le STM32 et le capteur.

Adresse	Nom du Registre	Fonction	Valeur de Réinitialisation
0x00	DEVID	Identifiant unique du périphérique	0xE5
0x2D	POWER_CTL	Contrôle l'activation du capteur (mode mesure/veille)	0x00
0x31	DATA_FORMAT	Configuration du format des données et de la plage d'accélération	0x00
0x32 - 0x37	DATAX0 - DATAZ1	Données d'accélération pour les axes X, Y, Z	0x00 (tous)
0x2C	BW_RATE	Contrôle la bande passante et le taux d'échantillonnage	0x0A
0x38	FIFO_CTL	Contrôle le fonctionnement du FIFO (Bypass, Stream, etc.)	0x00
0x30	INT_SOURCE	Identifie la source des interruptions	0x02

Table 2 – Registres du Capteur ADXL345

J'ai ensuite paramétré la communication en fonction des limites du capteur ADXL, qui a une fréquence maximale de 5 MHz.

Paramètre	Valeur recommandée	Description
Mode SPI	CPOL = 1 CPHA = 1	L'horloge est inactive à l'état haut, échantillonnage sur le front descendant
Fréquence d'horloge	5 MHz	Fréquence maximale pour la communication SPI avec le capteur
Durée de CS à SCK	500 ns	Délai minimum entre la mise à zéro de CS et l'activation de l'horloge SPI

Table 3 – Parametres SPI

3 Configuration du SPI sur le STM32

3.1 Broches du SPI2 sur STM32

Pour SPI2, nous utilisons d'autres broches, j'ai mis en place la configuration nécessaire à son bon fonctionnement, quelque chose de similaire serait fait pour les broches SPI1 équivalentes, déjà définies dans la table 1 ci-dessus.

Broche STM32 (SPI2)	Dans le protocole SPI	Fonctionnalité SPI	Configuration GPIO
PB12	CS	Sélection du périphérique	Mode Général, Push-Pull
PB13	SCK	Horloge SPI	Mode alternatif, Push-Pull, 50 MHz
PB15	MOSI	Permettre au maître d'envoyer les données	Mode alternatif, Push-Pull, 50 MHz
PB14	MISO	Permettre à l'esclave d'envoyer des données	Mode alternatif, Input Floating

TABLE 4 – Broches du SPI2

3.2 Configuration des Registres SPI

Vous trouverez ci-dessous les tableaux et la justification du choix des configurations des registres SPI_CR1 *et* SPI_CR2.

Bits	Nom	Description	Valeur à Configurer
2	MSTR	Sélection du mode maître ou esclave	0b 1
5 : 3	BR[2 : 0]	Vitesse de l'horloge SPI	0b 011
1	CPOL	Polarité de l'horloge (Idle High ou Low)	0b 1
0	CPHA	Phase de l'horloge (Front montant/descendant)	0b 1
9	SSM	Gestion logicielle du CS	0b 1
8	SSI	Permet de forcer le signal CS	0b 1

TABLE 5 – Registre SPI_CR1

Justification :

— **MSTR (Bit 2 : Sélection du mode maître/esclave)(0b1)** : Le STM32 est configuré comme maître SPI, car il doit initier les transferts de données avec le capteur. Ce rôle nécessite la mise à 1 du bit MSTR.

— **BR[2 :0] (Bits 5 à 3 : Vitesse de l'horloge SPI)(0b011)** : La vitesse de l'horloge SPI est choisie en fonction de la fréquence maximale supportée par le ADXL (5 MHz). Le diviseur 0b011 (division par 8 de 36 MHz, $\frac{36Mhz}{8} = 4,5MHz < 5MHz$) permet d'ajuster cette vitesse pour rester dans les limites spécifiées par le capteur.

— **CPOL (Bit 1 : Polarité de l'horloge)(0b1)** : La polarité *Idle High* est utilisée. Cette configuration dépend des spécifications du adxl, qui exigent cette polarité pour un fonctionnement correct.

— **CPHA (Bit 0 : Phase de l'horloge)(0b1)** : Le front descendant de l'horloge est utilisé pour capturer les données. Cette phase garantit la synchronisation correcte des données selon les exigences du capteur.

— **SSM (Bit 9 : Gestion logicielle du signal CS)(0b1)** : La gestion logicielle du signal `CS` (*Slave Select*) permet un contrôle précis par logiciel, ce qui est utile pour les protocoles personnalisés.

— **SSI (Bit 8 : Permet de forcer le signal CS)(0b1)** : Ce bit est activé pour forcer le signal `CS` (*Chip Select*) en état haut lorsque le `SSM` est utilisé, évitant ainsi toute sélection indésirable du capteur.

Bits	Nom	Description	Valeur à Configurer
2	SSOE	Activation automatique du signal CS	0b 1

TABLE 6 – Registre SPI_CR2

Justification :

— **SSOE (Bit 2 : Activation automatique du signal CS)(0b1)** : Le signal `CS` est géré automatiquement par le système SPI de la carte, lors de chaque transfert SPI, simplifiant la gestion du signal `CS` et assurant une meilleure synchronisation avec le capteur.

4 Implementation du Code SPI

Pour cette section, nous disposons des implémentations réelles du code du capteur, avec autant de généralisation que possible pour faciliter la réutilisation du code.

4.1 Transmission avec l'Analog Discovery 2

Tous les paramètres définis ci-dessus ont été utilisés, avec AD2 monté sur les broches SPI1 (il a également été testé sur SPI2 et a fonctionné parfaitement). Vous trouverez ci-dessous le circuit assemblé et les résultats obtenus avec des explications, respectivement :

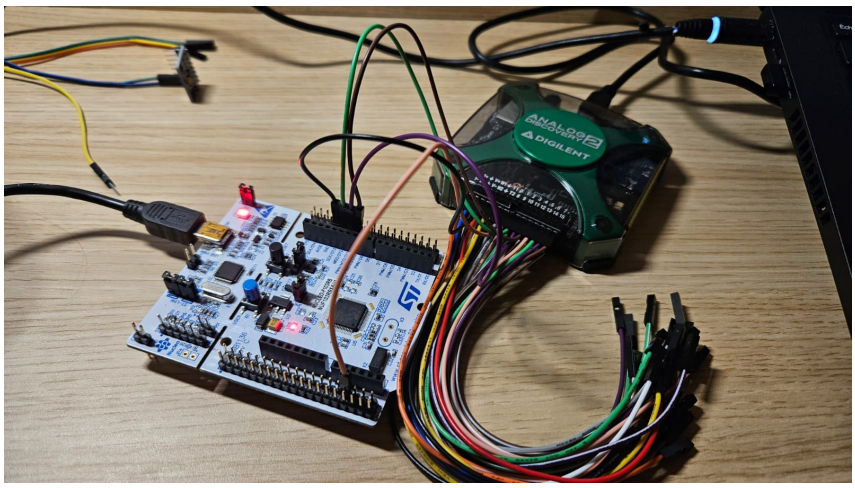

FIGURE 1 – Montage Analog Discovery 2

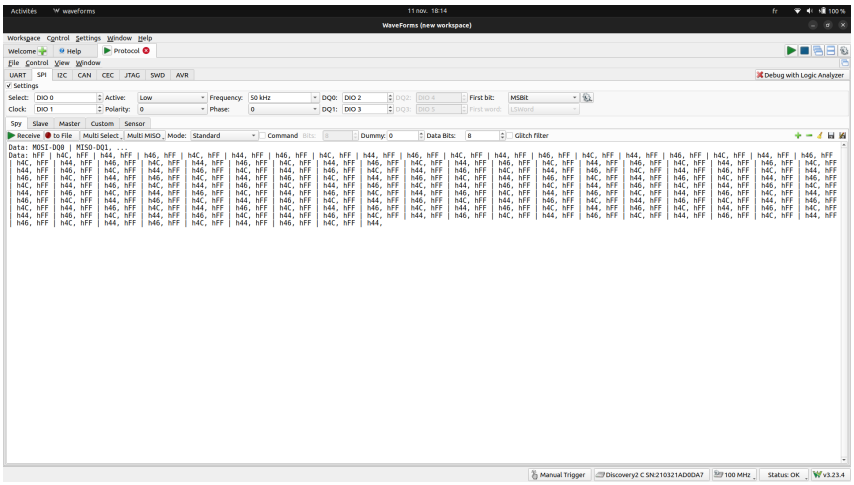

FIGURE 2 – Données reçues dans Waveforms

```
Data: MOSI-DQ0 | MISO-DQ1, ...
Data: hFF | h4C, hFF | h44, hFF | h46, hFF | h4C, hFF |
```

FIGURE 3 – Zoom sur l'image

Caracters envoyés :

$$ascii(D) = 0x44 = 68$$

$$ascii(F) = 0x46 = 70$$

$$ascii(L) = 0x4C = 76$$

Le code qui effectue le travail ci-dessus est le fichier `code_3.2.c`

4.2 Initialisation du capteur ADXL345

Pour l'initialisation, les procolos SPI et UART ont dû être configurés, respectivement SPI1 et USART2. Après avoir correctement connecté le capteur ADXL et le STM32, nous avons le circuit suivant :

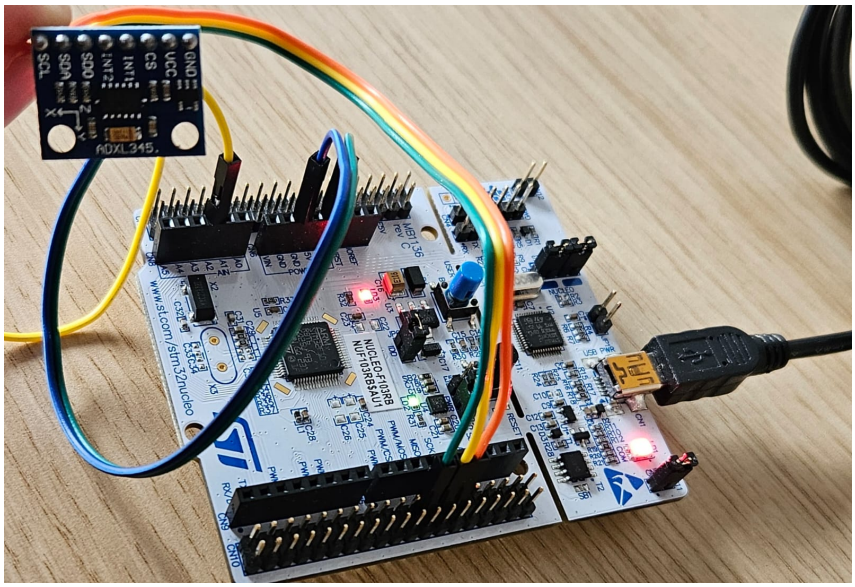

FIGURE 4 – Montage ADXL345

Après avoir créé la logique de décodage pour pouvoir lire un registre et l'envoyer en hexadécimal sur USART2, il a été possible de confirmer l'initialisation du capteur.

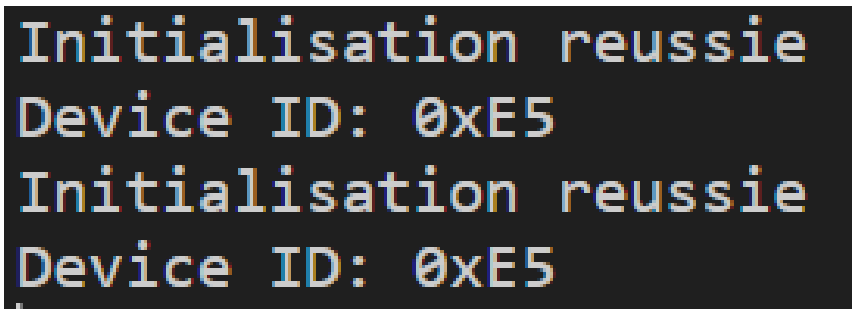

FIGURE 5 – Réponse via USART2

Le code qui effectue le travail ci-dessus est le fichier `code_3.2.c`

4.3 Verification de l'acquisition des données d'acceleration

Pour cette étape, j'ai créé deux codes de contrôle, l'un décodant et exprimant visuellement les données reçues à la suite de la lecture de tous les enregistrements :

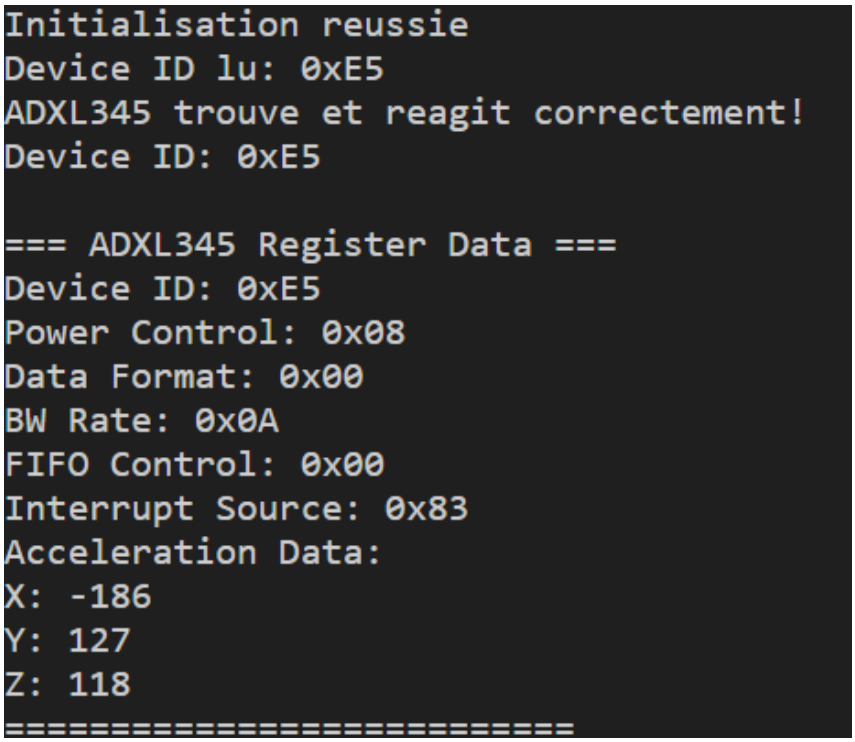

FIGURE 6 – Réponse décodé

Pour faciliter la compréhension, voici les valeurs brutes et la représentation hexadécimale correspondante avec les caractères obtenues à partir de la lecture :

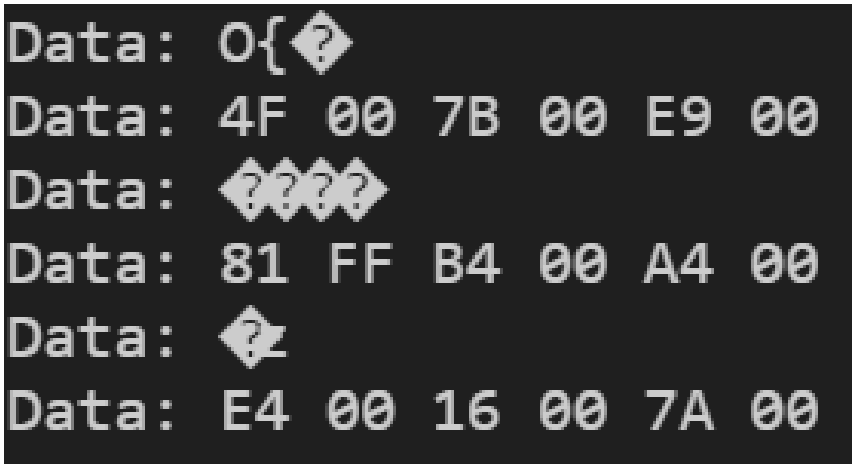

FIGURE 7 – Fausse Réponse Brute

En changeant l'encodage du moniteur série, nous pouvons voir ce que le STM32 envoie réellement comme message, voir l'image ci-dessous :

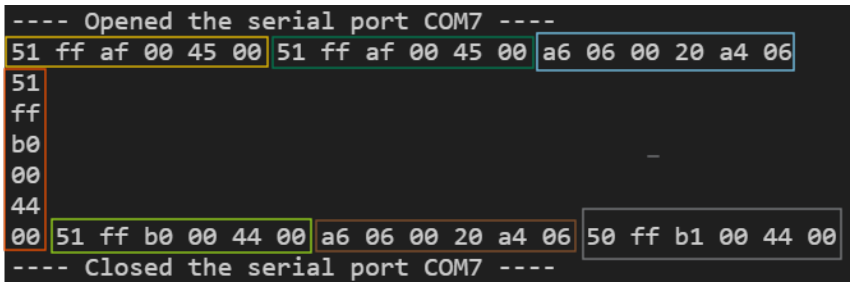

FIGURE 8 – Réponse Brute

5 Châine Complete

5.1 Implementation du Procotole

Pour traiter les données, un identifiant unique et un code de session sont indiqués. Comme ils n'ont pas été fournis, j'utiliserai des identifiants personnalisés basés sur mon numéro d'étudiant :

1. Mon numéro d'étudiant : 21**4215**08
2. ID unique choisi : 42 = 0x2A
3. Code de Session choisi : 15 = 0x0F

En suivant l'idée proposée par TP, je parviens à calculer la somme de contrôle et le paquet est formé correctement, parallèlement à ce que nous avons vu à la fin de la section précédente, nous pouvons voir que le nouveau paquet, avec l'octet de début et de fin, la somme de contrôle et les identificateurs a fait le paquet avec 12 octets, dont 6 sont les données d'accélération proprement dites, voir les images ci-dessous.

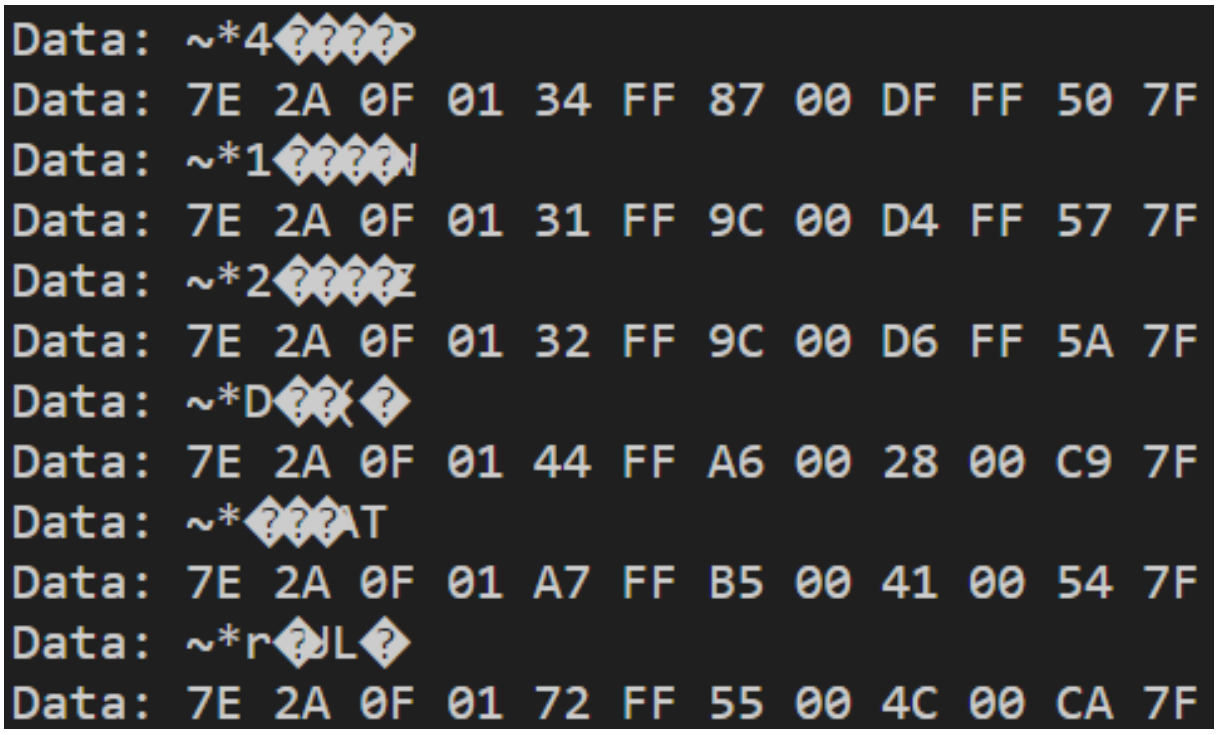

FIGURE 9 – Paquet final Debug

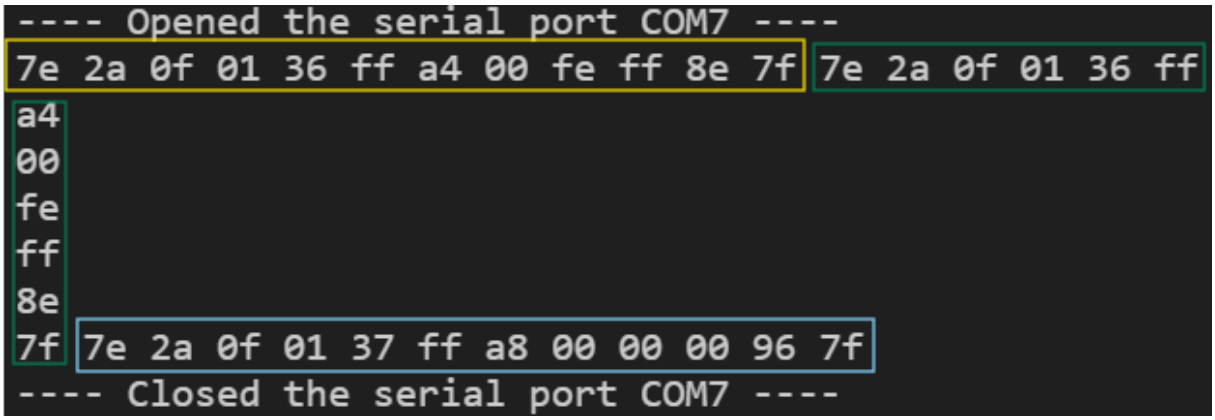

FIGURE 10 – Paquet final

Afin de vérifier si le paquet a été envoyé correctement, j'ai défini les valeurs X, Y et Z ci-dessous. Nous pouvons ensuite calculer la somme de contrôle et la manière dont le paquet devrait être décodé par le logiciel à l'avenir.

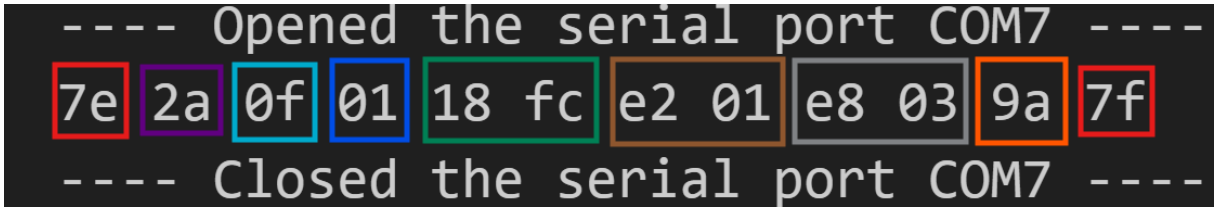

FIGURE 11 – Paquet final Fixé

1. Initial Byte = 0x7E
2. ID unique = 0x2A = 42
3. CodeSession = 0x0F = 15
4. ID Capteur = 0x01 = 1
5. Data X = 0x18 (LSB), 0xFC (MSB) = -1000
6. Data Y = 0xE2 (LSB), 0x01 (MSB) = 482
7. Data Z = 0xE8 (LSB), 0x03 (MSB) = 1000
8. Checksum = 0x9A = 154
9. Finale Byte = 0x7F

$$\text{Checksum} = 0x7E + 0x2A + 0x0F + 0x01 + (0x18 + 0xFC) + (0xE2 + 0x01) + (0xE8 + 0x03)$$

$$\text{Checksum} = 126 + 42 + 15 + 1 + (24 + 252) + (226 + 1) + (232 + 3)$$

$$\text{Checksum} = 126 + 42 + 15 + 1 + 276 + 227 + 235 = 922 = 0x39A$$

$$\text{Checksum} = 922 \bmod 256 = 154 = 0x9A$$

5.2 Recepction et Traitement dans l'ordi

En utilisant le paquet fixe que j'ai créé comme base, j'ai amélioré le code de traitement fourni avec plus de fonctions, vous permettant de vérifier les données reçues sur la base du protocole que j'ai créé et comment encoder et décoder l'information qui circule à travers lui.

En mode débogage, vous pouvez voir les valeurs reçues étape par étape, comme le montre la figure ci-dessous :

```
Index: 0, Byte reçu: 42
ID Étudiant : 42
Index: 1, Byte reçu: 15
Code de Session : 15
Index: 2, Byte reçu: 1
Code Capteur: 1
Index: 3, Byte reçu: 24
Index: 4, Byte reçu: 252
X: -1000
Index: 5, Byte reçu: 226
Index: 6, Byte reçu: 1
Y: 482
Index: 7, Byte reçu: 232
Index: 8, Byte reçu: 3
Z: 1000
Index: 9, Byte reçu: 154
Received Checksum: 154
Données valides. X: -1000 Y: 482 Z: 1000
Calculated Checksum: 154
calcul Checksum = 126 + 42 + 15 + 1 + 24 + 252 + 226 + 1 + 232 + 3 = 922
922 % 256 = 154
```

FIGURE 12 – Paquet reçu Debug

En mode normal, le terminal affiche les données comme suit :

```
ID Étudiant : 42
Code de Session : 15
Code Capteur: 1
X: -1000
Y: 482
Z: 1000
Received Checksum: 154
Données valides. X: -1000 Y: 482 Z: 1000
```

FIGURE 13 – Paquet reçu normal

Enfin, j'ai mis en place un moyen de visualiser les valeurs reçues par l'ordinateur en temps réel, de sorte que les valeurs reçues de chaque axe sont exprimées sous la forme de 3

barres centralisées, ainsi que les autres données du paquet dans le coin supérieur gauche. Les données ne sont affichées qu'une fois que le paquet a été correctement validé.

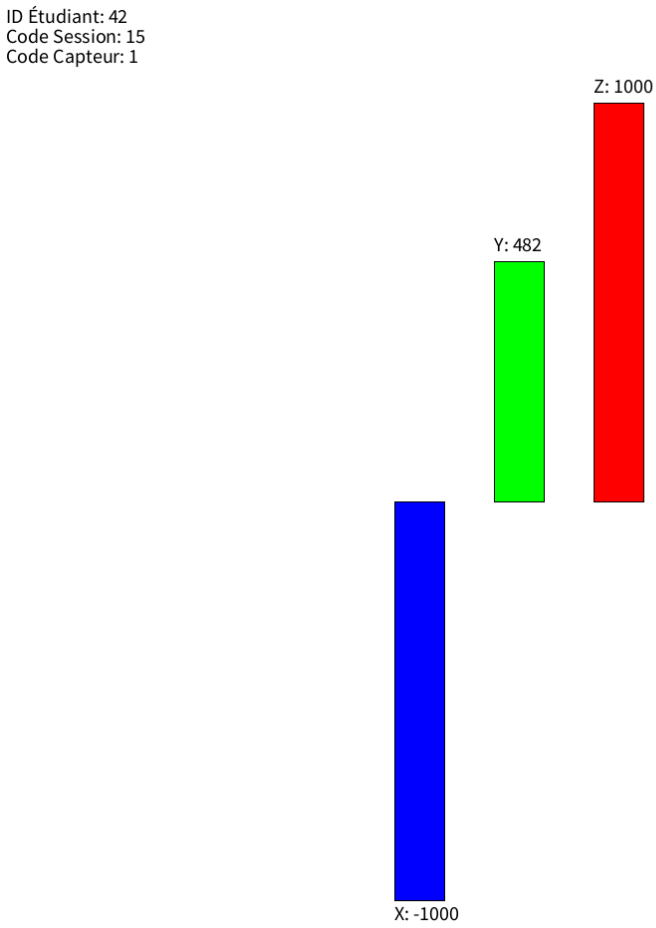

FIGURE 14 – Paquet reçu en visualization Processing

6 Conclusion

Ce TP a permis d'atteindre plusieurs objectifs importants dans la compréhension et l'implémentation de la communication SPI. La communication entre le STM32F103 et le capteur ADXL345 a été réalisée en respectant les contraintes techniques spécifiées. Le développement d'une bibliothèque générique et réutilisable pour la gestion du protocole SPI représente un atout significatif pour de futurs projets avec des microcontrôleurs STM32F103.

L'implémentation du protocole de communication complet, incluant la gestion des identifiants uniques, le calcul et la vérification des checksum, ainsi que la structuration des paquets de données, a démontré la maîtrise des concepts fondamentaux. La création d'une interface de visualisation en temps réel des données d'accélération sur les trois axes a permis de valider la fiabilité de la chaîne complète de communication. Les compétences acquises dans ce TP, particulièrement en programmation bas niveau et en gestion de protocoles de communication, constituent une base solide pour le développement de projets plus complexes utilisant des capteurs et des communications série.