PROJET DE FIN D'ETUDE

Département Informatique, Microélectronique et Automatique

« Conception par fabrication additive des pièces en plastique à partir d'un robot »

- Encadrants -

Rochdi MERZOUKI

Othman LAKHAL

- Etudiante -

Diana MARRUCHO

Réalisé du 10 janvier 2018 au 20 février 2018

Remerciements

Je tiens tout d'abord à remercier monsieur Rochdi MERZOUKI pour m'avoir proposé ce projet de fin d'étude rapidement lors de mon retour de mon semestre d'étude au Québec, et de son aide tout au long du projet.

Ensuite, je tiens à remercier tout particulièrement Othman LAKHAL pour sa participation, son aide, ses explications et sa patience durant ces six semaines malgré sa préparation de soutenance de thèse sur le projet Matrice.

Merci également à l'équipe du laboratoire Cristal pour l'accueil et la mise à disposition du matériel et à monsieur Gregory SANT pour ses explications sur l'imprimante 3D.

Sans ces personnes réunies, je n'aurais certainement pas pu avancer convenablement dans mon travail.

Sommaire

Table des illustrations

Introduction

Dans le cadre de ma dernière année en tant qu'étudiante en Informatique, Microélectronique et Automatique à Polytech Lille, nous avons l'opportunité de mettre en œuvre les connaissances acquise durant nos années d'études en un projet. En l'occurrence, revenant d'un semestre à l'étranger, mon projet de fin d'étude s'est déroulé sous une période limitée de six semaines.

Ayant effectué la demande d'un sujet en rapport avec la spécialisation Systèmes Autonomes du département auprès de monsieur MERZOUKI, plusieurs sujets m'ont été proposés et mon choix s'est porté sur la conception par fabrication additive. En effet, l'impression 3D est un domaine en pleine expansion, et en plein essor industriel et me paraissait intéressant afin de pouvoir mettre en œuvre mes connaissances auprès d'un sujet de recherche.

Mon rapport se déclinera en trois parties : une première où je présenterai le projet, la seconde reflètera le travail que j'ai effectué pour proposer des améliorations ou conseils dans la suite du projet qui va être repris à partir de mars par d'autres étudiants.

1. Présentation du projet

1.1. Contexte du projet et état de l'art

1.1.1. Contexte

Dans le cadre du programme CENTAURE d'accompagnement des PME et PMI de la région du Nord-Pas-de-Calais dans l'intégration, l'étude et la conception de systèmes mécatroniques et robotiques, Polytech Lille, le laboratoire Cristal, la société ALL-TRENDS ainsi que ERM se sont associées afin de réaliser un prototype industriel d'impression 3D robotisé à base de granulés en vue d'une mise en commercialisation au second trimestre 2018.

L'entreprise *ERM Automatismes* cherche à se diversifier et à proposer à ses clients de nouvelles formes d'impressions 3D et de fabrication additive. Pour se faire, elle souhaite proposer un robot industriel, en l'occurrence ici un KUKA, doté d'une tête d'impression réalisée par Grégory SANT et la société *ALL-TRENDS*. Par cette démarche, et afin d'être innovants, la réalisation du prototype utilisé sera décliné en versions éducative et industrielle et proposera des prouesses de rapidité tout en conservant une certaine précision, dans le repère cartésien mais aussi il bénéficiera aussi d'un avantage fourni par le KUKA : la possibilité de s'orienter.

Lancée par Gregory SANT en 2013, la société *ALL-TRENDS* vise à séduire des industriels comme ceux du BTP, du biomédical, de la plasturgie ou encore de la métallurgie, à utiliser l'impression 3D et donc à alléger certaines pièces. La technologie innovatrice de cette entreprise est de fabriquer des pièces 3D en polymère par impression additive à partir de granulés plastiques non-propriétaires : la technologie *e-Bridium®*. Cette technologie permet donc à l'utilisateur de choisir les matières avec lesquelles il souhaite travailler à un coût bien inférieur aux impressions traditionnelles tout en maximisant les performances et permet donc de travailler avec des matières très souples. Le projet en partenariat avec l'école Polytech Lille permet donc d'imprimer dans toutes les directions, avec des buses de diamètres diversifiés et propose un débit maximal de 700 grammes par heure, contre 700 grammes à la journée dans les imprimantes traditionnelles. Ce projet sera commercialisé sous le nom d'*e-Bridium Gigantic* et viendra compléter l'imprimante déjà disponible : la *e-Bridium 400*. La tête de buse a entièrement été réalisée par la société *ALL-TRENDS* et dont la pièce esthétique extérieure a été réalisée en impression 3D.

Figure 1 - Imprimante e-Bridium Gigantic

1.1.2. Etat de l'art

Avec son essor depuis une dizaine d'années, l'impression 3D est devenue un nouveau moyen de création d'objets, que se soit pour les particuliers ou pour les professionnels. En effet, plus besoin d'utiliser des procédés de soustraction, mais par ajout de couches de matière successives, il est désormais possible de réaliser toute sorte d'objet.

L'impression 3D remonte aux années 1980. A cette époque, le Dr. Kodama de l'*Institut Municipal de Recherche Industriel de Nagoya* souhaite alors réaliser un prototype à base de la fabrication « tranche-par-tranche ». est un procédé de fabrication additive qui transforme un objet modélisé en CAO en un objet solide.

Aujourd'hui, diverses solutions d'impression 3D existent. Ces différents procédés sont les suivants :

- La **technologie FDM** (Fused Deposition Modeling) ou encore **dépôt de fil fondu** :

  Cette technique est la plus couramment utilisée pour les imprimantes 3D. Elle consiste à chauffer un filament thermoplastique entre 170°C et 260°C (PLA, ABS…) à travers une buse d'impression extrudeuse puis de déposer sur un plateau la matière fondue par couche. Généralement, la tête d'impression se déplace selon les axes X et Y, et le plateau support selon l'axe Z.

  Le principe de fonctionnement de ce type d'imprimante est modélisé dans ***la Figure 2 - Principe de fonctionnement d'une imprimante 3D FDM*** ci-dessous :

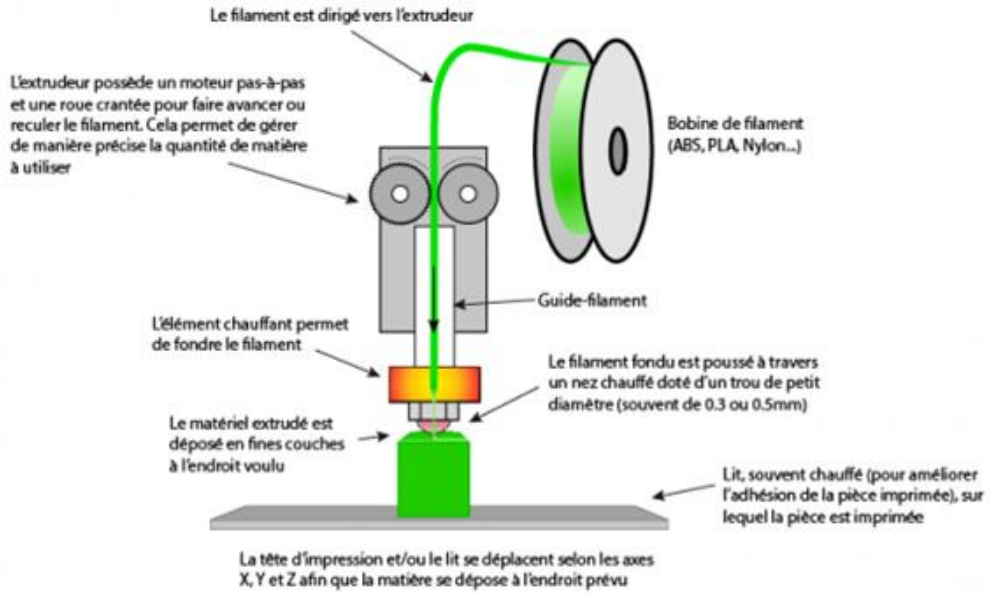


Figure 2 - Principe de fonctionnement d'une imprimante 3D FDM

- La **technologie SLA** (StereoLithography Apparatus) dite la **stéréolithographie** :

  Cette technique d'impression 3D est le nom donné par le créateur de l'impression 3D en 1982.

  Ce processus polymérise une résine liquide photosensible grâce à un laser ultra-violet. Le plateau étant placé dans un bac en résine, le laser va permettre de durcir instantanément la résine contenue dans le bain couche par couche avec un plateau se déplaçant selon l'axe Z négatif.

Généralement, il est nécessaire d'effectuer un post-traitement de pièce afin de retirer les restes de solvant puis un passage au four est nécessaire pour solidifier la pièce dans la plupart des cas.

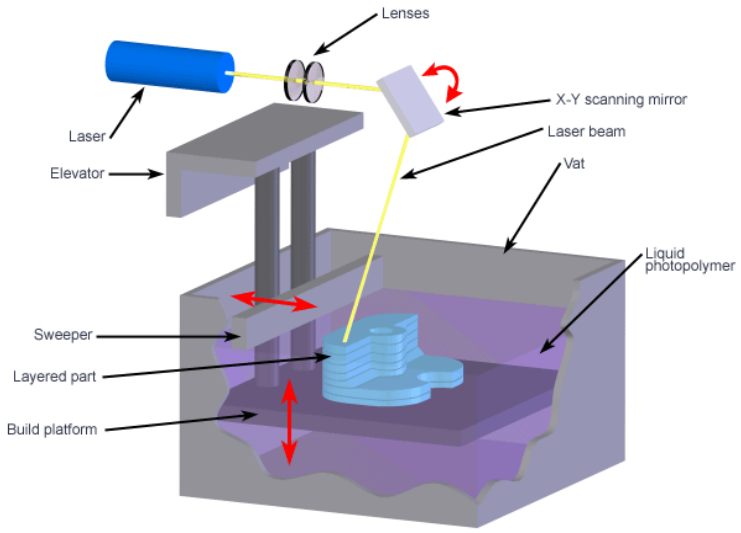


Figure 3 - Principe de fonctionnement d'une imprimante 3D SLA

- La **technilogie SLS** (Selective Laser Sintering) plus communément appelée le **frittage laser** :
  Même procédé de fabrication que la SLA à la seule différence que le matériau utilisé est une poudre extrêmement fine (plastique, de verre, céramique, métal, nylon…)
  Cette technique est souvent utilisée en prototypage car très couteuse et très puissante.

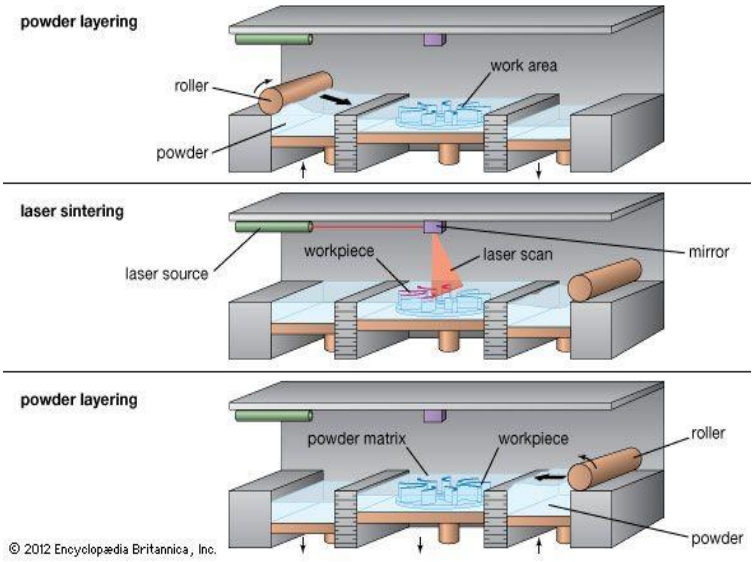


Figure 4 - Principe de fonctionnement d'une imprimante 3D SLS

- La **technologie DLP** (Digital Light Processing) :

  Procédé utilisant une résine liquide photosensible mais à la place d'un laser UV, un projecteur renvoie de la lumière sur un miroir décliné en petits miroirs qui, lorsque l'un d'eux s'incline, il s'oriente vers le bain de résine afin de durcir instantanément celle-ci.

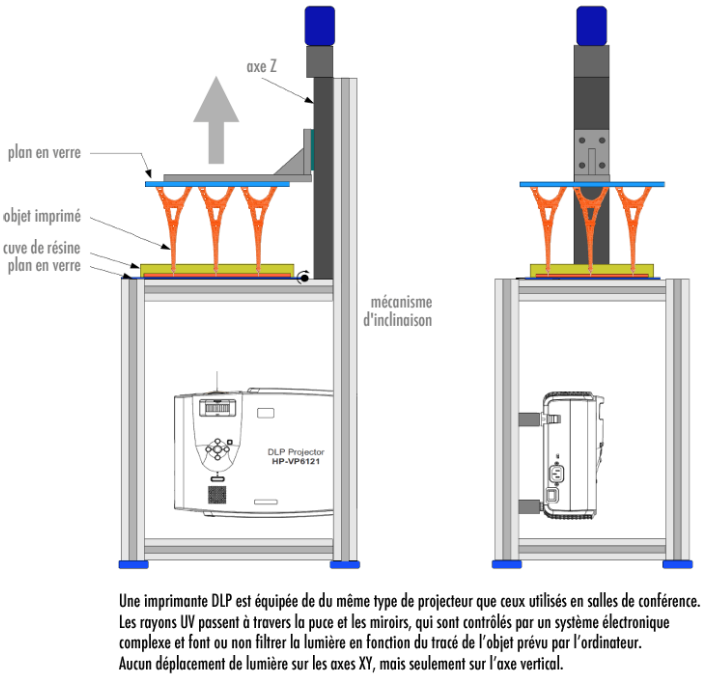


Figure 5 - Principe de fonctionnement d'une imprimante 3D DLP

- La **technologie SLM** (Selective Laser Melting) :

  Technologie se basant sur la fusion de particules de poudre métallique (et non pas de la fritter comme la SLS).

- La **technologie EBM** (Electron Beam Melting) :

  Sa particularité réside en son avantage de faire fondre la matière jusqu'à 1,000°C pour faire fondre des particules comme le titane. Elle se base sur la SLS, mais est lente et couteuse, et ses domaines d'application sont l'aérospatial, la défense et le médical.

- La **technologie LOM** (Laminated Object Manufacturing) par encollage de papier :

  A l'aide de feuilles (de papier, plastique ou métal laminé) encollées les unes au-dessus des autres sous l'effet de chaleur exercé par une presse, les couches se solidarisent avant qu'un laser ne viennent découper les feuilles pour leur donner la forme associée.

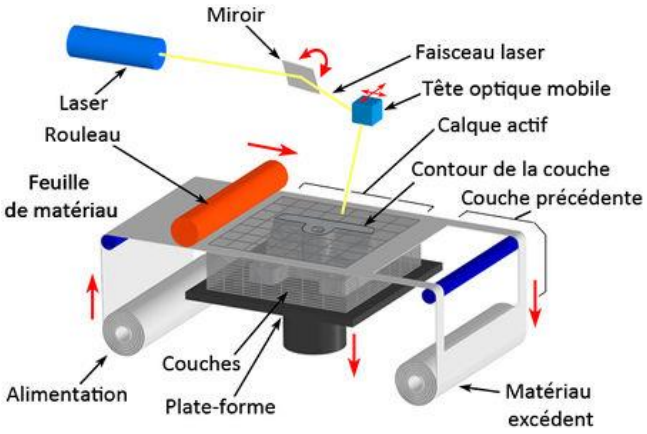


Figure 6 - Principe de fonctionnement d'une imprimante 3D LOM

- La **technologie BJ** (Binder Jetting) ou **projection de liant** :
  Cette technique nécessite l'utilisation d'une poudre et d'un agent de liaison. Cette poudre est déposée sur un plateau et les pièces 3D seront consolidées grâce à l'agent de liaison couche par couche.
- La **technologie MJ** (Material Jetting) ou encore **projection de matériaux** :
  Cette technique projette de la matière couche par couche afin de réaliser un objet 3D.

Après ce mapping des différents procédés d'impression 3D, il faut s'intéresser au robot réaliser dans le projet CENTAURE.

En effet, en ***Figure 7 - Robot industriel de fabrication additive*** ci-dessous, vous pouvez voir le prototype réalisé.

Figure 7 - Robot industriel de fabrication additive

Le robot Kuka est relié en fin de bras à une buse d'impression réalisée par *ALL-TRENDS*. Cette buse d'impression se compose d'un vis sans fin afin de pousser la matière vers la buse, d'un fuseau, de deux colliers chauffants haut et bas, d'un réservoir à granulés, d'une buse de diamètres variables et de deux ventilateurs. Le plateau utilisé pour cette imprimante 3D est un plateau fixe chauffant en verre.

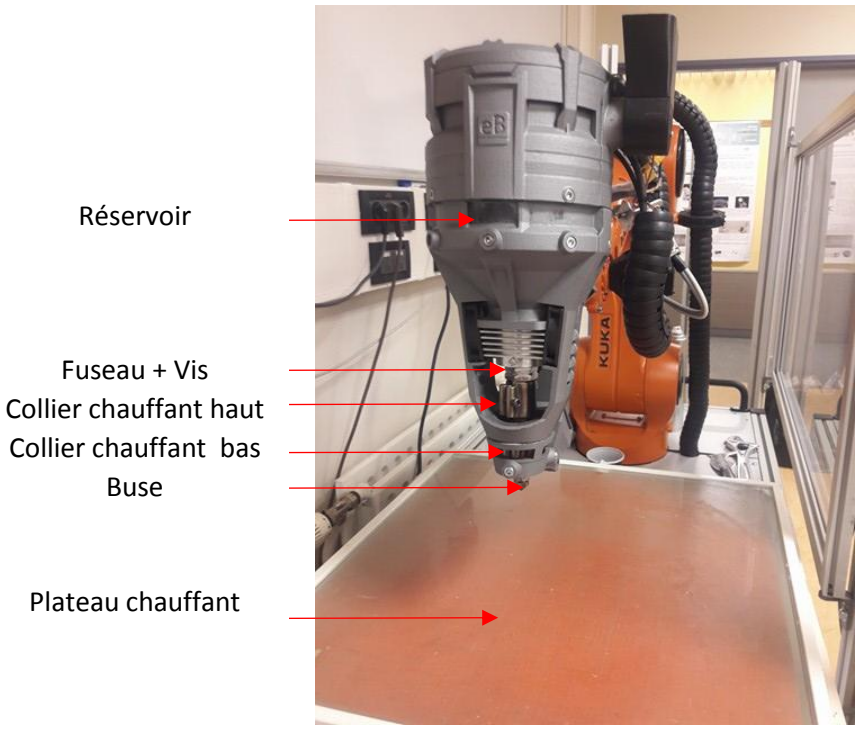


Figure 8 - Tête d'impression et plateau chauffant

Tout cela est contrôlé via un contrôleur *CompactRIO* de *National Instruments* contrôlé par le logiciel *Labview*.

Lors d'un précédent stage réalisé par un étudiant en Master MRT à Lille, une interface Homme-machine avait été réalisée via *Labview* visant à contrôler le robot et les températures des différents colliers et du plateau via un MODBUS.

Cette interface se présente en ***ANNEXE 1 – Interface de contrôle de la tête d'impression*** et ***ANNEXE 2 – Diagramme de contrôle de la tête d'impression***. Dans la première annexe, on remarque que l'espace est scindé en deux. En effet, dans la partie gauche, on effectue les contrôles du moteur de la tête, des colliers haut et bas et du plateau chauffant. Sur la droite de l'annexe 1, on peut contrôler l'état du système, s'il est en chauffe, s'il est initialisé…

Ci-dessous le compact RIO, ses modules, le câblage et les afficheurs de régulation de température :

Figure 9 - Régulateurs de température contrôlé via liaison Ethernet

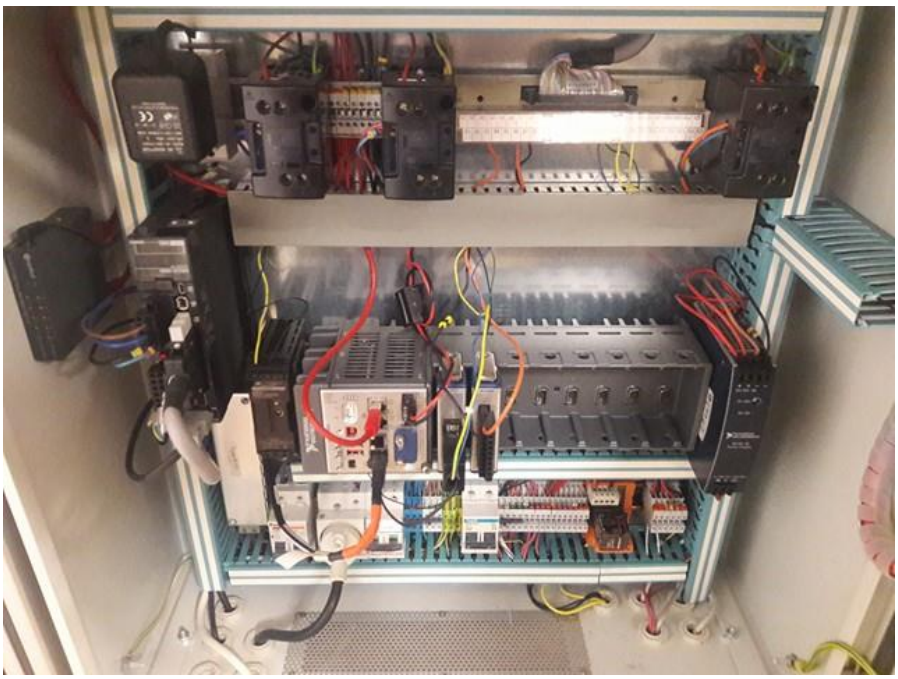


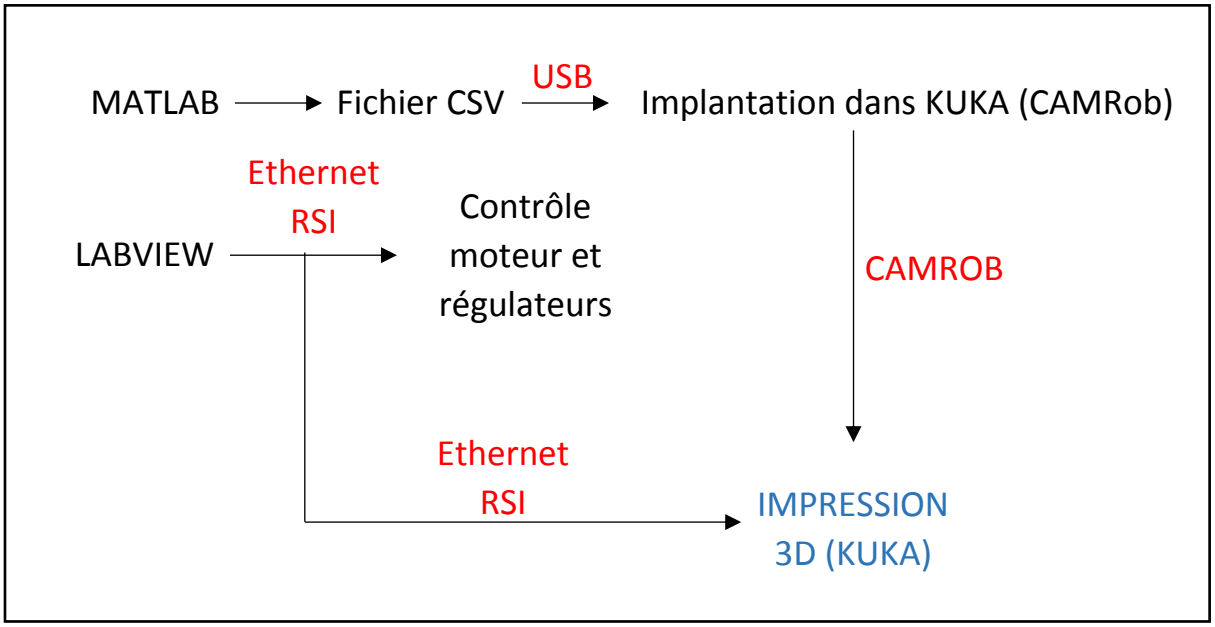
Figure 10 – CompactRIO et ses deux modules Mod1 et Mod2, DIO permettant l'acquittement des défauts, arrêt/démarrer, sens direct/indirect, et AO pour le contrôle moteur tête d'impression

1.2. Architecture actuelle

Avant le début de mon PFE, la génération d'un fichier CSV pouvant être lu par le logiciel **CAMROB** de KUKA était faite sur la génération de trajectoire via *MatLab* et permettait de générer des formes comme les suivantes :

Figure 11 - Pièces imprimées en PS avec MatLab

Donc dans ce premier temps, l'architecture matérielle et logicielle se présentait comme telle :



1.3. Cahier des charges initial et objectifs

Avant de débuter un projet, il faut définir le cahier des charges afin d'identifier au mieux le poste de travail.

Le but de ce projet est de concevoir des pièces en impression 3D par fabrication additive avec un robot industriel et une buse d'impression à granulé non propriétaire.

Pour se faire, j'ai repris le projet d'un étudiant en master MRT ayant déjà réalisé le câblage et l'interface permettant de contrôler les régulateurs de température des colliers et du plateau chauffant via *Labview*.

Les objectifs initiaux de mon projet étaient :

- Prendre en main le logiciel *Labview* ;
- Tester différents types de matériaux et d'en contrôler la régulation de température en fonction du type de granulé ;
- Intégrer le programme de commande déjà réalisé sur *CompactRIO* vers un *MyRIO* de *National Instruments*.

2. Travail effectué

2.1. Cahier des charges évolué

Au fur et à mesure de mon PFE, les objectifs ont été remaniés car en effet, pour les tests des différents matériaux, il aurait été utile de s'y connaître en polymères et donc en plasturgie, connaissances dont je ne disposais pas. De plus, la commande et donc réception du *MyRIO* ayant été conduite au 15 février, l'intégration n'a pu se faire dans les temps.

Donc, pour résumer, après modification des objectifs, voici les étapes de mon PFE :

- Réalisation d'un interpréteur G-Code afin de pouvoir imprimer des pièces pleines, de faire varier les vitesses et accélération du robot et donc d'effectuer un remplissage ;
- Intégrer une communication RSI permettant de transmettre directement les données depuis *Labview* vers le robot KUKA sans passer via CAMROB car coûteux et plus distribué ;
- Tests effectués les vendredis en présence de Gregory SANT et en collaboration avec Othman LAKHAL.

2.2. Réalisation de l'interpréteur G-Code

2.2.1. Prise en main de *Labview*

Durant la première semaine de trois jours de mon PFE, j'ai pris en main le logiciel *Labview* dont je disposais de peu de compétences.

J'ai effectué quelques essais sur la jambe de cheval de la salle C002 afin de configurer des entrées/sorties avec le logiciel.

Le vendredi après-midi, des essais ont été réalisés afin de tester la matière et les trajectoires. J'ai donc pu voir le fonctionnement du robot. Suite à la génération du fichier CSV par *Matlab*, les pièces imprimées étaient toujours basées sur des trajectoires tracées via le logiciel d'un unique dépôt par couche et de manière creuse. Afin de pouvoir contribuer à la suppression de *Matlab*, nous avons souhaité réaliser des pièces 3D via un Slicer après modélisation via un logiciel de CAO.

Après cette première semaine, l'installation de tous les modules complémentaires à *Labview*, j'ai pu commencer à réaliser une première ébauche d'interpréteur.

2.2.2. Récupération du code existant et présentation sur le robot Kuka

Afin de pouvoir récupérer les données du fichier G-CODE pour les indexer dans un tableau il faut donc :

- Charger le fichier à ouvrir dans la boîte de dialogue ;
- Lire le fichier ligne par ligne dans Labview ;
- Indiquer le format d'indexation (en l'occurrence ici chaine) et le charger dans un tableau d'une dimension.

En ***ANNEXE 3 – Chargement des lignes du G-Code,*** ANNEXE 3 –la face avant utilisateur présente le lien de chargement du fichier G-Code et le tableau en sortie de la conversion G-Code – Tableau faite grâce à la programmation du diagramme sur la droite de l'image.

2.2.3. Réalisation de l'interface G-Code – Comma Separated Values

2.2.3.1. Récupération des données du G-Code

Afin de savoir quelles données récupérer lors de l'extraction des valeurs à utiliser par le robot, il est nécessaire de connaître les codes-commandes G utiles.

Un G-Code se présente de la manière suivante :

```
;FLAVOR:RepRap
;TIME:2702
;Filament used: 29.2738m
;Layer height: 0.6
;Generated with Cura_SteamEngine 3.1.0
T0
M190 S20
M104 S265
M109 S265
M82 ; absolute extrusion mode
G28 ;Home
G1 Z15.0 F6000 ;Move the platform down 15mm
;Prime the extruder
G92 E0
G1 F200 E3
G92 E0
M83 ; relative extrusion mode
;LAYER_COUNT:42
;LAYER:0
M107
G0 F3600 X41.466 Y15.619 Z0.3
;TYPE:WALL-OUTER
G1 F1200 X40.707 Y18.45 E0.69972
G1 X37.891 Y23.327 E1.34445
G1 X38.046 Y23.607 E0.0764
G1 X38.222 Y23.982 E0.09889
G1 X38.365 Y24.374 E0.09962
```

Figure 12 - Exemple de G-Code généré ici par le Slicer CURA

Les commandes selon les M codes ne sont pas à prendre en compte lors de l'extraction des données.

Les commandes retenues sont donc les commandes des codes G0, G1 et G92, pour les déplacements selon X, Y et Z ainsi que la vitesse d'extrudage :

- G0 : commandes du mouvement en déplacement rapide ;
- G1 : commandes du mouvement en déplacement linéaire ;
- G92 : définition de la position.

Cependant, les Slicer ne délivrant pas les orientation A, B et C, il va falloir déterminer un système permettant d'utiliser toutes les fonctionnalités du robot KUKA.

Afin de récupérer les données selon les variable X, Y et Z, j'ai effectué de la même façon la programmation suivante :

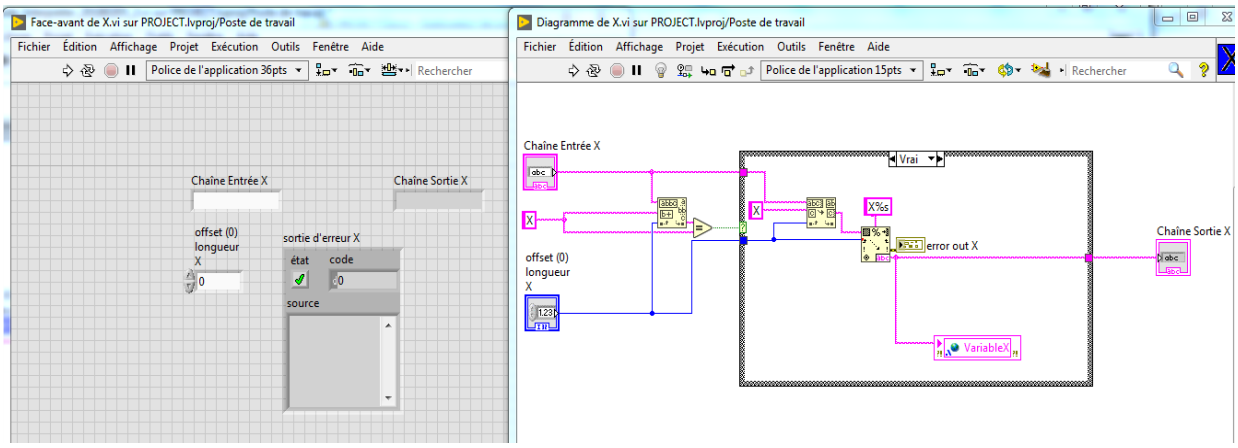


Figure 13 - Face avant et Diagramme de récupération des données selon X (identique pour Y et Z)

Ainsi, lors de la lecture de la ligne du tableau correspondante, dès lors que la lettre X (réciproquement Y ou Z) est rencontrée, alors celle-ci est récupérée et placée dans une variable globale qui permettra de charger le tableau du fichier CSV :

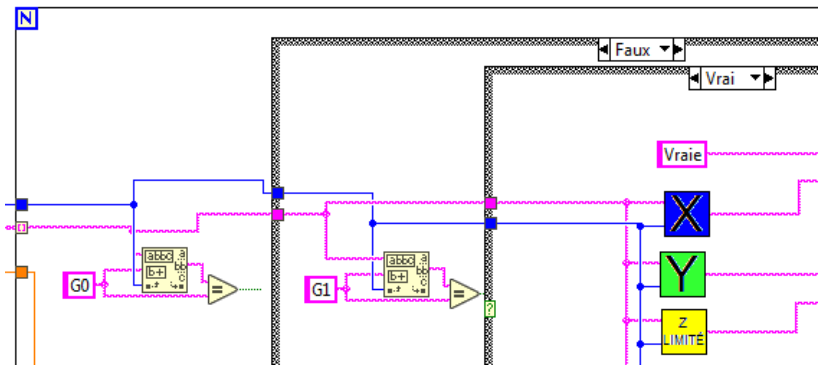


Figure 14 - Extrait du diagramme de l'interpréteur G-Code pour chargement des données X, Y et Z

2.2.3.2. Calcul de la vitesse du bras et de rotation des moteurs

Afin de pouvoir contrôler la vitesse du bras et d'en influencer le débit, donc la vitesse moteur, il est important de connaître la formule à utiliser.

Du coup, on récupère la variable F du fichier G-Code. Pour calculer la vitesse de de déplacement du bras et de la buse, je convertis les données en m/s et je limite la vitesse à 0.1m/s.

Ce calcul est réalisé dans le diagramme suivant :

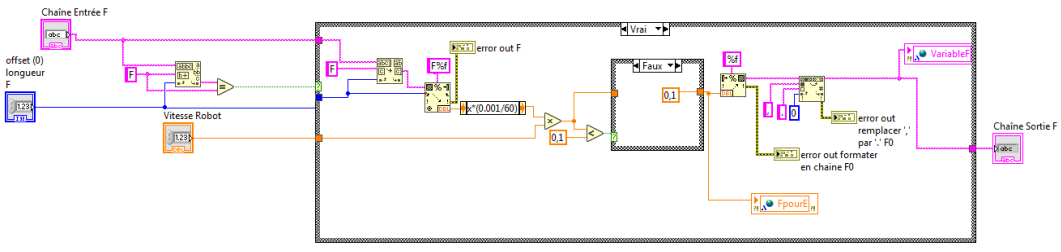


Figure 15 - Récupération de la variable F (vitesse d'avance du bras)

De plus, pour déterminer la variable de vitesse moteur, on utilise la commande F du G-Code. Pour calculer la valeur de la vitesse moteur, j'effectue le calcul suivant :

- On peut déterminer la formule en prenant l'exemple sur la vitesse de ration calculée pour un fraisage ;
- La formule du calcul utilisé pour le fraisage est la suivante :
  $V = \frac{Vitesse\ de\ coupe * 1{,}000}{diamètre\ buse * \pi}$
- Dans notre cas, on remplace la vitesse de coupe par la vitesse d'avance qui correspond à la variable F du fichier G-Code.

Cela donne donc le VI suivant :

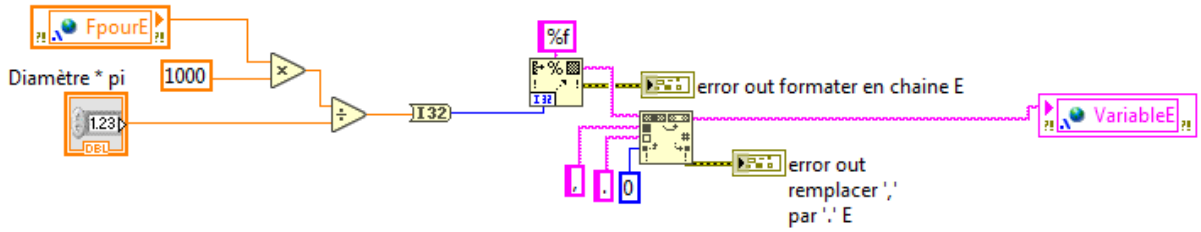


Figure 16 - Calcul de E à partir de F

Cette commande, si elle est égale à 0 sera imposée négative (-20 tours/min) afin d'éviter tout dépôt de matière inutile.

Le VI réalisé correspondant à la valeur E est donc le suivant :

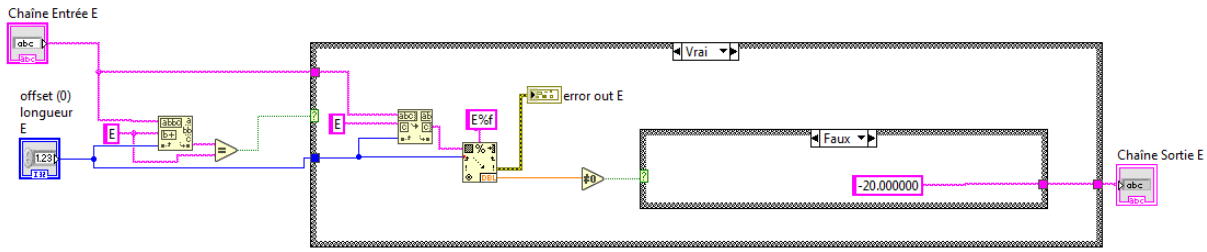


Figure 17 - Rotation moteur vis

2.2.3.3. L'interface finale

ANNEXE 4 – Initialisation des variables et tests de non-vides :

Dans un premier temps, il faut initialiser les variables globales utilisée dans le VI principal et les sous-VI. De plus, il faut tester si les valeurs entrées dans l'interface utilisateur sons vides ou non et bridé l'exécution du programme si les valeurs ne sont pas renseignées.

ANNEXE 5 – Chargement du G-Code :

Dans un second temps, on effectue le chargement des données G-Code et des paramètres entrés dans l'interface utilisateur dans un tableau de deux dimensions.

ANNEXE 6 – Récupération des données X, Y, Z, des orientations A, B, C, de la vitesse du bras F, du calcul de la vitesse moteur E, et de tous les paramètres :

Le logiciel CAMRob est paramétré de sorte à recevoir les données comme indiquées en ***Figure 18 - Format du tableau CSV*** :

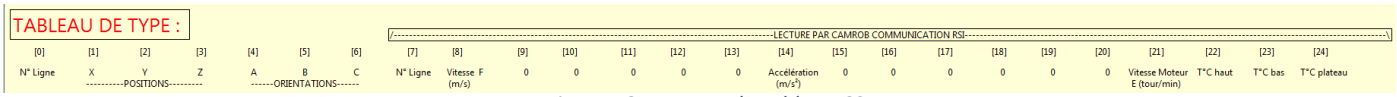


Figure 18 - Format du tableau CSV

Donc tous les paramètres seront chargés de manière à avoir en colonne :

- 0 et 7 : N° ligne de l'instruction ;
- 1 à 3 : positions X, Y et Z ;
- 4 à 6 : orientations A, B et C ;
- 8 : vitesse en m/s du bras du robot limitée à 0.1m/s
- 14 : accélération en m/s² fixée par l'utilisateur, inférieure à 4 ;
- 21 : vitesse moteur de la vis en tour/min ;
- 22 à 24 : températures en fonction du matériau du collier haut, du collier bas et du plateau ;
- 9 à 13, 15 à 20 : non utilisées par CAMRob.

ANNEXE 7 – Suppression des lignes inutilisées , compte du numéro des lignes et chargement du tableau dans un fichier CSV séparé par un ' ;'. :

Dans cette partie, je supprime les lignes dont les données sont vides, et je passe toutes les valeurs récupérées ayant une virgule en un point. En effet, CAMRob lit les données à l'anglo-saxonne.

Je compte ensuite les lignes du fichier pour les associer aux colonnes correspondantes dans le tableau csv, puis je charge le tableau dans un fichier CSV dont les données sont séparées par un point-virgule.

Voici donc ci-dessous l'interface utilisateur finale :

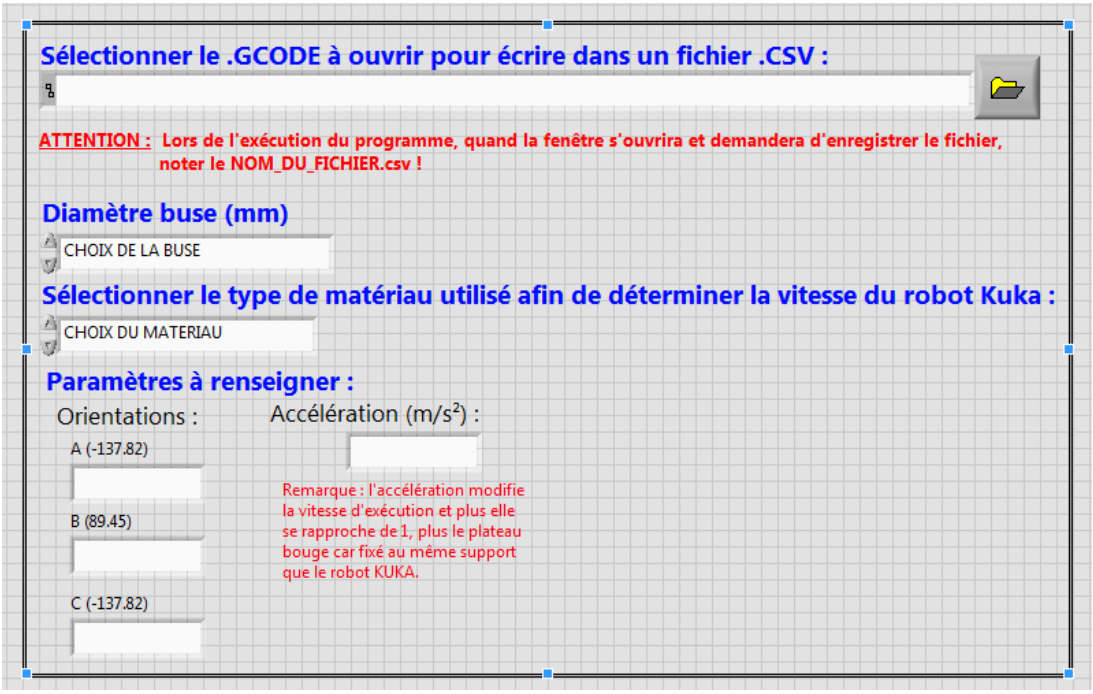


Figure 19 - Interface finale de l'interpréteur G-Code

2.2.3.4. Tests réalisés et apports à l'existant

Durant les semaines 2 à 5, nous avons réalisés des tests les vendredis de fichiers générés par le G-Code.

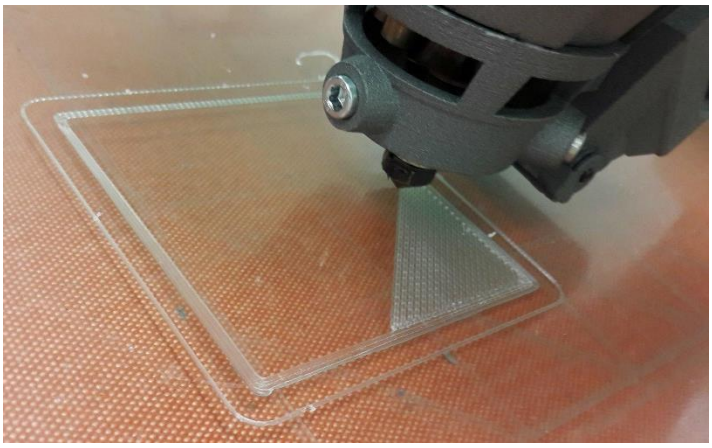


Figure 20 - Impression d'un support

Nous avons donc effectué des premières pièces pleines à l'aide du Slicer *CURA* dont une pièce cubique mais nous nous sommes vite aperçus que le remplissage effectué n'était pas des plus optimal : il commençait le remplissage au milieu, puis partait vers un extérieur, pour revenir ensuite au milieu et partir de l'autre côté. La solidarisation entre les deux passages au milieu n'était pas parfaite du fait du contact de la matière froide et chaude. (Voir figure ci-dessous)

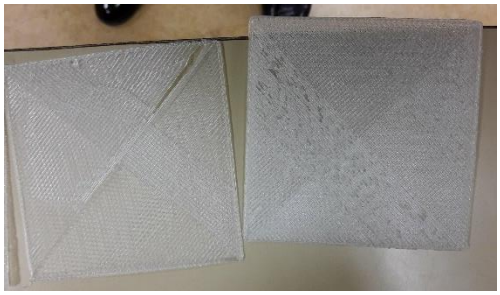


Figure 21 - Slicer CURA : Désolidarisation des pièces à la jointure centrale

Nous avons donc décidé de changer de slicer et de passer sur *SLIC3R* qui nous permet de choisir le remplissage et donc d'éviter un dépôt aléatoire de matière.

Ainsi, les pièces réalisées lors de ces essais les plus prometteuses sont les pièces ci-dessous même si les contours sont encore à améliorer :

Figure 22 - Pièces avec remplissage et déportée convenables

Les paramètres le plus proches de la meilleure application obtenue sont détaillés en ***ANNEXE 8 – Paramètres sur SLIC3R.***

Mon principal apport dans cette première partie de PFE a donc été d'offrir la possibilité de réaliser des pièces pleines, avec support et/ou déportées ainsi que de permettre de supprimer l'utilisation de *Matlab* dans la génération des trajectoire et d'utiliser n'importe quel logiciel de CAO ou Slicer (avec une préfrence pour SLIC3R dans les deux testés).

2.2.4. Difficultés rencontrées

Les principales difficultés rencontrées sont :

- les contraintes dues à l'environnement, en effet, les courants d'air ou autre modification influent sur la qualité du fil de sortie de l'extrudeuse ;
- l'influence du Slicer ;
- le bouchon de matière obligeant au démontage de la tête d'impression ;
- les contraintes logicielles ;
- l'absence de connaissance du matériau (en l'occurrence, nous avons uniquement travaillé sur le polystyrène PS).

2.3. Connexion RSI (*Robot Sensor Interface), programmation KRL*

2.3.1. RSI et KRL

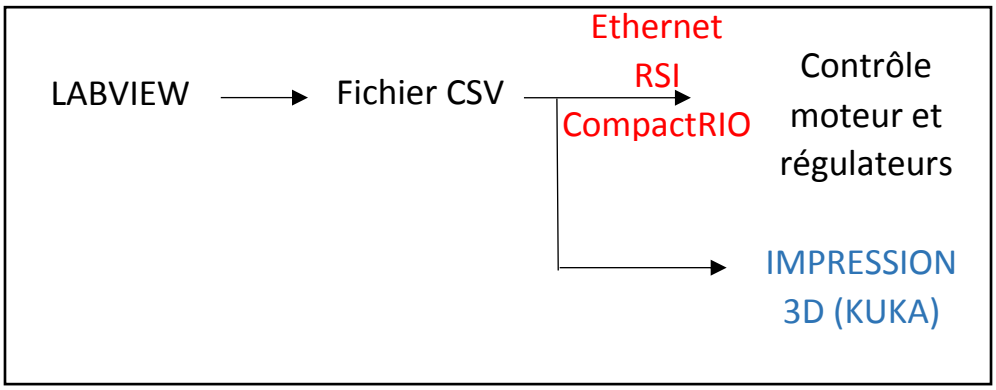
Dans une seconde partie de mon PFE, le but était de supprimer les manipulations entre la génération du fichier CSV et les commandes robot.

Ainsi, il a fallu prendre en main la communication RSI.

Le logiciel *RobotSensorInterface* permet de programmer le traitement des signaux entrants et sortants.

Afin de permettre l'envoi et la réception des données entre le PC et le robot KUKA, j'ai établi un transfert de données via câble Ethernet.

Cela dans l'objectif d'obtenir une architecture matérielle et logicielle comme suit :



2.3.1.1. Fichier RSI

Le fichier RSI permet de traduire l'échange des données effectué entre le PC (*Labview*) et le robot KUKA via une interface par bloc.

En entrée de la liaison Ethernet, j'ai inscrit la variable 'Next' qui permet de passer à l'instruction suivante une fois que le point est atteint, et en sortie, les variables à envoyer via la lisaion.

Il faut faire attention à indiquer les bons index et le bon fichier XML correspondant dans le bloc Ethernet.

Le fichier RSI est visible en ***ANNEXE 9 - FICHIER RSI***.

2.3.1.2. Fichier XML

Un fichier XML (ou eXtensible Markup Language) est un langage informatique de balisage qui permet de relier les entrées et sorties du fichier RSI au fichier Labview correspondant aux données que l'on souhaite envoyer au robot. Ce document est structuré et organisé, et permet de faciliter l'échange de données entre le PC et le robot.

PROJET3D_Config.src | PROJET3D_Config.xml

```
<ROOT>
   <CONFIG>
      <IP_NUMBER>192.168.20.10</IP_NUMBER>   <!-- IP-number of the external socket -->
      <PORT>49152</PORT>                     <!-- Port-number of the external socket -->
      <SENTYPE>ImFree</SENTYPE>          <!-- The name of your system send in <Sen Type="" > -->
      <ONLYSEND>FALSE</ONLYSEND>         <!-- TRUE means the client don't expect answers. Do not send anything to robot -->
   </CONFIG>
   <!-- RSI Data: TYPE=  "BOOL", "STRING", "LONG", "DOUBLE" -->
   <!-- INDX= "INTERNAL" switch on internal read values. Needed by DEF_... -->
   <!-- INDX= "nmb" Input/Output index of RSI-Object / Maximum of RSI Channels: 64  -->
   <!-- HOLDON="1", set this output index of RSI Object to the last value  -->
   <!-- DEF_Delay count the late packages and send it back to server  -->
   <!-- DEF_Tech: .T = advance .C = main run / .T1 advance set function generator 1 -->

   <SEND>
      <ELEMENTS>
         <ELEMENT TAG="DEF_RIst" TYPE="DOUBLE" INDX="INTERNAL" />
         <ELEMENT TAG="DEF_RSol" TYPE="DOUBLE" INDX="INTERNAL" />
         <ELEMENT TAG="DEF_Delay" TYPE="LONG" INDX="INTERNAL" />
         <ELEMENT TAG="DEF_Tech.C1" TYPE="DOUBLE" INDX="INTERNAL" />
         <ELEMENT TAG="Next" TYPE="DOUBLE" INDX="1" />
      </ELEMENTS>
   </SEND>
   <RECEIVE>
      <ELEMENTS>
         <ELEMENT TAG="DEF_EStr" TYPE="STRING" INDX="INTERNAL" />
         <ELEMENT TAG="DEF_Tech.T2" TYPE="DOUBLE" INDX="INTERNAL" HOLDON="0" />
         <ELEMENT TAG="X" TYPE="DOUBLE" INDX="1" HOLDON="0" />
         <ELEMENT TAG="Y" TYPE="DOUBLE" INDX="2" HOLDON="0" />
         <ELEMENT TAG="Z" TYPE="DOUBLE" INDX="3" HOLDON="0" />
         <ELEMENT TAG="A" TYPE="DOUBLE" INDX="4" HOLDON="0" />
         <ELEMENT TAG="B" TYPE="DOUBLE" INDX="5" HOLDON="0" />
         <ELEMENT TAG="C" TYPE="DOUBLE" INDX="6" HOLDON="0" />
         <ELEMENT TAG="ARRET" TYPE="DOUBLE" INDX="7" HOLDON="0" />
         <ELEMENT TAG="ACC" TYPE="DOUBLE" INDX="8" HOLDON="0" />
         <ELEMENT TAG="VEL" TYPE="DOUBLE" INDX="9" HOLDON="0" />
      </ELEMENTS>
   </RECEIVE>
</ROOT>
```

Figure 23 - Configuration du fichier XML

En éléments envoyés, j'ai inscrit la variable 'Next' qui permet de passer à la ligne suivante du fichier CSV lorsque le point a été atteint par le robot.

En éléments reçus, j'ai inscrit les variables données dans le G-Code et traduites via l'interpréteur utiles à la mise en mouvement du robot, à savoir les données X, Y, Z, A, B, C, Arret, Accélération et Vitesse. Les index associés à ses variables correspondent aux index énumérés dans le fichier RSI.

2.3.1.3. Fichier SRC

Le fichier source implémenté dans le pendant du robot KUKA permet d'introduire les instructions à faire par le robot en fonction des variables envoyées et reçues ($SEN_PREA[]).

Dans un premier temps, il faut déclarer toutes les variables à utiliser. Ensuite vient l'initialisation du robot, puis l'attribution du premier point et de la vitesse robot. De plus, j'attribue les bons outil et bas correspondant à mon système, et j'ouvre enfin la liaison RSI avec le fichier RSI expliqué en *partie 2.3.1.1.* . Pour finir, j'effectue une boucle permettant d'attribuer les valeurs du tableau au robot qui réceptionne ses $SEN_PREA[] et qui incrémente le 'Next' en fonction de l'atteinte des points.

Le code SRC est disponible en ***ANNEXE 10 – Fichier SRC***.

2.3.2. Interface Labview

Afin de pourvoir envoyer les informations via Labview, j'ai modifié l'interface déjà réalisée par Othman LAKHAL sur le sujet pour la conformer au nouveau fichier XML.

Ainsi, j'ai modifié l'entrée du fichier XML par le modèle de données suivant :

```
<Sen Type="ImFree">
<X>0</X>
<Y>0</Y>
<Z>0</Z>
<A>0</A>
<B>0</B>
<C>0</C>
<VEL>0</VEL>
<ACC>0</ACC>
<ARRET>0</ARRET>
<IPOC>0</IPOC>
</Sen>
```

Il a donc été nécessaire d'adapter les sous-VI du programme pour que les valeurs soient écrites.

2.3.3. Difficultés rencontrées et contribution

Durant les 2 dernières semaines de mon PFE j'ai effectué plusieurs manipulations sur le fichier Labview, ainsi que sur les fichiers SRC, XML et RSI.

Les principales difficultés rencontrées sont les suivantes :

- Durant presque une semaine, j'ai été bloquée suite à l'ajout inopportun d'un espace vide qui ne permettait donc pas la lecture du fichier RSI ;
- Le fichier source 8 jours avant d'être débloqué. En effet, j'avais effectué les essais sur le partage d'une seule donnée via la liaison RSI et le problème d'envoi/réception était correct. Cependant, lorsque j'ai ajouté toutes les valeurs à transmettre, le robot envoyait bien les instructions mais ne recevait aucune variable. Lundi 19 février j'ai donc décidé de reprendre le code ligne par ligne pour voir la différence, et en copiant le programme SRC initial ligne par ligne, j'ai supprimé l'erreur tout en conservant exactement le même code. Je soupçonne donc que des caractères inscrits dans mon fichier posaient problème mais je ne sais pas lesquels.
- Ensuite, toujours dans le fichier source, j'ai eu de la difficulté à trouver le moyen de changer les base et outil pour le système Robot+Buse, mais après lecture de la documentation, l'aide de Othman, j'ai finalement trouvé l'erreur ;
- Pour finir, la dernière difficulté rencontrée réside en l'affectation en temps réelle de la vitesse du robot. En effet, je souhaitais effectuer directement l'affectation via les variables $VEL.CP=$SEN_PREA[9], or celle-ci doit être modifiée via la commande BAS(VEL_CP,$SEN_PREA[9]). J'ai donc passé ma journée de mardi à résoudre ce problème.

Ainsi, grâce à mon travail, les données pourront être envoyées directement de Labview au Kuka sans passer par CAMRob.

3. Suite du projet

3.1. Compte-rendu du reste-à-faire de ma partie

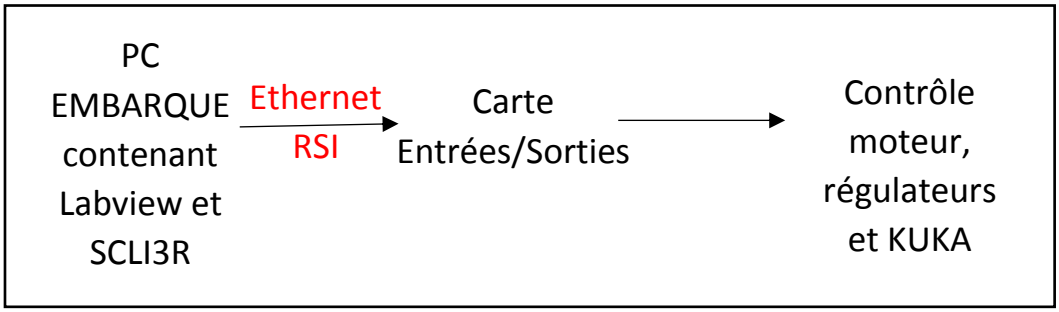
Dans une reprise de PFE, je conseille au futur étudiant travaillant sur la partie informatique industrielle d'effectuer dans un premier temps la liaison entre la sortie de mon VI G-Code_Interpreter_Final.vi, avec le fichier CIMMUNICATION_RSI_OK.vi puis d'effectuer la mise en relation avec le fichier Final_Project_20180220.vi ayant partiellement été modifié pour faciliter les échanges.

3.2. Architecture actuelle et proposition future d'amélioration

Lorsque nous avons reçu le MyRIO, nous nous sommes aperçus que la communication MODBUS n'étaient pas intégrée, or en industrie, l'habitude est de transmettre les données via ce mode de communication. Othman s'est donc renseigné pour de probables modules supplémentaires à rajouter au MyRio ou éventuellement modifier les régulateurs en communication analogique.

Lors de notre réunion le lundi 19 février, ce problème a été mis en évidence mais comme désormais la communication pourrait s'effectuer directement via le PC vers KUKA, la question s'est posée sur le fait de conserver un microcontrôleur ou juste une carte entrées/sorties qui communiquerait en MODBUS car tout est désormais régulé pas Labview.

Ainsi, l'architecture actuelle en **partie RSI *et KRL*** pourrait être simplifiée de la manière suivante :



Pour la suite du projet, un étudiant en Science des Matériaux à Polytech Lille et un étudiant IMA (ou CDD envisagé) seraient repreneurs du sujet.

Conclusion

Au bout de mes six semaines de recherche, il reste encore du travail à effectuer sur la partie informatique industrielle mais aussi concernant les températures de fusion des différents matériaux.

En effet, mon travail s'est principalement porté sur la réalisation d'un interpréteur G-Code vers un fichier CSV, pour ensuite être orienté vers la transmission directe des informations depuis l'interpréteur vers le robot sans utiliser d'interface supplémentaire.

Le prototype reste à finaliser notamment avec l'implantation de l'enceinte thermorégulée, l'implantation ou non d'un MyRIO ou d'une carte d'entrées/sorties.

Ce projet m'a permis d'en apprendre un peu plus sur la recherche et je suis contente d'avoir pu participer au développement actuel de cette future imprimante à fabrication additive industrielle.

Bibliographie

Documentation sur le G-Code :

http://reprap.org/wiki/G-code/fr

Documentation sur le Kuka (KRL, RSI, SRC, XML…):

Fichiers PDF :

KRL-Reference-Guide-v4_1.pdf

KST_CAMRob_KRC_30_en.pdf

KST_RSI_33_fr.pdf

Documentation Labview :

http://www.ni.com/getting-started/labview-basics/f/online-help

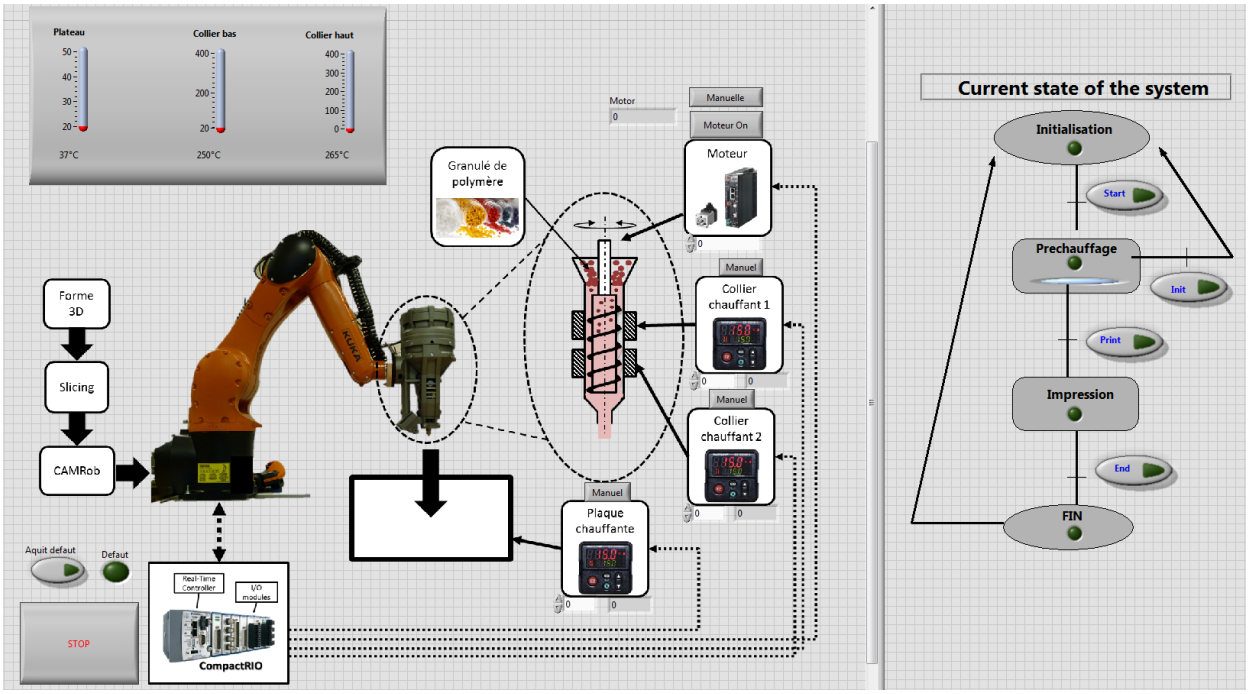
Documentation sur les imprimantes 3D :

https://www.sculpteo.com/fr/impression-3d/technologies-dimpression-3d/

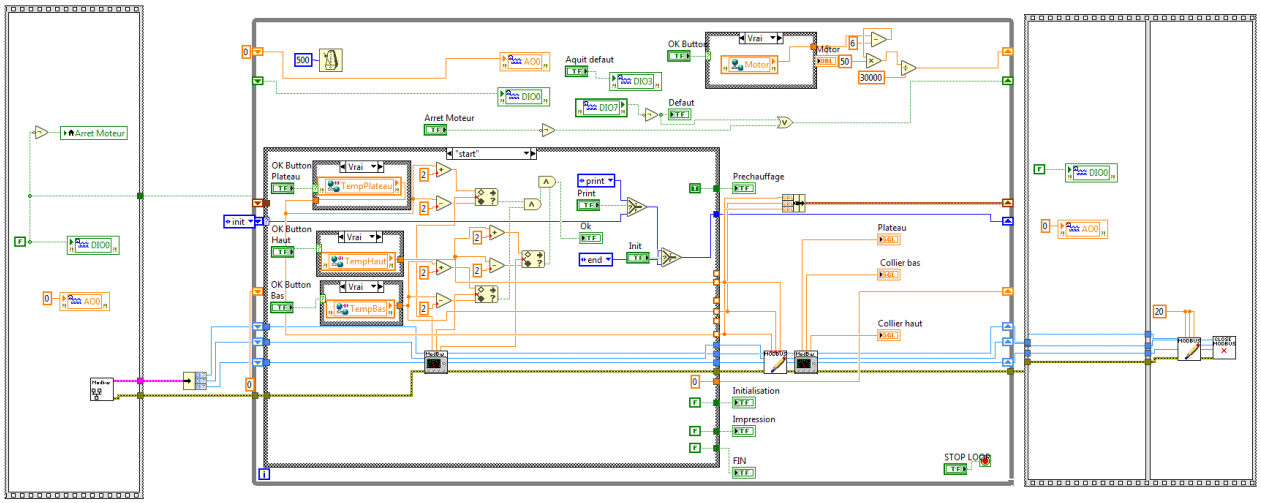
http://www.additiverse.com/tout-savoir-sur-les-technologies-dimpression-3d/

ANNEXES

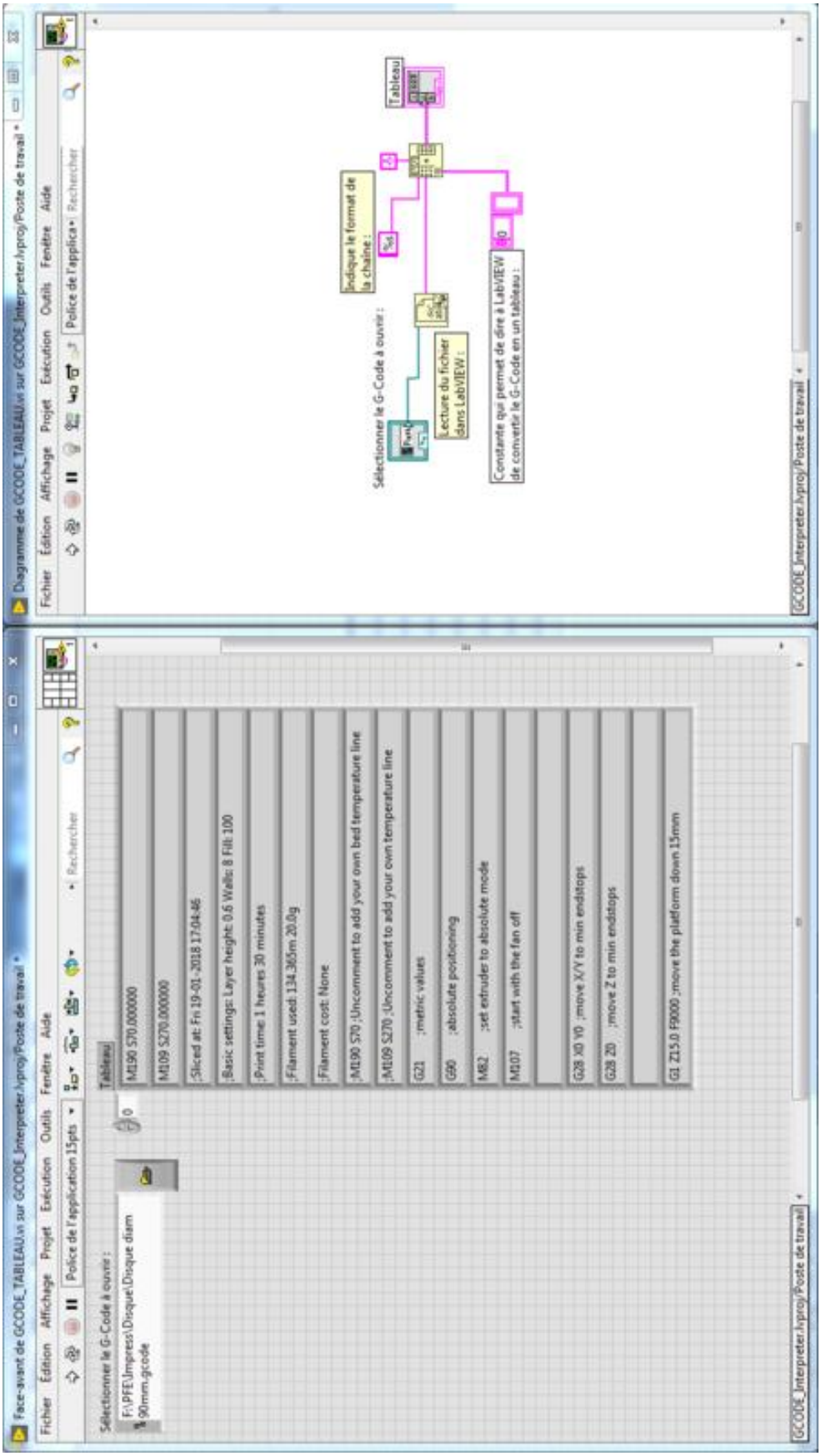
ANNEXE 1 – Interface de contrôle de la tête d'impression



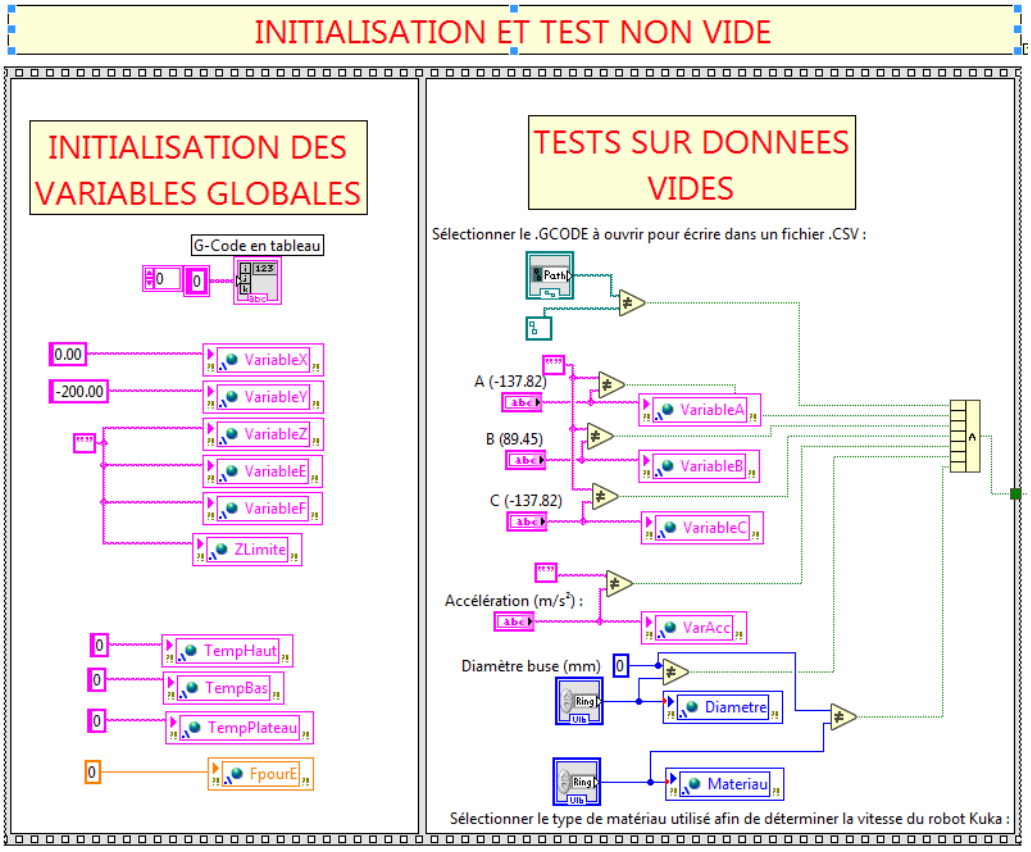
ANNEXE 2 – Diagramme de contrôle de la tête d'impression



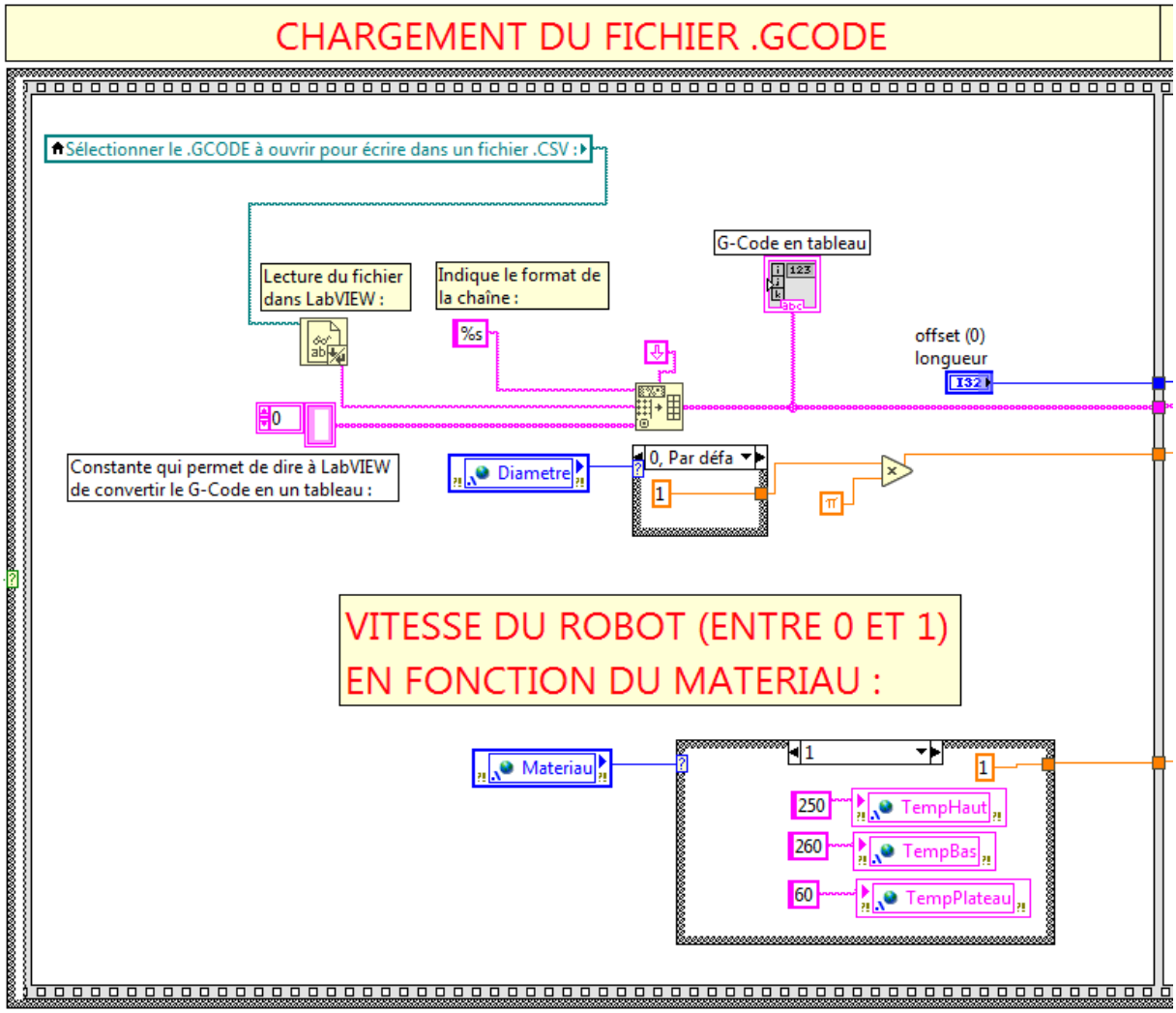
ANNEXE 3 – Chargement des lignes du G-Code



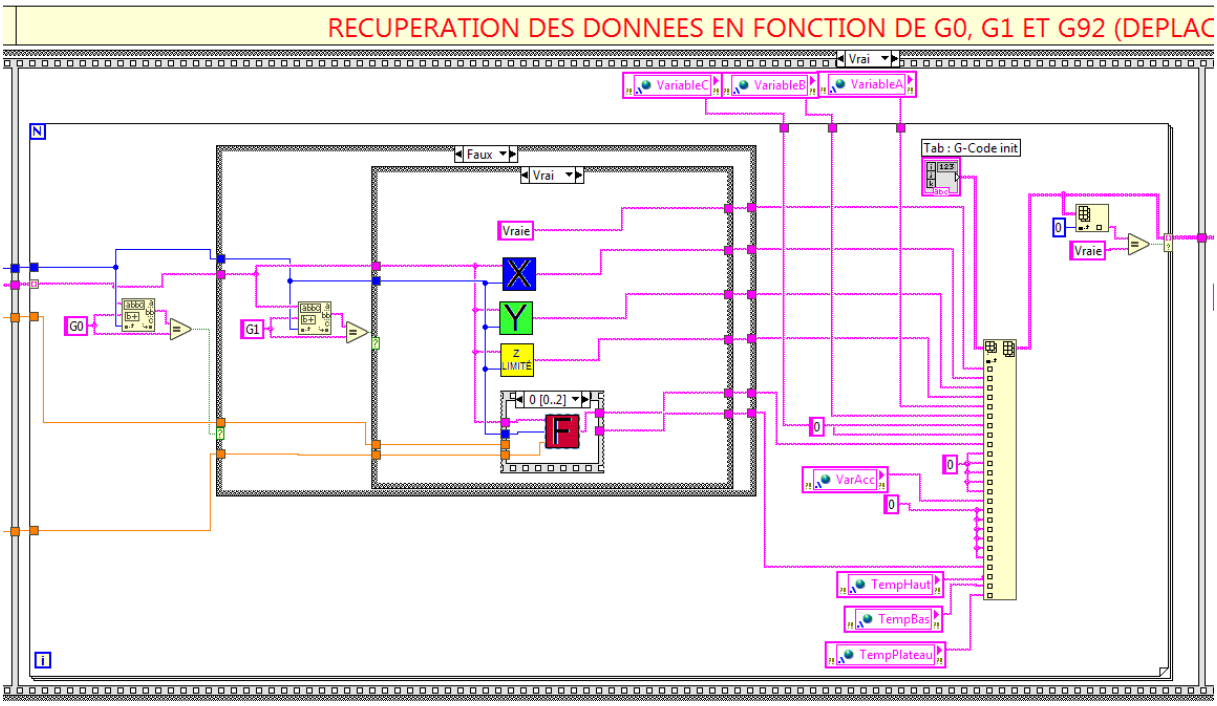
ANNEXE 4 – Initialisation des variables et tests de non-vides



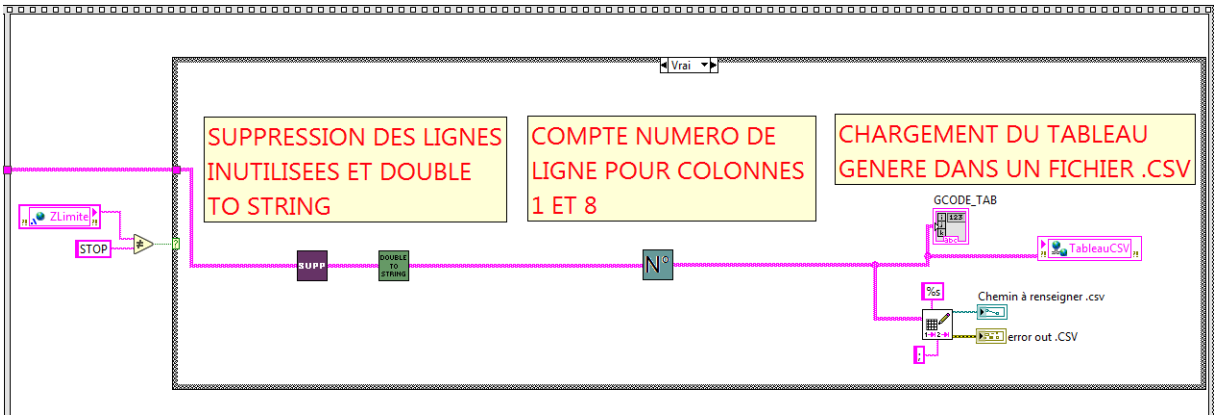
ANNEXE 5 – Chargement du G-Code



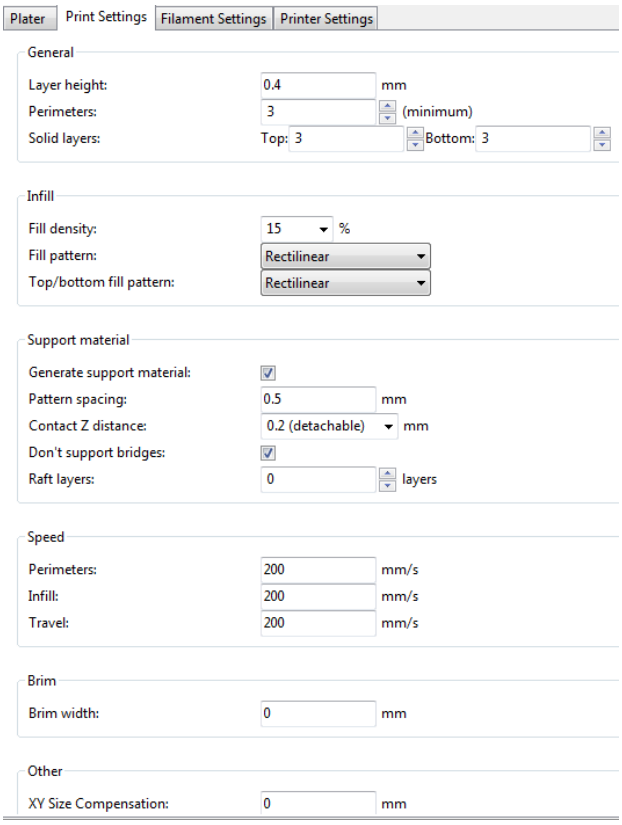
ANNEXE 6 – Récupération des données X, Y, Z, des orientations A, B, C, de la vitesse du bras F, du calcul de la vitesse moteur E, et de tous les paramètres

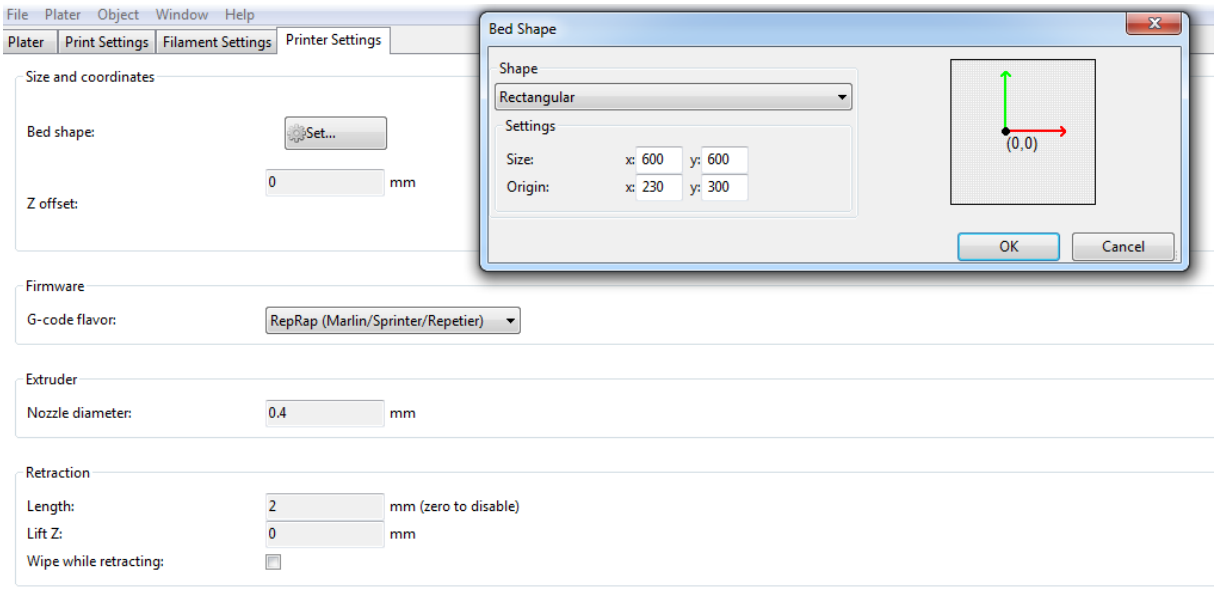


ANNEXE 7 – Suppression des lignes inutilisées , compte du numéro des lignes et chargement du tableau dans un fichier CSV séparé par un ‘ ;’.



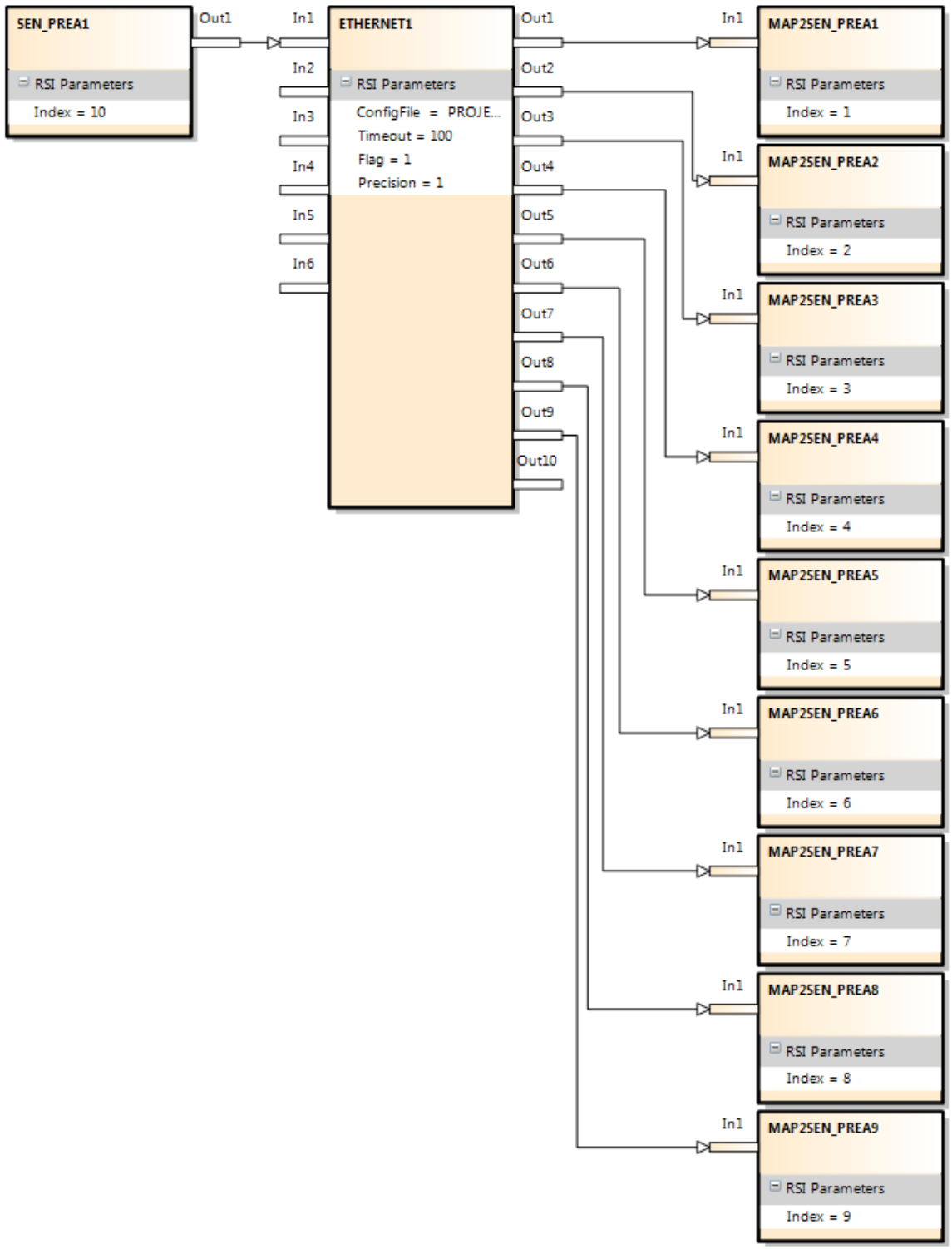
ANNEXE 8 – Paramètres sur SLIC3R





ANNEXE 9 - FICHIER RSI

PROJET3D_Config.rsi



ANNEXE 10 – Fichier SRC

PROJET3D_Config.src | PROJET3D_Config.xml

```
&ACCESS RVP
&REL 6
&PARAM TEMPLATE = C:\KRC\Roboter\Template\vorgabe
&PARAM EDITMASK = *
DEF RSI_PROJET3D( )
; ===========================================
;
; RSI : ETHERNET communication between Labview and Kuka Robot Interface
; Realtime UDP data exchange with server application
;
; ===========================================

; Declaration of KRL variables
DECL INT ret    ; Return value for RSI commands

DECL INT CONTID ; ContainerID
DECL REAL VARX
DECL REAL VARY
DECL REAL VARZ
DECL REAL VARA
DECL REAL VARB
DECL REAL VARC
DECL REAL ARRET
DECL REAL VARACC
DECL REAL VARVEL
DECL POS LINTARGET
DECL POS POINT1

; INITIALISATION :
;FOLD INI
  ;FOLD BASISTECH INI
    GLOBAL INTERRUPT DECL 3 WHEN $STOPMESS==TRUE DO IR_STOPM ( )
    INTERRUPT ON 3
    BAS (#INITMOV,0 )
  ;ENDFOLD (BASISTECH INI)
  ;FOLD USER INI
    ;Make your modifications here

  ;ENDFOLD (USER INI)
;ENDFOLD (INI)

; VALEURS DES VARIABLES ASSOCIEES AU PREMIER POINT A ATTEINDRE
VARX=100 ; POSITION X
VARY=0 ; POSITION Y
VARZ=200 ; POSITION Z
VARA=-133 ; ORIENTATION A
VARB=90 ; ORIENTATION B
VARC=-133 ; ORIENTATION C
ARRET=0 ; VARIABLE STOP
VARACC=0 ; ACCELERATION
VARVEL=0 ; VITESSE

```

```
; PREMIER POINT A ATTEINDRE
LINTARGET.X=VARX
LINTARGET.Y=VARY
LINTARGET.Z=VARZ
LINTARGET.A=VARA
LINTARGET.B=VARB
LINTARGET.C=VARC

$VEL.CP=0.1
$APO.CVEL=100
$ACC.CP=0.1

; HOME
; MOVE TO START POSITION EN PTP
PTP {A1 0, A2 -90, A3 90, A4 0, A5 0, A6 0}

; SELECTION DE LA BASE ET DE L'OUTIL (ATTENTION NE CHANGE PAS L'ADDICHAGE SUR LE PENDANT KUKA)
$TOOL=tool_data[2]
$BASE=base_data[2]

; POINT DE SECURITE ET D'EXTRUDAGE
PTP {A1 22, A2 -80, A3 100, A4 48, A5 -31, A6 -44}

; CREATION DU CONTEXTE RSI
ret = RSI_CREATE("PROJET3D_Config.rsi",CONTID,TRUE)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; DEBUT DE LA CONNEXION RSI
ret = RSI_ON(#ABSOLUTE, #IPO)
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; ATTEINTE DU POINT DE SECURITE ET D'EXTRUDAGE A LA VITESSE $VEL.CP
POINT1=LINTARGET
LIN POINT1 C_VEL

LOOP
    ;POINT A ATTEINDRE EN LINEAIRE TANT QUE LA BOUCLE N'EST PAS FINIE
    LINTARGET.X=$SEN_PREA[1]
    LINTARGET.Y=$SEN_PREA[2]
    LINTARGET.Z=$SEN_PREA[3]
    LINTARGET.A=$SEN_PREA[4]
    LINTARGET.B=$SEN_PREA[5]
    LINTARGET.C=$SEN_PREA[6]

    ARRET = $SEN_PREA[7]

    ; AFFECTATION DES VARIABLES
    VARACC = $SEN_PREA[8]
    VARVEL = $SEN_PREA[9]
    VARVEL=ABS($SEN_PREA[9])
                    _
    ; ASSOCIATION DES VALEURS DE VITESSE ET D'ACCELERATION AVEC LES VARIABLES CORRESPONDANTES
    BAS(#VEL_CP, VARVEL)
    ;$VEL.CP=VARVEL
    $APO.CVEL=100

    BAS(#ACC_CP, VARACC)
    ;$ACC.CP=VARACC

    LIN LINTARGET C_VEL

    ; INCREMENTATION
    $SEN_PREA[10]=$SEN_PREA[10]+1

    ; ARRET DE L'EXECUTION
    IF $SEN_PREA[7]>0 THEN
        EXIT
    ENDIF
ENDLOOP

; Turn off RSI
ret = RSI_OFF()
IF (ret <> RSIOK) THEN
  HALT
ENDIF

; HOME
PTP {A1 0, A2 -90, A3 90, A4 0, A5 0, A6 0}

END
```

ANNEXE 11 – VI De communication RSI