GROUPE N° 4
Tâche N° 1

Régné 100%

Intégration Arduino

1. Etat initial de la tâche :

Au démarrage du semestre nous disposons de codes basiques réalisés l'année dernière. Les codes sont décomposés en deux parties, une pour le capteur accéléromètre, une autre pour la gestion de l'asservissement du moteur. Les différents schémas de câblage sont disponibles Les codes devront être très largement améliorés et il sera nécessaire de découvrir une solution pour rassembler le tout en une seule carte arduino. La vitesse de traitement maximale devra être mesurée.

2. Objectifs :

Objectif principal: Réaliser le code Arduino pour répondre aux besoins du projet en utilisant une seule carte Arduino.

- Compréhension et intégration des codes pour retrouver le fonctionnement de l'année précédente

- Amélioration du code, optimisation

- Intégration du code capteur avec le code moteur pour ne garder qu'une carte (100%)

- Gestion du port série, dans les deux sens de communication (20%)

3. Démarche de travail/plannification/problèmes rencontré

Problèmes rencontrés	Solution supposée / apportée
Code arduino capteur ne correspond pas au capteur possédé, le capteur utilisé ne possède pas de tension de référence les schémas sont donc faux. Le comporte des multiples fonctions inutiles qui ne font qu'alourdir le traitement.	nettoyage du code et adaptation au capteur utilisé cette année pour le groupe 4
Pas de communication avancée avec l'IHM pour la mise en place de paramètres spécifiques sur l'arduino, seulement la valeur de l'accéléromètre est transmise.	attente de l'avancement de l'IHM pour connaître les besoins
Code moteur fonctionnel mais ne réalisant aucun asservissement, l'asservissement n'est pas fonctionnel, la valeur de commande se bloque à la valeur maximale et ne change plus.	recherche de code d'asservissement existant, adaptation au moteur utilisé, calcul des coefficients d'asservissement.
Deux cartes nécessaire pour le fonctionnement actuel	Réflexion sur le nombre de timers et d'interruptions disponible, nombre d'entrée sortie suffisantes
Problème de contrôle des Timer pour les différentes tâches	Utilisation, configurations à plus bas niveau des 3 timers pour l'utilisation simultanée de l'accéléromètre et du moteur asservi en vitesse, ainsi que la communication temporelle avec le PC

GROUPE N° 4
Tâche N° 1

Régné 100%

Intégration Arduino

4. Présentation des résultats obtenus

Les débuts de la tâche d'intégration du code Arduino ont été le nettoyage du code, l'adaptation du code au capteur utilisé pour l'année 2021 au sein du groupe 4. En effet, celui si ne possède pas de pin Vref et le code effectuait un appel trop complexe et inutile de fonctions pour une tâche aussi simple que la lecture d'un pin et l'envoie de cette donnée sur le port série. Nous avons dans un premier temps câblé le capteur sur la carte. Grâce aux fils rigides disponibles, la solution de fils plus fin et sans gaine plastique (fils émaillés) est vite explorée et celui-ci arrivera avec la commande des composants électroniques. Le fil commandé (0.01mm) est beaucoup trop fin pour la réalisation de soudure simple et correcte ainsi que pour une continuité optimale. Nous conseillons pour les années suivantes d'acheter plus épais tout en restant dans un diamètre qui permet une légèreté et une absence de rigidité : 0.1mm devrait être satisfaisant.

J'effectue ensuite l'affichage des données sous forme de graphiques pour une meilleure interprétation des données lues. Cet affichage est généré grâce au traceur série délivré par arduino.

Je suis ensuite passé sur la compréhension du code de l'asservissement du moteur. Le code disponible ne possédait pas d'asservissement. Après de nombreuses recherches je me suis donc concentré sur l'utilisation d'une librairie d'asservissement PID. Cette librairie m'a permis de correctement asservir le moteur, j'ai donc repris le code à partir de zéro.

Figure 1, Code d'origine fourni pour le contrôle moteur et l'acquisition de l'accéléromètre

Voir le code Code/Moteur_asservi_accelerometre pour le moteur asservi avec l'acquisition de l'accéléromètre, le code est commenté.

GROUPE N° 4
Tâche N° 1

Régné 100%

Intégration Arduino

Je me suis ensuite concentré sur la possibilité de réunir tout le code sur une seule carte, car les tests du fonctionnement du moteur asservi était très satisfaisant.

Voici le schéma applicable pour l'arduino (à la place de la STM32F411), celui-ci représente toutes les liaisons carte/capteurs nécessaires :

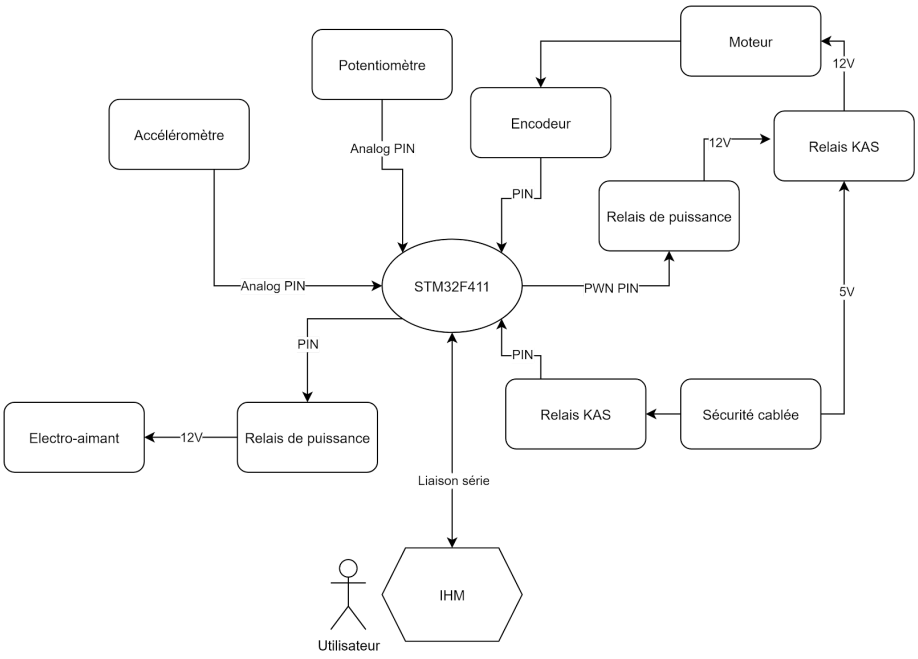

Figure 2, Schéma représentant les interactions entre les différents composants du système

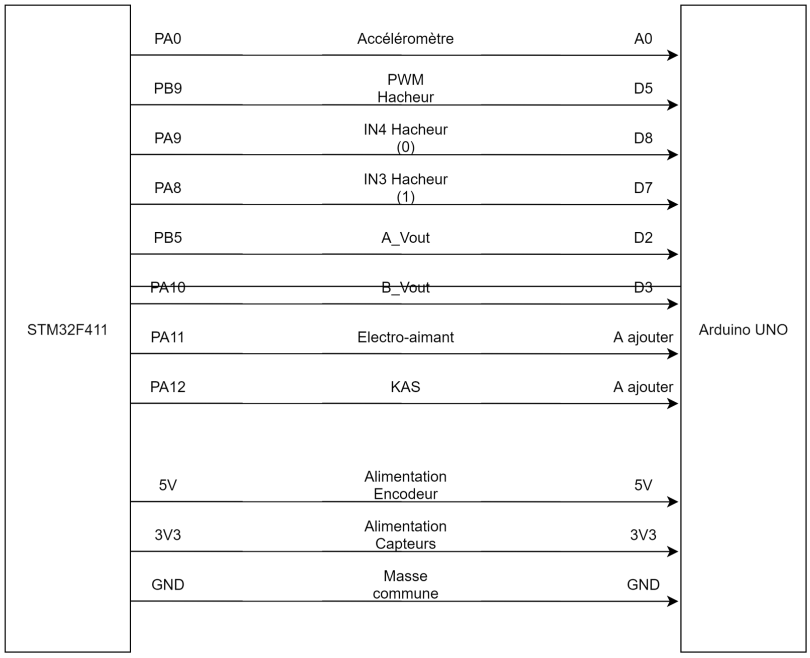

Figure3, Correspondance des Pin entre STM32 et Arduino, pour le câblage de la carte Arduino sur le PCB à la place de la carte STM32

GROUPE N° 4
Tâche N° 1

Régné 100%

Intégration Arduino

L'utilisation d'une seule carte nécessite principalement la gestion de la lecture et de l'asservissement du moteur en simultané. Il est donc nécessaire de mettre en place différents timers en paramétrant leur fréquence d'interruption pour permettre le bon fonctionnement de l'asservissement du moteur et de l'échantillonnage des données de l'accéléromètre. Ici j'ai choisi d'asservir le moteur à 10Hz et de lire l'accéléromètre à 1kHz. L'accès simple au timer ne permet pas d'en utiliser deux avec des fréquences de fonctionnement différentes. J'ai donc dû implémenter les deux timers en écrivant directement dans leur registres comme le montre le code ci-dessous. Nous utilisons donc les timers 1 et 2, en supplément de l'interruption en entrée de la carte sur le pin 2 sur lequel est branché l'encodeur.

```
//Réglages des timers 1 et 2 pour fréuence asservissement et échantillonage accéléromètre
cli();//stop interrupts

//set timer1 interrupt at 10Hz
TCCR1A = 0;// set entire TCCR1A register to 0
TCCR1B = 0;// same for TCCR1B
TCNT1  = 0;//initialize counter value to 0
// set compare match register for 10hz increments
OCR1A = 1562;// = (16*10^6) / (10*1024) - 1 (must be <65536)
// turn on CTC mode
TCCR1B |= (1 << WGM12);
// Set CS12 and CS10 bits for 1024 prescaler
TCCR1B |= (1 << CS12) | (1 << CS10);
// enable timer compare interrupt
TIMSK1 |= (1 << OCIE1A);

TCCR2A = 0;// set entire TCCR2A register to 0
TCCR2B = 0;// same for TCCR2B
TCNT2  = 0;//initialize counter value to 0
// set compare match register for 8khz increments
OCR2A = 249;// = (16*10^6) / (1000*64) - 1 (must be <256)
// turn on CTC mode
TCCR2A |= (1 << WGM21);
// Set CS21 bit for 8 prescaler
TCCR2B |= (1 << CS21) | (1 << CS20);
// enable timer compare interrupt
TIMSK2 |= (1 << OCIE2A);

sei();//allow interrupts
```

Après cette implémentation confirmée, le moteur reste bien asservi et les données de l'accéléromètre sont correctement lues. Nous réalisons un horodatage des valeurs pour assurer aucune perte de données du côté du PC et une bonne synchronisation pour les analyses.

La prochaine amélioration nécessaire pour rendre l'utilisation d'arduino complète consiste à établir une transmission correcte du PC vers la carte Arduino pour ainsi permettre à l'utilisateur de régler certains paramètres directement via l'IHM. Nos tentatives n'ont pas donné de bons résultats et nous avons dû nous concentrer sur la réalisation de cette fonctionnalité sur STM32.

GROUPE N° 4
Tâche N° 1

Régné 100%

Intégration Arduino

5. Bilan des résultats par rapport aux objectifs fixés

Nous avons donc répondu à l'objectif principal qui était de rassembler le code sur une seule carte. L'asservissement fonctionne correctement malgré une certaine lenteur à trouver la vitesse de commande. La transmission des valeurs de mesures de l'accéléromètre.

La carte Arduino offre une forte simplicité de codage et des performances plus que satisfaisantes pour les objectifs demandés. Il est de plus possible d'utiliser une seule carte Arduino comme présenté précédemment. Contrairement à la STM32 qui offre les mêmes performances, Arduino est simple de prise en main et de nombreux tutoriels ou forum sont très largement disponibles pour réaliser ou comprendre les fonctions spéciales nécessaires.

6. Perspectives d'évolution

Quant à l'évolution du code, la transmission séries n'a pas pu être mise en fonction faute de temps pour le débuggage. Il faudra donc continuer le code Code/Moteur_asservi_test_transmission_PC_Carte qui comprend les bases de la réception séries. (Le code de références reste cependant Code/Moteur_asservi_accelerometre)

Une autre piste d'amélioration concerne l'amélioration de la fonction PID, voir son remplacement par une fonction d'asservissement personnel dont la compréhension et les réglages des paramètres seront plus accessibles.