Elsa COUSTHAM - Marie DEFIVES
Fabien MONTAGNE - Clara ZYSKIND

Rapport Technique :

Chaîne de mesure Infrarouge

Notre projet est uniquement basé sur une manipulation du LEnsE qui sera détaillée juste en-dessous. Notre but était de moderniser cette manipulation notamment en changeant l'interface graphique. Actuellement celle-ci est en Labview, langage peu modifiable et peu pratique pour les élèves de l'IOGS. Qui dit changer un langage de programmation dit aussi changer le reste du montage tel que le mode de communication entre les différents acteurs du système. D'où la problématique suivante : "Comment moderniser une interface graphique et connectique du LEnsE ?"

I. Description :

1- Partie Optique :

Notre projet s'articule autour d'une manipulation de travaux pratiques de troisième année à l'Institut d'Optique. Celle-ci a pour objectif de mesurer avec précision la distance focale d'un objectif en germanium en fonction de sa température.

Pour définir précisément la focale de l'objectif, nous utilisons le système suivant :

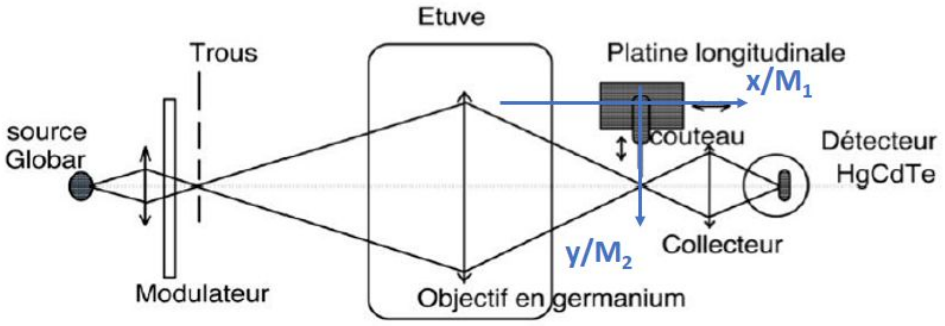


Figure 1 : Schéma du montage expérimental

L'objectif à étudier en germanium est transparent dans l'infrarouge et non dans le visible. On utilise donc un corps noir comme source devant lequel on place un modulateur afin de pouvoir récupérer le signal dans la composante fréquentielle correspondant en sortie sur le détecteur. Le signal intéressant porte la "signature" de la modulation, permettant ainsi de l'isoler du bruit ambiant. Le détecteur est une photodiode infrarouge en HgCdTe, il fonctionne en détection synchrone avec le modulateur. L'objectif est placé dans une étuve dont on peut faire varier la température.

On peut placer après le modulateur des trous qui vont servir d'objet à l'objectif, en sortie du système on vient placer un couteau sur une plaque. Celle-ci va nous permettre de positionner le couteau comme on le souhaite dans l'axe x, selon l'axe optique, asservi par le moteur M1 (on utilisera l'abus de langage "position du moteur 1" pour parler de la position du couteau selon l'axe x) . De même selon l'axe y, dans le plan transversal à l'axe optique, contrôlé par le moteur M2 (on utilisera le même abus de langage à savoir "position du moteur 2" pour parler de la position du couteau selon l'axe y).

Le déplacement du couteau va venir couper une partie du faisceau lumineux et on va pouvoir tracer des profils d'intensité en fonction du positionnement des moteur.

Exemple : En déplaçant le moteur M1, on place le couteau aux emplacements bleus puis oranges (les traits noirs représentent le faisceau lumineux). Le couteau va venir bloquer une partie du faisceau et réduire l'intensité, en fonction du déplacement du moteur M2.

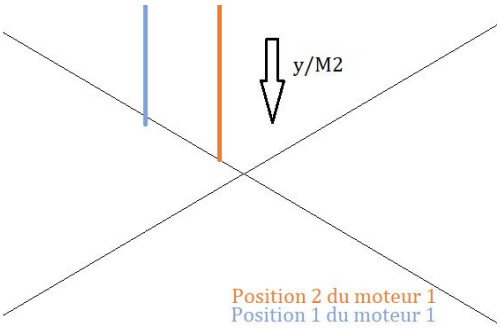


Figure 3 : Schéma du principe du déplacement des moteurs

On peut tracer l'intensité en fonction de la position du moteur M2, en changeant de courbe pour chaque valeur de la position du moteur M1. On obtient alors une allure de marche d'escalier et plus la transition entre le palier haut et le palier bas est brutale, plus on se rapproche du plan focal. Idéalement, en se positionnant exactement dans le plan focal, on aurait une pente verticale.

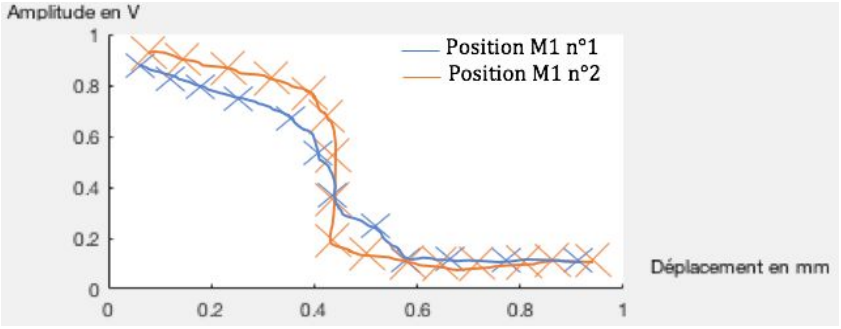


Figure 4 : Intensité en fonction de la position des moteurs

On peut ainsi déterminer la position du plan focal. En faisant varier le pas entre chaque courbe et leur nombre de courbes (pas et nombre de mesures du moteur M1) et du pas entre chaque point et le nombre de points (pas et nombre de mesures du moteur M2), on peut augmenter la précision de la position du plan focal.

2- Partie Électronique :

Notre projet se décompose en 5 blocs électroniques. Les deux premiers sont les blocs moteurs qui sont contrôlés par la carte Nucléo via un câble RS232. Ce type de liaison est une contrainte propre à notre projet lié au système permettant l'asservissement en position des moteurs. On retrouve ce même système de connexion et de contrainte pour la récupération des données du détecteur infrarouge. Ces trois sous-systèmes sont reliés à la carte Nucléo, celle-ci

étant connectée à un ordinateur qui la pilote via le programme Matlab. Ce programme s'occupe de l'interface Homme-Machine et du traitement des données acquises transmises par la carte.

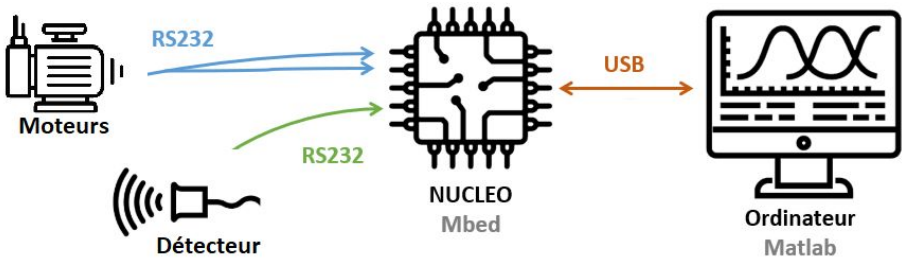
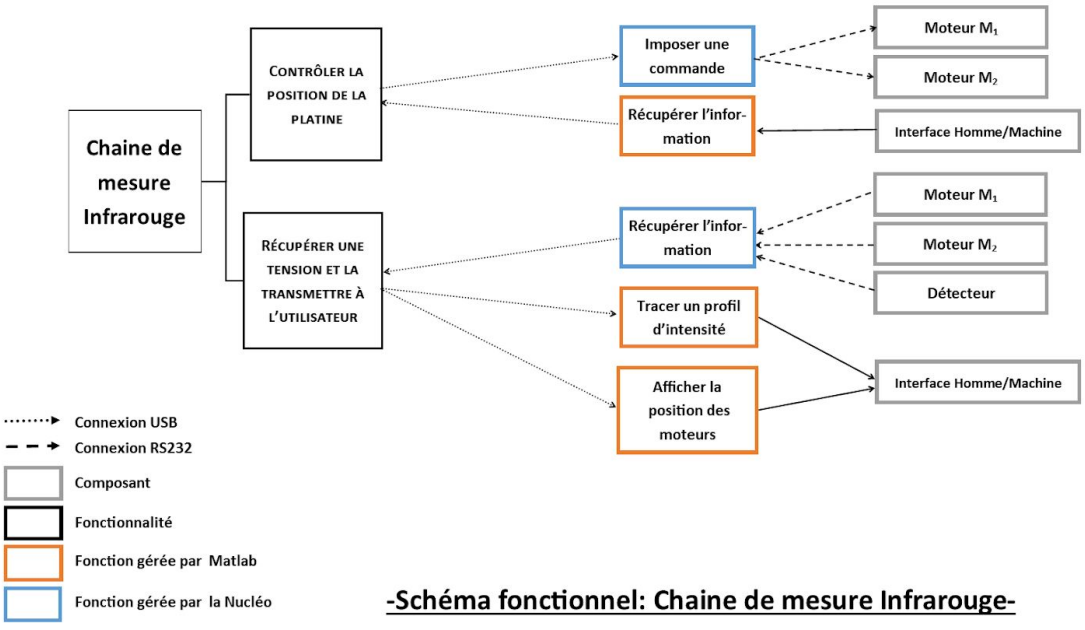


Figure 5 : Représentation des différents blocs électroniques

II. Fonctionnalités :

Deux grandes fonctionnalitées sont à noter dans la chaîne de mesure infrarouge:

- Recevoir une information de l'utilisateur pour permettre de diriger le système (haut du schéma fonctionnel) .
- Récupérer les mesures et les traiter avant de les transmettre à l'utilisateur (bas du schéma fonctionnel).



-Schéma fonctionnel: Chaine de mesure Infrarouge-

→ L'utilisateur souhaite contrôler la position du couteau (de manière manuelle ou automatique), il indique la commande à l'interface Homme/machine. Matlab va alors traiter l'information pour qu'elle puisse être comprise par la carte Nucléo et qui va à son tour traduire le message en une commande pour les moteurs. Cette dernière sera transmise par une connexion RS232.

→ L'utilisateur souhaite récupérer une information sur la position des moteurs ou sur l'intensité captée par le détecteur. La carte Nucléo va alors récupérer l'information via un cable RS232 et la traduire pour que l'interface gérée par Matlab puisse la lire. L'interface va par la suite soit afficher la position des moteurs, soit réaliser et afficher un graphique indiquant l'intensité reçue par le détecteur en fonction de la position transversale du couteau.

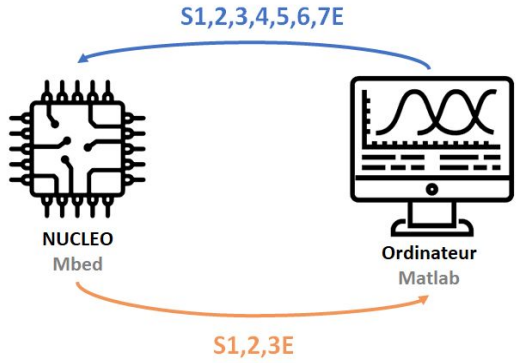
III. Programmation :

Cette partie du projet se découpe facilement suivant 3 axes : la partie Matlab qui va traiter les données résultantes de l'expérience et échanger avec l'utilisateur, la partie Mbed de la carte Nucléo qui va traiter les données d'entrée de l'expérience et échanger avec les actionneurs du système et la communication entre ces deux parties.

1- Communication:

Il n'existe qu'un seul fil reliant l'interface homme/machine et la carte Nucléo. Par exemple, lors de l'envoi d'une information à l'interface, si aucune précaution n'est prise, Matlab ne peut faire la différence entre une indication de position de moteur (M1 ou M2) et une information sur l'intensité captée par le détecteur. Et le même problème se pose lorsqu'une information est envoyée à la carte Nucléo.

Nous avons donc choisis de communiquer par des chaînes de caractères dont le début sera marqué par un 'S' (*start*) et une fin par 'E' (*end*).

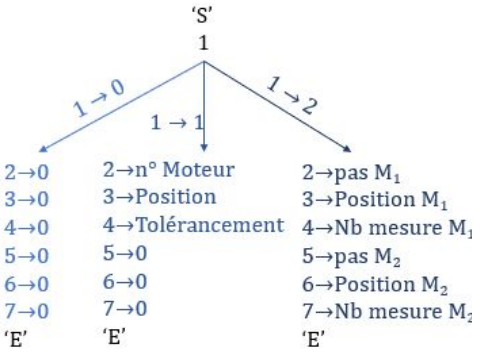


Un message contenant 2 lettres ('S' et 'E') et 7 nombres sera envoyé par Matlab à la carte Nucléo. Et un message de 2 lettres ('S' et 'E') et 3 nombres sera envoyé par la carte à Matlab.

De la Nucléo à l'ordinateur, le message se présente de la manière suivante :

'S' caractère *start*, début de la transmission
1→ Position du moteur M_1
2→ Position du moteur M_2
3→ Intensité reçue par le détecteur
'E' caractère *end*, fin de la transmission

L'ordinateur envoie un message la carte Nucléo de 9 caractères (dont les caractères de début et fin). Il se présente de la manière suivante:



Le message envoyé dépend ce que souhaite l'utilisateur. En fonction du premier caractère se situant après le 'S' la carte Nucléo n'utilisera pas les mêmes fonctions.

→ S'il souhaite **connaître la position** des moteurs et l'**intensité** reçu par le détecteur, Matlab enverra un message commençant par un **0** (le reste du message sera remplie de zéros mais pourrait être complété avec n'importe quels caractères car ils ne seront pas lus).

→ S'il souhaite **imposer une position** à un des moteurs, son message commencera par **1** et sera suivi du numéro du moteur, sa position et une valeur de tolérancement.

→ S'il souhaite **faire une série de mesures et récupérer l'intensité et la position des moteurs** pour chaque mesure, Matlab enverra un message commençant par **2** et contenant toutes les informations suivantes : pas du moteur M_1, position initiale de M_1, nombre de mesure M_1, pas du moteur M_2, position initiale de et nombre de mesure M_2.

2- Code Matlab :

Notre code est directement relié à l'interface graphique, faite sur *Guide*. Il commence par un code d'initialisation directement généré par *Guide.* Nous allons expliquer ensuite les différentes fonctionnalités de ce code :

Cases ou boutons : Lorsque l'on crée une case ou un bouton sur l'interface graphique, il se crée automatiquement 2 fonctions que nous pouvons utiliser :

- Fonction de type *'Opening' :* Permet de changer l'aspect visuel du bouton (couleur…)

- Fonction de type *'CallBack'* : Sert à recevoir et envoyer des messages de type S,...,E, à afficher les positions des moteurs M1 et M2 (notées *M1* et *M2*) ou à afficher les graphiques de l'amplitude (notée *Intensite*) reçue par le détecteur en fonction du déplacement du couteau en mm.

Affichage des courbes : Pour afficher des courbes sur l'interface graphique, on configure l'action après avoir cliqué sur les boutons "Graphe manuel" ou 'Graphe automatique" en éditant les fonctions *Graphe_manuel_Callback* et *Graphe_automatique_Callback*. Ces 2 fonctions diffèrent, nous allons les expliquer.

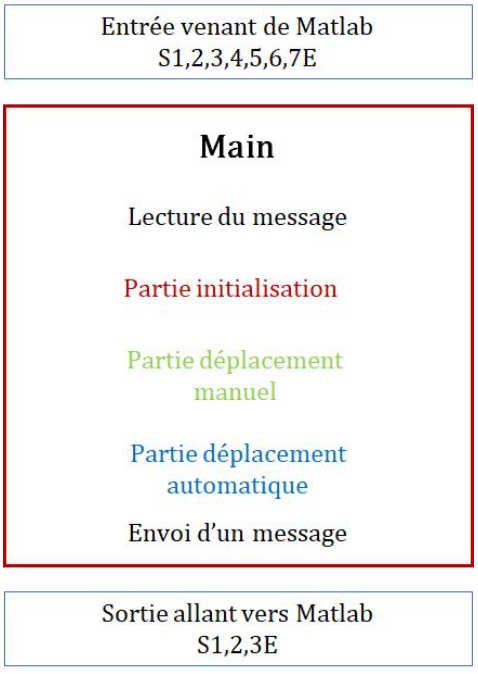
- Fonction *Graphe_manuel_Callback* : On lit le message envoyé par MBED commençant par le caractère 'S' et terminant par le caractère 'E'. Il n'y a qu'un seul message. On récupère les informations lues dans des variables notées *M1*, *M2* et *Intensite*. Puis on trace point par point l'intensité reçue en fonction de la position du couteau (contenue dans *M1* ou *M2* selon le choix du moteur de l'utilisateur) à l'aide d'un *plot(x,handles.y,'*')*.

- Fonction *Graphe_automatique_Callback* : Ici on doit lire plusieurs messages, commençant tous par 'S' et terminant par 'E'. Il faut donc collecter les données dans des vecteurs et on termine la lecture quand on a récupéré tous les points/amplitudes.
  Pour chaque valeur de M1 (on note n ce nombre de valeurs) on a N mesures faites par M2 donc la boucle doit s'arrêter quand j = N*n. On va donc collecter toutes les valeurs de déplacement du couteau longitudinal *M1 et* transversal *M2* dans les vecteurs *vectM1* et *vectM2* ainsi que les valeurs de *Intensite* dans *VectIntensite*.
  Puis pour tracer les n graphes qui se superposent, on fait une boucle de j allant de 1 à n. En abscisse on a les positions de M2, identiques quelque soit la position de M1. En ordonnée on a les amplitudes qui, pour chaque position de M2, varient d'une valeur de M1 à une autre, d'où le 'j' dans le code.
  On trace le tout à l'aide de *plot(handles.x1,handles.y1)* et on met un *hold on* pour que les courbes se superposent.

3- Code Mbed :

Le code Mbed implanté dans la carte Nucléo a pour objectif de récupérer les informations envoyées par le logiciel Matlab par une connexion *serial USB*, de contrôler en conséquence les deux moteurs et de lire et envoyer les informations de positions et d'intensité à l'ordinateur.

Le programme s'articule autour de la réception d'un message envoyé par Matlab, par l'utilisation d'une fonction d'interruption qui se lance à la réception d'un caractère et va scanner le message pour s'assurer de notre forme (à savoir "S1,2,3,4,5,6,7E"). Après l'utilisation d'un des modes la sortie est toujours de la même forme est contient toujours les deux positions et l'intensité sur le détecteur ("S1,2,3E" avec 1 = position M1, 2 = position M2 et 3 = intensité sur le détecteur).

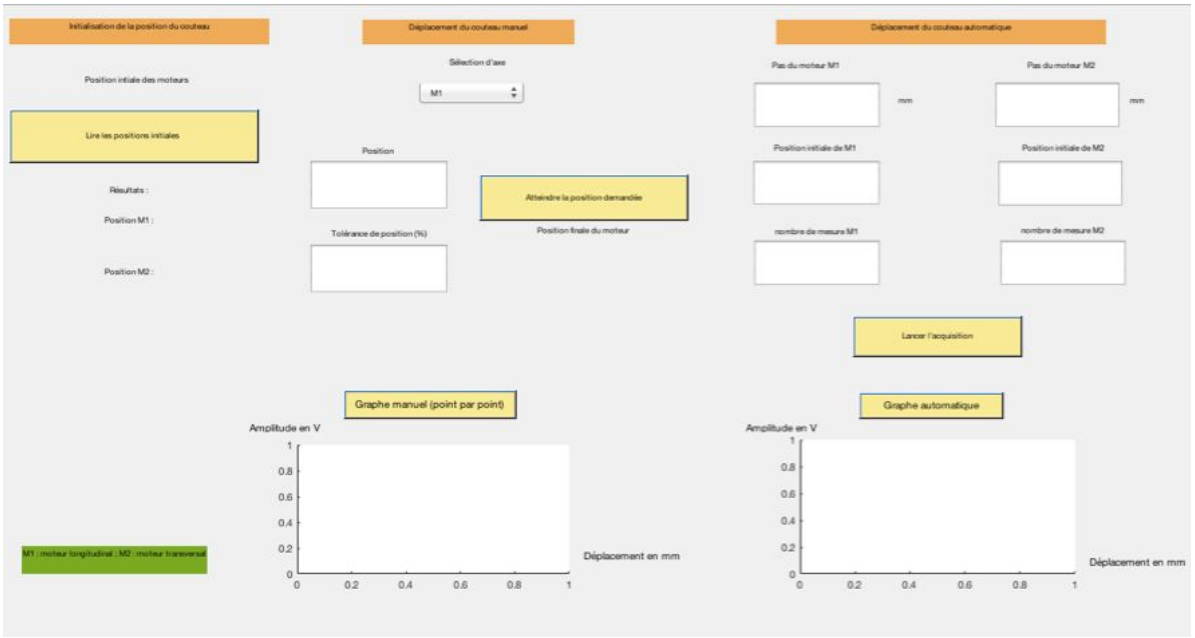


Il existe trois modes de fonctionnement :

- Si 1 → 0 : partie initialisation : le système récupère et transmet les positions des moteurs et l'intensité du détecteur.
- Si 1 → 1 : partie déplacement manuel : dans cette partie, la carte Nucléo reçoit une commande de position d'un moteur, ainsi qu'une valeur de tolérancement en pourcentage de cette position. Nous ne maîtrisons pas encore le fonctionnement du bloc moteur pour transmettre l'information du tolérancement. On commande donc le moteur choisi par l'utilisateur à la position qu'il a définit et on renvoie toujours à l'ordinateur par le câble USB la position actuelle des moteurs et l'intensité reçu par le détecteur.
- Si 1 → 2 : partie déplacement automatique : dans ce mode, on reçoit les informations de position initiale, pas du moteur et du nombre de mesures pour chaque moteur. On réalise ensuite une mesure pour chaque position du moteur M1 et M2, en envoyant en temps réel les coordonnées du couteau et l'intensité sur capteur. C'est grâce à cette fonction que matlab peut tracer les graphes.

IV. Notice utilisateur :

Cette interface graphique comporte 3 parties : Une première où on initialise la position du couteau, une deuxième partie où l'on peut déplacer le couteau de manière "manuelle" et une troisième partie où l'on déplace le couteau de façon automatisée. Les 3 parties peuvent être utilisées indépendamment, selon le choix de l'utilisateur.



1ère partie : Initialisation de la position du couteau

Ici, l'utilisateur peut tout d'abord cliquer sur le bouton jaune Lire les positions initiales.

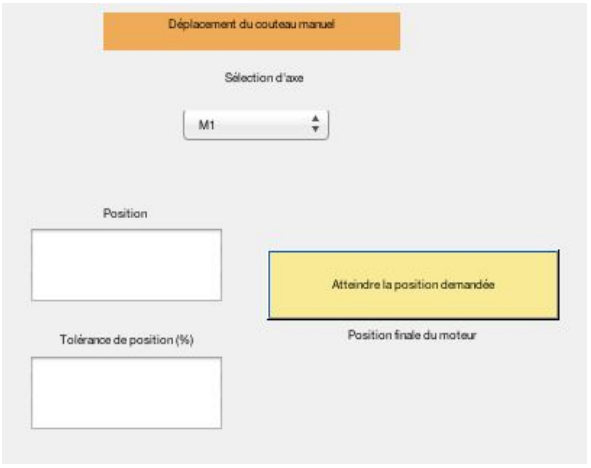
Il s'affiche alors dans les résultats la position du moteur M1 et celle du moteur M2

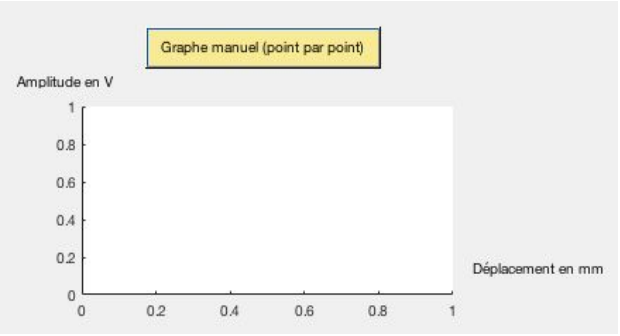
2ème partie : Déplacement du couteau manuel

L'utilisateur commence par choisir l'axe qu'il veut étudier : l'axe longitudinal (moteur M1) ou l'axe transversal (moteur M2).

Ensuite, il peut rentrer la position qu'il souhaite que ce moteur atteigne dans la première case, et le tolérancement souhaité sur cette position.

Il clique enfin sur le bouton Atteindre la position demandée, et la position finale du moteur s'affiche : c'est celle de la position choisie ± tolérancement choisi.





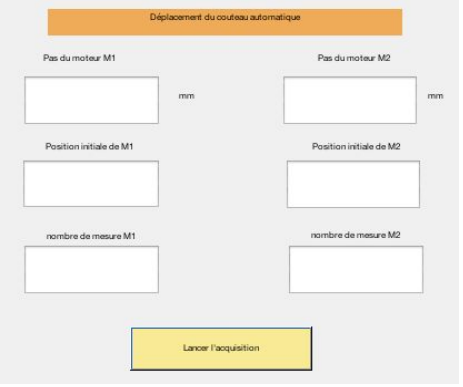
Alors, pour chaque position, l'utilisateur peut cliquer sur le bouton Graphe manuel (point par point).

Pour chaque position rentrée, une croix donnant l'amplitude en volt en fonction du déplacement en mm du couteau apparaît. À chaque fois qu'il clique sur le bouton Graphe manuel (point par point), les points se superposent sur le graphique.

3ème partie : Déplacement du couteau automatique

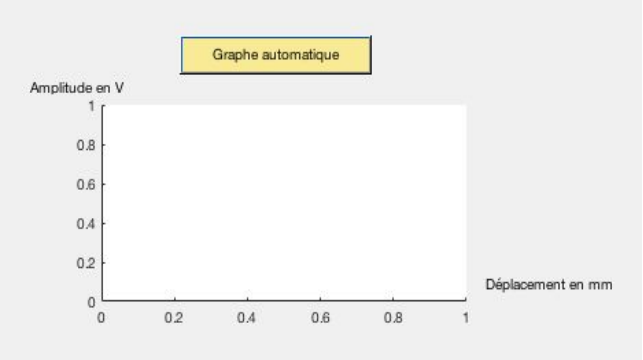
Cette partie est la plus intéressante de l'interface. L'utilisateur peut d'abord rentrer le pas, la position initiale et le nombre de mesures souhaitées pour les moteurs M1 et M2.

Il peut ensuite cliquer sur le bouton Lancer l'acquisition, et le couteau commence son déplacement.



Par exemple, on choisit pour le moteur M1, qui contrôle l'axe longitudinal, une position initiale à 10 mm puis on choisit de faire 3 mesures avec un pas de 1 mm, donc le couteau aura une position finale longitudinale de 25 mm. Et pour M2 on choisit un pas de 1 mm et 10 mesures.

Attention, il faut attendre que l'acquisition soit terminée avant de cliquer sur Graphe automatique .



Ensuite, l'utilisateur clique sur le bouton Graphe automatique. C'est le déplacement transverse du moteur M2 qui nous intéresse.

Ainsi dans notre exemples, on aura 3 courbes superposées (correspondant aux 3 différentes positions de M1), avec chacune 10 points (correspondant aux 10 différentes positions de M2).

V. Bilan des compétences :

Lors de ce projet, nous avons pu développer des nouvelles compétences typiques à notre étude.

a) Communiquer avec une série de type RS232

Voici le premier champ de compétences développées, c'était également une contrainte très forte de notre cahier des charges. En effet les instruments du LEnsE utilisent cette interface.

Pour connaître les fonctions de base permettant la communication entre Nucléo et RS232 nous avons utilisé ce lien : http://lense.institutoptique.fr/Nucleo-configurer-une-communication-point-a-point-de-type-rs232-2/.

Les données ne voyageant que par un seul câble, elles sont envoyées les unes après les autres. Il est nécessaire de définir une forme précise de message. Cela s'appelle "parser" l'information. Pour réaliser une fonction de parsage, nous nous sommes appuyés sur le tutoriel suivant : https://os.mbed.com/cookbook/Serial-Interrupts. Grâce à ces tutoriels nous avons donc pu apprendre les particularités de la liaison RS232 et les inclure dans notre projet.

b) Interface sur Matlab : utilisation de GUIDE

Voici le coeur de la seconde exigence du cahier des charges. Mr. Avignon voulait que l'interface puisse être utilisée et modifiée facilement par tous les membres de l'IOGS. Nous avons donc choisi de programmer cela en Matlab en utilisant le module GUIDE. La partie IV. Notice Utilisateur consiste en une notice de cette interface. Nous avons appris comment créer l'interface : boutons, menus déroulants, saisie de données, affichages de courbes... et comment récupérer et exploiter les données rentrées par l'expérimentateur.

Nous nous sommes appuyés sur ces tutoriels :
https://fr.mathworks.com/help/matlab/migrate-guide-apps.html?s_tid=CRUX_lftnav
https://fr.mathworks.com/help/matlab/creating_guis/gui-that-accepts-parameters-and-generates-plots.html

c) Liaison série Matlab - Nucléo

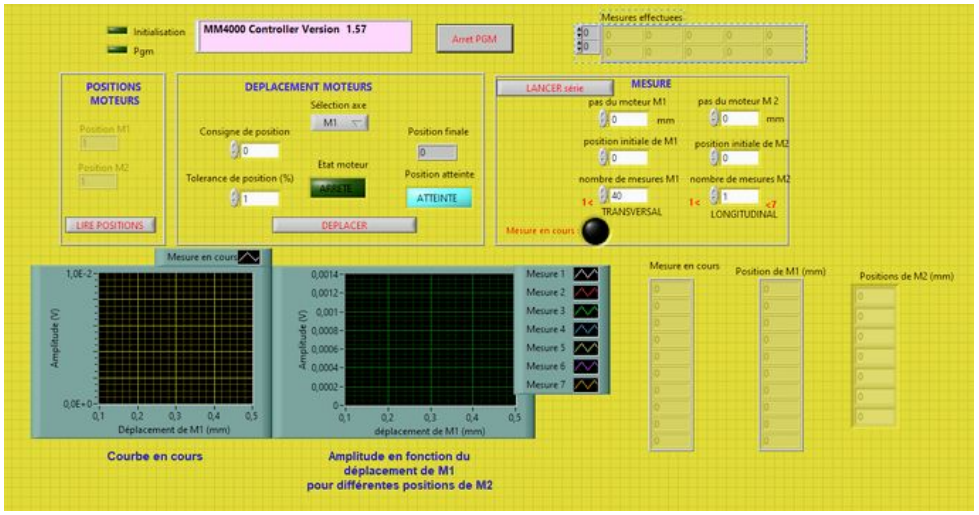
Il faut désormais faire communiquer les outils Matlab et Nucléo. La communication se fait par liaison série. Le parsage ayant déjà été vu précédemment nous nous sommes penchés sur la spécificité de communication entre ces deux appareils. Pour se faire nous nous sommes référés à https://fr.mathworks.com/help/matlab/serial-port-devices.html

Pour conclure, ce projet nous a permis de développer de nouvelles compétences sur Nucléo et d'utiliser de nouveaux outils (tel que GUIDE).

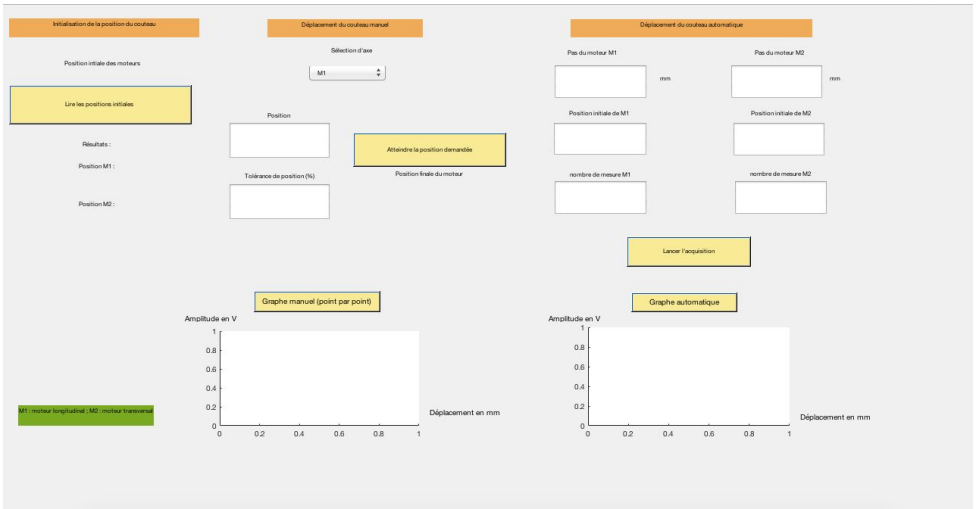
VI. Réalisation :

Globalement, nous avons réussi à respecter le plus gros des contraintes. En effet, il fallait tout d'abord changer l'interface graphique. L'expérience du LEnsE avait une interface graphique en LabView, peu visuelle pour les élèves.

Voici l'interface graphique initiale :



Et celle que nous avons réalisé à l'aide de Matlab (par le biais de GUIDE) :



- Celle-ci est plus épurée et plus simple d'utilisation (voir Partie utilisateur à la partie V.)
- Pour échanger les informations, nous avons utilisé le langage Matlab (qui gère l'interface graphique) et le langage C (qui gère la carte Nucléo) au lieu du langage Labview. Nous avons donc réussi à utiliser des langages plus simples d'utilisation par les élèves de l'Institut d'Optique, et les codes sont également modifiables.
- Nous avons également créé un bloc contenant une carte Nucléo reliée à l'expérience par 3 ports RS232 et à l'ordinateur par un port USB, alors qu'on avait auparavant ces ports RS232 directement branché à l'ordinateur (contrôlé via Labview). La contrainte vis à vis de l'interface RS232 est donc respectée.
- Enfin nous avons réussi à gérer la répartition des tâches entre les différent membres de l'équipe, malgré la distance due au confinement. Nous avons pu bien avancer dans notre projet même si les essais n'ont pas pu être faits.

Cependant, il reste quelques objectifs que nous n'avons pas atteint, soit par manque de temps, soit en raison du contexte actuel.

- En effet, un des objectifs était de tracer la FTM du système optique grâce aux mesures faites par le détecteur : nous avons décidé de ne pas nous pencher sur cette étape afin de finaliser le reste de l'étude.
- De plus, il serait intéressant de réussir à récupérer toutes les mesures lors d'une exécution automatique dans un fichier de type excel. L'utilisateur pourrait interpréter les mesures autrement qu'avec les graphiques représentés par l'interface.
- De même, il peut être judicieux de se pencher sur la rapidité de notre système. Le manque d'accessibilité aux locaux ne nous a pas permis d'étudier cette partie.
- Enfin le tolérancement de la position des moteurs est un projet futur que nous aurions voulu aborder. Le tolérancement correspond à l'erreur relative de position des moteurs par rapport à la consigne donnée par l'utilisateur. Pour le moment nous savons pas si cette erreur est préjudiciable au bon fonctionnement du système et si elle est modifiable.

VII. Retour sur expérience :

En ce contexte de confinement, nous avons pu expérimenter le travail à distance sur l'enseignement Protis. Vous trouverez dans ce document un retour sur expérience abordant les difficultés et les solutions trouvées ainsi que la gestion du travail en équipe à distance.

1) Bien évidemment travail à distance et manipulations ne riment pas. La première difficulté rencontrée fut donc la perte de contexte. Nous travaillions sur le projet "Chaîne de mesure infrarouge" qui était directement relié à une manip du LEnsE. Cela a causé un flou dans l'équipe au début de la première séance de télétravail, nous ne savions pas comment aborder le projet.

   Solution : Après avoir discuté entre nous et avec un enseignant nous avons pu cibler les points du projets qui restaient inchangés tels que l'interface graphique.

2) Une contrainte du projet était l'adaptation à une interface RS232 et la communication entre Nucléo et Matlab. Nous n'avons pu aborder "en vrai" aucun de ses aspects. Par exemple, nous devions tracer des graphes à partir de données récupérées sur la Nucléo mais sans test nous ne savons pas si cette récupération est efficace. De même à propos des câbles RS232 nous ignorons si nos hypothèse de données envoyées par le RS232 sont justifiées (reçoit t-on une valeur d'intensité normalisée, un bit indiquant la bonne transmission des données ?). Par ailleurs aucune difficulté mécanique n'a été examiné : avec quelle précision les moteurs atteignent leur position ? Comment l'expérimentateur doit remplir la case "Tolérancement" ?

   Solution : Nous avons décidé de coder "dans le vide". Il est donc probable que la façon de récupérer les infos de Nucléo à Matlab et inversement ne fonctionne pas.

3) Quelques problèmes techniques que chaque personne a dû rencontrer au moins une fois, micro qui ne fonctionne pas, perte de connexion, plus de son etc…

   Solution : Prendre son mal en patience. Lors d'une explication au moins 3 membres sur 4 avaient une connexion suffisante. Il s'agissait alors de ré-expliquer au membre manquant.

Gestion du travail en équipe :

Nous n'avons pas rencontré de réelles difficultés à ce niveau. Grâce aux sessions individuelles de e-campus nous pouvions parler, réaliser des schémas, partager des applications entre nous… Bien-sûr cela aurait été plus sympathique de se voir en vrai. Les tâches étaient facilement séparées selon le plan établi à la première séance de la phase 2. Nous nous divisions entre Skype et e-campus comme si nous avions changé de paillasse, utilisant chacun notre espace de travail à plusieurs centaines de kilomètres les uns des autres.