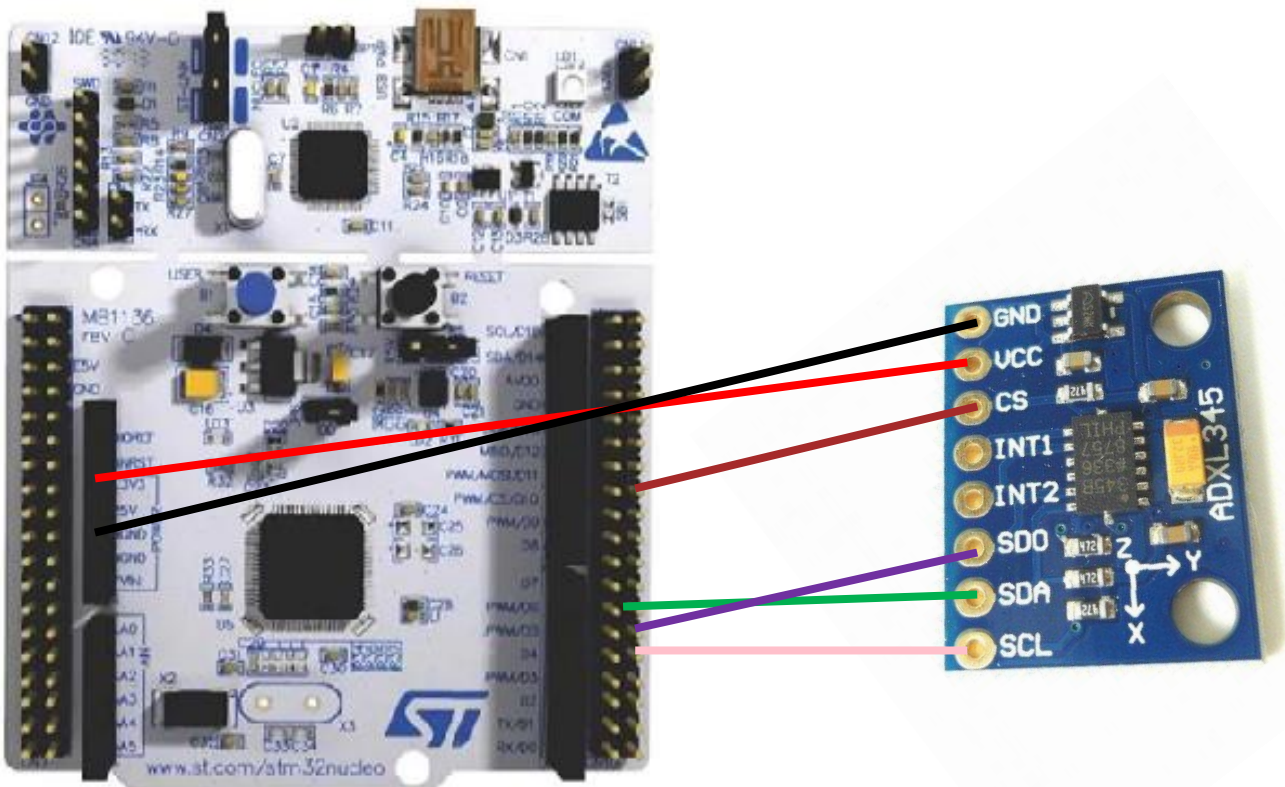


Compte-rendu du TP3/DM :

Communication SPI avec un accéléromètre et visualisation des données



Étudiant : Ayoub LADJICI

Spécialité : Électronique – Informatique

Enseignant référent : Arouna DARGA

Table des matières

I. Introduction.....	3
II. Analyse fonctionnelle.....	4
1. Tableau des entrées/sorties	4
2. Tableau des registres essentiels du capteur ADXL345	4
3. Tableau des paramètres SPI	5
III. Configuration du SPI sur le STM32	5
1. Broches du SPI2 sur STM32	5
2. Configuration des registres SPI.....	6
IV. Implémentation du code SPI2 avec validation par mesures	6
1. Implémentation du code SPI2.....	6
2. Test de transmission avec l'Analog Discovery 2	6
3. Initialisation du capteur ADXL345.....	7
4. Vérification de l'acquisition des données	8
V. Chaîne complète de transmission et visualisation des données d'accélération.....	8
1. Implémentation d'un protocole de transfert des données.....	8
2. Réception et vérification sur PC (Processing).....	9
3. Affichage des résultats	10

I. Introduction

L'objectif de ce TP consiste à mettre en place toutes les étapes d'une chaîne de communication, de l'acquisition des données d'un capteur d'accélération ADXL345, en passant par la transmission en utilisant un microcontrôleur STM32 jusqu'à la visualisation des données sur un terminal série Putty ou bien sous forme graphique sur le logiciel Processing.

Nous avons utilisé le protocole de communication SPI (Serial Peripheral Interface), qui est le plus simple pour des communications courte distance entre les microcontrôleurs et d'autres appareils. Le SPI offre une communication bidirectionnelle entre un maître (STM32) et un esclave (ADXL345). Cela nécessitera 4 fils pour générer ce processus, un fil pour sélectionner l'esclave (car il est possible que le maître communique avec plusieurs esclaves toutefois cela n'est pas notre cas), un fil pour générer l'horloge et avoir une communication synchrone, un fil pour la sortie du maître vers l'esclave (la ligne MOSI) permettant d'envoyer un message, une commande, un fil pour la sortie de l'esclave vers le maître (la ligne MISO) permettant de répondre à la commande du maître. Enfin, si le maître veut entamer une communication, il devra descendre la ligne SS et s'il souhaite mettre fin à la communication, il devra la remonter.

Nous avons également utilisé la communication série UART (ou plutôt l'USART du STM32) pour transmettre les données d'accélération vers des interfaces comme Putty ou Processing.

Pour observer les signaux échangés entre les différents composants, nous avons utilisé une Analog Discovery, qui est un très bon outil pour le débogage et la validation des différentes étapes.

II. Analyse fonctionnelle

1. Tableau des entrées/sorties

Broche ADXL345	Fonction	Correspondance STM32
CS	Chip Select (SPI Enable)	SPI2_NSS (PB12)
SCL/SCLK	Horloge SPI	SPI2_SCK (PB13)
SDA/SDI/SDIO	Données SPI (Entrée/Sortie)	SPI2_MOSI (PB15)
SDO/ALT_ADDRESS	Données SPI (Sortie)	SPI2_MISO (PB14)
INT1	Interruption 1	PC6 (au choix)
INT2	Interruption 2	PC8 (au choix)
VS	Alimentation	3.3V
GND	Masse	GND

2. Tableau des registres essentiels du capteur ADXL345

Adresse	Nom du Registre	Fonction	Valeur de réinitialisation
0x00	DEVID	Identifiant unique du périphérique	0b11100101
0x2D	POWER_CTL	Contrôle l'activation du capteur (mode mesure/veille)	0b00000000
0x31	DATA_FORMAT	Configuration du format des données et de la plage d'accélération	0b00000000
0x32 - 0x37	DATA0 - DATAZ1	Données d'accélération pour les axes X, Y, Z	0b00000000
0x2C	BW_RATE	Contrôle la bande passante et le taux d'échantillonnage	0b00001010
0x38	FIFO_CTL	Contrôle le fonctionnement du FIFO (Bypass, Stream, etc.)	0b00000000
0x30	INT_SOURCE	Identifie la source des interruptions	0b00000010

3. Tableau des paramètres SPI

Paramètre	Valeur recommandée	Description
Mode SPI	CPOL = 1, CPHA = 1 (page 8 doc)	L'horloge est inactive à l'état haut, échantillonnage sur le front montant
Fréquence d'horloge	Maximum 5 MHz (page 8 doc)	Fréquence maximale pour la communication SPI avec le capteur
Durée de CS à SCK	10 ns (durée du front descendant de CS à front descendant de SCLK)	Délai minimum entre la mise à zéro de CS et l'activation de l'horloge SPI

III. Configuration du SPI sur le STM32

1. Broches du SPI2 sur STM32

Broche STM32 (SPI2)	Nom correspondant dans le protocole SPI	Fonctionnalité SPI	Configuration GPIO
PB12	CS	Permet de sélectionner un esclave	Output mode, max speed 50 MHz, General purpose output push-pull
PB13	SCK	Permet de synchroniser les données	Output mode, max speed 50 MHz, Alternate function output push-pull
PB15	MOSI	Permet au maître d'envoyer des données à l'esclave	Output mode, max speed 50 MHz, Alternate function output push-pull
PB14	MISO	Permet au maître de recevoir des données de l'esclave	Input mode, Floating input (reset state)

2. Configuration des registres SPI

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

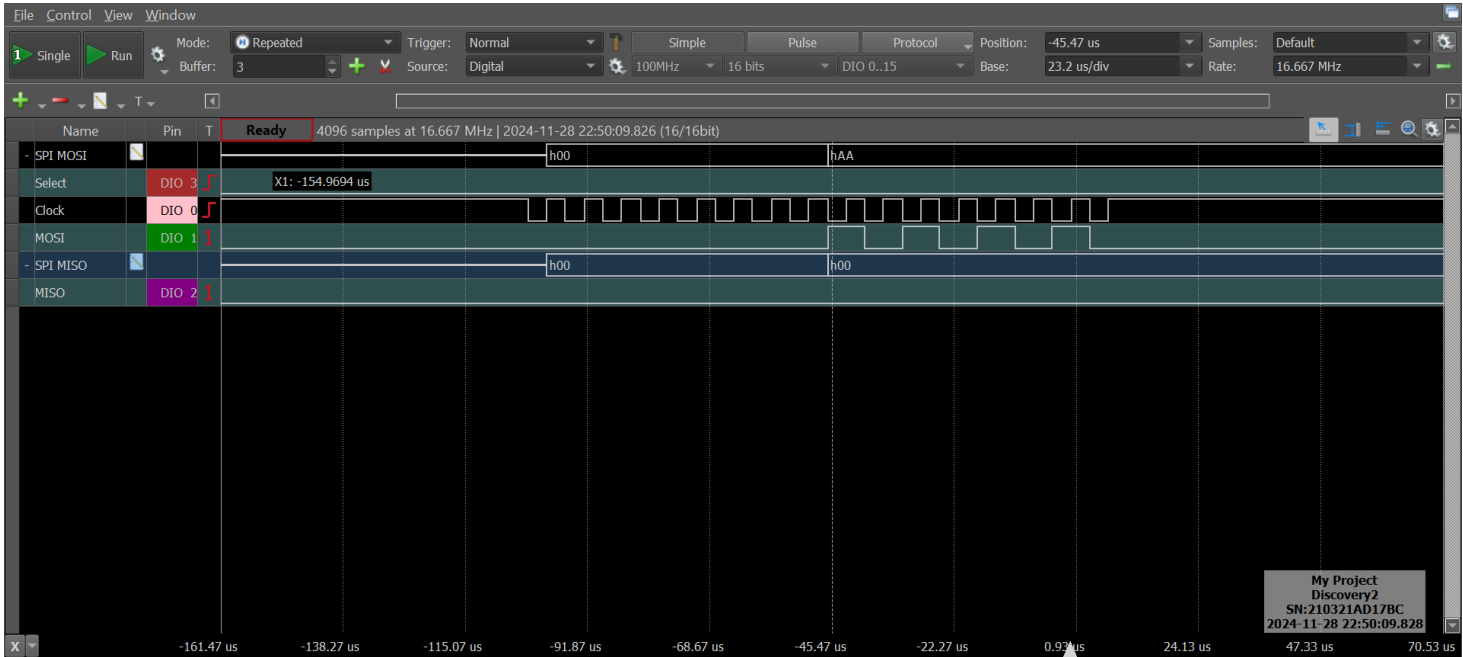
IV. Implémentation du code SPI2 avec validation par mesures

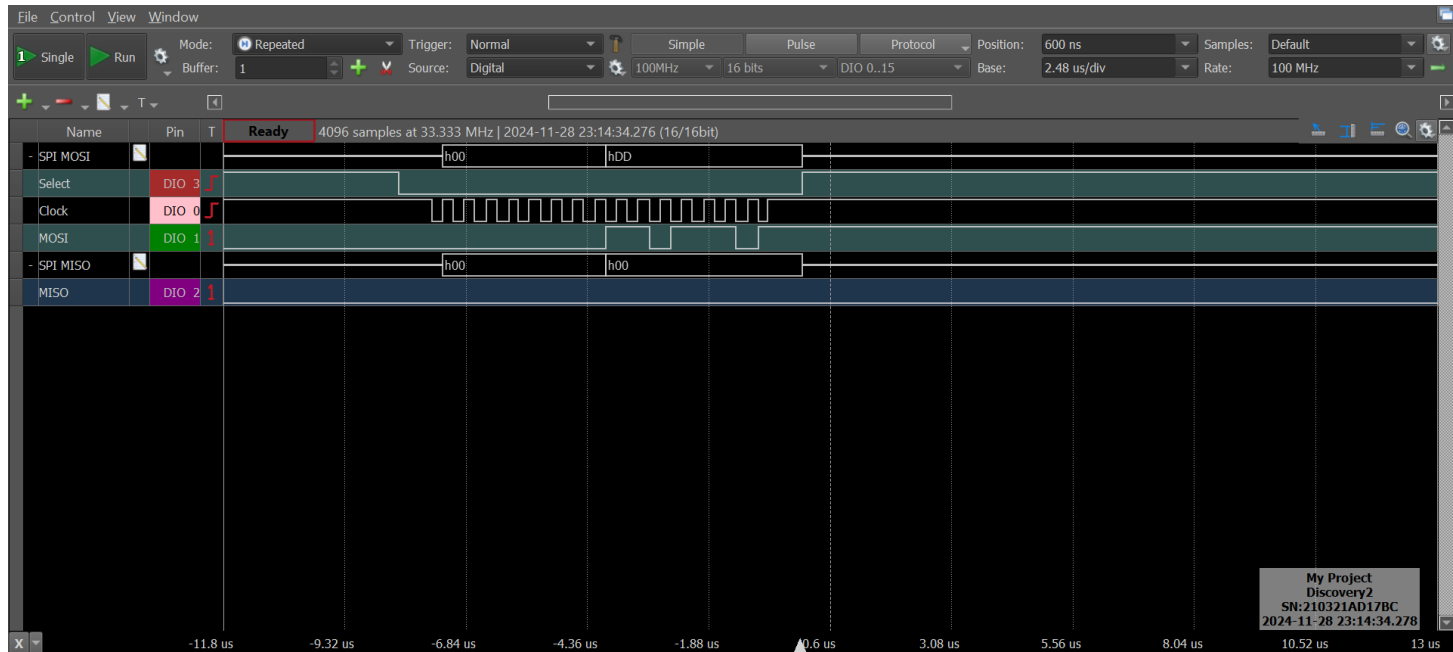
1. Implémentation du code SPI2

Vous pouvez trouver tous mes codes de ce DM dans le dossier sources. Le fichier de configuration du SPI2 sur le STM32 se nomme « initSPI2.c »

2. Test de transmission avec l'Analog Discovery 2

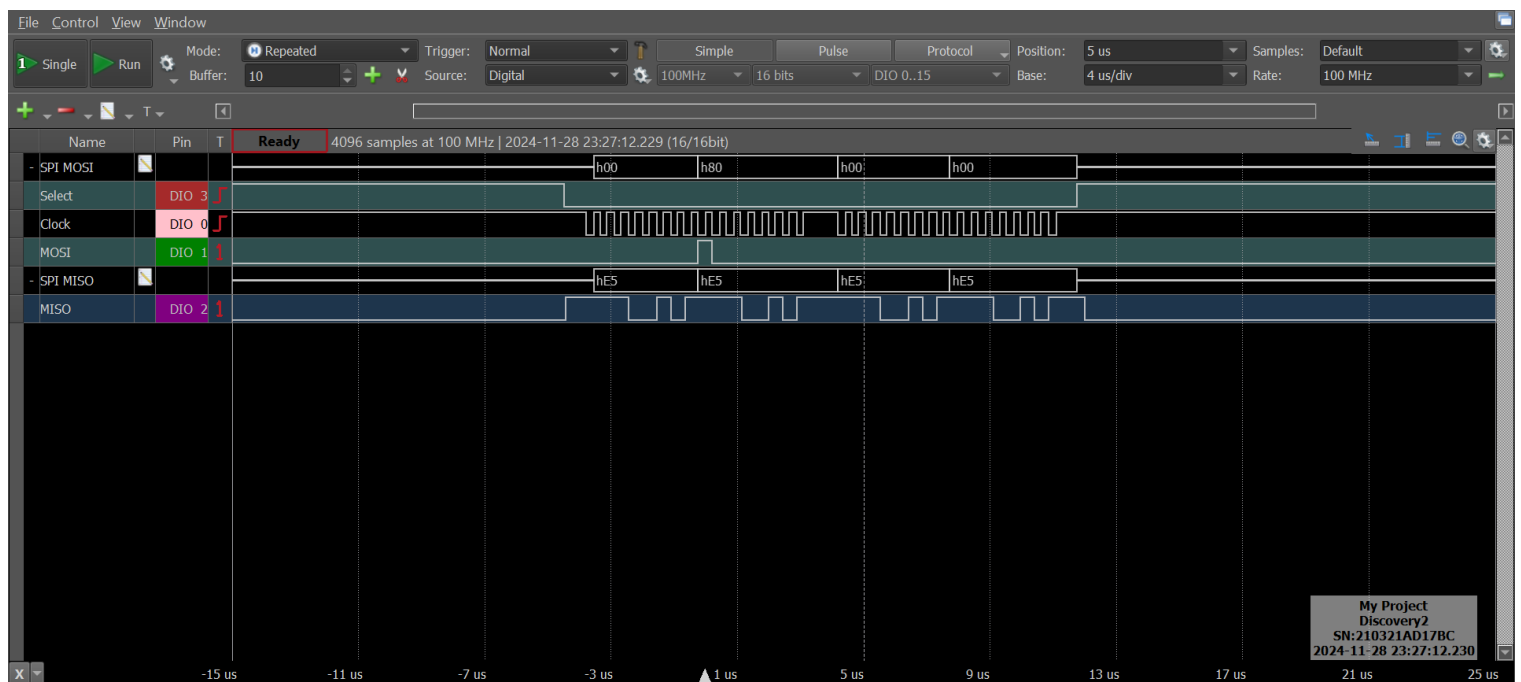
J'ai testé la communication SPI en envoyant un octet de test (0xAA) puis après chaque transmission, on met un délai pour permettre au périphérique de traiter la demande, et on incrémente la valeur à envoyer pour tester une séquence de plusieurs octets.





3. Initialisation du capteur ADXL345

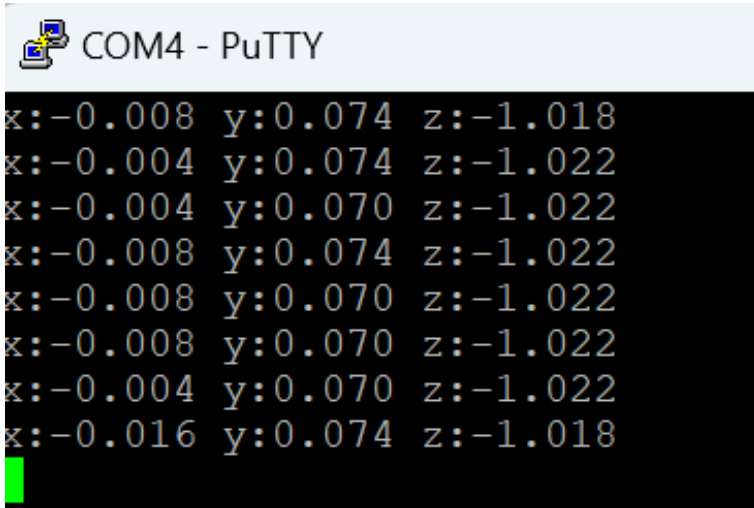
J'envoie une commande de lecture du registre DEVID, qui est localisé à l'adresse 0x00 pour lire l'identifiant du périphérique. L'esclave, en l'occurrence l'ADXL345, nous renvoie bien l'ID attendue qui est 0xE5 sur la ligne MISO.



De plus, j'ai utilisé un autre moyen de débogage dans le cas où WaveForms était surchargé, en écrivant une condition d'allumage de la LED PA5 si l'ID reçue est bien 0xE5.

4. Vérification de l'acquisition des données

En configurant l'USART2 sur mon STM32, j'ai pu communiquer les données de chaque axe de mon accéléromètre ADXL345 et les visualiser sur le terminal PuTTY. Sachant que j'ai positionné mon ADXL345 de sorte à ce qu'il soit immobile et que l'axe z soit dirigé vers le haut. Les données obtenues sont sensiblement bonnes, il n'y a quasiment pas d'accélération sur l'axe X et Y tandis que sur l'axe Z, nous avons une valeur proche de -1g montrant bien que la gravité agit en sens opposé.



```
COM4 - PuTTY
x:-0.008 y:0.074 z:-1.018
x:-0.004 y:0.074 z:-1.022
x:-0.004 y:0.070 z:-1.022
x:-0.008 y:0.074 z:-1.022
x:-0.008 y:0.070 z:-1.022
x:-0.008 y:0.070 z:-1.022
x:-0.004 y:0.070 z:-1.022
x:-0.016 y:0.074 z:-1.018
```

V. Chaîne complète de transmission et visualisation des données d'accélération

On passe directement à l'étape 2, étant donné que nous avons déjà fait l'acquisition et la lecture des données lors de la précédente partie.

1. Implémentation d'un protocole de transfert des données

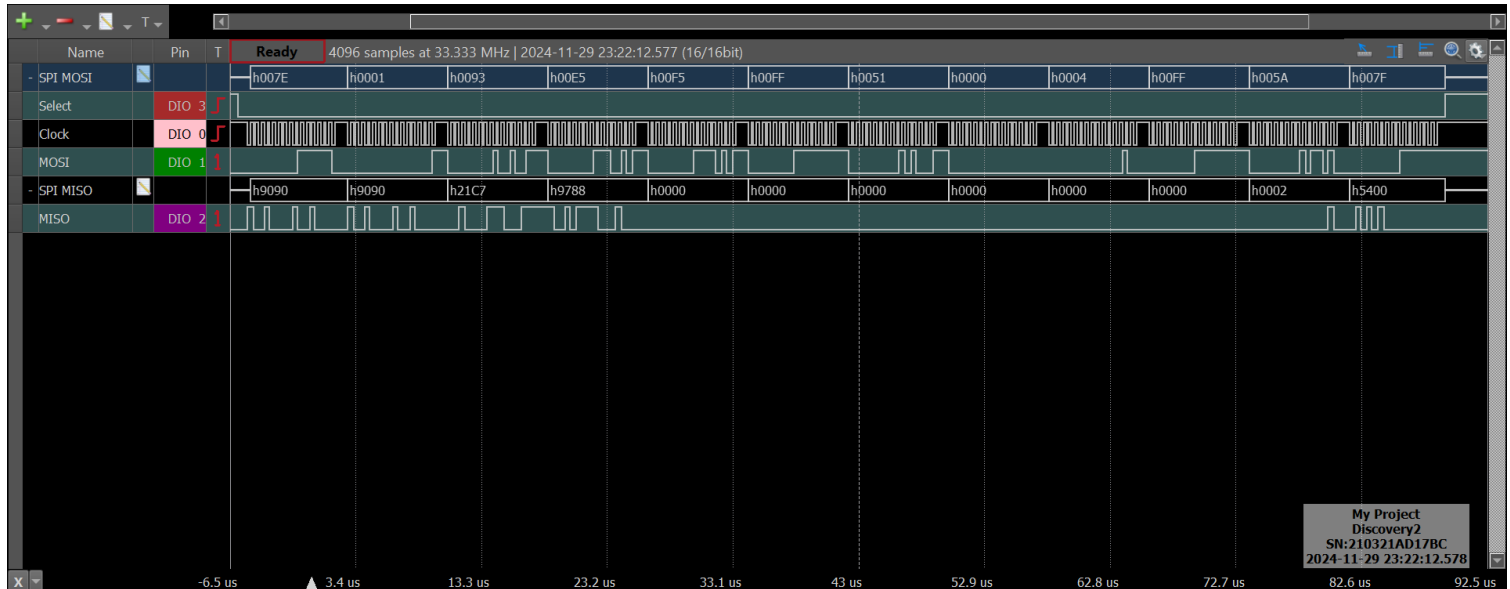
On va créer notre propre protocole de transmission personnalisé en prenant plusieurs paramètres à prendre en compte. Notre paquet de données sera composé de :

- ✓ ID Étudiant : 0x01
- ✓ Code de Session : 0x93 (choisi aléatoirement)
- ✓ ID Capteur : 0xE5
- ✓ Données X, Y, Z qui seront lues en temps réel à partir du capteur ADXL345
- ✓ Checksum qui est calculé en faisant la somme de tous les champs sauf le checksum et l'end byte puis en appliquant le modulo 256.
- ✓ Start Byte : 0x7E
- ✓ End Byte : 0x7F

Mon paquet de données aura la forme suivante :

[0x7E, 0x01, 0x93, 0xE5, DATA0, DATA1, DATA0, DATA1, DATAZ0, DATAZ1, 0x7F]

Avant de passer à la partie Processing, je me suis assuré que l'envoi de mon paquet de données personnalisé par le STM32 via SPI2 est bien correct. En lançant la capture des signaux SPI dans WaveForms, on constate que la ligne MOSI correspond parfaitement à la trame qu'on doit envoyer.



2. Réception et vérification sur PC (Processing)

Comme on peut le voir sur la capture d'écran ci-dessous des données reçues sur la console Processing, le code commence d'abord par lister les ports série disponibles pour permettre à l'utilisateur de vérifier le port connecté au STM32. Ensuite, il lit les données envoyées par le microcontrôleur grâce à la communication UART. Il affiche l'ID de l'étudiant et le code de session, en décimal, qui correspondent exactement à la valeur qu'on a écrit en hexadécimal. S'il a réussi à afficher les données de l'accéléromètre, cela signifie que le checksum reçu correspond bien au checksum calculé sur Processing, d'où le message « Données valides ». Toutefois, je n'ai pas réussi à afficher les données dans la forme que je souhaitais comme sur Putty.

```
Ports disponibles :
[0] "COM4"
Port sélectionné : COM4
ID Étudiant : 1
Code de Session : 147
Données valides. X: -51.9207 Y: 77.8752 Z: 109.8201
ID Étudiant : 1
Code de Session : 147
Données valides. X: -35.9463 Y: -88.8576 Z: 55.906498
```

3. Affichage des résultats

Nous pouvons désormais afficher les valeurs d'accélération sous formes de barres horizontales avec un label et une couleur pour différencier les 3 axes.

