FABING Loan,
LAKSSIOUAR Hajar,
SERMET Leonine

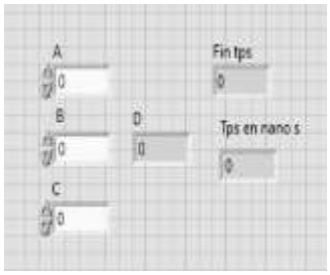
COMPTE RENDU PROJET FPGA

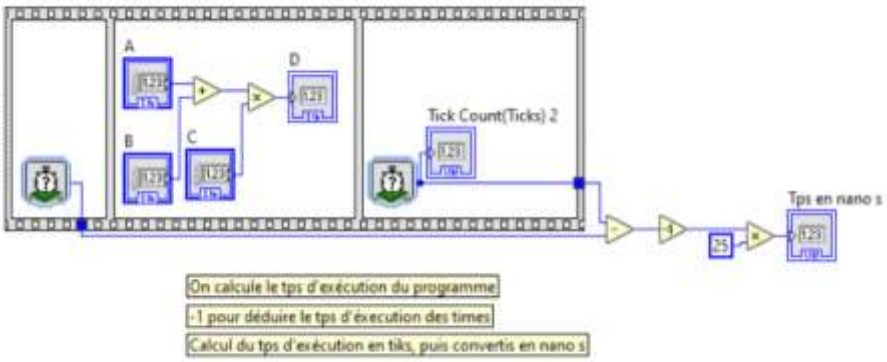
Pour ce projet, nous avons eu différents exercices à exécuter.

Exercice 1 :

L'exercice 1 nous apprend à prendre en main Labview avec un module FPGA.
Celui-ci doit exécuter un programme simple. Nous devons calculer $(a - b) * c$, tout en calculant le temps d'exécution du programme.

Interface Labview :





Conclusion :

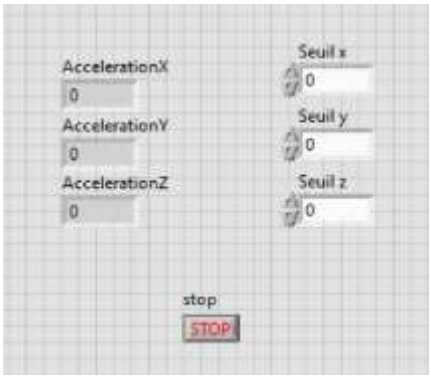
Grâce à ce premier exercice, nous avons remarqué que la compilation du programme est beaucoup plus longue avec le module FPGA. Celle-ci peut prendre plusieurs minutes. Cependant, l'exécution du programme est toujours aussi rapide.

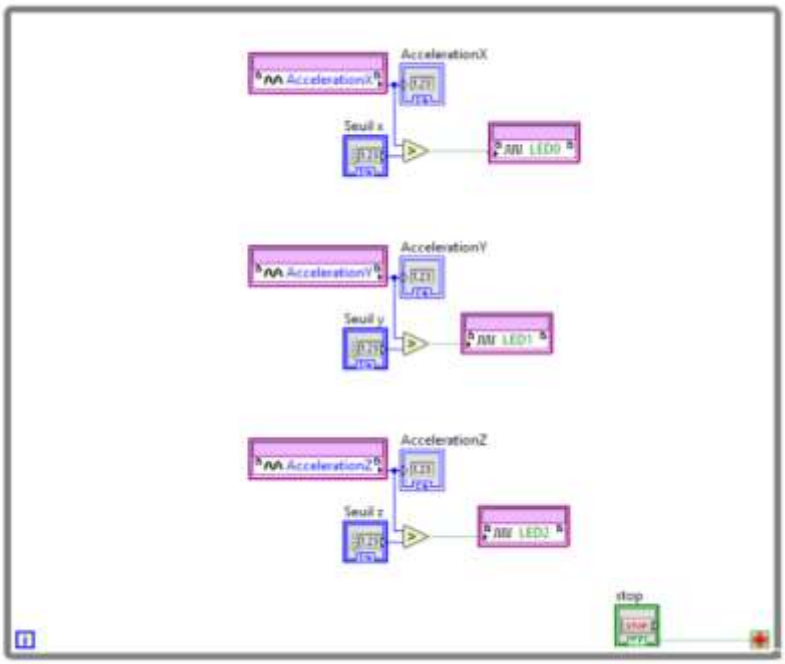
Exercice 2 :

Le deuxième exercice, nous amène à utiliser les 3 LEDs du module FPGA.
Nous devons comparer les accélérations des axes x, y et z, à des seuils. Si les accélérations sont plus grandes que les seuils, alors nous allons les LEDs respectives du module.

Axes d'accélérations	LED à allumer
$\vec{x}$	LED 0
$\vec{y}$	LED 1
$\vec{z}$	LED 2

Interface Labview :





Exercice 3 :

Pour ce troisième exercice, nous cherchons à reproduire la connexion entre le module FPGA et la carte lsm303agr.
Pour cela, nous devons utiliser la liaison I2C, sans utiliser les drivers.

1) Pour commencer, nous avons brancher nos deux modules ensembles, puis à notre ordinateur.

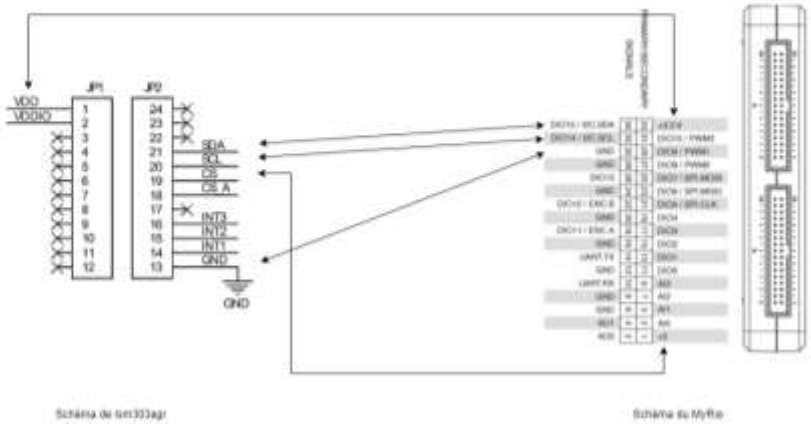
Notre matériel :

3 ordinateurs

National instruments MyRio

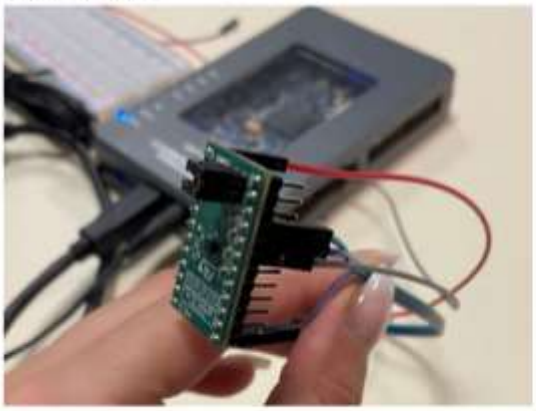
Micro composant lsm303agr

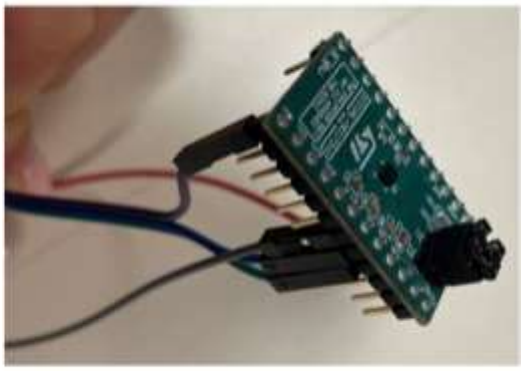
Connexion entre la MyRio et la lsm303agr :



Nos photos :

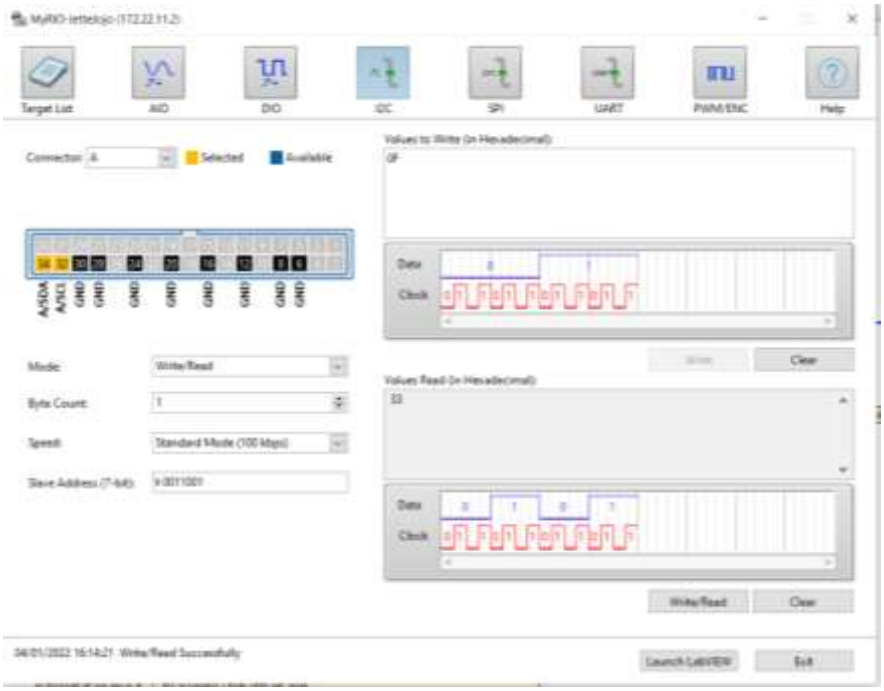
En plus détaillé :





Documentation réalisé avec draw.io

2) On vérifie également la connexion entre le module FPGA et le module LSM303agr.



Nous pouvons alors voir quand rentrant l'adresse de notre esclave ainsi que la valeur de notre écriture, nous retrouvons bien notre réponse.

WHO_AM_I_A	R	0F	000 1111	00110011	Dummy register

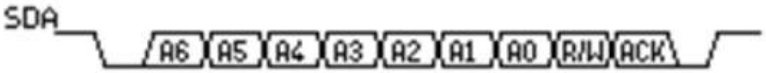
3) Nous avons ensuite codé nos fonctions : START et STOP.

Pour cela, nous avons codé la prise en main du bus. Nous nous sommes inspirés du schéma ci-dessous.

Nous avons plus détaillé notre fonction dans notre vi START.

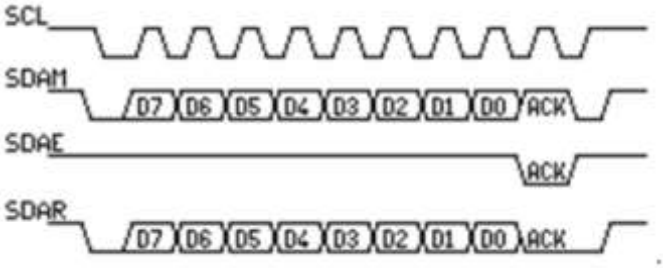
Pour la fonction STOP, nous sommes partit à l'inverse de la fonction START. Nous avons forcé les états de SCL et de SDA pour relâcher le bus.

4) Après avoir coder nos deux premières fonctions, nous avons coder la fonction ADRESSE, qui permet de transmettre l'adresse du maitre à l'esclave.



L'adresse est unique.
Sur le bit D0, le maitre indique s'il souhaite écrire ou lire les données.

5) Ensuite, nous avons coder la transmission d'octets qui peut être représenter par le schéma suivant.



6) Pour finir nous avons essayé de coder la fonction MAIN, qui correspond à la fonction globale du projet, où nous pouvons retrouver l'appel de toutes les fonctions.

7) Pour la suite, il nous manquera la fonction LIREetECRIRE, que nous n'avons pas eu le temps de finir.

Pour documenter ce projet, nous avons utiliser Github.
Ceci nous à permis de créer un projet, et de *push* toutes les modifications de nos VI. Ceci nous permet de suivre notre avancer mais nous donne également la possibilité de sauvegarder chaque version de nos fichiers, au cas où nous avions la nécessité de revenir en arrière.

Github a été associé à Github Desktop. Cette application note toutes les modifications des fichiers. Nous pouvions par la suite, les *commit* pour les sauvegarder, puis les *push* vers le Github.