République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE SAAD DAHLEB BLIDA 1

Faculté des Sciences et de Technologie

Département d'Electronique

MEMOIRE DE MASTER

Spécialité : Electronique des Systèmes Embarqués

Conception et réalisation électromécanique d'une imprimante 3D, avec son interface graphique son système d'exploitation et son Firmware embarqués.

Par

Riad BENHALLA & **Nawfel KRITLI**

Promoteur

Madame **Hamida BOUGHERIRA**

Blida, Juillet 2020

Année Universitaire
2019-2020

بسم الله الرحمان الرحيم

Remerciements

Tout d'abord, nous remercions Dieu Tout-Puissant de nous avoir donné la santé et la volonté pour faire cette humble tâche.

Nous exprimons nos profonds remerciements à notre promoteur Madame le professeur "BOUGHRIRA Hamida" pour l'aide compétente et son encouragement qu'elle nous a apportée, nous avons eu la chance que vous soyez la directrice du mémoire.

Nous remercions sincèrement tous les enseignants, conférenciers et toutes les personnes qui ont guidé nos réflexions sur leurs paroles, leurs écrits, leurs conseils et leurs critiques, et ont accepté de nous rencontrer et de répondre à nos questions pendant nos recherches.

Nos remerciements s'adressent ainsi aux : M. le président et les membres de jury d'avoir accepté de juger et évaluer notre travail.

Enfin, nous voulons remercier toutes les personnes qui nous ont aidé à réaliser ce projet, en particulier M. Brahimi Sidali pour son aide précieuse, surtout dans cette période éprouvante.

Qu'il nous soit enfin permis de remercier toute nos familles pour leur amour et leur soutien constant. Nous leur dédions ce mémoire.

Table des matières

Liste des figures

Liste des tableaux

Acronymes et abréviations

FDM : Fused Deposit Modeling.

3D : Trois Dimension.

DSP : Digital Signal Processor.

CNC : Computer Numerical Control.

STL : Stereolithography.

PCB : Printed Circuit Board.

SLA : Stéréo Lithographie Apparatus.

RepRap : Replicating Rapid prototyper.

CLIP : Continuous Liquid Interface Production.

HP : Hewlett-Packard.

CAO : Conception Assistée par Ordinateur.

FFF : Fused Filament Fabrication.

ABS : Acrylonitrile Butadiene Styrene.

PLA : Polylactic Acid.

UV : Ultraviolet.

MIT : Massachusetts Institue of Technology.

DMLS : Direct Metal Laser Sintering.

OS : Operating System.

CPU : Central Processing Unit.

RTOS : Real Time Operating System.

GUI : Graphical User Interface.

LCD : Liquid Cristal Display.

Mosfet : Metal-oxide-semiconductor field-effect transistor.

NC : Numerical Control.

PWM : Pulse Width Modulation.

UART : Universal Asynchronous Receiver Transmitter.

NTC : Negative Temperature Coefficent.

RTD : Resistance Temperature Detector.

CMOS : Complementary Metal Oxide Semiconductor.

AVR : Alf and Vegard’s RISC processor.

RISC : Reduced Instruction Set Computer.

MIPS : Million Instructions Per Seconds

ALU : Arithmetic and Logic Unit.

CISC : Complex Instruction Set Computer.

ICSP : In Circuit Serial Programming.

RAM : Random Access Memory.

ASCII : American Strandard Code for Information Interchange.

NO : Normally Open.

ATX : Advanced Technology Extended.

IDE : Integrated Development Environment.

AWT : Abstract Window Toolkit.

PID : Proportional Integral Derivative.

Mutex : Mutual Exclusion.

RBPI : Recheche Binaire Par Intervalle.

DDM : Démarrages Des Moteurs.

ADC : Analog to Digital Converter.

Coef : Coefficient.

P-Sensor : Proximity Sensor.

POT : Potentiomètre.

GND : Ground.

PS : Power Supply.

ARD : Arduino.

PS Motor : Première alimentation ATX.

PS Heater : Deuxième alimentation ATX.

ALIM : Alimentation.

ARD Mosfet : Pin de l'Arduino pour controler le Mosfet.

ARD Proximity Sensor : Pin de l'Arduino pour lire le capteur de proximité.

ARD NTC : Pin de l'Arduino pour lire la résistance thermique.

Résumé

Le présent travail a pour objectif la conception et la réalisation d'une imprimante 3D de type FDM avec interface graphique, logiciels générateurs de commandes, et système d'exploitation temps réel et firmware embarqués, personnalisés. Il s'est articulé autour de trois axes principaux : mécanique, électronique et informatique. Nous avons créé notre propre design mécanique et électronique, puis construit le système électromécanique. Nous avons ensuite développé et implémenté nos propres algorithmes et programmes non embarqués ainsi que algorithmes et Firmware embarqués, pour le fonctionnement de l'imprimante, et la correction de problèmes divers, tels que les vibrations, la stabilité, l'accélération des moteurs, le contrôle de température, pour une amélioration globale de la qualité d'impression. Notre imprimante 3D est complètement fonctionnelle, et la qualité de l'impression obtenue est satisfaisante et comparable à celle des imprimantes disponibles sur le marché.

Abstract

The 3D printing and more specifically FDM is the result of a unique technological mix, a challenge for researchers and engineers of all fields. In this project, we approached the fields of mechanics, computer science and electronics.

To build the mechanical structure of our 3D printer, we first had to study the constraints and problems of the CNC machine in general, understand what are the physical phenomena associated with the movements and acceleration of a mass. Subsequently we had to design the kinematics of the three axes of the machine, in addition to the axis of the extruder. After finishing the design of the machine, we used our old 3D printer, and a CNC machine (milling machine) to build the new printer.

Identifying constraints and problems that can be encountered and finding solutions to them before they arise is a huge time and money saver.

In theory, 3D printers and CNC machines are capable of moving at high speeds, but in reality the greater the acceleration, the more vibrations there are. These are harmful and drastically affect the quality of the print. The larger the moving mass and its acceleration, the more vibrations there are. There are two types of vibrations: free vibrations and forced vibrations.

We selected carefully then presented and discussed briefly the theory behind the vibrations. We also gave theoretical and practical solutions to these problems.

We found the majority of essential equipment for mechanical design (belt, pulley, etc ...). Unfortunately, we did not find aluminum plates to buy locally. As we did not find aluminum, we opted for wood. The wood will be used to make the frame and the frame. We also decided to print with our old printer a few parts of Prusa i3 MK3s, which we specially modified for our new 3D printer.

The realization will follow this approach:

- Design the frame, the frame and the support of the printing platform on Adobe Illustrator.
- Design the 3D model in Fusion 360.
- Cut these with a CNC machine.
- Modify then print the Prusa models.
- Prepare the rods, belts and printing plate.
- Put it all together.

In second step of the projet, we approached the electronic part in order to control the different motors, extruder, temperature, fan, etc. For this we had to study the electronic components available on the market and then select what suited us the most, in order to design electrical circuits for hardware control.

The different electrical circuits produced are:

- Temperature control circuit.
- Motor control circuit.
- LCD display control circuit.
- Sensor circuit.
- Thermal resistance circuit.

In order to build the software part of the machine, we made some rules and a certain philosophy. This philosophy prompted us to create several programs that run on a computer and act as "preprocessing" programs. Finally, we have developed a graphical computer interface that allows simple and easy communication between the user and the machine. The purpose of the latter is to send and receive various commands to the machine.

The 3D printer that we designed being built, we developed the software that we have created for its operation. They are classified into two types: PC software and embedded software. All programs, ideas and methods are personal creations.

There are nine software programs on PC:

1- Graphical user interface (GUI): it allows the user on the one hand to manage the selected 3D printing, to initialize and configure the printer, and to view the printer parameters in real time. On the other hand it also allows access to other software that we have created (G-Code Simulator, STL Viewer and G-Code Preprocessor).
2- G-Code Preprocessor: it allows you to generate the files necessary for the successive stages of printing.

3- G-Code Simulator: this is a set which contains the following programs: G-Code Simulator S, G-Code Simulator D, G-Code Simulator Preprocessor S and G-Code Simulator Preprocessor D.
4- G-Bézier code: It allows to use Bézier curves.
5- STL Viewer: It allows viewing STL files.
6- G-Code Futurs Mouvement: It generates files essential for printing.

Software embedded on the printer, the Firmware, responsible for transforming the command files generated on the PC, into digital commands, executed by the Atmega 2560 microcontroller, for the electromechanical system.

Finally and that is most of the work, we designed Firmware to control the whole machine using algorithms that we created. This Firmware is a mix between baremetal programming and a real-time operating system. The Firmware has a scheduler, which ensures the execution of the tasks, the latter are coded as efficiently as possible. For this, many characteristics of the operating systems have been abandoned in favor of the speed of execution. The tasks of the scheduler call for functions that require algorithms, we have designed these to maximize machine performance using our unique philosophy.

ملخص

الهدف من هذا العمل هو تصميم وبناء طابعة ثلاثية الأبعاد من نوع FDM بواجهة رسومية ، وبرمجيات لتوليد الأوامر، ونظام تشغيل في الوقت الفعلي وبرامج ثابتة مضمنة ، مخصصة. تمحور هذا العمل حول ثلاثة محاور رئيسية: الميكانيك والالكترونيات وتكنولوجيا المعلومات. ابتكرنا تصميمنا الميكانيكي والإلكتروني ، ثم بنينا النظام الكهروميكانيكي. ثم قمنا بتطوير وتنفيذ الخوارزميات الخاصة بنا والبرامج الغير مضمنة بالإضافة إلى الخوارزميات المضمنة والبرامج الثابتة ، لتشغيل الطابعة ، وتصحيح المشاكل المختلفة ، مثل الاهتزازات والاستقرار وتسريع المحركات ، التحكم في درجة الحرارة ، لتحسين شامل لجودة الطباعة. طابعتنا ثلاثية الأبعاد تعمل بكامل طاقتها ، وجودة الطباعة الناتجة مرضية ويمكن مقارنتها بجودة الطابعات المتوفرة في السوق .

Introduction générale

Depuis la nuit des temps, l'homme a utilisé son imagination et son intelligence pour construire et concevoir des choses à partir d'une matière première. Sculptant la pierre pour faire des outils, utilisant l'argile pour faire des pots ou taillant des arbres pour faire des maisons. Cependant le monde a récemment vu une révolution industrielle, celle du numérique.

Grâce à la révolution technologique en informatique industrielle, la commande des machines électriques a subi des progrès significatifs. Ce qui a permis le développement de solutions numériques efficaces avec une possibilité d'implanter des algorithmes de plus en plus complexes. Ces commandes sont en majorité basées sur les microprocesseurs, les DSP (Digital Signal Processor) et les microcontrôleurs. En effet, ce progrès a abouti à la conception de la machine à Commande Numérique par Calculateur (CNC). Cette machine procure une grande rapidité ce qui rend cette procédure très efficace et productive [1].

L'impression 3D transforme une information digitale en un objet physique à travers l'utilisation d'un logiciel de modélisation, elle est en train de révolutionner notre quotidien [2]. Longtemps réservée aux industries de pointe, elle s'est récemment démocratisée avec l'arrivée sur le marché d'imprimantes moins onéreuses et plus rapides, ainsi qu'un choix plus varié de matériaux imprimables [3]. Tous les processus d'impression 3D requièrent des outils logiciels, du matériel et de la matière.

Actuellement, les imprimantes 3D sont disponibles sur le marché à des prix allant de 250 à 100000 euros. Il est aussi possible d'acheter des kits à moins de 100 euros dont le montage n'est pas complexe.

Cependant, nous on s'est fixé comme but la conception et la réalisation d'une imprimante 3d dans tous ces aspects : électronique, mécanique et informatique.

Comparativement aux machines vendues en kits, la nôtre est unique dans la mesure où elle ne ressemble à aucune autre tant sur les plans du design, des circuits électroniques, des pièces mécaniques et du software.

Lors de la réalisation de cette imprimante, nous avons rencontré beaucoup de problèmes notamment ceux relatifs à la conception software, aux algorithmes et à la mécanique.

Ce mémoire s'articule autour des points suivants :

Le chapitre 1 porte sur les généralités de l'impression 3D, comme son évolution, son histoire, les technologies existantes, etc... Ensuite nous ferons le point sur les éléments nécessaires à la compréhension de ce mémoire tél que le langage G-code, les systèmes d'exploitation temps réel, format STL, etc…

Le chapitre 2 est axé sur deux parties : la mécanique et l'électronique. Nous présenterons à la fin les PCB conçus. Dans la partie mécanique nous aborderons les problèmes rencontrés et les solutions que nous avons proposées, ainsi que la structure et la cinématique de la machine. Pour ce qui est de l'électronique, nous traiterons toute la circuiterie de l'imprimante 3D et les problèmes rencontrés.

Danse le chapitre 3, nous parlerons:

- Des différents programmes que nous avons créés.
- Des principes de base d'un Firmware d'imprimante 3D, puis en déduire les points important qui doivent être pris en compte lors de la conception software.
- De l'architecture du Firmware et nous allons nous intéresser plus avec plus détail sur : les modules qui le composent, les ressources hardware et software qu'il utilise, les règles de conception, l'ordonnanceur, les tâches et de quelques algorithmes.
- Du fonctionnement du Firmware, de l'initialisation jusqu'au corps du programme passant par les problèmes qui peuvent survenir et les techniques de prévention qu'on a mis en place.

Dans le chapitre 4, nous parlerons des problèmes rencontrés. Des solutions et des algorithmes que nous avons créées. Enfin nous discuterons les résultats obtenus.

Chapitre 1. Les généralités sur les imprimantes et l'impression 3D

1.1. Introduction

Dans ce chapitre nous présentons un état de l'art de l'impression et l'imprimante 3D : son histoire, les techniques d'impression et les majeurs types de filament utilisés. Nous allons donner quelques points essentiels concernant le principe de fonctionnement des imprimantes 3D.

1.2. Histoire de l'impression 3D

Le premier brevet sur l'impression 3D (dite « fabrication additive ») fut déposé le 16 juillet 1984. Les dépositaires sont français : Jean-Claude André, Olivier de Witte, et Alain le Méhauté pour le compte de l'entreprise CILAS ALCATEL. La même année, aux Etats-Unis, le 1er août 1984, c'est l'américain Chuck Hull qui dépose le brevet sur la technique d'impression 3D de stéréolithographie (SLA pour StéréoLithographie Apparatus). Ce brevet donnera non seulement le nom de l'extension du fichier d'impression STL, mais donnera aussi naissance à une entreprise leader : 3D Systems, géant de la fabrication d'imprimantes 3D. 3D Systems lancera fin 1988 la première imprimante 3D, la SLA-250.

En 1988, une autre entreprise américaine, la société Stratasys fondée par Scott Crump, lance sur le marché une nouvelle technologie reposant elle aussi sur le dépôt couche par couche en fabrication additive : le procédé FDM pour Fused Deposition Modeling ou dépôt de fil fondu en français. Cette technique donnera naissance par la suite aux imprimantes domestiques personnelles telles que nous les connaissons aujourd'hui.

En 2006 apparaît pour la première fois un projet d'imprimante 3D open source qui ouvrira la voie aux futures imprimantes domestiques : le projet RepRap a été initié dès 2004 par le Dr Adrian Browyer alors professeur en génie mécanique de l'Université de Bath au Royaume-Uni. L'idée à la base de ce projet est de pouvoir construire par soi-même une imprimante 3d en technologie de dépôt de fil fondu. C'est le début de ce qu'on a pu appeler par la suite le mouvement Makers.

En 2014, l'évolution technologique s'attaque à une contrainte de taille. La société chinoise Win Su annonce la fabrication de maisons en impression 3D, à bas prix. Le secteur de la construction et de l'immobilier s'intéresse de près à ces nouvelles technologies permettant notamment la conception 3D et production de formes difficiles à produire dans des processus de construction traditionnels.

En 2015 la société Carbon3D annonce une nouvelle technologie révolutionnaire permettant de multiplier par 7 la rapidité en impression 3D. Baptisée CLIP, la technologie sur l'utilisation de résine, lumière et d'oxygène pour polymériser l'objet. Ce principe constitue une avancée importante dans le monde de la fabrication additive. Les premières imprimantes 3D sont attendues sur le marché en 2016. Début 2015 c'est aussi Hewlett Packard (HP) qui annonce se positionner sur le marché des imprimantes 3D professionnelles avec une technologie brevetée appelée Multi jet Fusion. [4]

On peut voir l'évolution de l'impression 3D (Figure 1.1) sur le court de l'histoire depuis que l'idée a surgit jusqu'au futur projet qui concerne le sujet.

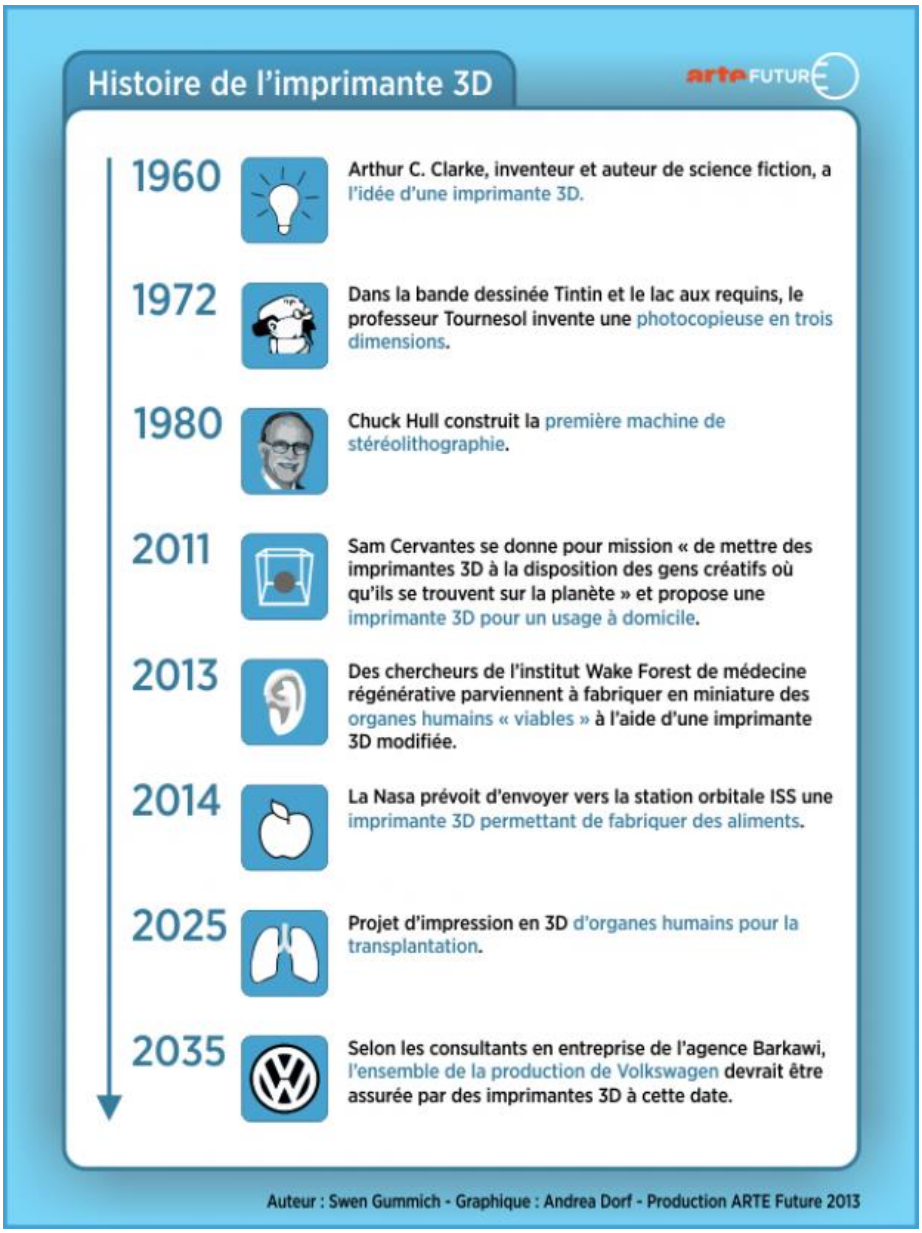

Figure 1.1: Histoire de l'imprimante 3D

1.3. Prototypage rapide

Le prototypage est une étape essentielle de la conception d'un produit. Cependant, les va-et-vient et les délais de production d'un prototype, font de la phase de prototypage un goulet d'étranglement qui freine le projet. Berman identifie trois principales phases de mise en œuvre de l'impression 3D. La première étape comprend le prototypage rapide et la fabrication du pont, où les modèles 3D sont utilisés pour améliorer la conception et le développement de produits de masse dans les entreprises [5].

Le prototypage rapide est l'art de réaliser dans un temps très court la représentation physique d'un modèle CAO en trois dimensions. Les techniques utilisées pour la fabrication des pièces sont des polymérisations de matière par couches successives dans des machines de prototypage rapide. L'usinage à grande vitesse est un autre moyen de prototypage rapide [6].

Grace au prototypage rapide à et l'impression 3D, les ingénieurs et designers peuvent créer rapidement des prototypes à partir d'un fichier numérique, et effectuer aisément des corrections multiples, basées sur des tests et des observations en conditions réelles [7].

1.4. Utilisation des imprimantes 3D

Aujourd'hui, l'impression 3D continue à se développer et à offrir de nombreux avantages, car elle permet l'utilisation de matériaux différents (et d'une grande variété) simultanément. Au cours des dernières années sont apparues de nombreuses nouvelles applications et expériences avec l'impression 3D, à la fois pour du prototypage et pour des produits finis. Il existe de nombreux exemples allant de la conception de la mode à la construction de maisons, ou à la production de biens de consommation personnalisés, de modèles d'impression 3D (Figure 1.2), jusqu'à la nourriture. Émerge également une recherche ciblée sur l'impression 3D pour les applications médicales comme la bio-impression (organes et cellules), pour les implants ou encore les os [5].

L'impression 3D est également de plus en plus utilisée en pré-production. Elle intervient dans la phase de conception de pièces outils pour améliorer l'efficacité du processus de production industriel. Dans ce domaine, elle permet de faire baisser les coûts de production grâce à l'accélération du processus de fabrication des moules, des maîtres modèles ou via la fabrication directe d'accessoires. D'après l'armée américaine, l'impression tridimensionnelle réduit de 97 % les coûts de production et de 83 % le temps de production [8]. Conjointement à cela, la multiplication des technologies duales en entreprises et les avancées technologiques ainsi que l'essor de l'impression 3D ont créé une synergie entre le domaine de l'armement et le domaine civil [9].

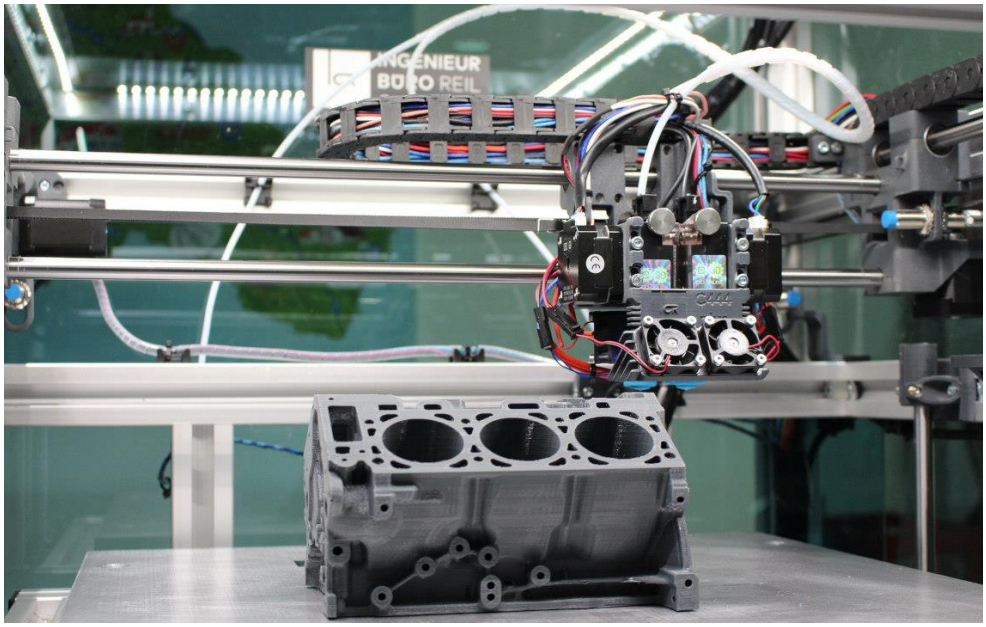

Figure 1.2: Exemple d'une imprimante 3D en action.

Après l'industrie, le secteur de la santé est le plus représenté. En effet, la fabrication additive est une technologie en plein essor dans le domaine médical, les fabricants de prothèses auditives et dentaires sont ceux qui utilisent aujourd'hui le plus ce procédé. Les audioprothésistes emploient déjà massivement (depuis plus de 10 ans déjà) cette technologie qui présente l'avantage de produire des pièces sur mesures parfaitement adaptés à la morphologie du patient.

Les laboratoires dentaires et d'orthodontie utilisent de plus en plus l'impression 3D pour sa précision et sa production accrue. Selon les imprimantes 3D, il est possible de fabriquer des moulages

de dents (Figure 1.3), des gouttières, des bridges ou des couronnes provisoires parfaitement ajustées à la denture du patient. Preuve de cette utilisation massive, selon l'expert Britannique Phil Reeves, en 2010 ce ne sont pas moins de 10 millions de prothèses auditives, 500 000 implants dentaires et 17 millions de gouttières qui ont été imprimées en 3D dans le monde.

Figure 1.3: Prothèse dentaire imprimée en 3D

Avec l'odontologie, la bijouterie est le domaine qui a recourt le plus à l'impression 3D (Figure 1.4), à tel point qu'il n'utilise quasiment plus que ce procédé. Bien qu'il soit possible d'imprimer directement des métaux précieux tels que l'or, cette technologie est le plus souvent utilisée pour fabriquer des moules à cire perdue. Ces derniers se destinent à donner la forme au moule final (moule réfractaire) dans lesquels sont coulés les métaux en fusion.

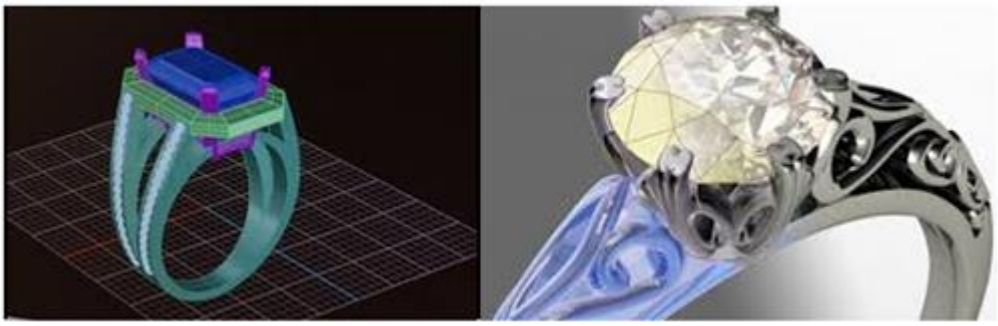

Figure 1.4: Bague dans un slicer

1.5. Types d'impressions 3D

L'impression 3D fonctionne selon plusieurs procédés, qui diffèrent selon le type d'imprimante 3D utilisée. On peut classer ces procédés dans trois grands groupes :

- Le dépôt de matière.
- La solidification par la lumière.
- L'agglomération par collage.

Ces trois procédés fonctionnent selon le même principe de base, c'est à dire superposer des couches de matières selon les coordonnées d'un fichier 3D. La différence se situe dans la manière de dépôts et traitement des couches, ainsi que le type de matériau utilisé.

1.5.1. L'impression 3D par dépôt de matière FDM

Le terme « Filament Deposition Manufacturing » ou FDM ayant été breveté par la société Stratasys, les sociétés concurrentes ont dû un autre nom. La technique d'impression 3D par dépôt de matière fondue est ainsi ont dû lui affecter un autre nom : Fused Filament Fabrication(FFF). Comme ce nom l'indique, l'imprimante déroule un fil de polymère, comme de l'ABS ou du PLA, et le chauffe à haute température via une buse (extrudeuse ou tête d'extrusion) pour ensuite le déposer en couches successives en suivant le fichier 3D CAO. [10] (Figure 1.5)

Il s'agit de la technologie de fabrication additive la plus utilisée en raison de son aptitude à réaliser des échantillons ayant une forme géométrique difficile dans des conditions de périodes de temps raisonnables et sans aucune exigence d'outillage [11].

La technique FDM utilise un filament en plastique ou un fil métallique fourni dans une buse d'extrusion. La buse d'extrusion chauffe le matériau fourni et se déplace dans les directions horizontale et verticale avec un mécanisme à commande numérique tout en contrôlant l'écoulement du matériau sortant de la buse [12].

Cette technique consiste à déposer couche par couche un filament de matière thermoplastique fondu à 200°C (en moyenne) lequel en se superposant donne forme à l'objet. La tête d'impression se déplace selon les coordonnées X, Y et Z (longueur, largeur et hauteur) transmises par un fichier 3D correspondant au modèle 3D de l'objet à imprimer. Limitée pendant longtemps à des matériaux de type plastique tels que les classiques PLA et l'ABS, l'impression 3D voit arriver de nouveaux filaments composites à base de métal (cuivre, bronze…), de fibres de carbone et même de bois. Plus rarement certaines machines utilisent des cires ou des polycarbonates. Aujourd'hui l'industrie agroalimentaire et la médecine s'emparent peu à peu de cette technique pour imprimer des aliments et des cellules en adaptant la tête d'extrusion. [10]

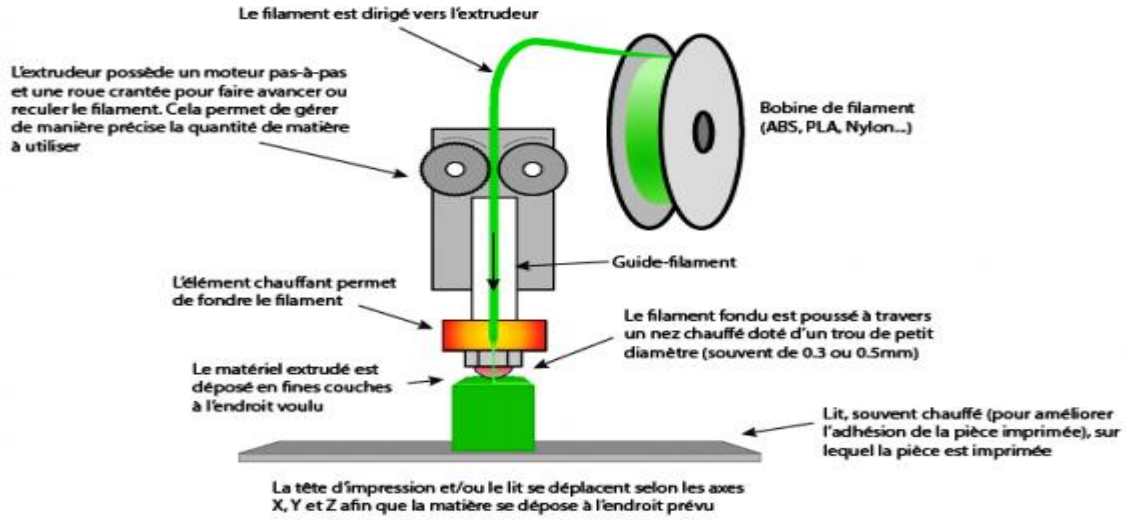

Figure 1.5: Principe de fonctionnement d'une imprimante 3D FFF

1.5.2. La solidification par lumière SLA

Appelée aussi SLA (Stéréolithographie Apparatus) cette technique consiste à solidifier un liquide photosensible par le biais d'un rayon laser ultraviolet. Les imprimantes fonctionnant par SLA (Figure 1.6) ont quatre parties principales: un réservoir qui peut être rempli avec un liquide photopolymère, une plate-forme perforée qui est descendue dans le réservoir, un rayonnement ultraviolet (UV) et d'un ordinateur commandant la plate-forme et le laser [13].

Pour créer une pièce en stéréolithographie, il est nécessaire d'avoir un fichier au format STL ou SLA. Ce format est une triangulation du modèle CAO, la qualité du résultat dépend en grande partie de ce fichier et de sa précision. Le rayon laser piloté décrit le parcours pour former la pièce et polymérise la matière qu'il insole et ceci, couche après couche [6].

Tout comme la FDM, l'imprimante va dans un premier lieu analyser le fichier CAO, puis, en fonction de la forme de l'objet, elle va lui ajouter des fixations temporaires pour maintenir certaines parties qui pourraient s'affaisser. Puis le laser va commencer par toucher et durcir instantanément la première couche de l'objet à imprimer. Une fois que la couche initiale de l'objet a durci, la plate-forme est abaissée, est ensuite exposée une nouvelle couche de surface de polymère liquide. Le laser trace à nouveau une section transversale de l'objet qui colle instantanément à la pièce durcie du dessous. Ce processus se répète encore et encore jusqu'à ce que la totalité de l'objet soit formé et soit entièrement immergé dans le réservoir. La plateforme va ensuite se relever pour faire apparaitre l'objet fini en trois dimensions. Après qu'il a été rincé avec un solvant liquide pour le débarrasser de l'excès de résine, l'objet est cuit dans un four à ultraviolet pour durcir la matière plastique supplémentaire.

Parmi les inconvénients de cette méthode, on note un coût plus élevé que la FDM et un panel de matériaux et des coloris plus limité du fait des polymères utilisés comme matière première. Les solvants et les liquides polymères dégageant par ailleurs des vapeurs toxiques durant l'impression.

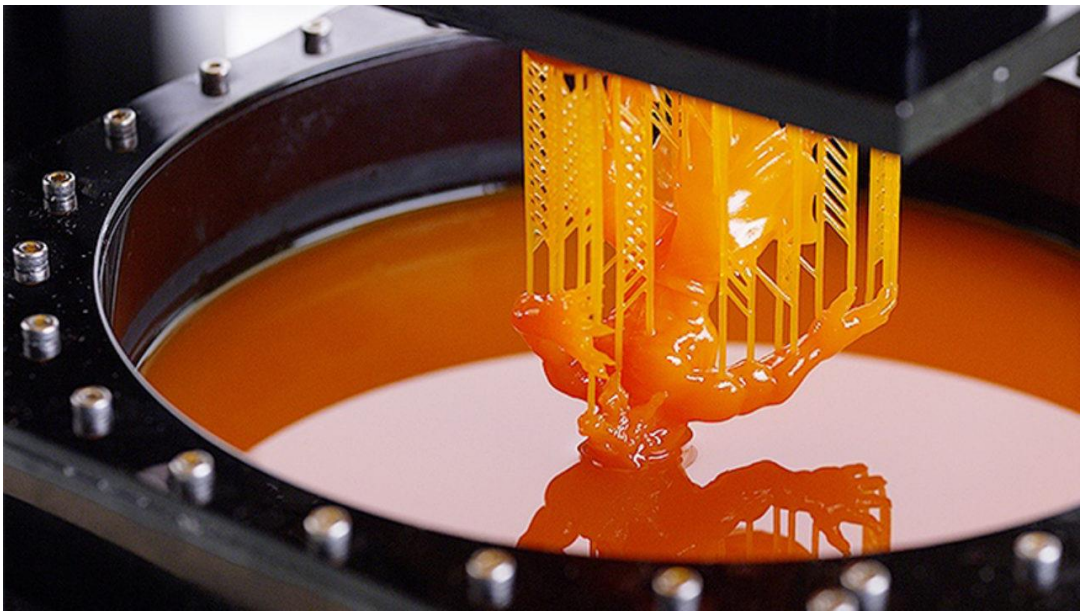

Figure 1.6: Impression 3D SLA

1.5.3. L'agglomération de poudre par collage

Initialement développée au Massachusetts Institute of Technology (MIT) en 1993, la 3DP ou Three-Dimensional Printing (Figure 1.7) constitue la base du processus d'impression 3D de Z Corporation. Le procédé consiste en l'étalement d'une fine couche de poudre de composite sur une plateforme. La tête d'impression va alors déposer sur celle-ci de fines gouttes de glue colorées, lesquelles combinées entre elles, permettent d'obtenir un large panel de couleur. La plateforme s'abaisse au fur et à mesure que les couches de poudre sont collées jusqu'à obtenir l'objet final [13].

Pour la finition il faut aspirer l'excédent de poudre, brosser et/ou poncer la pièce, puis la chauffer pour finaliser la solidification. La 3DP a l'avantage d'être rapide et de proposer une large gamme de couleurs. Jusqu'à 6 fois moins chère qu'une imprimante 3D SLA son prix est plus attractif malgré une précision et une qualité d'impression parfois inférieure. Parmi les inconvénients, sans traitement post-impression, les pièces sont plus fragiles et leur surface est plus rugueuse.

Figure 1.7: Objet 3D imprimé par collage de poudre

1.6. Matériaux pour l'impression

On distingue 2 grandes familles de matériaux que sont les plastiques (Figure 1.8) et les métaux (Figure 1.9), auxquels il faut ajouter les céramiques (silice, alumine, plâtre…) et les matières organiques (cires, tissus et cellules). Nous allons voir ici les caractéristiques et les utilisations des principaux matériaux utilisés pour imprimer en 3D.

1.6.1. Les matériaux plastiques

Figure 1.8: Bobines et objets en plastiques

1.6.1.1. PLA

Le PLA est l'un des polymères les plus utilisés la FDM en raison de sa facilité d'imprimabilité. Plusieurs applications du PLA sont dans les domaines pharmaceutique, alimentaire, industries chimiques et textiles. C'est un acide organique qui peut être obtenu par différentes méthodes de synthèse chimique. Les propriétés du matériau PLA sont bonnes par rapport à de nombreux autres matériaux bio-sources [11].

Il a la particularité d'être très peu soumis au phénomène de warping et ne se déforme pas. Ce plastique permet d'avoir des impressions sans perte de dimensions lors du refroidissement. Cependant il est à éviter pour des objets qui doivent résister à de fortes températures. En effet, il a tendance à se déformer au-delà de 60°C. [14]

1.6.1.2. ABS

L'ABS est un polymère thermoplastique populaire qui au cours des dernières années a gagné en popularité pour son utilisation dans la technique FFF. Il présente une excellente rugosité, une bonne stabilité dimensionnelle, une bonne aptitude au traitement, ainsi qu'une résistance et une stabilité chimiques. L'ABS est très approprié pour le moulage par injection et est donc utilisé dans une grande variété de produits manufacturés qui comprennent des composants automobiles, des équipements sportifs, des boîtiers électriques ainsi que des jouets tels que des briques Lego. l'ABS est un matériau couramment utilisé pour l'impression 3D [15].

1.6.1.3. Les cires

La cire est composée de polymère et fond comme les autres plastiques. Ce matériau est principalement utilisé en bijouterie et en dentisterie pour la création de moules où sont ensuite coulés d'autres matériaux comme du métal ou de la céramique [14].

1.6.2. Les poudres métalliques

Figure 1.9: Objets métalliques imprimés en 3D

Le plus souvent les objets en métal sont imprimés à partir de poudres métalliques fusionnées avec un laser (procédé DMLS). En combinant l'impression 3D à l'optimisation topologique, on obtient des pièces beaucoup plus légères que des pièces traditionnelles et même plus résistante, grâce à des géométries optimisées et l'absence de fixations. Cette capacité intéresse particulièrement les constructeurs dans les domaines aéronautique et spatial, et de l'automobile, dans un souci de performance et d'économie de carburant [14].

1.6.2.1. L'acier inoxydable

Plus connu sous l'appellation « d'inox », l'acier inoxydable a été le premier métal à être commercialisé pour la fabrication additive. Comme son nom le suggère il possède des propriétés mécaniques de haute résistance à la corrosion [14].

1.6.2.2. L'aluminium

Dans cette catégorie on retrouve l'AlSi10Mg d'EOS, un alliage qui par sa composition (magnésium et de silicium) est à la fois très léger et très solide. Il est le plus souvent utilisé pour la fabrication de moules à parois fines et aux géométries complexe. Il existe aussi l'alumide, une poudre à base de polyamide et d'aluminium. Il offre l'avantage de pouvoir réaliser des pièces à la fois très solides et très flexibles avec une importante résistance à la chaleur [14].

1.6.2.3. Le titane

La fabrication additive est de plus en plus employée lorsqu'il s'agit de travailler ce matériau, les plus gros utilisateurs étant les secteurs de l'automobile, l'aéronautique et de la médecine. Les alliages de titane tels que le Ti6Al4V sont plus solides que le matériau original pur. Cet alliage biocompatible trouve notamment des applications dans la médecine. Il est utilisé par exemple pour fabriquer des implants en titane sur mesure, sa porosité naturelle permettant aux cellules osseuses de le coloniser efficacement [14].

1.7. Processus d'impression 3D

L'impression 3D nécessite une représentation particulière de l'objet à imprimer. Des logiciels permettant de convertir la représentation 3D d'un objet, en fichier STL par un slicer, puis en fichier G-code. La (Figure 1.10) illustre ce processus.

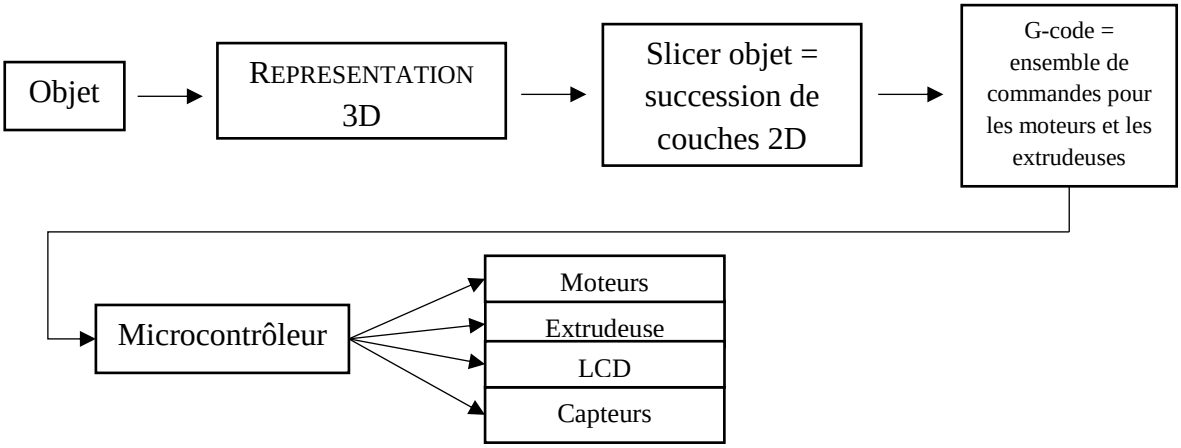

Figure 1.10: Schéma synoptique du processus d'impression 3D

1.8. Fichier STL

Pour créer une pièce en trois dimensions, il est nécessaire d'avoir un fichier au format STL (Figure 1.11) ou SLA. Les données des solides 3D (ou maillages fermés) sont converties en une représentation maillée à facettes constituée d'un ensemble de triangles et enregistrée au format STL [16].

Cette représentation numérique est constituée d'une mosaïque de triangles dont chacun a une position connue de ses 3 sommets [17]. La qualité du résultat dépend en grande partie de ce fichier et de sa précision [6].

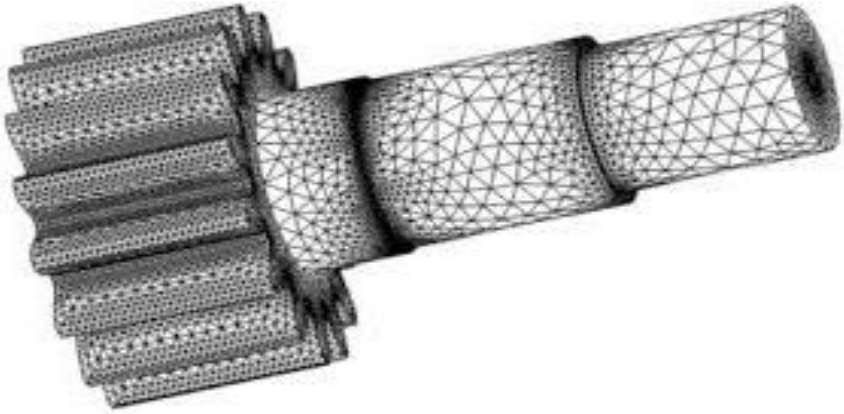

Figure 1.11: Affichage d'un fichier STL d'un objet.

1.9. Slicing

Le découpage (Figure 1.12) est le processus de transformation du modèle 3D en parcours d'outil pour l'imprimante 3D. La plupart des gens l'appellent découpage car la première chose que fait le slicing engine est de couper le modèle 3D en fines couches horizontales [18]. Le logiciel de découpage ou slicer converti un fichier STL en fichier G-code.

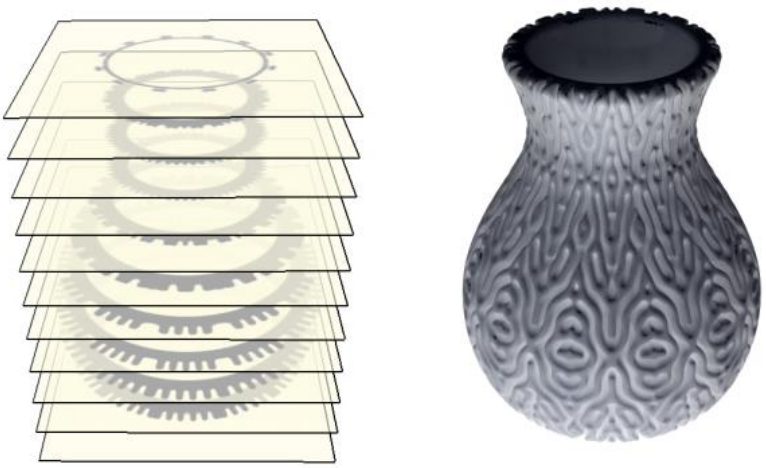

Figure 1.12: Exemple d'une opération slice d'un vase

1.10. G-code

Le G-code (Figure 1.13) c'est le nom commun du langage de programmation à commande numérique par ordinateur (CNC) le plus utilisé. Il est principalement utilisé dans la fabrication assistée par ordinateur pour contrôler les machines-outils automatisées. Le fichier de G-code contient les instructions basées sur les paramètres que vous choisissez à partir de slicer et calcule la quantité de matériel dont l'imprimante aura besoin et combien de temps il faudra pour imprimer [18]. Les instructions de G-code fournies au contrôleur de la machine indiquent aux moteurs où se déplacer, à quelle vitesse se déplacer et quel chemin suivre.

```
M107
M190 S45 ; set bed temperature
M104 S175 ; set temperature
G28 ; home all axes
G1 Z5 F5000 ; lift nozzle
M109 S175 ; wait for temperature to be reached
G21 ; set units to millimeters
G90 ; use absolute coordinates
M82 ; use absolute distances for extrusion
G92 E0
G1 Z0.400 F7800.000
G1 E-2.00000 F2400.00000
G92 E0
G1 X50.367 Y88.236 F7800.000
G1 E2.00000 F2400.00000
```

Figure 1.13: Exemple d'une suite d'instruction G-code

1.11. Système d'exploitation temps réel

Un système d'exploitation ou Operating System (OS) est un programme qui agit comme un intermédiaire entre l'utilisateur et la machine. Son but est de fournir un environnement dans lequel un utilisateur peut exécuter des programmes et son rôle est de coordonner l'exécution simultanée de plusieurs taches utilisateurs. Il doit répartir les ressources du système telles que le CPU, la mémoire, les périphériques. Il a aussi pour rôle de fournir des services tels que la communication entre processus, la synchronisation, l'affichage… [19]

Un système d'exploitation temps réel, en anglais RTOS pour *real-time operating system*, est un système d'exploitation pour lequel le temps maximum entre un stimulus d'entrée et une réponse de sortie est précisément déterminé. Un RTOS facilite la création d'un système temps réel, mais ne garantit pas que le résultat final respecte les contraintes temps réel, ce qui exige le développement correct du logiciel. Un RTOS utilise des ordonnanceurs spécialisés afin de fournir aux développeurs des systèmes temps réel les outils et les primitives nécessaires pour produire un comportement temps réel souhaité dans le système final.

Les calculateurs embarqués étant généralement dotés de peu de mémoire et d'une puissance de calcul, les systèmes d'exploitation temps privilégient l'encombrement (empreinte mémoire) et la simplicité. Le temps réel n'est pas forcément rapide, il est déterministe. En effet, à quelques exceptions prêtes, les méthodes de validation temporelle sont conservatives : lorsqu'on valide le système, on valide le système pour le pire cas. Enfin, une métrique important caractérisant un RTOS est la latence noyau : cette durée caractérise le pire délai pouvant s'écouler entre un événement de déclenchement d'une tâche, et la prise en compte effective du déclenchement par le noyau. L'architecture interne d'un RTOS est faite de sorte à minimiser ce délai, au détriment de la vitesse moyenne des traitements [19].

1.12. Interface graphique

L'interface graphique, appelée communément la GUI (pour Graphical User Interface), est le lien entre l'utilisateur et la machine, en l'occurrence une imprimante 3D. La conception de l'interface graphique d'une application consiste essentiellement à créer des instances des classes représentant les différents éléments nécessaires, modifier les caractéristiques de ces instances de classe, les assembler et prévoir le code de gestion des différents événements pouvant intervenir au cours du fonctionnement de l'application [20].

1.13. Conclusion

Dans ce chapitre nous avons fait une introduction sur les éléments généraux de l'impression 3D imprimantes points qui vont nous aider à établir un cahier des charges précis de l'imprimante 3D que nous allons concevoir et réaliser au cours de notre projet. Nous décrivons les étapes de conception mécanique et électronique dans le chapitre suivant.

Chapitre 2. Conception électromécanique de l'imprimante 3D

2.1. Introduction

Dans ce chapitre nous allons présenter toute la partie électronique : expliquer et discuter les schémas électriques réalisés. Nous aborderons aussi la partie mécanique avec les problèmes rencontrés et les solutions que nous avons proposées ainsi que la structure et la cinématique de la machine.

2.2. Cahier des charges

La conception électromécanique (Figure 2.1) de l'imprimante nécessite un cahier des charges bien précis à suivre :

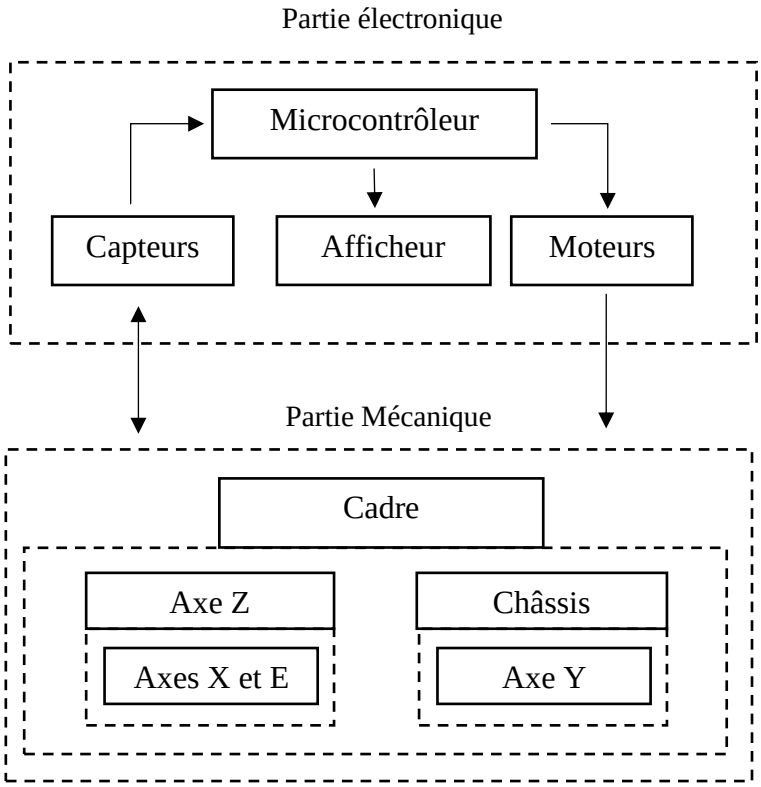

Figure 2.1: Schéma synoptique de la conception électromécanique

Pour la partie mécanique, notre but est de construire une imprimante 3D possédant les caractéristiques suivantes :

- Imprimante de type FDM.
- Trois dégrées de liberté.
- Doit pouvoir extruder du filament.
- Extrudeuse de type Bowden.
- La taille totale de l'imprimante : 44 x 74 x 44 cm.
- Cadre de la machine de taille : 44 x 44 cm pour supporter l'axe Z.
- Châssis pour supporter toute l'imprimante.
- Support du plateau d'impression de taille : 22.6 x 22.6 cm.
- Plateau d'impression en verre.

- La taille de travail des axes X, Y et Z est respectivement de 20, 20 et 40 cm.
- Axes X et Y ont chaque un : une courroie, une poulie, deux tiges glissantes, un moteur pas à pas avec un pignon monté sur l'arbre et trois roulements linaires.
- Axe Z a deux tiges glissantes, deux tiges filetées, deux accouplements flexibles, deux écrous, quatre roulements linéaires circulaires et deux moteurs pas à pas.
- L'axe E (de l'extrudeuse) possède MK8 monté sur un moteur pas à pas.
- Support pour tous les moteurs pas à pas.
- Support pour la poulie.
- Transporteurs de l'axe X et Y.
- Supports des tiges glissantes et filetées.
- Support pour l'extrudeuse.

En ce qui concerne la partie électronique :

- Circuits pour contrôler la température de l'extrudeuse.
- Circuit pour contrôler les moteurs.
- Circuit pour afficher sur un LCD.
- Circuits pour gérer les capteurs de fins de course et le capteur de proximité.

La (Figure 2.2) illustre le schéma synoptique de la partie électronique.

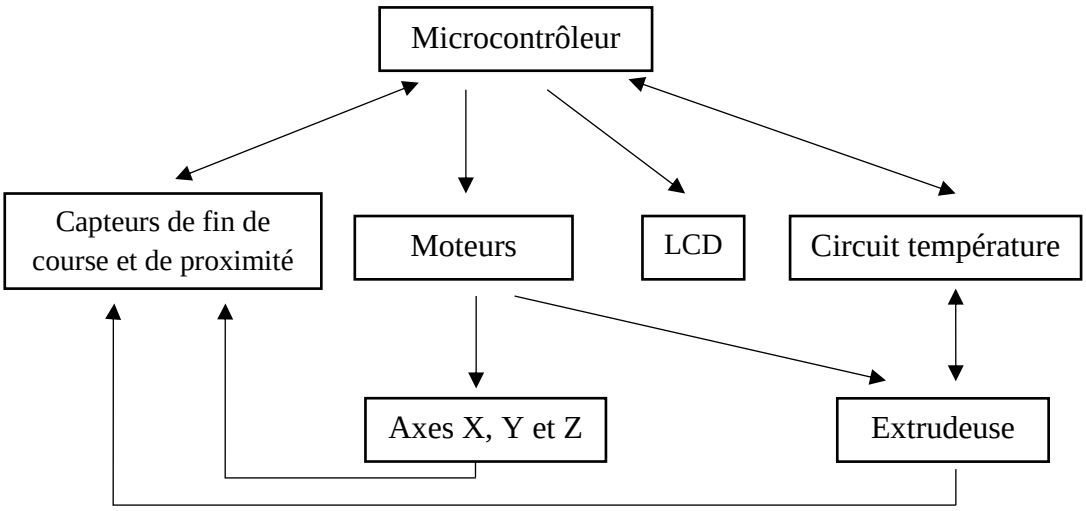

Figure 2.2: Schéma synoptique de la partie électronique

2.3. Flot de conception

2.3.1. Flot de conception mécanique

La partie mécanique est la première partie qu'on doit réaliser. Elle regroupe toute la théorie et la pratique à propos de la conception de la structure mécanique et de la cinématique de la machine. La Figure 2.3 représente l'organigramme du flot de conception mécanique.

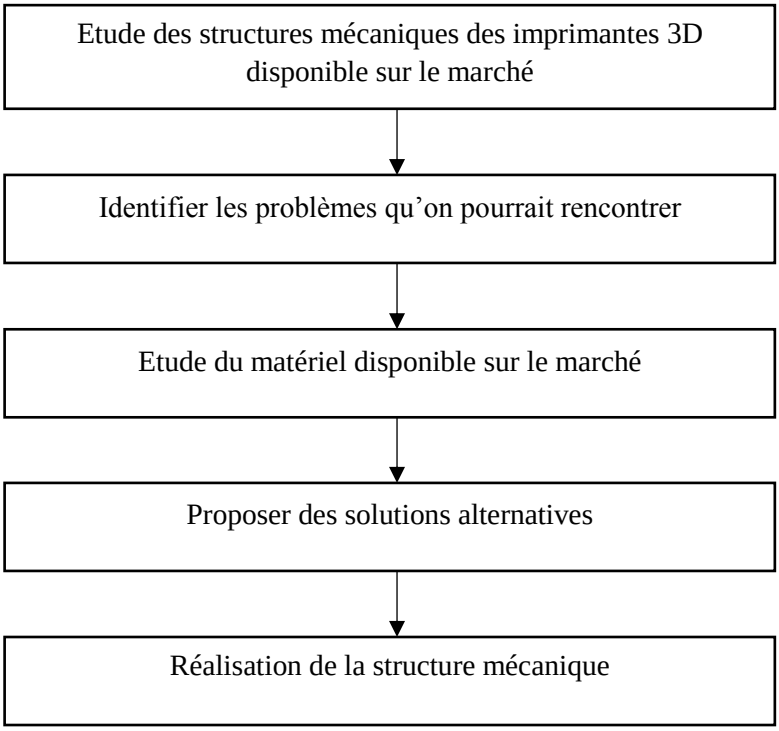

Figure 2.3: Organigramme du flot de conception mécanique

2.3.1.1. Etude des structures mécaniques des imprimantes 3D existantes

Pour construire la structure mécanique de notre imprimante 3D, il a fallu étudier tout d'abord les contraintes et les problèmes de la machine à commande numérique en générale, comprendre quels sont les phénomènes physiques associés aux mouvements et à l'accélération d'une masse.

Cette étude permet de trouver ou d'en déduire des astuces à partir des modèles qui connaissent le plus de succès.

Il existe des milliers de modèles en ligne, tous différents les uns des autres. Le nombre immense de machine qui existe rend l'étude difficile, par conséquent nous avons mis au point des critères pour trier les résultats. Les imprimantes doivent :

- Etre de type FDM.
- Ne pas avoir une structure mécanique sous forme de cube.
- Utiliser des matériaux disponibles localement.
- Avoir une bonne notation de la part des clients.
- Avoir une structure mécanique réalisable.

Après avoir fait le tri, il ne reste qu'une seule famille d'imprimantes : les Prusa. L'entreprise Prusa3D utilise des imprimantes pour fabriquer d'autre imprimante c'est ce qu'on appelle ferme d'impression 3D. La structure mécanique de la Prusa i3 MK3S (un bestseller de l'entreprise) est présenté dans la (Figure 2.4).

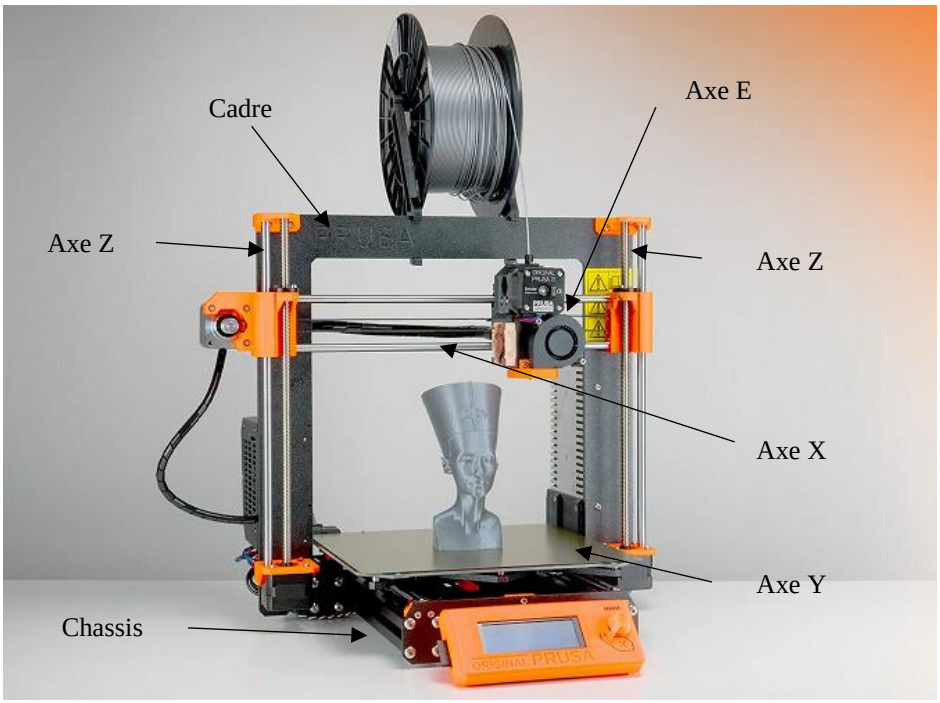

Figure 2.4: Prusa i3 MK3S

On va s'inspirer de la Prusa i3 MK3S pour la conception de notre imprimante 3D.

2.3.1.2. Identifier les problèmes qu'on pourrait rencontrer

L'identification des contraintes et les problèmes qu'on peut rencontrer et ainsi trouver des solutions à ces derniers avant qu'ils ne surviennent est un énorme gain de temps et d'argent. Les problèmes qu'on va rencontrer se résume aux points suivants :

- Impossibilité de copier la Prusa i3 MK3S car le matériel utilisé dans cette dernière n'est pas disponible en Algérie, par conséquent on doit refaire la structure mécanique.
- Problèmes de vibrations.

On remarque que les ingénieurs qui conçoivent les imprimantes 3D essayent de réduire au maximum le poids des transporteur X, Y et Z. Ils privilégient l'utilisation des courroies pour leur poids et leurs pas par millimètres. Cela permet aussi de réduire les vibrations.

2.3.1.3. Etude du matériel disponible sur le marché

On a trouvé la majorité du matériel essentielle pour la conception mécanique (courroie, poulie, etc…). Malheureusement on n'a pas trouvé des plaques d'aluminium pour en acheter localement.

2.3.1.4. Proposer des solutions alternatives

Comme on n'a pas trouvé d'aluminium on a opté pour le bois. Le bois servira pour la réalisation du cadre et du châssis. On a aussi décidé d'imprimer grâce à notre ancienne imprimante (que nous avions réalisée en mini projet) quelques pièces de Prusa i3 MK3s qu'on a modifié spécialement pour notre nouvelle imprimante 3D.

2.3.1.5. Réalisation de la structure mécanique

La réalisation va suivre cette démarche :

- Designer le châssis, le cadre et le support du plateau d'impression sur Illustrator.
- Découper avec une machine CNC ces derniers.
- Modifier puis imprimer les modèles de Prusa.
- Préparer les tiges, courroies et plateau d'impression.
- Assembler le tout.

2.3.2. Flot de conception électronique

La seconde partie du travail est la conception électronique. La démarche consiste à choisir et étudier les composants électroniques nécessaires à l'impression 3D ainsi que concevoir et tester les circuits électriques de contrôle. L'organigramme présent dans la Figure 2.5 illustre le flot de conception électronique.

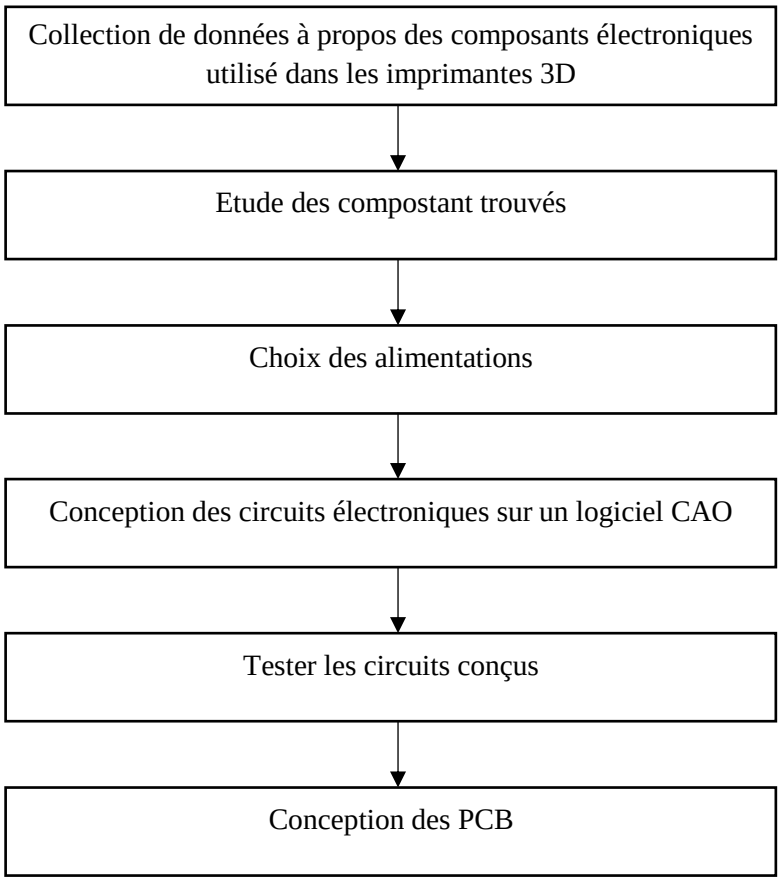

Figure 2.5: Organigramme de flot de conception électronique

2.3.2.1. Etude et collection de données à propos de l'électronique des imprimantes 3D

La première étape de la conception électronique de l'imprimante 3D est d'identifier, lister, télécharger les fiches techniques puis de s'assurer de la disponibilité des composants utilisés dans les imprimantes 3D. Dans notre imprimante 3D on a besoin de :

- Cinq moteurs pas à pas qui ont une bonne vitesse et une bonne force (torque).
- Cinq drivers pour contrôler ces moteurs pas à pas.

- Quatre capteurs de fin de courses.
- Un capteur de proximité.
- Un Mosfet qui peut faire passer un courant suffisant pour l'alimentation de l'extrudeuse.
- Une résistance thermique qui doit supporter une chaleur supérieure à 200 dégrée.
- Un microcontrôleur.
- Un ventilateur qui fonctionne avec 12 volt.
- Une diode (de roue libre) qui doit supporter les pics de courant du ventilateur.
- Un LCD.

On doit s'assurer ensuite que tous ces composants sont compatibles entre eux, par exemple un moteur pas à pas qui fonctionne avec 1.5A ne peut être contrôlé par un driver qui donne un courant max égale à 0.6 A.

2.3.2.2. Choix des types d'alimentation

Le choix du type d'alimentation dépend du courant total des circuits qu'on va concevoir.

2.3.2.3. Conception et test des circuits électriques sur logiciel CAO

En ce qui concerne la conception des circuits électriques, on va les concevoir et les simuler sur Proteus, puis les tester sur une plaquette d'essais.

2.3.2.4. Design des PCB

La dernière étape de la conception électronique est de concevoir les circuits imprimés PCB, afin de construire nos cartes électroniques.

2.4. Conception mécanique

2.4.1. Introduction

La partie mécanique est très importante car rien ne peut remplacer un bon design qui respecte les lois de la physique et de la mécanique, peu importe le software ou la qualité des drivers utilisés. On ne peut tout simplement pas se permettre de négliger cette partie malgré que ce ne soit pas notre domaine de prédilection.

Dans cette partie nous présenterons la structure mécanique, le matériel utilisé, les problèmes rencontrés, le design de l'imprimante 3D et les calculs nécessaires pour la compréhension de la partie software.

Pour ce qui est des contraintes, nous allons insister sur les vibrations qui constituent l'un des problèmes majeurs des machines à commande numérique notamment les imprimantes 3D à vitesses rapides.

2.4.2. Structure mécanique de l'imprimante 3D

2.4.2.1. Modèle 3D de l'imprimante

Le modèle 3D de notre l'imprimante a été conçus avec le logiciel Fusion 360 (annexe). La Figure 2.6, Figure 2.7, Figure 2.8, Figure 2.9 et Figure 2.10 montrent le modèle 3D de notre imprimante 3D.

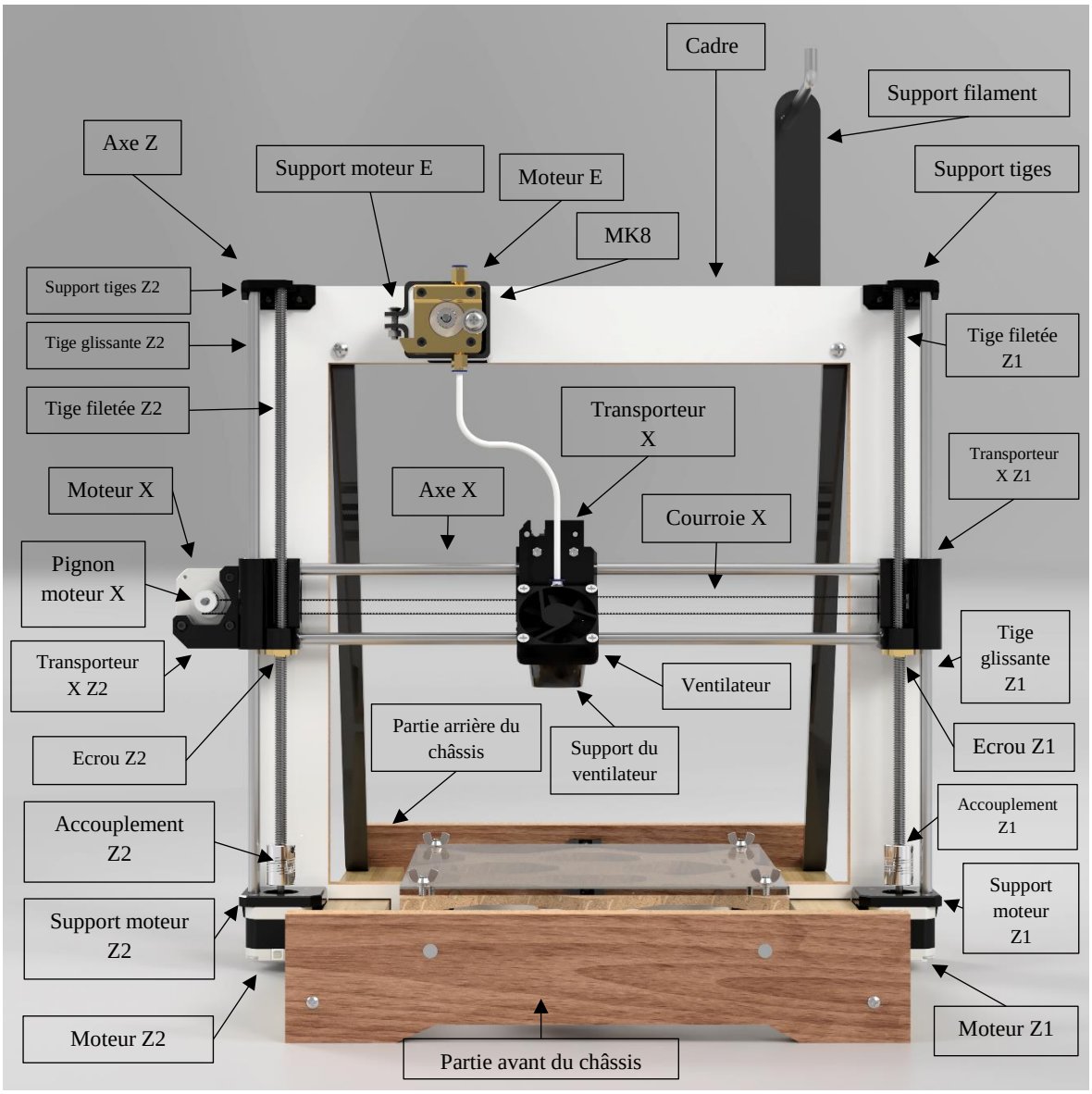

Figure 2.6: Vue frontale du modèle 3D de notre imprimante 3D dans Fusion 360

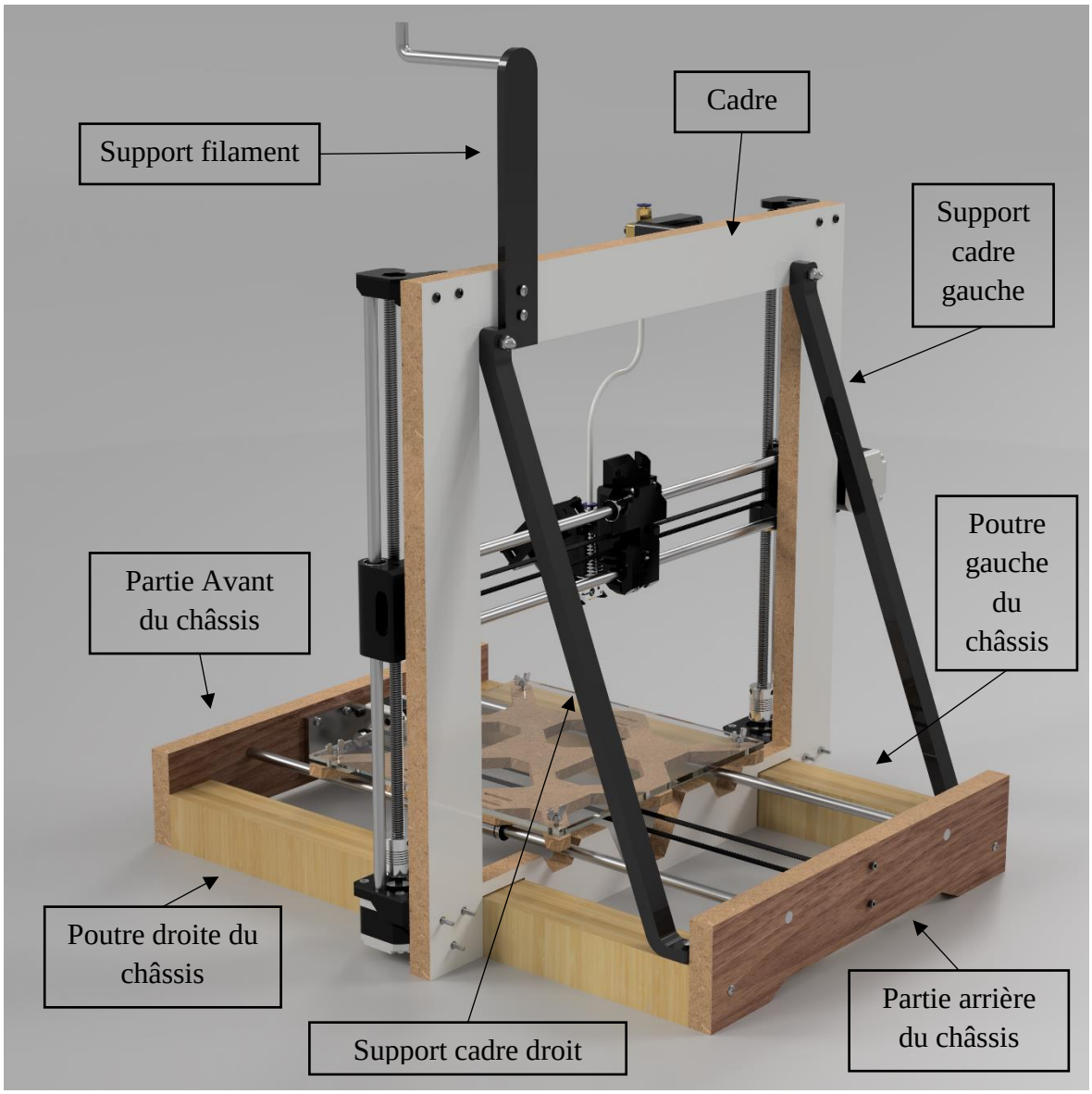

Figure 2.7: Modèle 3D de notre imprimante 3D dans Fusion 360 vu sous un autre angle

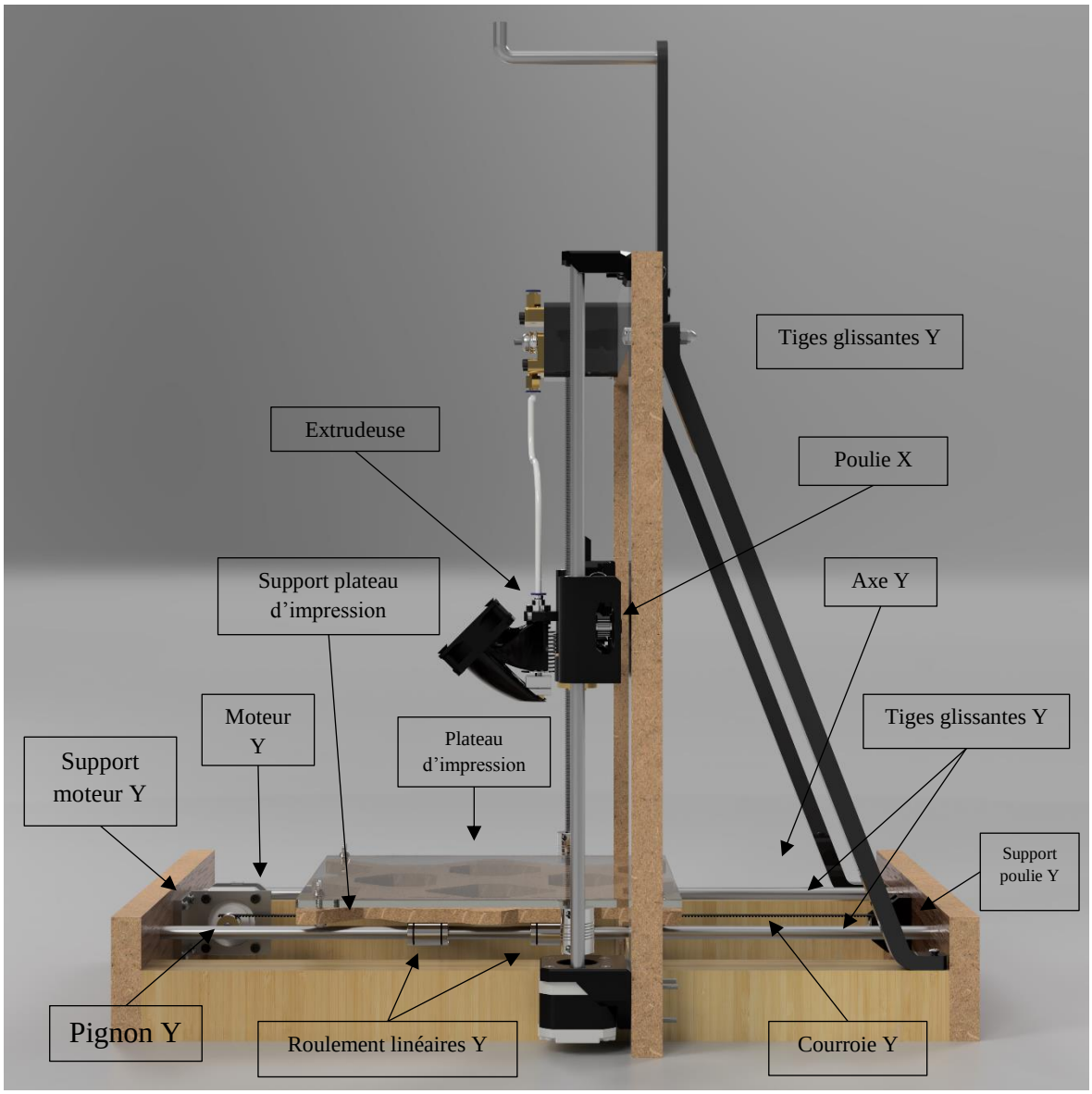

Figure 2.8: Vue latérale du modèle 3D de notre imprimante 3D dans Fusion 360

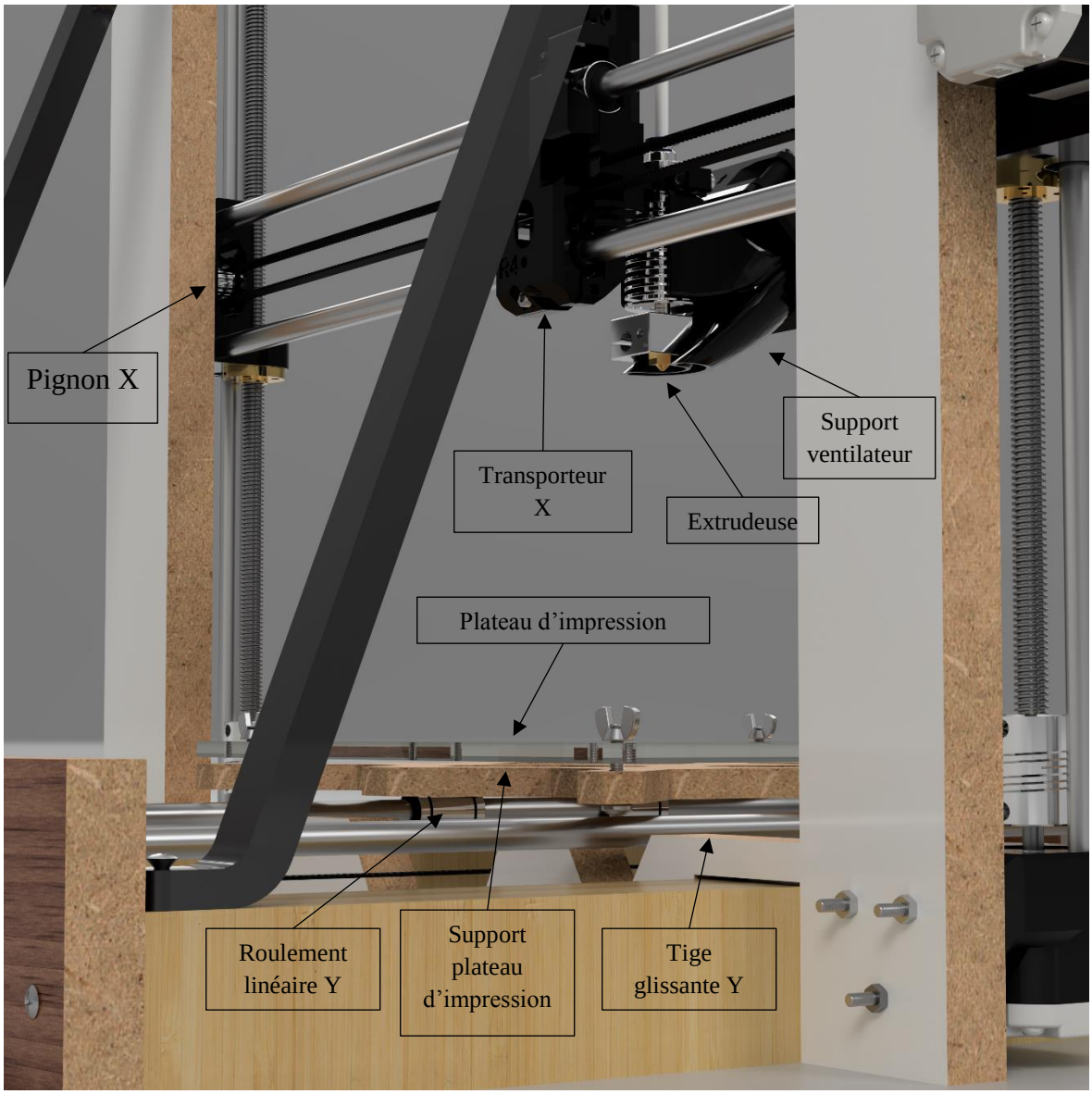

Figure 2.9: Vue zoomée du modèle 3D de notre imprimante 3D dans Fusion 360

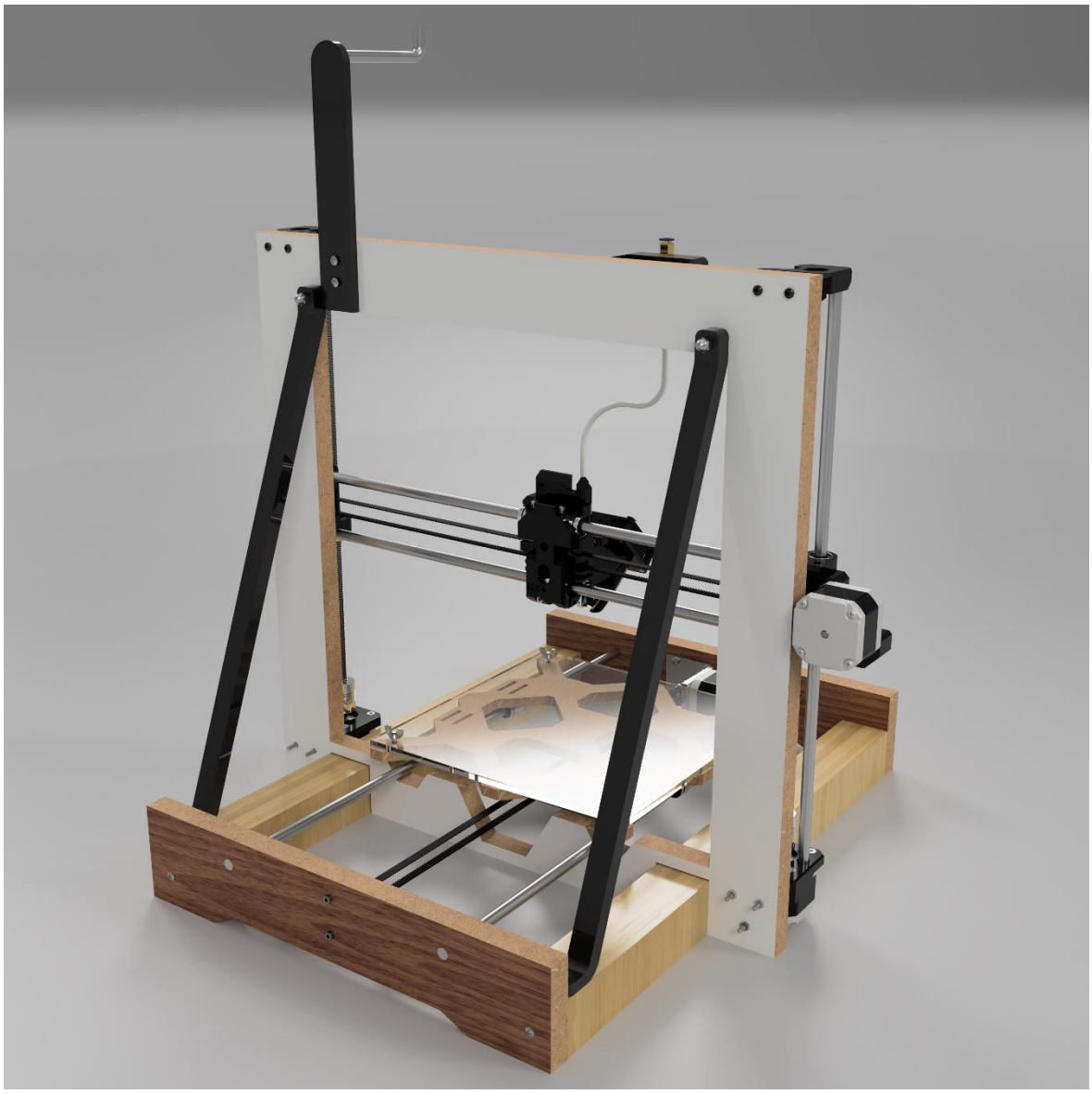

Figure 2.10: Vue arrière du modèle 3D de notre imprimante 3D dans Fusion 360

2.4.2.2. Description fonctionnelle

- Cadre : supporte l'axe Z.
- Châssis : supporte le cadre et l'axe Y.
- Supports du cadre : permet d'attacher et de supporter le cadre avec le châssis.
- Support du filament : supporte la bobine du filament.
- Poutres du châssis : supportent le cadre.
- Les axes X, Y et Z : donnent trois dégrées de liberté.
- L'axe E : permet l'extrusion du filament.
- L'axe X : supporte le transporteur X.
- L'axe Y : supporte le support du plateau d'impression.
- L'axe Z : supporte l'axe X et les transporteurs Z.
- Supports des moteurs : fixe les moteurs.
- La combinaison courroies, pignons, poulies : transforment les mouvements rotationnels en mouvements linéaires.

- Support poulie : supporte la poulie.
- Tige glissante axe X : supporte le mouvement du transporteur X.
- Courroie X : responsable du mouvement du transporteur X.
- Tige glissante Y : supporte le mouvement du support du plateau d'impression.
- Courroie Y : responsable du mouvement du support du plateau d'impression.
- Supports tiges glissantes et filetées : supportent les tiges de l'axe Z.
- Transporteur X : responsable du mouvement de l'extrudeuse sur l'axe X.
- Transporteurs Z : responsables du mouvement de l'extrudeuse sur l'axe Z, ainsi que le support de l'axe X.
- Tige filetée : C'est le composant mâle d'un système vis/écrou destiné à l'assemblage de pièces ou à la transformation de mouvement. Son complément, pièce femelle est l'écrou.
- La combinaison tige filetées, accouplements, écrous : transforment les mouvements rotationnels en mouvements linéaires.
- Accouplements : éliminent les désalignements du moteur avec la tige filetée.
- Ecrous : supportent les transporteurs Z.
- Roulements linéaires : ils permettent un mouvement linéaire avec un faible coefficient de frottement. Ils ont un coefficient de frottement très faible et une résistance accrue.
- MK8 : mécanique permettant au moteur pas à pas d'extruder du filament.

2.4.3. Vibrations

En théorie les imprimantes 3D et les machines à commandes numériques sont capables de se déplacer avec de grandes vitesses, mais en réalité plus l'accélération est grande plus il y'a des vibrations. Ces dernières sont néfastes et influent drastiquement sur la qualité de l'impression. Plus la masse qui se déplace et son accélération sont grandes, plus on a de vibrations. Il y a deux type de vibrations : les vibrations libres et les vibrations forcés.

2.4.3.1. Vibrations libre

La Figure 2.11 présente le modèle d'une masse-ressort-amortisseur que nous utilisons pour représenter le phénomène de vibrations dans notre imprimante .

K : Rigidité du ressort.
C : Coefficient d'amortissement.
V : Vélocité.
M : Masse.
Wn : Fréquence angulaire.
Fn : Fréquence naturelle non amortie.
Phi : Déphasage.
ζ : Rapport d'amortissement.
X : Magnitude initiale.

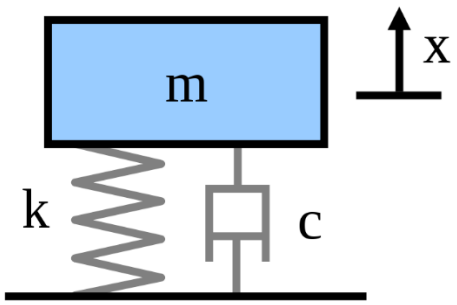

Figure 2.11: Modèle d'une masse-ressort-amortisseur

L'application de la loi de Newton nous donne cette équation différentielle :

$$m\ddot{x} + c\dot{x} + kx = 0.$$

La solution de cette équation dépend de la valeur d'amortissement. Si l'amortissement est suffisamment petit, le système vibre toujours - mais finalement, après un certain temps, il cessera de vibrer. Ce cas est appelé sous-amortissement. Si on augmente l'amortissement jusqu'au point où le système n'oscille plus, le système atteint le point d'amortissement critique. Si l'amortissement est augmenté au-delà de l'amortissement critique, le système est sur-amorti (Figure 2.12). La valeur que doit atteindre le coefficient d'amortissement pour que le système devienne critique est :

$$\zeta = \frac{c}{2\sqrt{\mathrm{km}}}.$$

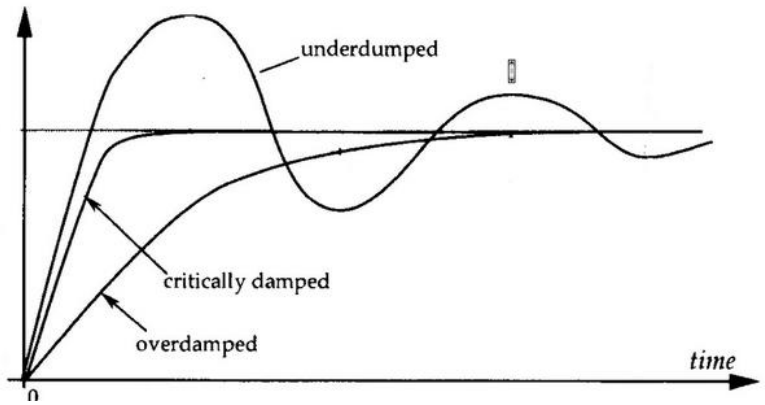

Figure 2.12: Graphes de systèmes sous-amorti, critique et en sur-amorti

La solution au système sous-amorti pour le modèle amortisseur masse-ressort est la suivante :

$$x(t) = Xe^{-\zeta\omega_n t}\cos\left(\sqrt{1-\zeta^2}\omega_n t - \phi\right), \qquad \omega_n = 2\pi f_n.$$

On remarque que plus le rapport d'amortissement est grand plus le système se stabilise rapidement (Figure 2.13). L'amplitude des vibrations diminue avec le temps en fonction de la constante d'amortissement du système. La fréquence des vibrations est principalement dominée par

la rigidité et la masse. Elle est aussi légèrement influencée par la constante d'amortissement visqueux, qui est très faible dans les structures mécaniques [21].

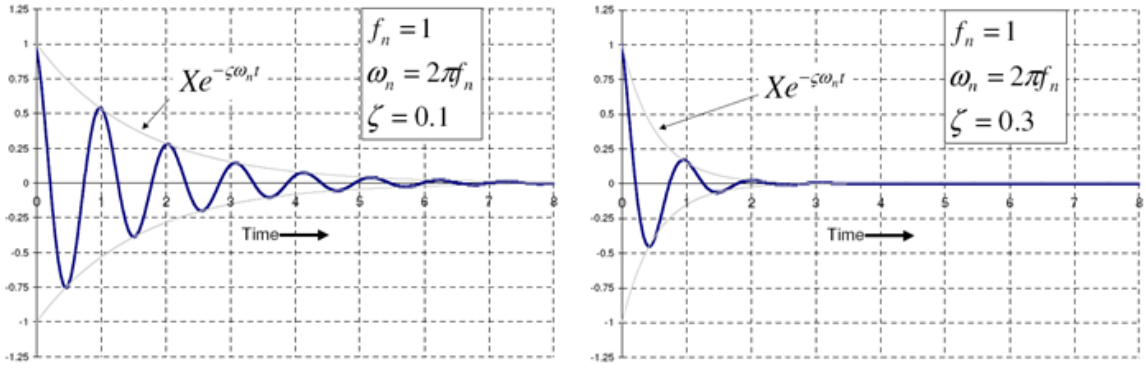

Figure 2.13: Deux graphe de deux systèmes sous-amorti avec un ζ différent

2.4.3.2. Vibrations forcées

Lorsqu'une force externe F(t) est présente, le système (Figure 2.14) subit des vibrations forcées. Si la force est constante, le système éprouve une vibration libre ou transitoire de courte durée, puis se stabilise [21].

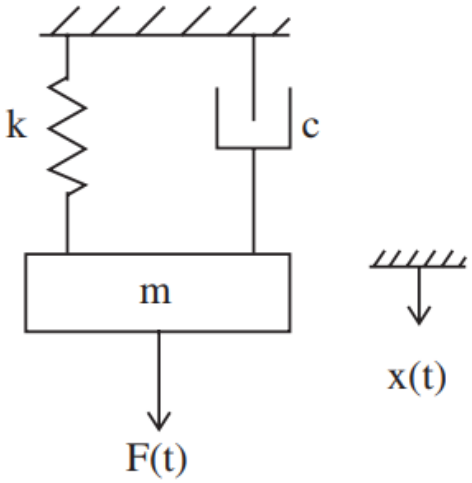

Figure 2.14: Système masse-ressort-amortisseur avec des vibrations forcé

L'équation du système devient alors :

$$m\ddot{x} + c\dot{x} + kx = F_0 \sin(2\pi f t).$$

L'amplitude de la vibration est exprimée par cette relation :

$$X = \frac{F_0}{k} \frac{1}{\sqrt{(1-r^2)^2 + (2\zeta r)^2}}.$$

Avec

$$r = \frac{f}{f_n}.$$

Dans un système légèrement amorti lorsque la fréquence de forçage s'approche de la fréquence naturelle ($f / f_n \approx 1$), l'amplitude de la vibration peut devenir extrêmement élevée. Ce phénomène est appelé résonance. Dans les systèmes de palier de rotor, toute vitesse de rotation qui excite une fréquence de résonance est appelée vitesse critique.

Si une résonance se produit dans un système mécanique, elle peut être très nocive, entraînant éventuellement une défaillance du système. Par conséquent, l'une des principales raisons de l'analyse des vibrations est de prédire quand ce type de résonance peut se produire, puis de déterminer les mesures à prendre pour l'empêcher de se produire. Comme le montre le graphique d'amplitude, l'ajout d'amortissement peut réduire considérablement l'amplitude de la vibration.

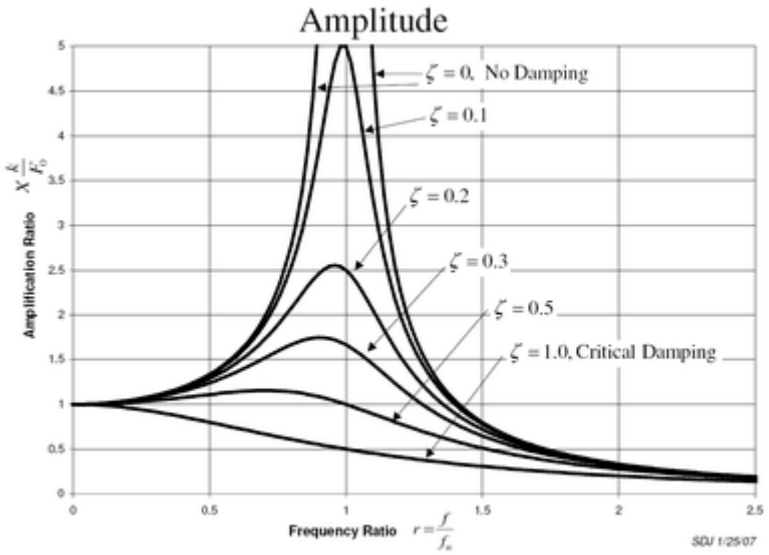

Figure 2.15: Amplitude des vibrations en fonction du ration de la fréquence

En outre, l'amplitude peut être réduite si la fréquence naturelle peut être décalée de la fréquence de forçage en modifiant la rigidité ou la masse du système. Si le système ne peut pas être changé, la fréquence de forçage peut être décalée (par exemple, changer la vitesse de la machine générant la force).

2.4.3.3. Source des vibrations

Pour chaque pas effectué par le moteur pas à pas, la partie tournante, ou rotor, oscille autour de la nouvelle position avant de s'arrêter [22].

Bien que la vibration soit inobservable la plupart du temps, la vibration se traduit par du bruit, qui pourrait être amplifié par d'autres mécanismes connectés au moteur. Il est important de comprendre la source des vibrations avant d'essayer de corriger le problème [23].

À l'intérieur d'un moteur pas à pas, il y a un petit espace d'air entre le rotor et le stator dans lequel circule le flux magnétique, et le seul frottement qui existe est dans le roulement à billes. Lorsque le moteur reçoit l'ordre de se déplacer et de s'arrêter, il n'y a pas suffisamment de friction dans le roulement à billes pour arrêter le dépassement. Cela signifie que l'arbre du moteur fait un dépassement (overshoot) puis un sous-dépassement (undershoot) à chaque pas en raison des forces d'inertie. Le temps qu'il faut à l'arbre du moteur, ou rotor, pour arrêter ce dépassement et ce sous-dépassement est appelé «temps de stabilisation» [23].

Une fois que la fréquence correspond à la fréquence naturelle du moteur, l'oscillation deviendra une résonance et provoquera du bruit. Lorsque la résonance domine le champ magnétique entre les stators et les rotors, le moteur perdra probablement la synchronisation [24].

Oriental Motor a fait une expérience pour illustrer la relation entre la vitesse et les vibrations du moteur pas à pas. Grace à un Tachogenerator (un composant électromécanique qui mesure la vitesse d'un moteur) attaché à un moteur pas à pas ils ont pu observer les vibrations du rotor (plus exactement ils ont observé la vitesse des vibrations qui sont des mouvements, chaque mouvement ayant une vitesse). Ensuite ils ont varié la vitesse du moteur pour voir son impact sur les vibrations. La Figure 2.16 illustre les résultats trouvés.

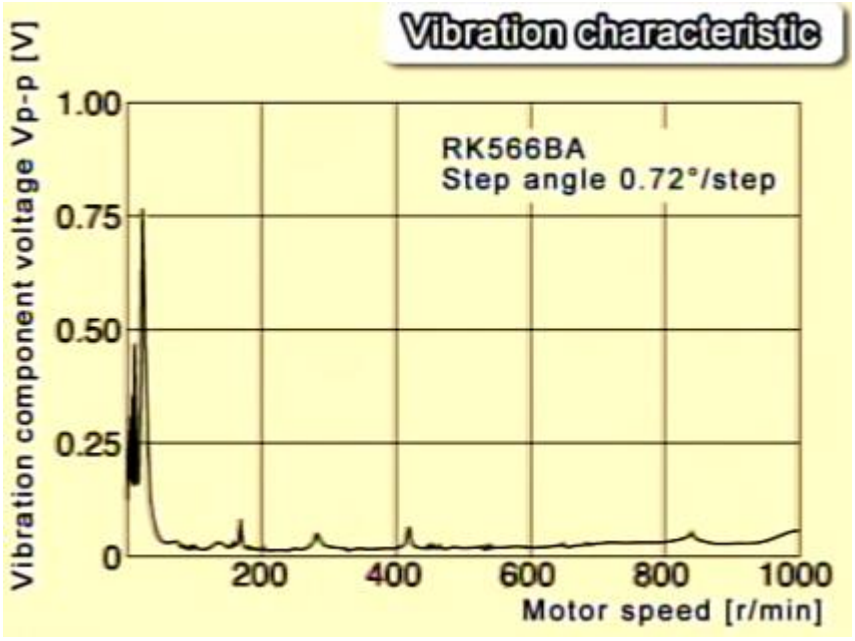

Figure 2.16: Graphe représentant les vibrations en fonction de la vitesse du moteur

D'après les résultats obtenus, on remarque que plus la vitesse est grande plus les vibrations diminuent. On observe un pic des vibrations aux alentours de la vitesse critique (la fréquence de résonance).

La solution serait donc de se diriger vers des vitesses plus rapide et d'éviter la fréquence de résonance, ce n'est malheureusement pas toujours possible. Plus la vitesse du moteur est grande, plus la fréquence des impulsions reçues par le moteur est grande. Cela donne un temps très court pour les inductances du moteur pour se charger et comme on le sait les inductances, vu leurs natures, résistent aux changements brusque de courant. Par conséquent les inductances ne se chargent pas complètement et on peut remarquer la diminution du moment de force (torque).

2.4.3.4. Réduire les vibrations

Nous allons à présent présenter d'autres solutions plus efficaces qu'on peut appliquer dans notre système afin de réduire les vibrations.

1- Eviter les vitesses critiques : Si on peut connaitre la fréquence de résonance on peut l'éviter.
2- Utiliser la technologie Microstepping : Cette technique consiste à diviser l'angle du pas du moteur en plusieurs petits angles en contrôlant le voltage appliqué au bobines du moteurs grâce à un driver.

3- Réduire le courant des moteurs : Le moteur produira moins de couple avec un courant plus faible. En conséquence, moins d'énergie sera produite pour déplacer le rotor (c.-à-d. $D\tau / d\theta$ plus faible, rigidité de couple) [24].
4- Réduire le rapport inertie de la charge sur l'inertie du moteur : La fréquence de résonance peut être augmentée en rigidifiant la constante de ressort du mécanisme ou en réduisant le rapport inertie moteur sur l'inertie de la charge [25]. On peut réduire ce ratio en réduisant la masse des transporteurs ou en ajoutant des « stepper motor dampers » (Figure 2.17).

Figure 2.17: Exemple d'un motor dampers

2.4.4. Cinématique de la machine

2.4.4.1. Mouvement de l'axe X et Y

La combinaison d'une courroie, d'un pignon placé sur un moteur d'un côté et d'une poulie de l'autre côté permet de transformer les rotations du moteur pas à pas en déplacements linéaires comme le montre la Figure 2.18.

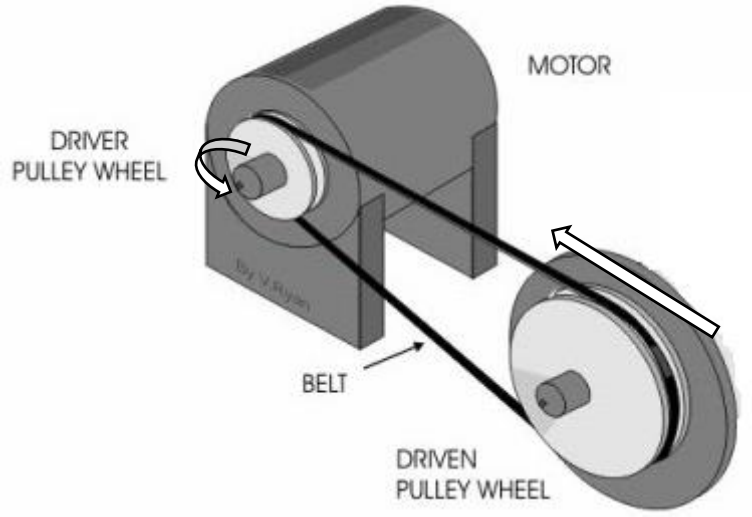

Figure 2.18: Mouvement d'une courroie placée sur un moteur

Si on connait la circonférence « C » de la poulie qui est placé sur le moteur pas à pas et que le moteur tourne de « θ » degrés, alors on peut calculer le déplacement de la poulie « P » avec cette formule-là :

$$P1 = \frac{\theta \times C}{360} \qquad (1)$$

2.4.4.2. Mouvement de l'axe Z

L'axe Z diffère des axes X et Y, il possède deux moteurs pas à pas, deux accouplements, deux écrous, deux tiges filetées et deux axes en acier inoxydables parallèles pour la stabilité de la machine.

Alors pour que le mouvement rotationnel du moteur devient un déplacement linéaire on rajoute un écrou sur la tige filetée, comme ça l'écrou avance ou recule en fonction de la direction de rotation du moteur comme le montre la Figure 2.19.

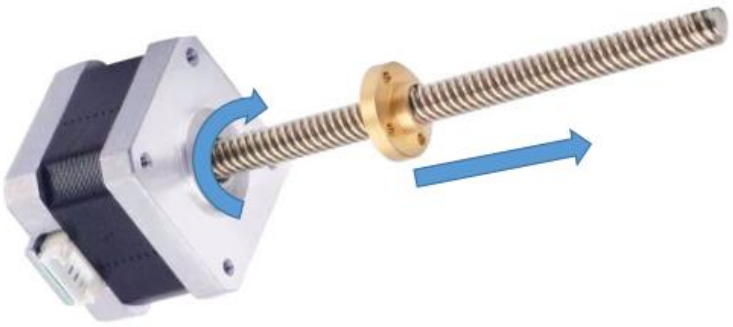

Figure 2.19: Mouvement d'un écrou placé sur une tige filetée.

Pour connaitre le déplacement « P » en fonction du degré de rotation du moteur pas à pas, on doit connaitre le pas de la tige « T », afin d'appliquer cette formule:

$$P2 = \frac{\theta \times T}{360} \qquad (2)$$

2.4.4.3. Mouvement de l'extrudeuse

Le mouvement de l'extrudeuse ressemble à celui de l'axe X et Y sauf qu'il n'y a pas de courroie, on peut calculer la longueur du filament « P » qui va s'extruder par la formule suivante:

$$P3 = \frac{\theta \times C}{360} \qquad (3)$$

2.4.5. Calcul des pas par millimètre

On a parlé des mouvements de l'axe X, Y, E et de l'extrudeuse et évoqué les formules pour calculer les déplacements en fonction du degré de rotation du moteur. Pour plus de précision et pour des calculs plus simples, il est plus sage de remplacer la rotation θ par des pas. Car comme on l'a expliqué dans la partie électronique de ce chapitre, les drivers des moteurs pas à pas fonctionnent avec des impulsions pour émettre un signal vers le moteur pas à pas afin de tourner d'un pas dans la direction voulue.

Le Nema 17 à 200 pas par tour, c'est-à-dire qu'il faut envoyer 200 impulsions à un driver pour que le moteur tourne d'un seul tour. Si on prend en compte les micro-stepping du driver qui est à 1/32 pas alors pour faire un seul tour il faudrait 6400 impulsions.

Revenons alors à l'axe X et Y, on avait dit que les moteurs avaient un pignon et une courroie. La poulie possède 20 dents espacées de 2 mm, cela donne une circonférence de 40mm.

On peut reformuler l'équation (1) de toute à l'heure en remplaçant un tour du moteur qui est égale à 360 degrés par son équivalent en impulsions qui est égale à 6400, sans oublier θ par son équivalent en impulsions « n » comme suit :

$$P1 = \frac{n \times 40}{6400} \rightarrow n = P \times 160$$

On pourra ainsi dire qu'on a 160 pas par millimètre.

On n'a qu'à faire de même avec l'équation (2) pour avoir le pas par millimètre de l'axe Z. On prend la même démarche sauf que cette fois on a « T » le pas de la tige, il est égale à 8 mm. Ça signifie que chaque rotation complète de la tige fera avancé l'écrou de 8 mm. La formule devient comme ceci :

$$P2 = \frac{n \times 8}{6400} \rightarrow n = P \times 800$$

Pour le dernier cas, on a la circonférence du pignon qui est égale à 11 mm donc on a :

$$P3 = \frac{n \times 11}{6400} \rightarrow n = P \times 581$$

2.4.6. Calcul de la vitesse de déplacement

En ce qui concerne la vitesse « v » de déplacement des moteurs, elle dépend du nombre de pas par millimètre « p » ainsi que la fréquence ou la période du signal « T » qui contrôle le driver. On applique cette formule-là :

$$v = p \times \frac{1}{T}$$

2.4.6.1. Calcul de la fréquence de résonance de l'axe X:

Si on peut connaitre la fréquence de résonance on peut l'éviter. Pour cela il faut appliquer cette formule :

$$f = \sqrt{(h / (8\pi \mu S))} \qquad [26]$$

h : Couple de maintien.
S : L'angle du pas en radian.
μ : Moment d'inertie du moteur et de la charge.

En ce qui concerne le moteur que nous utilisons il possède un angle de pas égal à 1.8°. En tenant compte du microstepping l'angle devient égal à 0.05625°. Le moteur possède un couple de maintien égal à 320 mN.m et une inertie égale à 3.8 Kg.mm². Quant à l'inertie du rotor « μ », c'est

la somme de toutes les inerties du système [27] . Dans cet exemple nous allons calculer la fréquence de résonance de l'axe X.

$$\mu = \text{Moment d}'inertie\ de\ l'axe\ X + \text{Moment d}'inertie\ du\ moteur$$

Pour les systèmes qui possèdent une transmission moteur-poulies-courroie, comme notre axe X, on doit appliquer cette formule :

$$Moment\ d'inertie\ de\ l'axe\ X = m \times r^2 \qquad [28]$$

m : La masse du système.
r : Le rayon de la poulie.

La masse du système (l'axe X) est estimée à 500 grammes. Le diamètre de la poulie est égal à :

$$r = \frac{Nombre\ de\ dents \times Le\ pas}{2\ \times\ \pi} = 6.36\ mm$$

Alors :

$$\mu = 0.5\ kg\ \times (6.36\ mm)^2 + \ 3.8\ Kg.mm^2 = \ 24.0248\ Kg.mm^2$$

Ce qui donne :

$$f = \sqrt{\frac{\left(\ 320000\ Kg.\frac{mm^2}{s^2}\right)}{(\ 8\pi\ \times (\frac{0.05625 \times \pi}{180}) \times\ 24.0248\ Kg.mm^2\ )\ )}}$$

La fréquence de résonance de l'axe X avec le Microstepping est égale à :

$$f = \ 735\ Hz$$

La fréquence de résonance de l'axe X sans Microstepping est égale à :

$$f = 130\ Hz$$

On remarque que la fréquence de résonance s'est décalée, mais elle existe toujours.

2.4.7. Conception de l'imprimante

Afin de réduire les erreurs manuelles lors de la réalisation de la machine, nous avons décidé de découper les pièces de l'imprimante avec une machine CNC. Pour cela il faut concevoir les pièces sur le logiciel Adobe Illustrator, les transformer en fichier NC (c'est un fichier qui ressemble aux fichiers G-code) grâce à l'aide du logiciel Aspire et enfin transférer ce fichier vers NC Studio (Interface graphique pour les machine CNC) pour qu'il soit découpé par la fraiseuse.

Par la suite nous avons modifié les pièces 3D qu'on a téléchargé à partir du site Prusa3D afin qu'elles soient compatibles avec notre matériel. Les fichiers sont alors importés vers Fusion 360 pour la modification puis sont envoyés vers le logiciel Cura pour la transformation en G-code. Nous avons utilisé notre ancienne imprimante 3D (celle du mini projet) pour imprimer ces dernières (Figure 2.20). Les figures des fichiers qu'on a imprimés sont disponible dans l'annexe D.

Figure 2.20: Pièces qu'on a imprimé à côté de notre ancienne imprimante 3D

2.4.7.1. Le châssis et le cadre

La Figure 2.21 montre le schéma synoptique général de la partie mécanique.

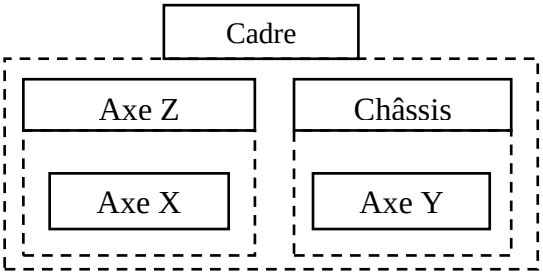

Figure 2.21: Schéma synoptique général de la partie mécanique

La Figure 2.22, Figure 2.23 et Figure 2.24 montrent les fichiers qu'on a conçu avec le Logiciel Adobe Illustrator.

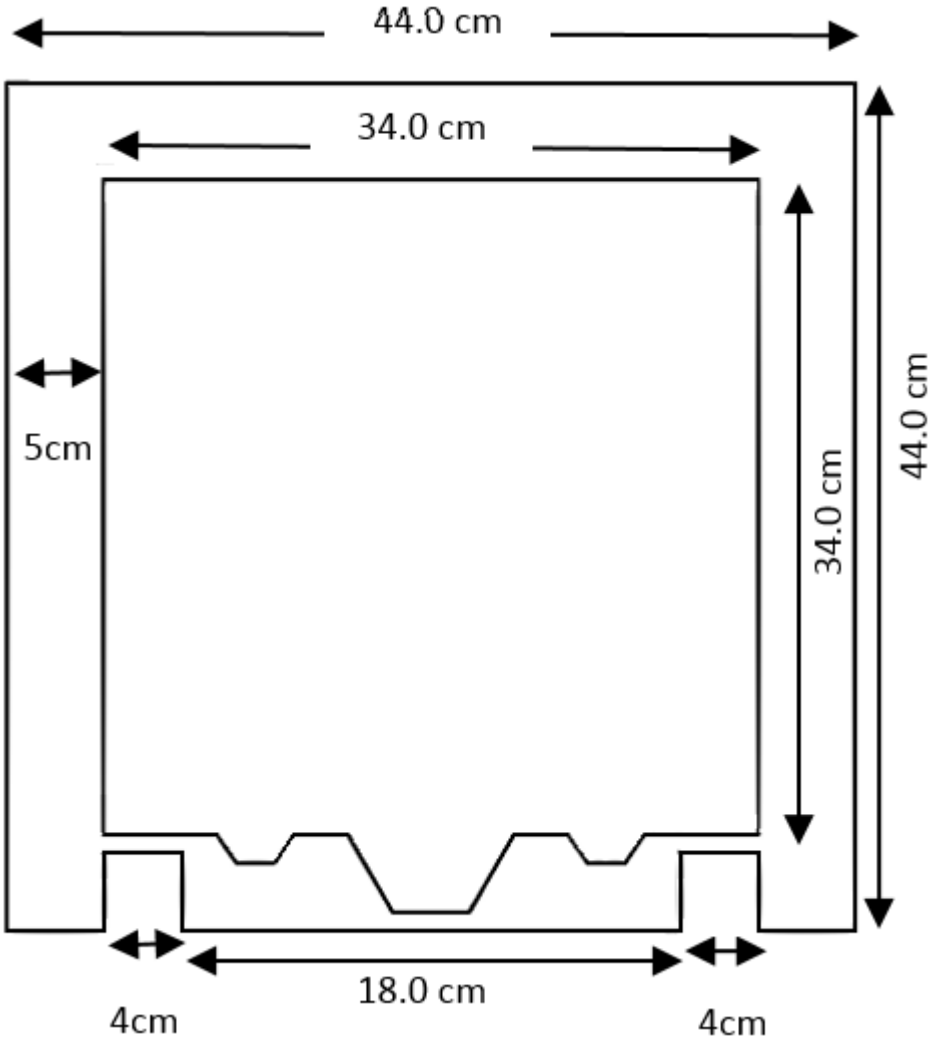

Figure 2.22: Cadre de l'imprimante 3D

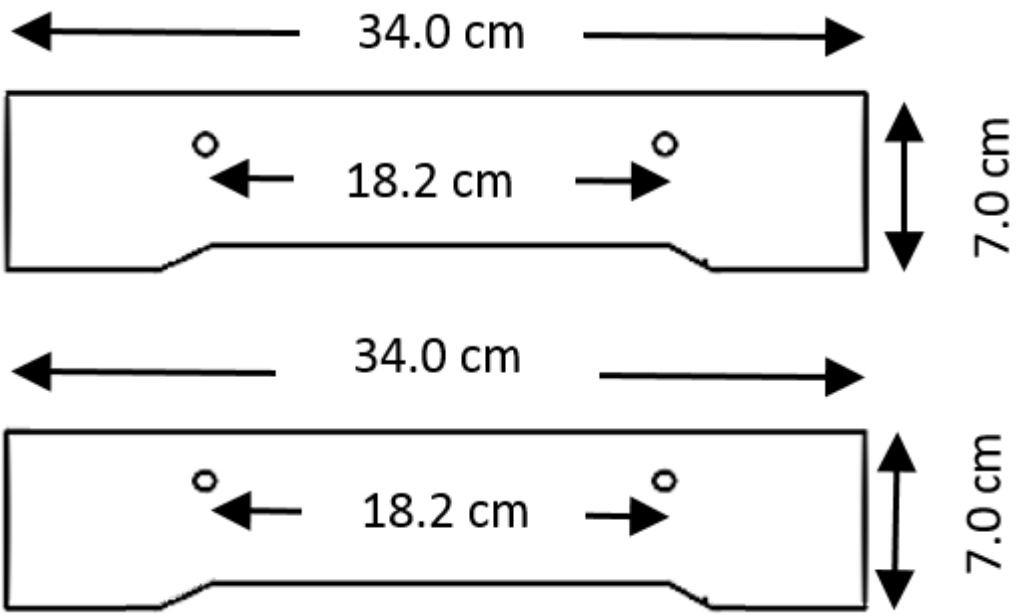

Figure 2.23: Parties avant et arrière du châssis

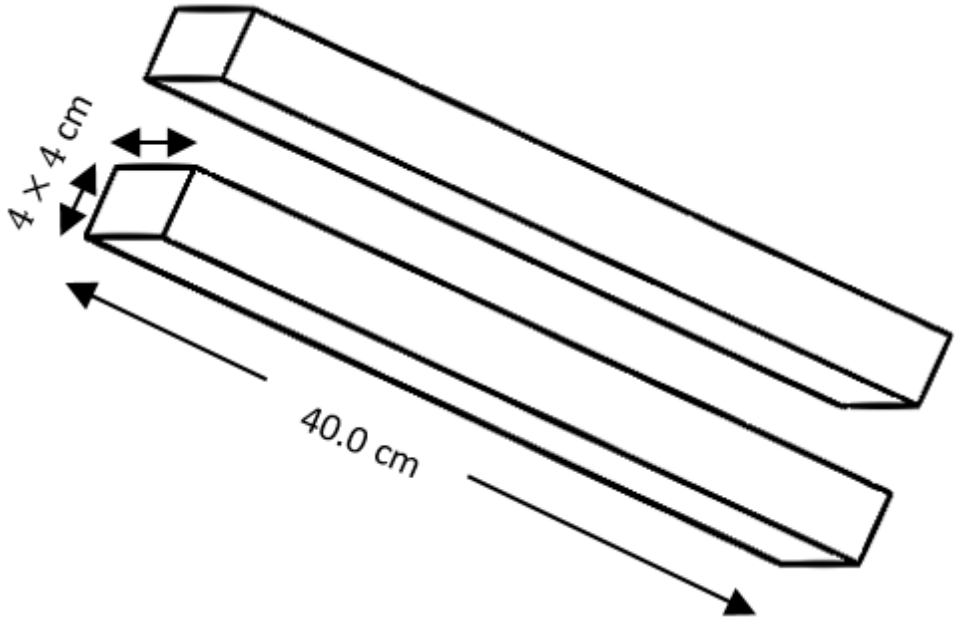

Figure 2.24: Poutres du châssis

La Figure 2.25 montre le cadre et le châssis découpés avec la machine CNC et assemblés.

Figure 2.25: Le Châssis et le cadre assemblé

2.4.7.1. L'axe Y

La Figure 2.26 illustre le schéma synoptique de l'axe Y.

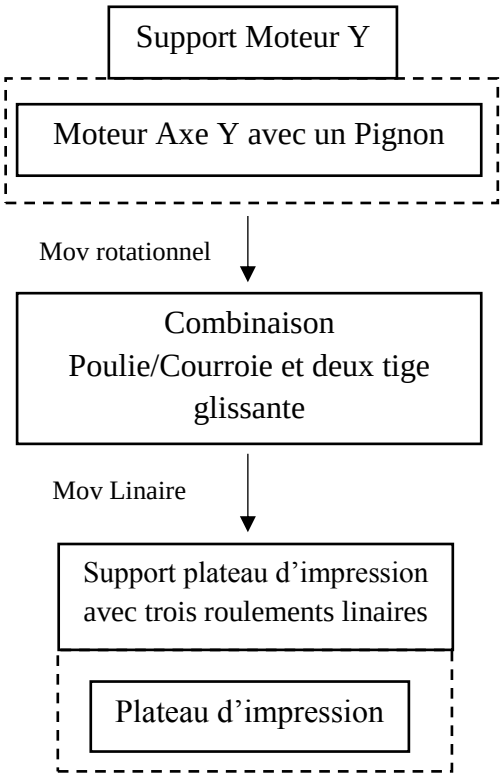

Figure 2.26: Schéma synoptique de l'axe Y

La Figure 2.27 montre le dessin qu'on a réalisé avec le logiciel Adobe Illustrator.

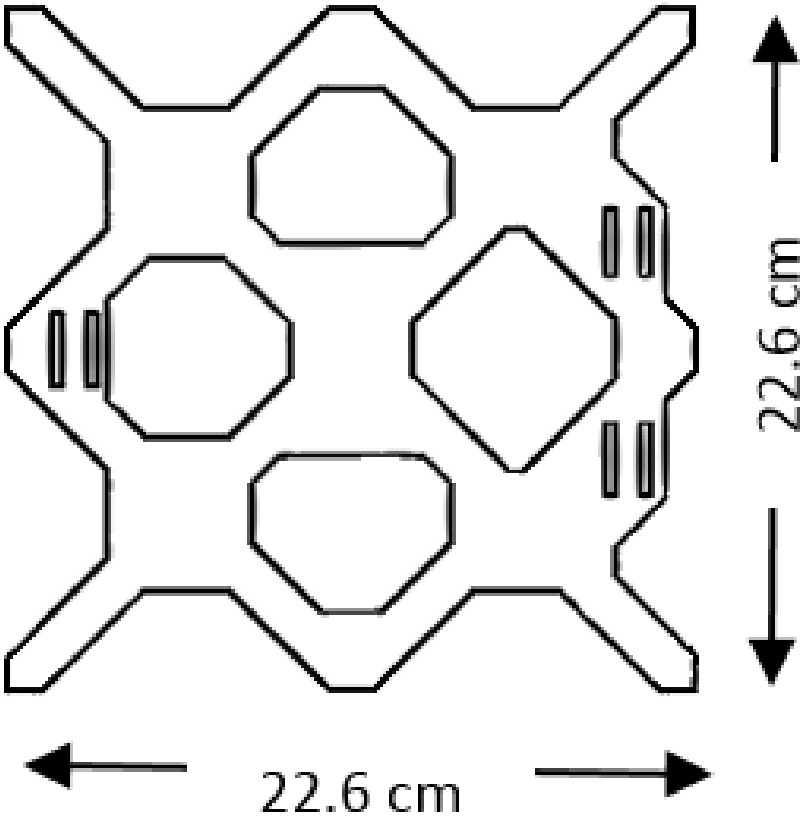

Figure 2.27: Support du plateau d'impression

La Figure 2.28 montre l'axe et le châssis découpés avec la machine CNC et assemblés.

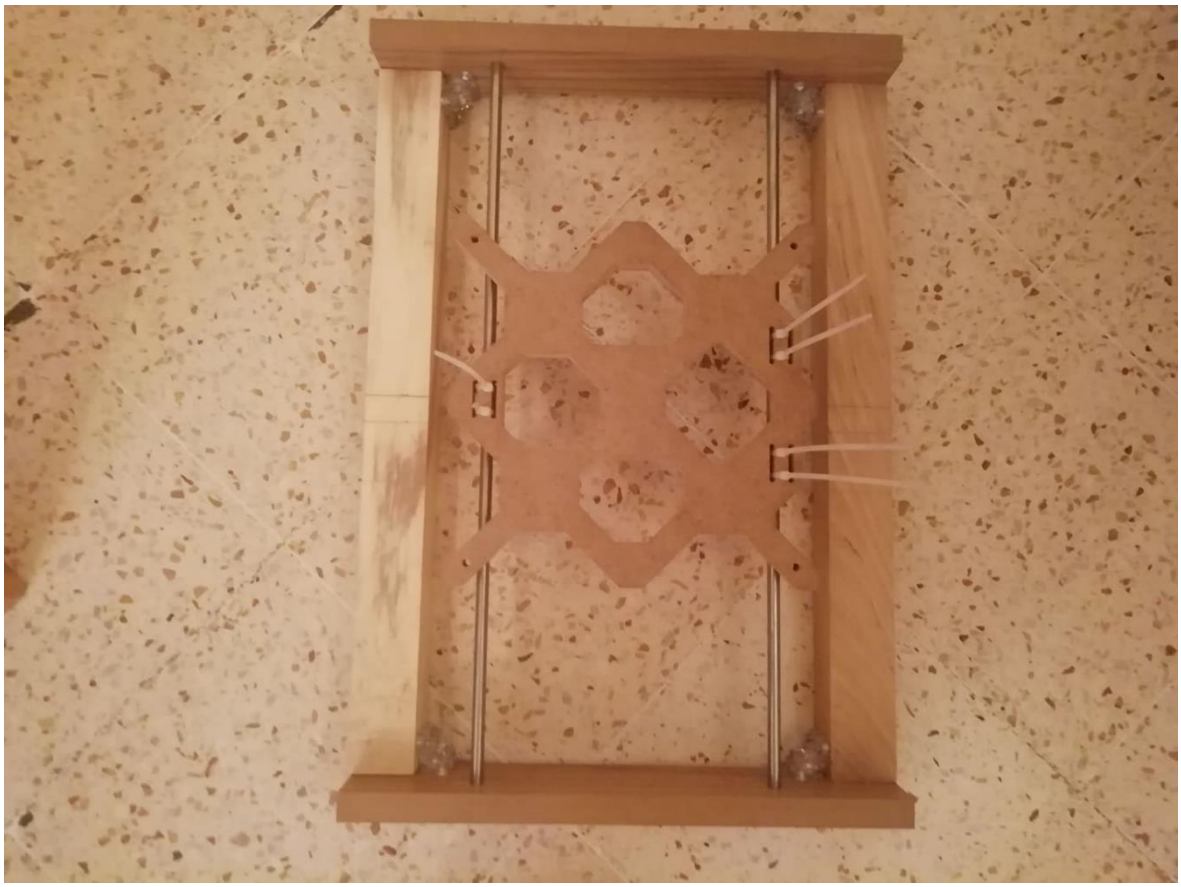

Figure 2.28: Châssis et l'axe Y assemblés

Support du moteur fait manuellement en acier (Figure 2.29).

Figure 2.29: Support de moteur en acier.

2.4.7.2. L'axe X

La Figure 2.30 illustre le schéma synoptique de l'axe X.

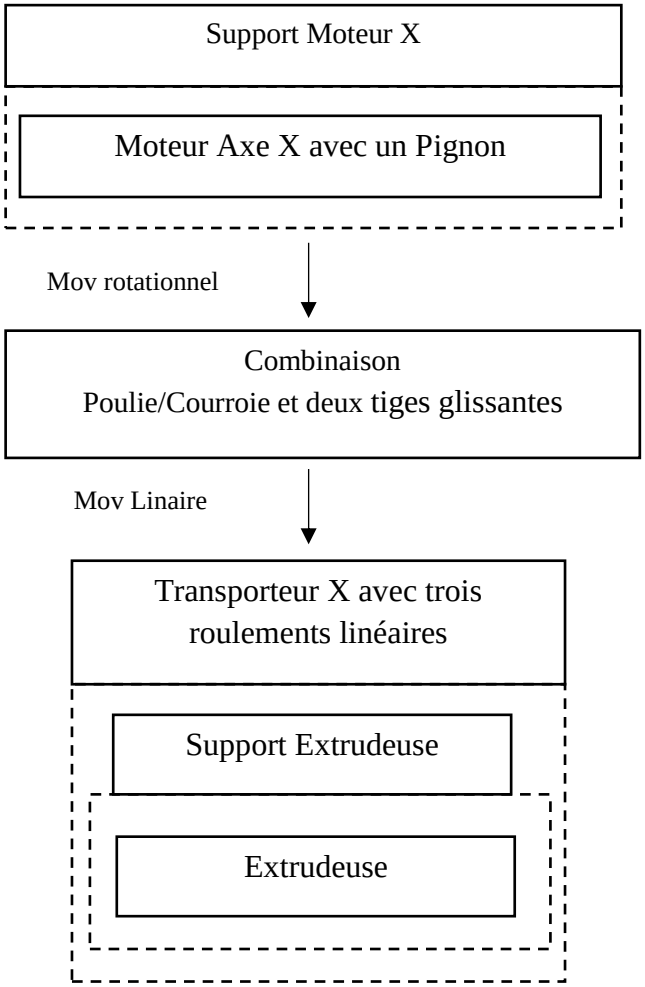

Figure 2.30: Schéma synoptique de l'axe X

La Figure 2.31 monte l'axe X assemblé.

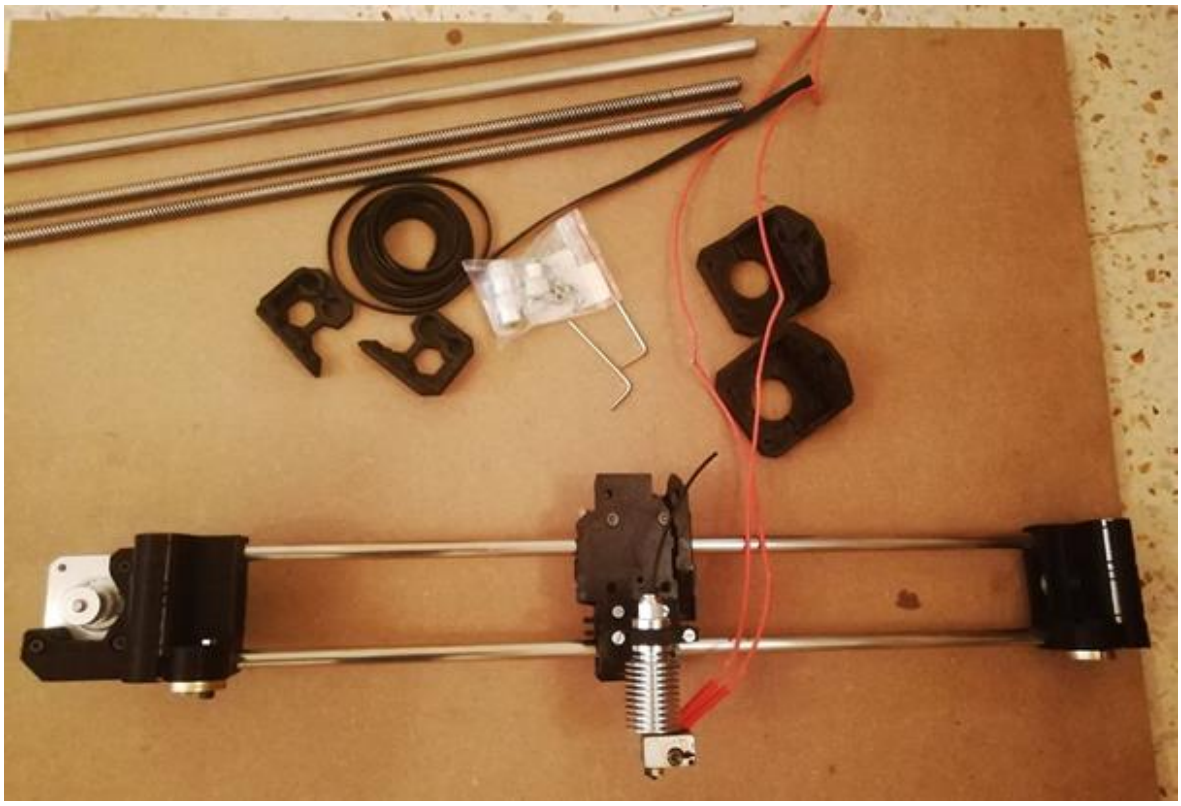

Figure 2.31: Axe X assemblé

La figure du transporteur X est présente dans l'annexe D.

2.4.7.3. L'axe Z

La Figure 2.32 illustre le schéma synoptique de l'axe Z.

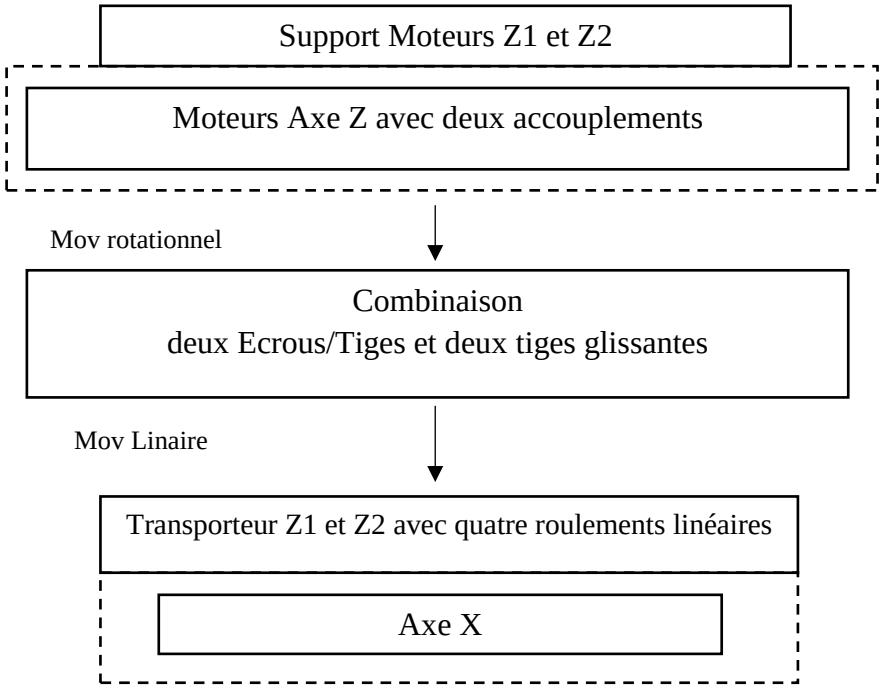

Figure 2.32: Schéma synoptique de l'axe Z

La Figure 2.32 montre le châssis, le cadre, l'axe X, Y et Z découpés et assemblés.

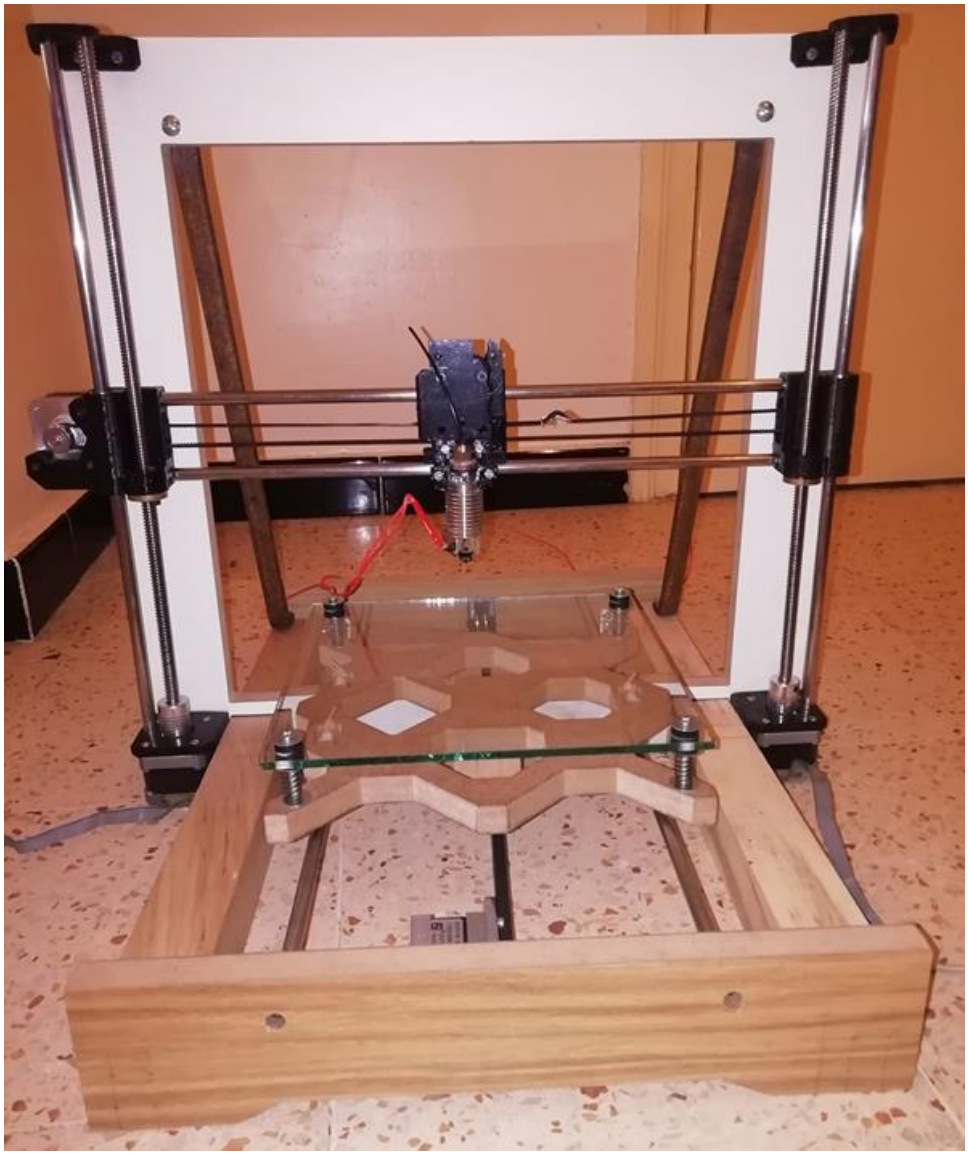

Figure 2.33: Châssis, cadre, axe X,Y et Z assemblés

Support fait manuellement en acier (Figure 2.34).

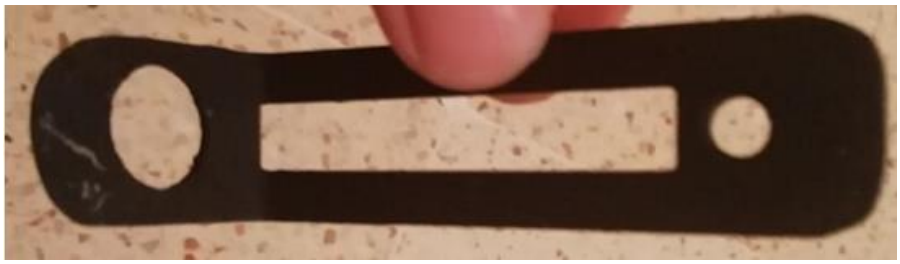

Figure 2.34: Support du capteur de proximité.

Les figures du transporteur Z1 et Z2, les supports des tiges filetés sont présentes dans l'annexe D.

2.4.7.4. L'axe E (l'extrudeuse)

La Figure 2.35 illustre le schéma synoptique de l'axe E.

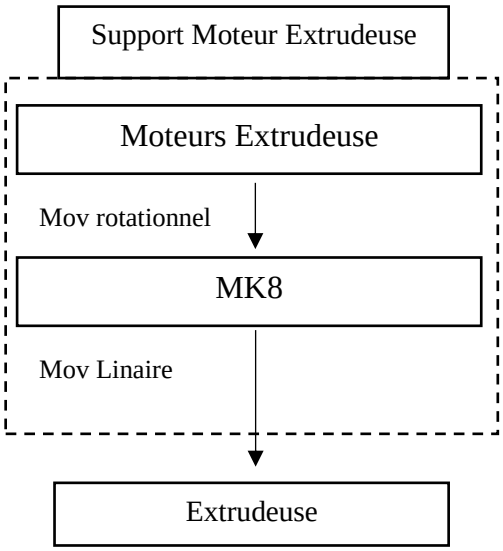

Figure 2.35: Schéma synoptique de l'axe E

Support fait manuellement en acier (Figure 2.36 et Figure 2.37).

Figure 2.36: Support du moteur de l'extrudeuse.

Figure 2.37: Support du moteur de l'extrudeuse assemblé.

2.5. Conception électronique

2.5.1. Introduction

Dans cette partie nous allons présenter les circuits électroniques, que nous avons conçus et réalisés pour commander et contrôler notre imprimante 3D.

2.5.2. Création des schémas électriques

Le schéma synoptique de la Figure 2.44 montre les différents éléments à contrôler dans l'imprimante 3D et l'ensemble des circuits électroniques que nous avons réalisés.

Les différents circuits électriques réalisés sont :

- Circuit de contrôle de la température.
- Circuit de contrôle des moteurs.
- Circuit de contrôle d'affichage LCD.
- Circuit des capteurs.
- Circuit de la résistance thermique

Ces circuits sont connectés au microcontrôleur ATmega 2560 (implanté sur la carte Arduino Mega (Figure B.4) pinout (Figure B.1)), que nous avons programmé pour contrôler et coordonner le fonctionnement de l'imprimante.

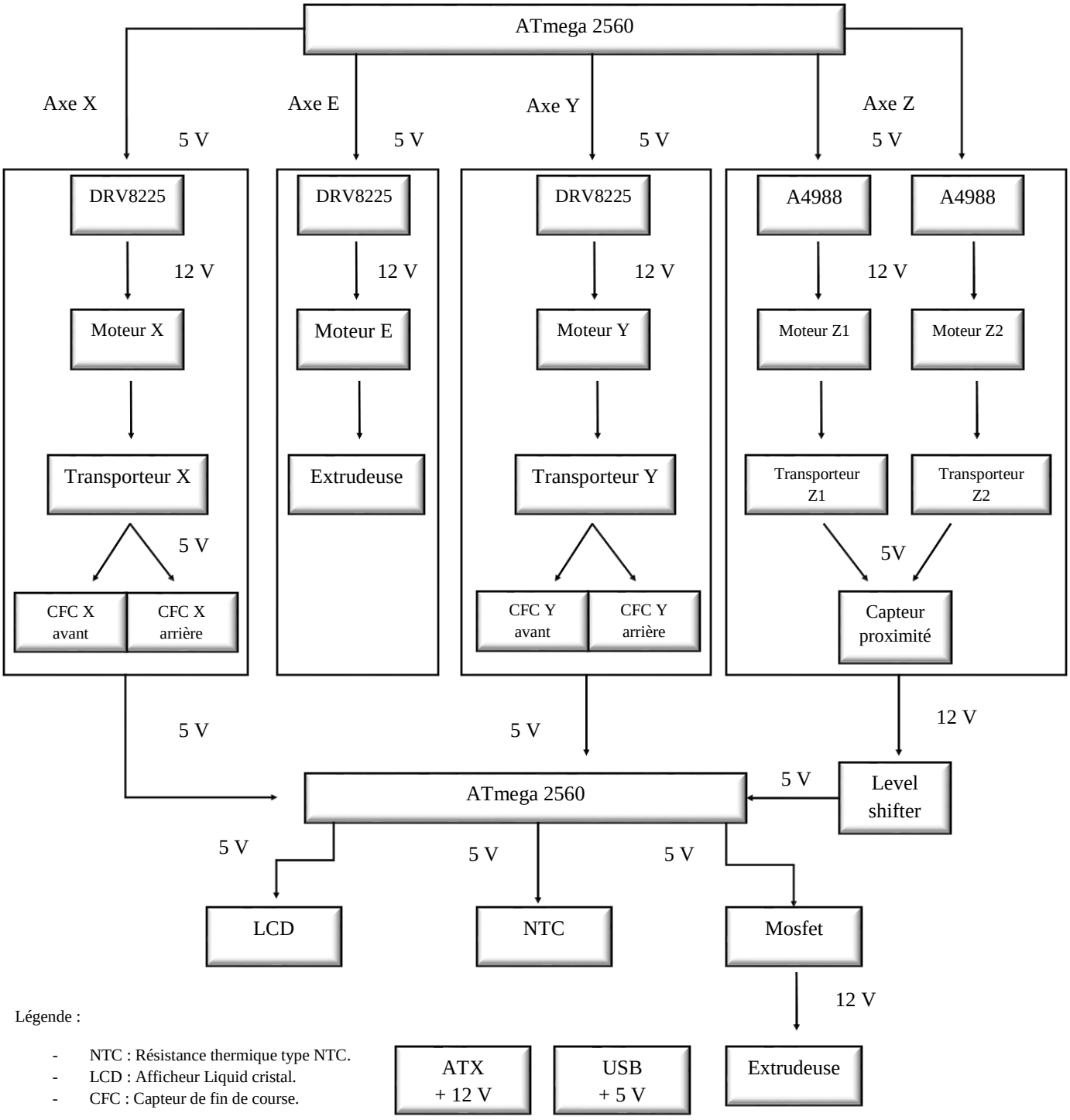

Figure 2.38: Schéma synoptique de l'électronique de notre imprimante

2.5.2.1. Contrôle de la température

La température est un élément clé de l'impression 3D, une mauvaise température (trop grande ou trop petite) risque de donner un résultat négatif voir même un échec de l'impression. Une fluctuation de température, donne une impression hétérogène.

Afin de contrôler au mieux cette température nous avons utilisé un Mosfet qui servira de switch, pour laisser passer un courant réglable vers l'extrudeuse. La Gate du Mosfet est contrôlée par un signal PWM provenant du microcontrôleur. Le circuit de contrôle de la température est illustré dans la Figure 2.39.

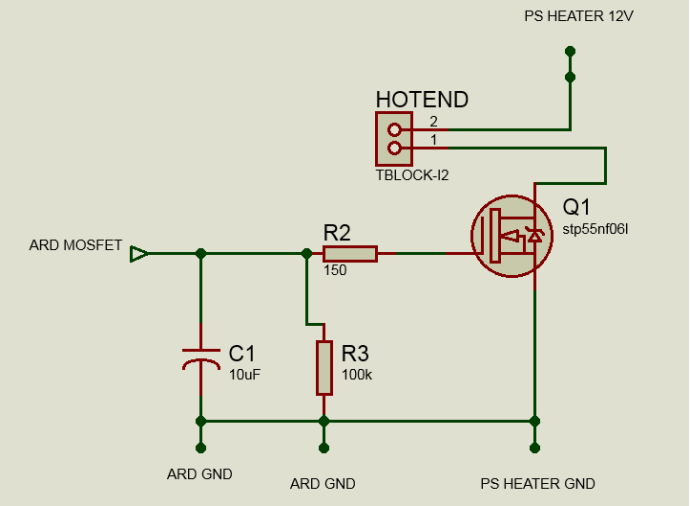

Figure 2.39: Circuit de contrôle de la température.

2.5.2.2. Contrôle des moteurs

Afin de contrôler les moteurs pas à pas nous avons opté pour le DRV8825 de Pololu (Figure 2.40), ce dernier est très facile a utilisé.

Les pins du DRV8825 seront branchés comme suit :

- STEP pour contrôler les pas du moteur pas à pas, à chaque impulsion provenant de du microcontrôleur le moteur tournera d'un seul pas.
- DIR pour spécifier la direction de rotation du moteur pas à pas est connectée au microcontrôleur.
- ENABLE pour activer et désactiver les moteurs. C'est un pin PULL-UP. Cela permet d'avoir les moteurs activer dès l'alimentation du DRV8825.
- M0, M1, M2 afin de spécifier le mode du micro-pas, nous n'utiliserons que le mode 1/32 pas, donc il est inutile de contrôler ces pins-là, alors on les a mis sur VCC.
- VDD et GND moteurs : proviendront de l'alimentation ATX, on place un condensateur entre le VDD et GND comme le recommande le constructeur.
- FAULT et GND logique : proviendront de l'Arduino Mega.
- SLEEP et RESET : le RESET a une résistance interne PULL-UP donc on va connecter les pins SLEEP et RESET entre elles car on n'aura jamais besoin de les contrôler.

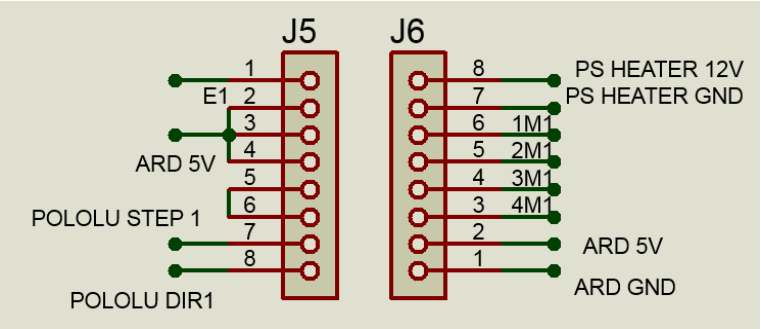

Figure 2.40: Circuit du DRV8825

2.5.2.3. Affichage sur LCD

Pour l'affichage sur le LCD nous avons utilisé un LCD 16x2 contrôlé par un HD44780. L'affichage LCD permet à la machine d'éviter un conflit de données dans la transmission série, car on peut afficher des données sur le LCD sans utiliser l'UART. De plus c'est seulement ce LCD-là qui est disponible.

Le LCD sera configuré sur une transmission 8-bit afin de gagner du temps, on n'utilisera pas le BUSY FLAG du HD44780 donc on sera toujours en mode WRITE. On aura besoin que des 8 pins DATA, ENABLE, REGISTER SELECT, VSS, VEE branché sur un potentiomètre pour régler le contraste de l'afficheur, GND, LED+ et LED- pour le rétroéclairage. La Figure 2.41 présente le schéma électrique du circuit LCD.

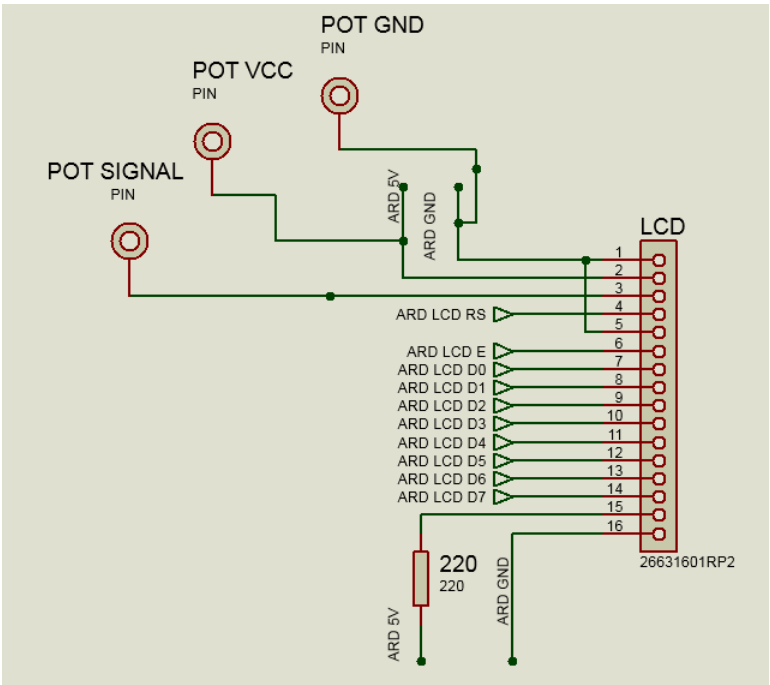

Figure 2.41: Circuit du LCD

2.5.2.4. Capteurs

L'imprimante comporte deux types de capteurs : capteurs de fin de course mécanique et capteur de proximité inductif.

Les capteurs de fins de course mécanique aux nombre de quatre seront utilisés pour détecter les limites de l'imprimante. Ils seront directement reliés à des entrées PULL-UP du microcontrôleur, ceci dit on n'utilisera pas de circuit pour le « bouncing » mécanique des capteurs (rebond lors d'un contact). On pourra régler ce problème avec le software.

Le capteur de proximité inductif Lj12a3-4-z/bx peut détecter un objet métallique à environ 4 mm de distance, il est donc utilisé sur l'axe Z. L'axe Z étant le plus fragile il est imprimé en PLA, si par malheur un choc survient il se brise. Il est alors judicieux d'arrêter l'imprimante 4 mm avant qu'elle ne touche le plateau d'impression.

Le Lj12a3-4-z/bx fonctionne en 12 volt donc on doit adapter le 12 volt en 5 volt pour qu'on puisse récupérer et utiliser le signal fourni. La Figure 2.42 illustre le schéma qu'on a mis en place.

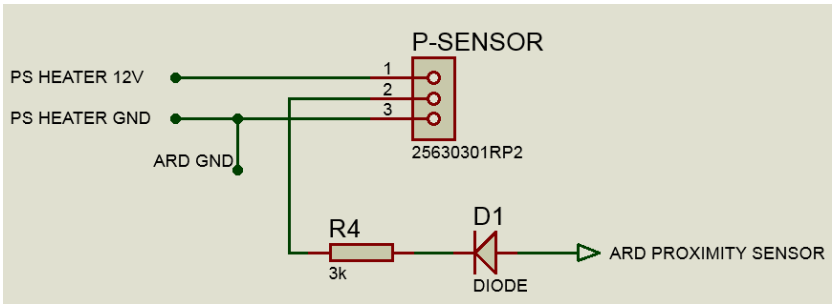

Figure 2.42: Circuit du capteur de proximité

Après avoir consulté la fiche technique et fait quelques calculs on a trouvé que la sortie du capteur est un collecteur ouvert connecté à une résistance PULL-UP d'une valeur de 9 KOhm. Dès que le capteur détecte un objet métallique, le transistor à la sortie conduira le courant et le voltage VCE deviendra alors égal à 0 volt, dans le cas contraire il sera de 12 volt. Alors c'est pour cela qu'à l'entrée de l'Arduino il y a une diode qui empêchera le courant de passer de l'alimentation 12 volt vers l'entrée de l'Arduino mais ça n'empêchera pas de mettre à zéro l'entrée si un objet est détecté. L'entrée de l'Arduino est configurée sur une entrée PULL-UP, ainsi si le capteur ne détecte rien on évitera un pin flottant. La résistance est juste mise par mesure de sécurité.

2.5.2.5. Circuit de la résistance thermique

Pour connaitre la température de l'extrudeuse, il faut utiliser une résistance thermique. Pour cela, nous avons opté pour une NTC 1000K. Un simple pont diviseur de tension a fait l'affaire. Cependant, parfois on détecte un pic de tension (un bruit) qui perturbe plus ou moins le bon fonctionnement de l'algorithme de régulation de la température. Alors pour résoudre ce problème, on a ajouté un condensateur, ce qui a beaucoup aidé à l'élimination de ce bruit. Le circuit de la NTC est présenté dans la Figure 2.43.

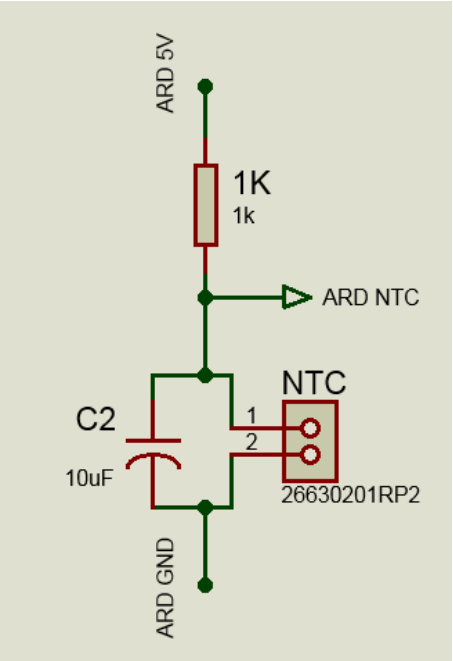

Figure 2.43: Circuit de la résistance thermique

2.5.3. Schéma électrique de toute l'imprimante 3D

La Figure 2.44 monte le schéma électrique complet que nous avons conçus.

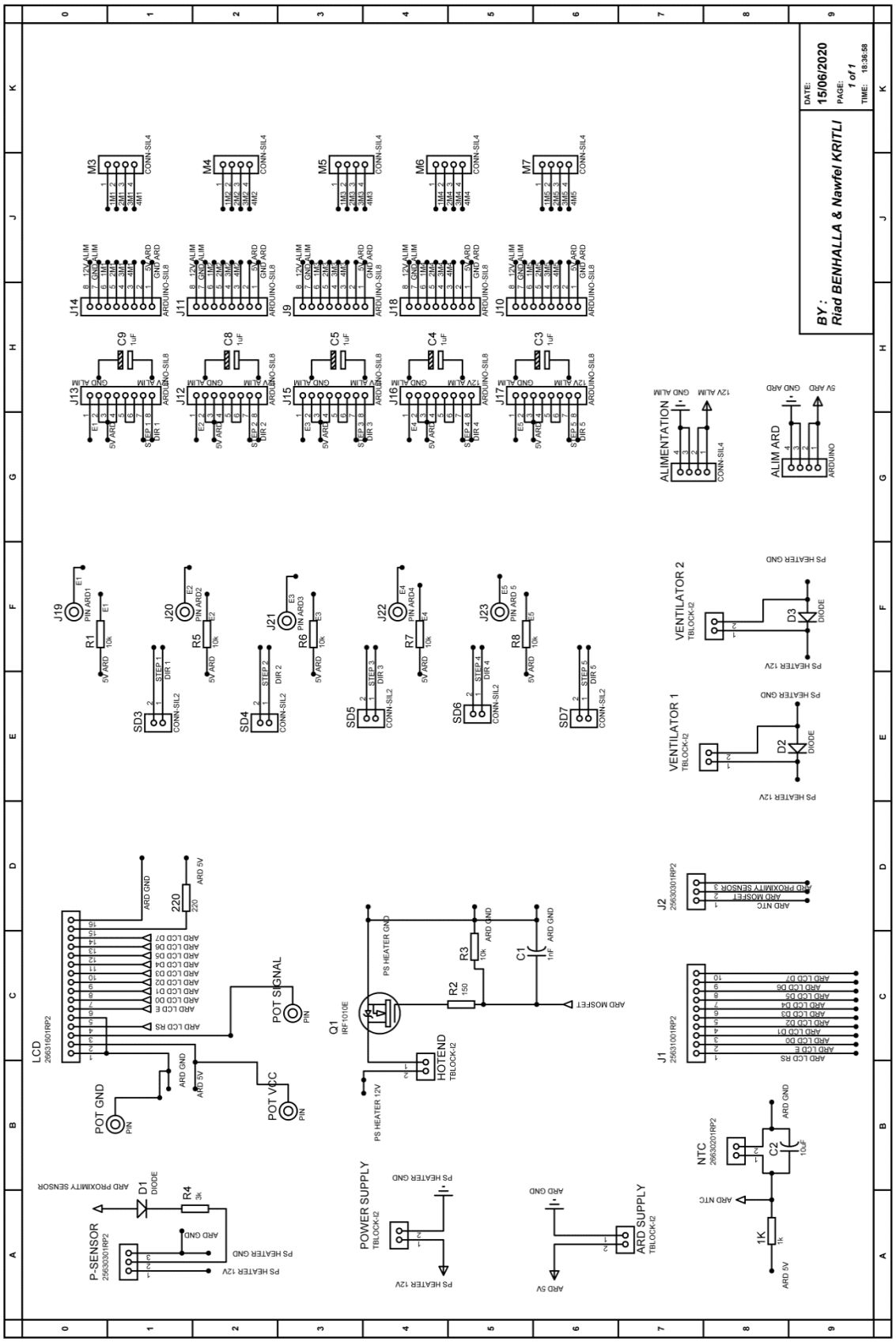

Figure 2.44: Schéma électrique complet réalisé

2.5.4. Bruit électrique

Le bruit est un problème très récurant et nuisible en électronique. Pour la réalisation des schémas qu'on a présentés précédemment, on a été obligés d'utiliser beaucoup de fils électriques. Ces derniers ont agi comme des antennes, provoquant un bruit qui donne parfois des résultats imprévisibles et difficile à détecter, spécialement sans matériels adéquats.

Afin de résoudre ce problème nous avons conçue deux PCB. La première a pour but d'englober toute la circuiterie des contrôleurs des moteurs pas à pas. Quant à la deuxième c'est pour l'affichage LCD, le contrôle de température, le capteur de proximité, le circuit pour la résistance NTC et les moteurs des ventilateurs.

2.5.5. Réalisation de la première PCB

Cette PCB (Figure 2.45 et Figure 2.46) comprend 5 drivers DRV8825 pour le contrôle des moteurs pas à pas de l'axe X, Y, Z1, Z2 et du moteur de l'extrudeuse.

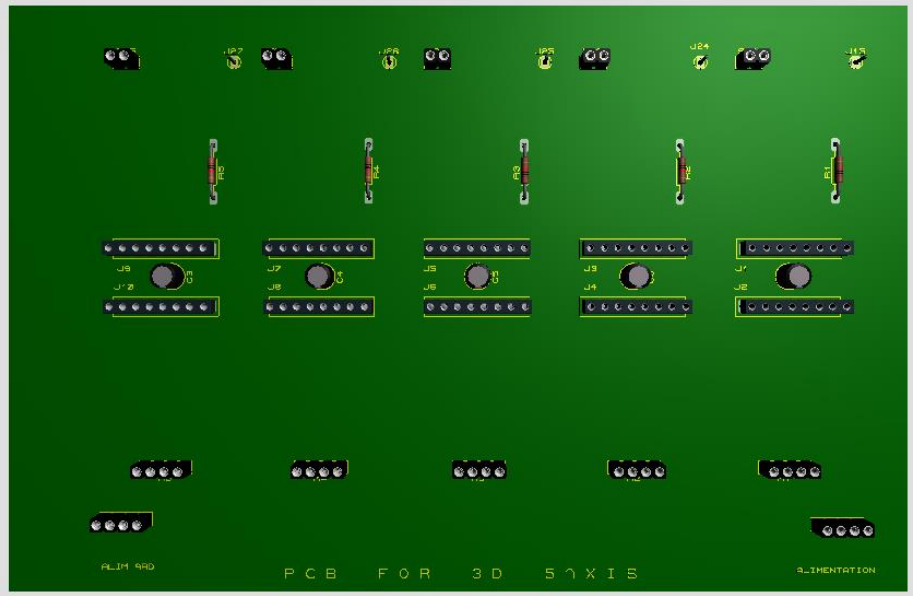

Figure 2.45: Vu 3D de la PCB numéro 1

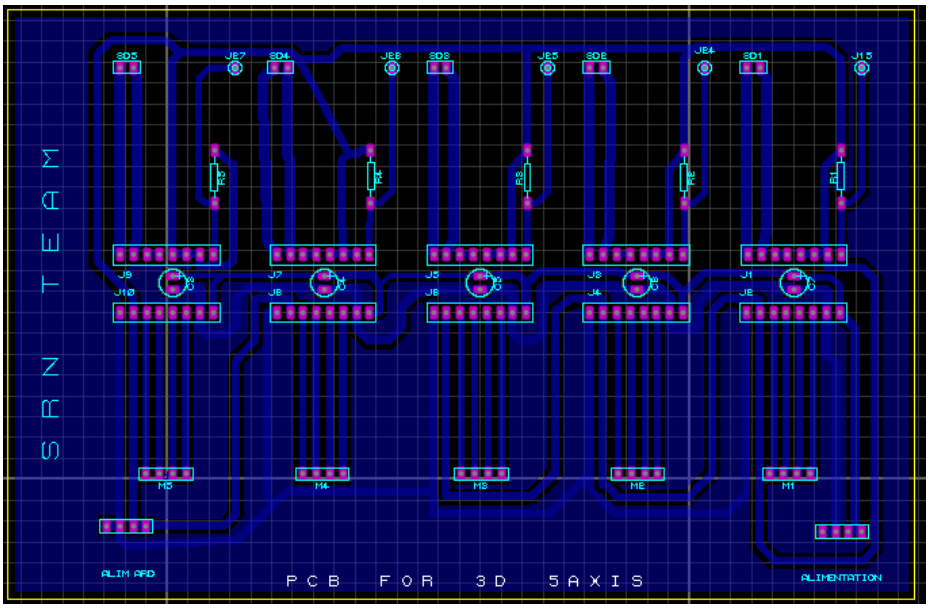

Figure 2.46:Layout de la PCB numéro 1

2.5.6. Réalisation de la deuxième PCB

Dans cette PCB (Figure 2.47 et Figure 2.48) il y a un LCD, un potentiomètre pour régler le contraste du LCD, le circuit pour la résistance NTC, le circuit du contrôle de la température avec le Mosfet, le circuit du capteur de proximité inductif et deux diodes de roue libre pour l'utilisation des ventilateurs.

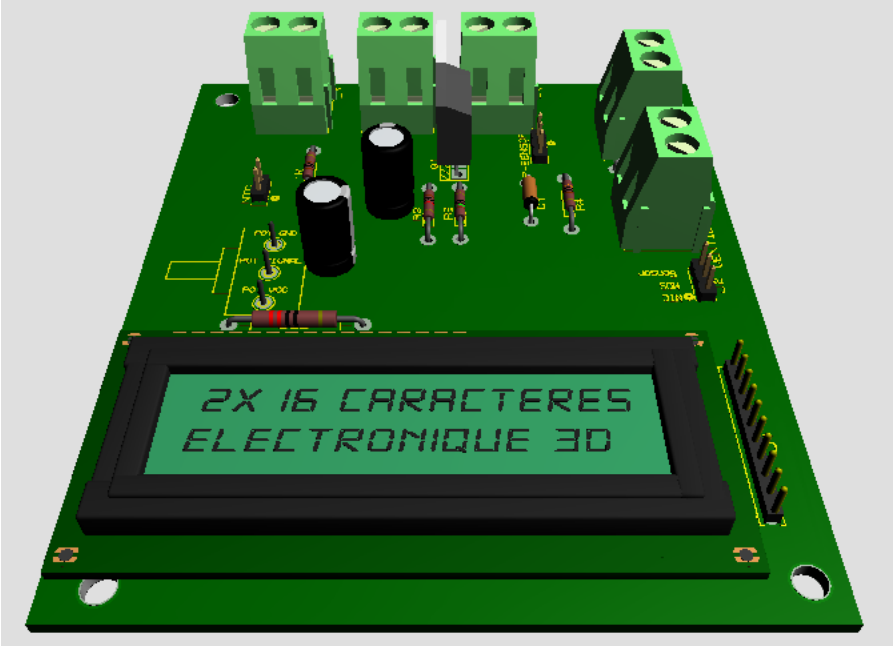

Figure 2.47: Vu 3D de la PCB numéro 2

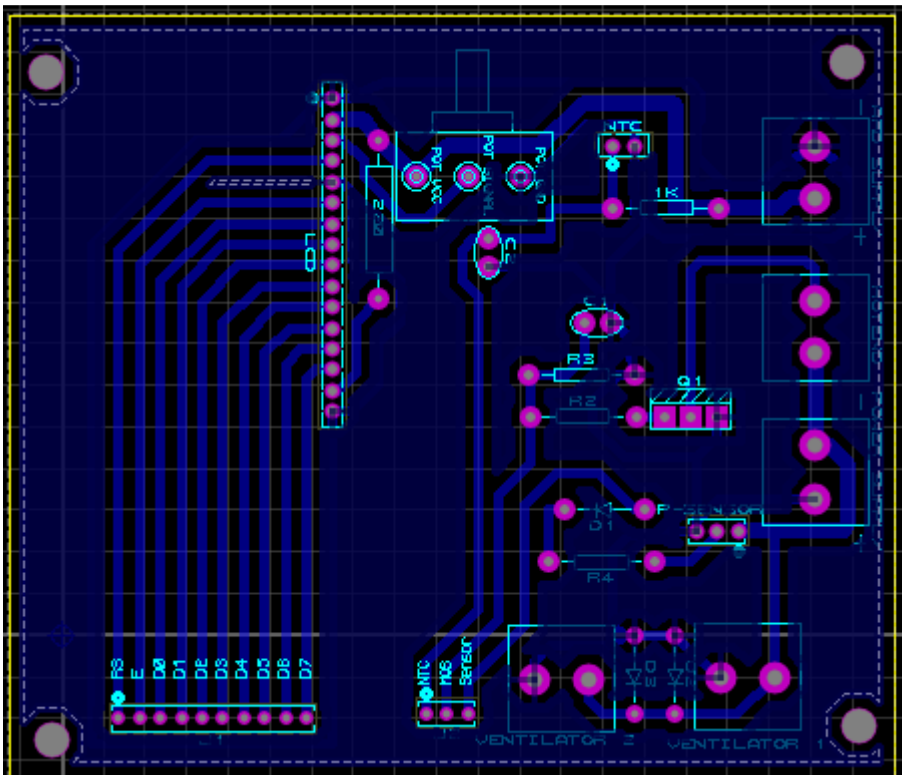

Figure 2.48:Layout de la PCB numéro 2

2.6. Conclusion

Dans ce chapitre nous avons vu la partie électronique de l'imprimante 3D, des circuits sur Proteus jusqu'à a la réalisation des PCB. Nous avons aussi discuté du problème de bruit que nous avons rencontré.

Nous avons aussi vu la partie mécanique en commençant par les contraintes qu'imposent les lois de la physique sur l'imprimante 3D, puis nous avons expliqué la cinématique de la machine. Ensuite nous avons vu le design de la machine qu'on a conçu, les objets que nous avons imprimés grâce à notre ancienne imprimante et pour finir l'assemblage de la machine.

L'imprimante étant terminée, il faut la programmer pour la faire fonctionner. Dans le prochain chapitre nous allons passer à la partie software.

Chapitre 3. Développement du software et du Firmware de l'imprimante 3D

3.1. Introduction

L'imprimante 3D que nous avons conçue (chap 2) étant construite, nous développons dans le présent chapitre, les logiciels que nous avons créés pour son fonctionnement. Ils sont classés en deux types: des logiciels sur PC et un logiciel embarqué.

Les logiciels sur PC sont au nombre de neuf :

1- Interface graphique utilisateur (GUI) : elle permet à l'utilisateur d'une part de gérer l'impression 3D sélectionné, d'initialiser et de configurer l'imprimante, et de visualiser en temps réel les paramètres de l'imprimante. D'autre part elle permet aussi d'accéder à d'autres logiciels qu'on a créés (G-Code Simulator, STL Viewer et G-Code Preprocessor).
2- G-Code Preprocessor : il permet de générer les fichiers nécessaires aux étapes successives de l'impression.
3- G-Code Simulator : c'est un ensemble qui contient les programmes suivants : G-Code Simulator S, G-Code Simulator D, G-Code Simulator Preprocessor S et G-Code Simulator Preprocessor D.
4- G-Code Bézier : Il permet d'utiliser les courbes de Bézier.
5- STL Viewer : Il permet de visionner des fichiers STL.
6- G-Code Futurs Mouvement : Il permet de générer des fichiers indispensables a l'impression.

Un logiciel embarqué sur l'imprimante, le Firmware, chargé de transformer les fichiers de commande générés sur PC, en commandes numériques, exécutées par le microcontrôleur Atmega 2560, pour le système électromécanique.

Afin d'obtenir une impression 3D de qualité et résolution acceptables, nous avons répertorié les problèmes pouvant survenir à différentes étapes de l'impression, et introduits des correcteurs sur PC et dans le Firmware.

Tous les programmes, idées et méthodes sont des créations personnelles.

3.2. Flot de conception software

La conception du software (Figure 3.1) se présente comme suit :

- Etude des Firmware déjà existant dans le but d'avoir un aperçu et de comprendre le fonctionnement du logiciel embarqué sur imprimante 3D. Certains Firmware comme « Marlin » on un nombre de « predirective instructions » très important, ce qui rend le code non seulement difficile à comprendre mais demande beaucoup temps. D'autres Firmware, proposent une documentation médiocre, mais leur philosophie (à propos de la conception du Firmware) ne correspond pas au hardware que nous avons créé. Par exemple certains Firmware requièrent l'ajout d'un deuxième microcontrôleur plus performant que l'ATmega2560, ce qui est impossible dans notre cas. Alors on doit créer notre propre architecture Firmware sans avoir un modèle ou une littérature à suivre.
- Etude et choix des outils nécessaires pour la création du Firmware, cela va des IDE et langages de programmation qu'on doit utiliser jusqu'au choix du standard c++.
- Développement d'une philosophie et des règles pour la conception du Firmware.

- Créer des algorithmes qui vont tourner dans les programmes crées (programmes embarqués et non embarqués).
- Création d'une architecture Firmware pour notre imprimante 3D.
- Création d'autres programmes non embarqués (donc qui tourne sur ordinateur).
- Création d'une interface graphique.

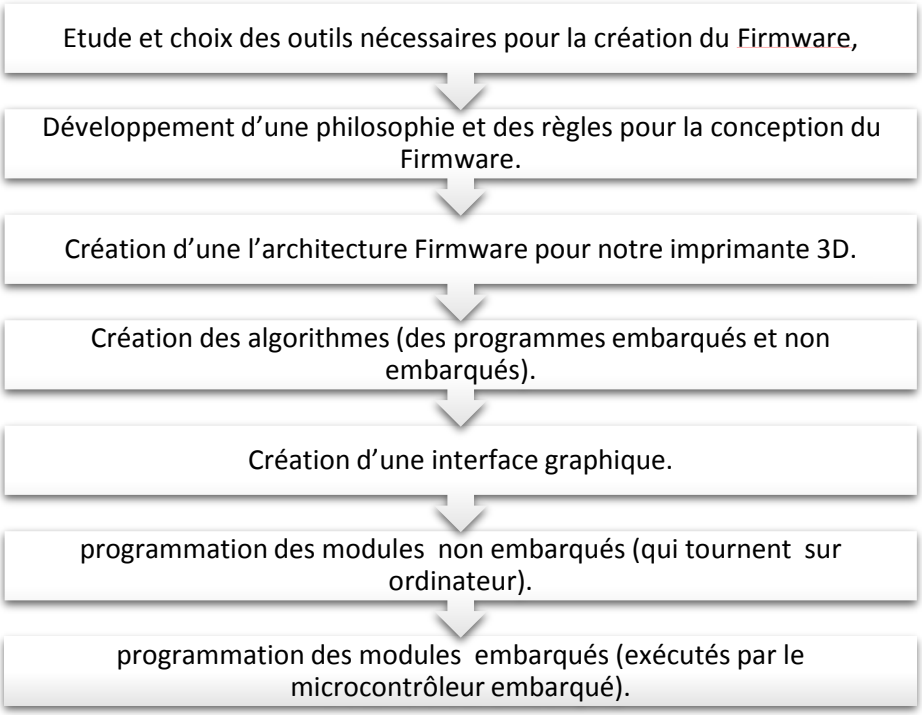

Figure 3.1: Flot de conception du software de l'imprimante 3D

3.3. Identification des modules principaux à programmer

3.3.1. Programmes sur PC

L'impression 3D est initiée à partir du PC. Nous avons créé une interface graphique qui permet d'accéder aux programmes de prétraitement qui préparent l'impression 3D, puis d'envoyer au système embarqué, les commandes pour le système électromécanique. Grâce à l'interface graphique, il est également possible d'acquérir, par le biais du port série (par des sous programmes de communication des informations mises à jour en temps réel, et fournies par le microcontrôleur, sur l'état de l'impression.

Ainsi, sur le PC on accède aux programmes grâce aux commandes évènementielles de l'interface graphique. La fenêtre d'accueil permet d'accéder directement à quatre commandes et sept options (Figure 3.2).

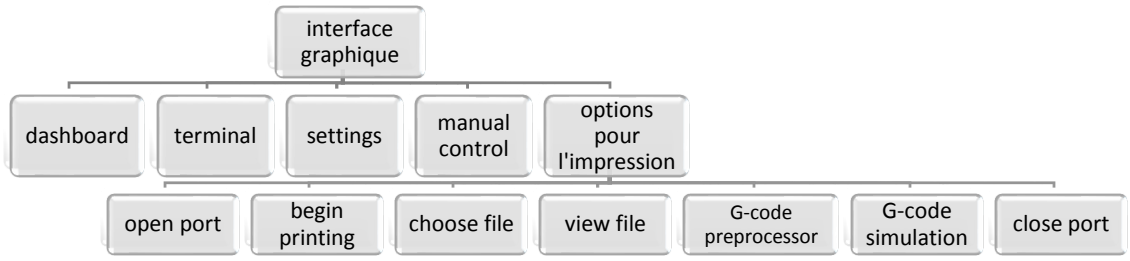

Figure 3.2: Schéma synoptique du GUI

D'autres programmes qu'on a créés ne sont pas accessible depuis l'interface graphique (Figure 3.3).

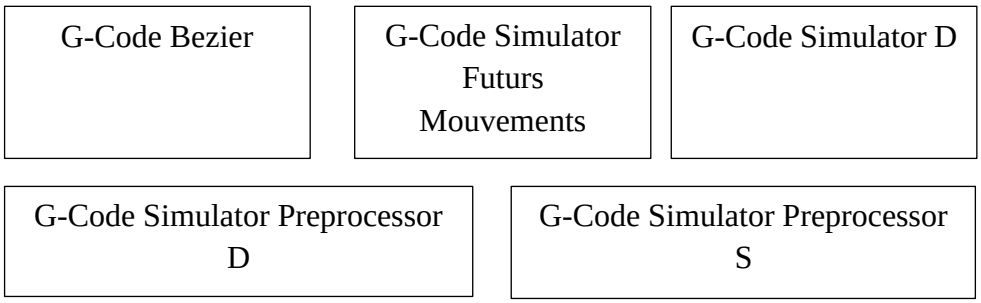

Figure 3.3: Les autres programmes qu'on a crées

3.3.2. Programmes exécutés sur l'imprimante:

Les programmes exécutés au niveau de l'imprimante ont pour rôle de commander et contrôler les éléments électroniques et électromécaniques disponibles sur l'imprimante et forment le Firmware. Ils sont classés en deux types (Figure 3.4):

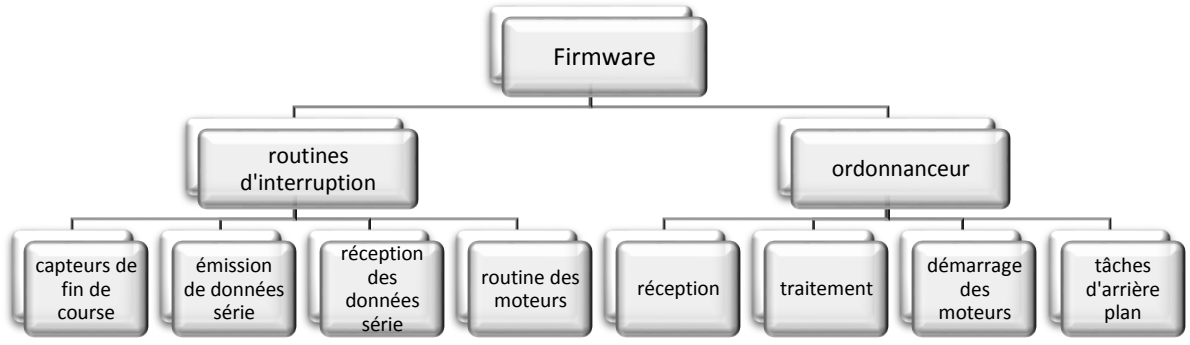

Figure 3.4: Schéma synoptique du Firmware

L'ordonnanceur: chargé de synchroniser et coordonner quatre tâches.

Les programmes d'interruptions: chargés d'interrompre le microcontrôleur pour lui faire exécuter des tâches de plus haute priorité, requise par l'état des capteurs, et actionneurs, le PC etc…

3.3.3. Programmes à développer sur PC et sur l'imprimante

La Figure 3.5 résume les principaux programmes à développer pour la gestion de l'imprimante, et l'impression 3D.

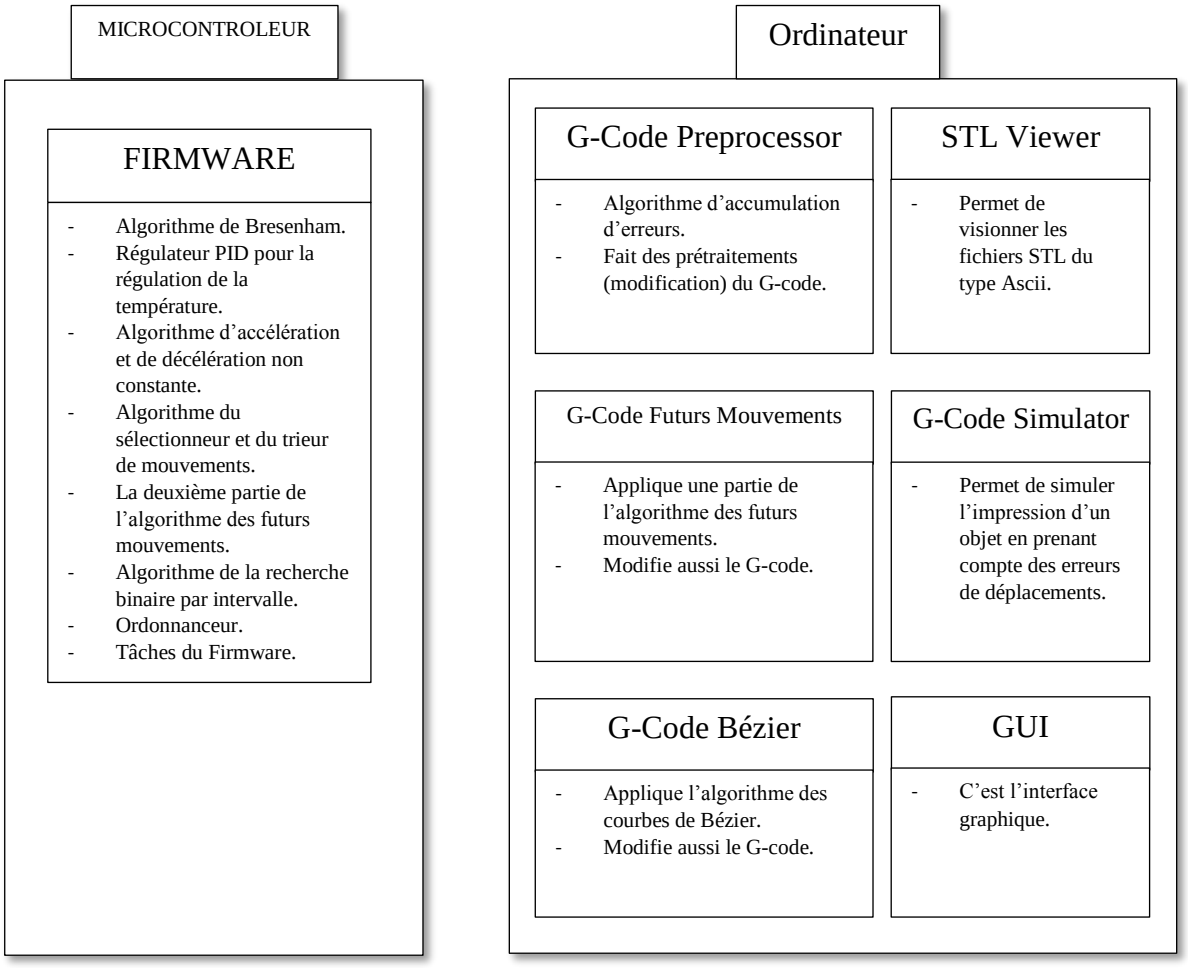

Figure 3.5: Schéma synoptique des programmes crées

3.4. Programmes qui tournent sur PC

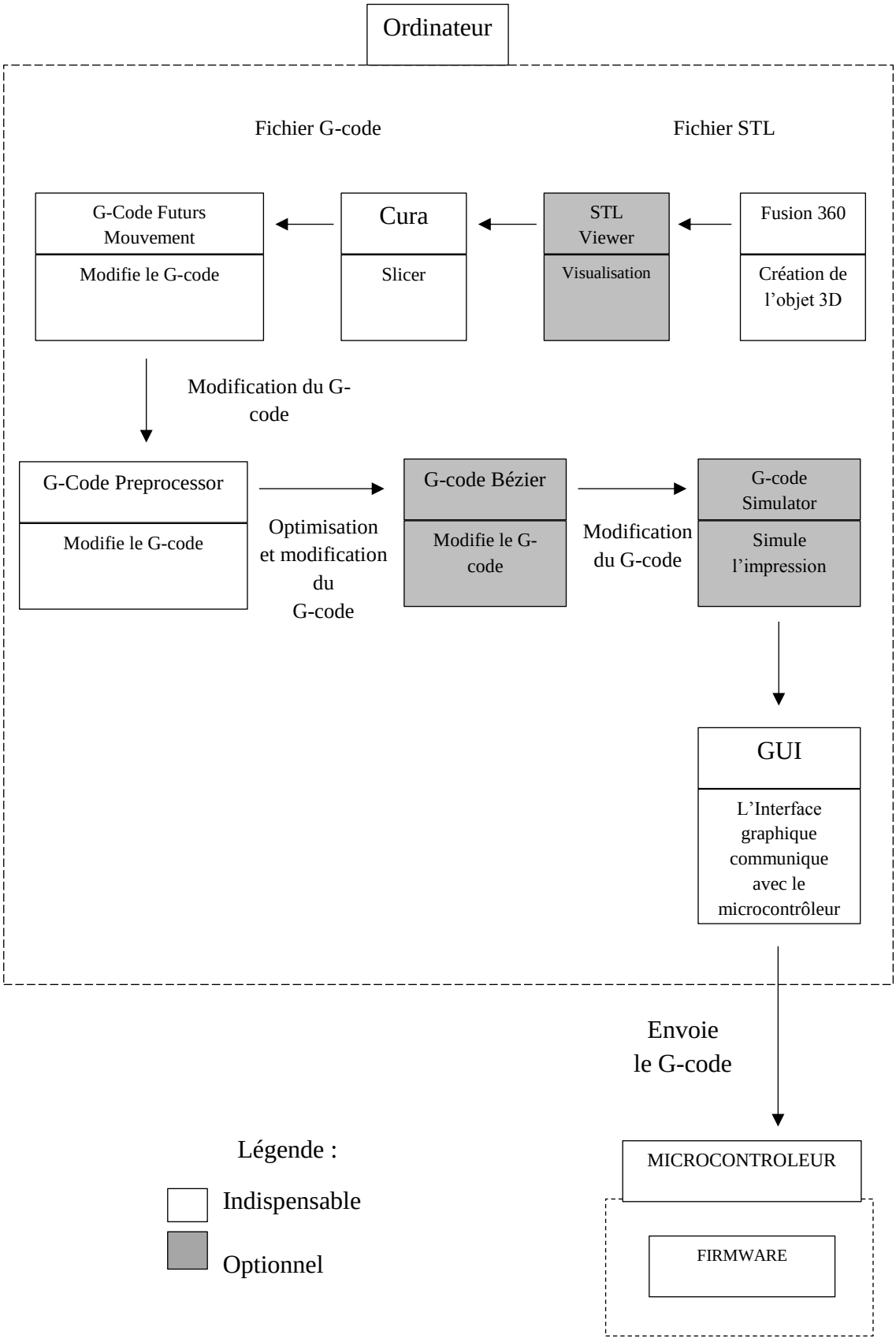

Figure 3.6: Schéma synoptique des étapes d'impression

3.4.1. Interface graphique (GUI)

Nous avons créé une interface graphique ergonomique écrite en Java (Swing), elle permet à l'utilisateur de contrôler la machine de façon interactive et simple, sans avoir à taper et à envoyer les commandes de manière manuelle ou connaitre les détails techniques.

La bibliothèque Swing a été conçue pour pallier aux principales insuffisances de la bibliothèque AWT (Abstract Window Toolkit). Cette amélioration a été obtenue en écrivant entièrement cette bibliothèque en Java sans pratiquement faire appel aux services du système d'exploitation. Seuls quelques éléments graphiques (fenêtres et boîtes de dialogue) sont encore en relation avec le système d'exploitation. Pour les autres composants, c'est le code de la bibliothèque Swing qui détermine entièrement leurs aspects et comportements. La bibliothèque Swing contient donc une quantité impressionnante de classes servant à redéfinir les composants graphiques. Il ne faut cependant pas penser que la bibliothèque Swing rend complètement obsolète la bibliothèque AWT. Beaucoup d'éléments de la bibliothèque AWT sont d'ailleurs repris dans la bibliothèque Swing [20].

Une interface graphique conçue avec AWT ou Swing est composée de quatre types d'objets différents [29] :

- Les composants de fenêtre tels que les cadres, les boites de dialogue, les barres de menus...
- Les conteneurs qui sont eux-mêmes des composants pouvant contenir des composants graphiques et également des conteneurs.
- Les composants graphiques qui sont les éléments tels que les boutons, les listes déroulantes, les listes de choix, les champs de texte, les cases à cocher, les boutons d'options…
- Les gestionnaires de présentation qui permettent de définir la disposition des composants à l'intérieur des conteneurs.

L'interface graphique (Figure 3.7) a été divisée en quatre parties ou plus précisément quatre « panel » : le Tableau de bord, le Terminal, les Paramètres et le Contrôle manuel. On peut choisir à quel panel on veut accéder en cliquant sur l'un deux sur la barre de gauche, en plus de ces quatre panels il y a aussi sept options en dessous : Ouvrir un port, Démarrer l'impression, Choisir un fichier, Voir un fichier, Préprocesseur du G-code, Simulateur du G-code, Fermer le port.

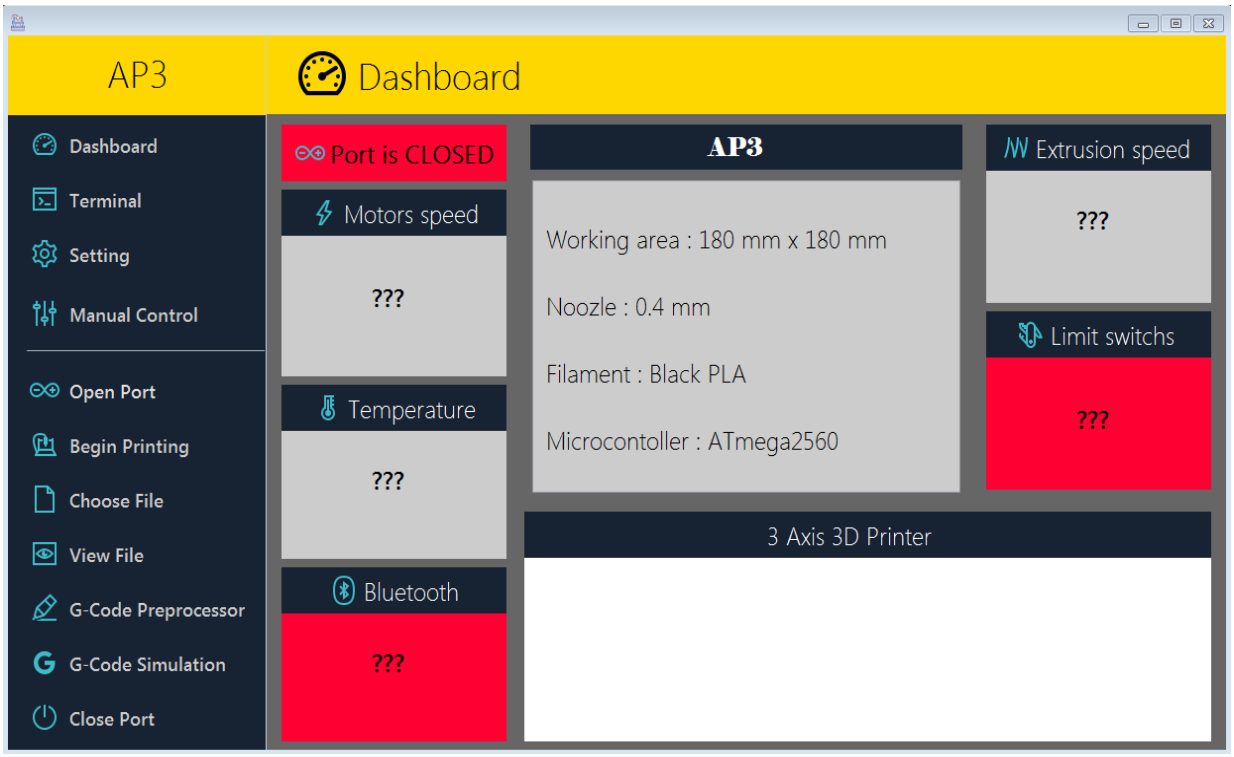

Figure 3.7: Le tableau de bord de GUI

Seul 3 programmes sur 8 implémentés sur PC peuvent être accessible depuis l'interface graphique.

Les programmes qui ne sont pas accessible depuis la GUI sont : G-Code Bezier, G-Code Simulator D, G-Code Preprocessor S, G-Code Preprocessor D, G-Code Futurs Mouvements.

3.4.2. STL Viewer : visualisation des objets en 3D et maillage

Afin de mieux comprendre comment un Slicer tel que Cura (ce que nous utilisons) transforme un fichier STL en G-code, on a écrit ce programme en Processing (mode Java) qui permet de visualiser un fichier STL (Figure 3.8). La visualisation du fichier STL dans la fenêtre de l'interface graphique par la commande view file -> STL, permet une visualisation 3D de l'objet à imprimer en 3D, et de la résolution de son maillage. Ce programme utilise le fichier STL de l'objet 3D créé par le logiciel fusion 3D.

Figure 3.8: Affichage STL

L'organigramme de la Figure 3.9 montre comment les coordonnées des triangles du maillage sont extraites et affichées.

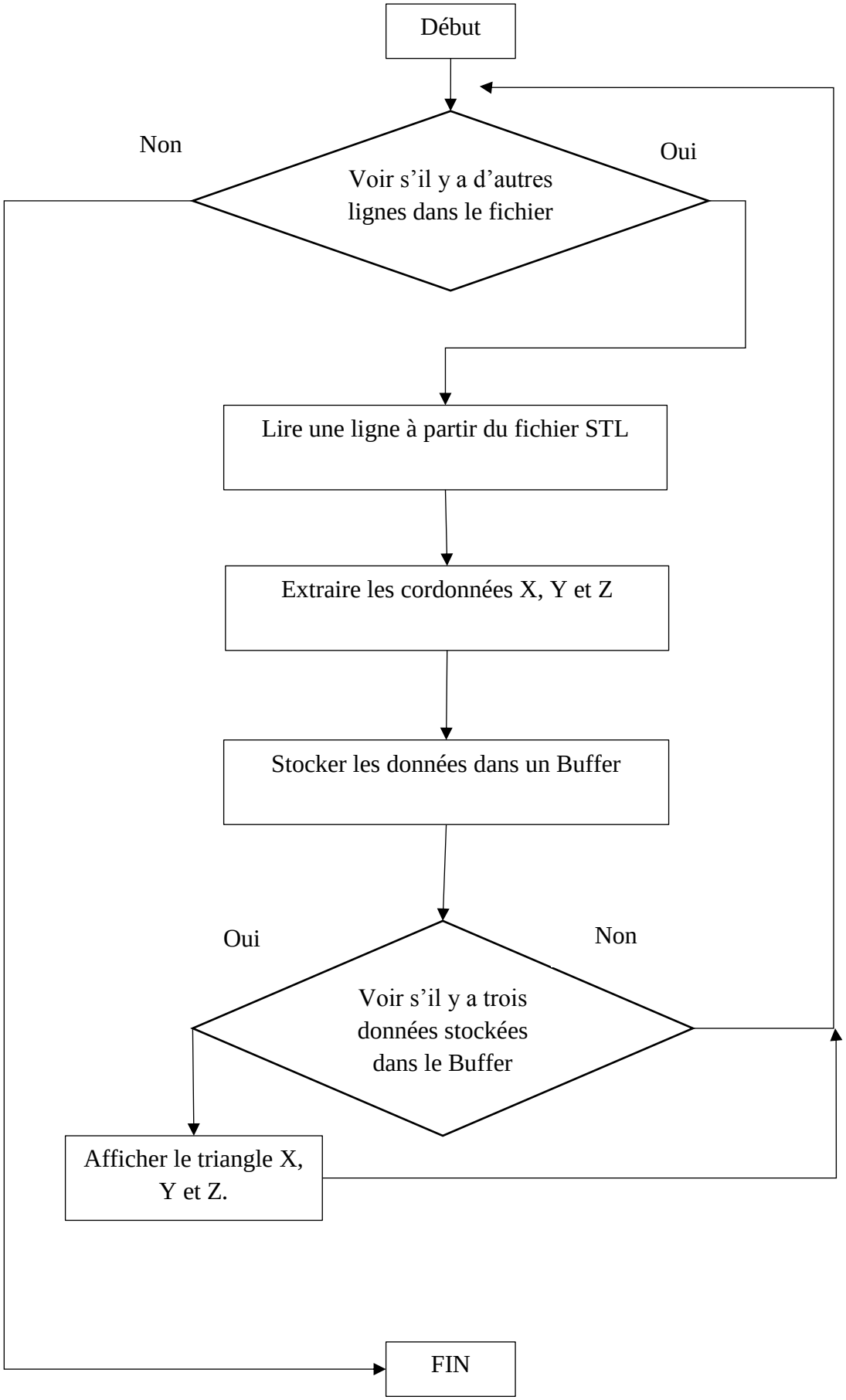

Figure 3.9: Organigramme du programme STL Viewer

3.4.3. G-code Simulator

Le fichier G-code contient les commandes à envoyer aux moteurs de l'imprimante pour le déplacement X, Y, Z et E de la tête d'impression, et de l'extrudeuse, respectivement.

Avant d'imprimer l'objet en 3D, il est nécessaire de visualiser une simulation du processus d'impression ligne par ligne, ou de l'objet fini, afin de palier à des problèmes éventuels.

Nous avions besoin d'un simulateur de G-code propre à notre machine, afin de le customiser et ajouter de nouvelles fonctions. La fonctionnalité principale qu'on a voulu avoir dans notre simulateur de G-code est d'afficher l'objet à imprimer en tenant compte des erreurs provenant du manque de précision de la machine. Pour cela on a écrit quatre programmes en Processing (Mode Java).

1- Le premier programme G-code Simulator S : permet de simuler ligne par ligne sans prendre en compte les erreurs.
2- Le deuxième programme G-code Simulator D : permet de d'afficher directement le résultat de l'impression sans prendre en compte les erreurs.
3- Le troisième programme G-code Simulator Preprocessor S : permet de simuler ligne par ligne en prenant en compte les erreurs.
4- Le quatrième programme G-code Simulator Preprocessor D : permet de d'afficher directement le résultat de l'impression en prenant en compte les erreurs.

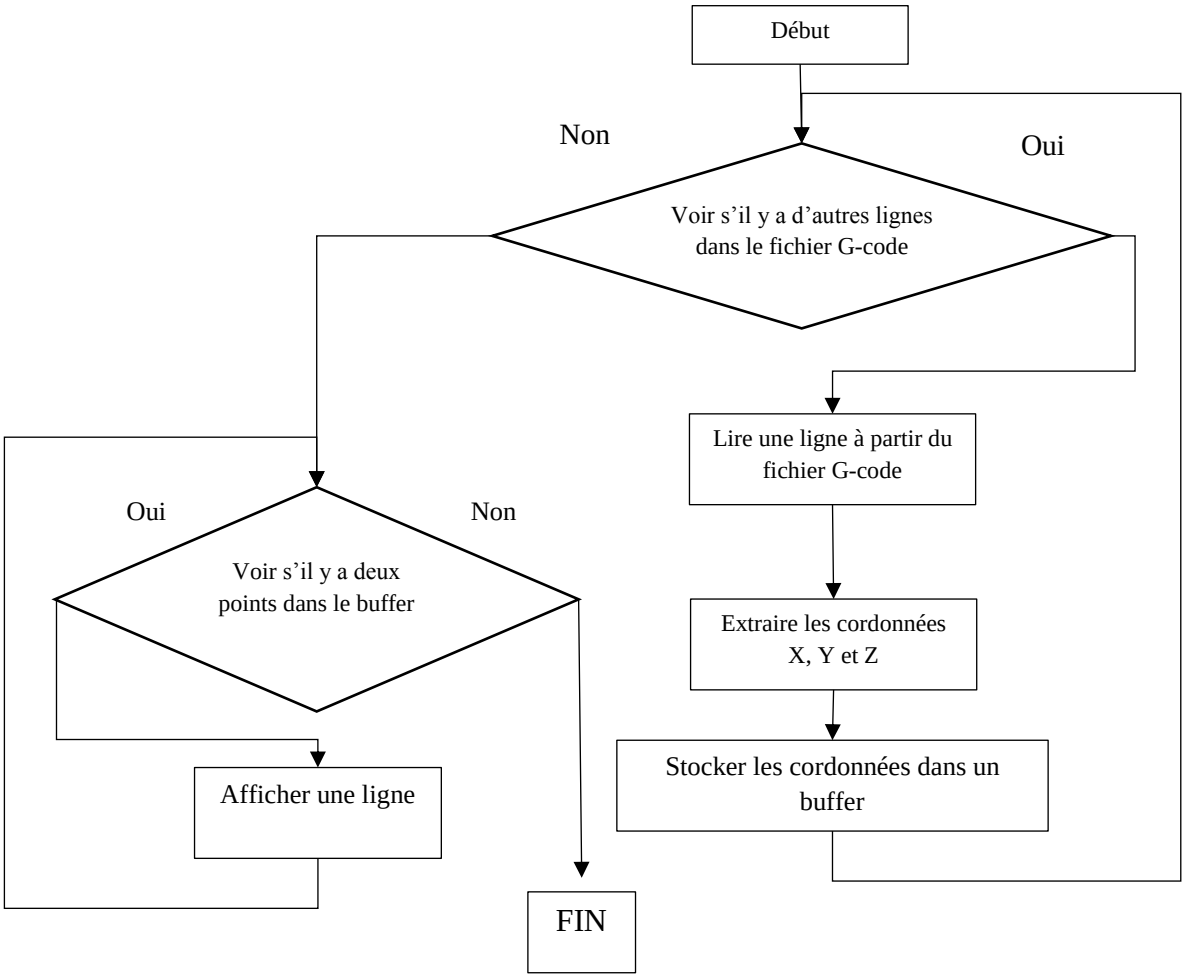

Figure 3.10: Organigramme du programme G-Code Simulator S

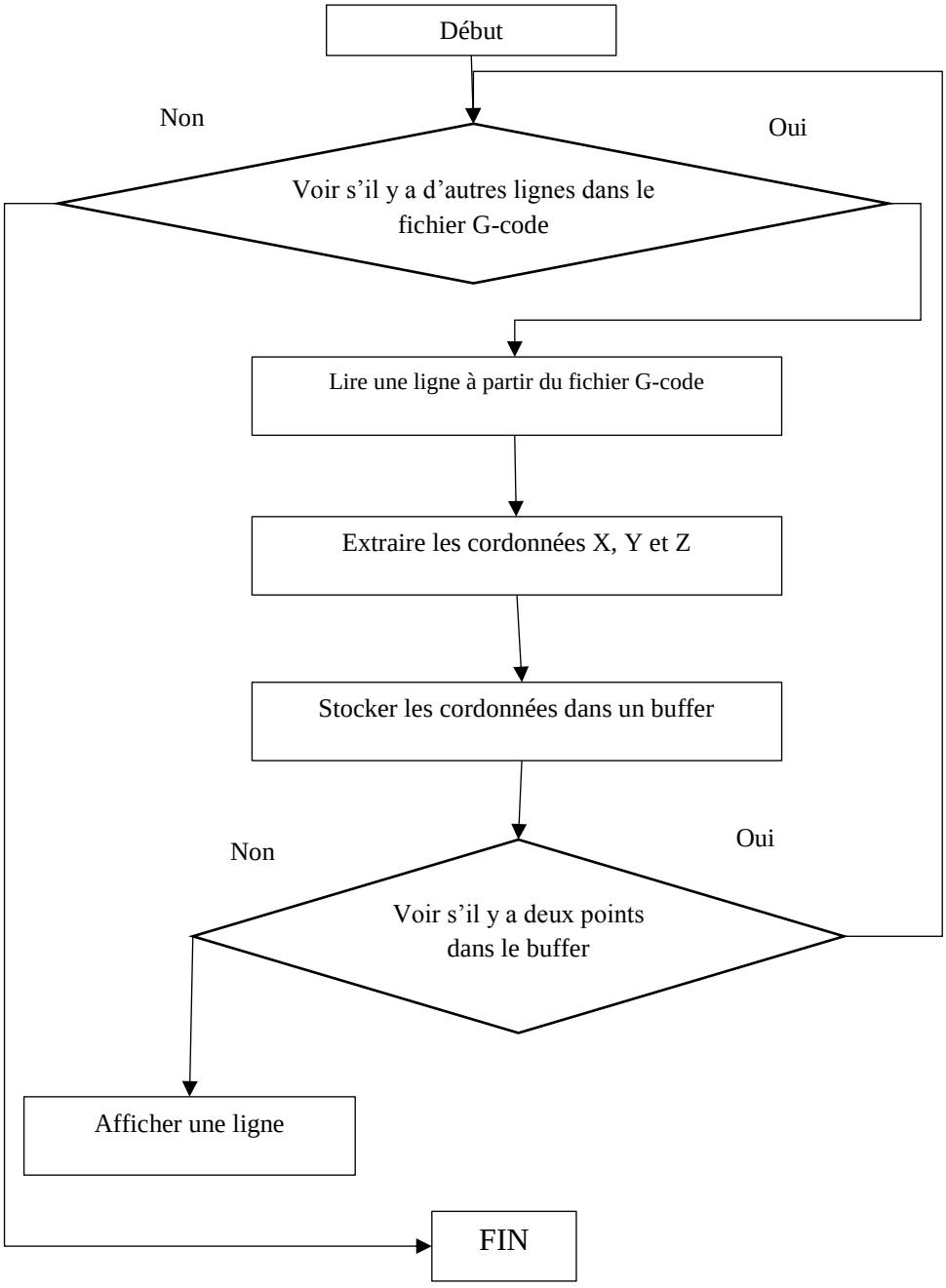

Figure 3.11: Organigramme du programme G-Code Simulator D

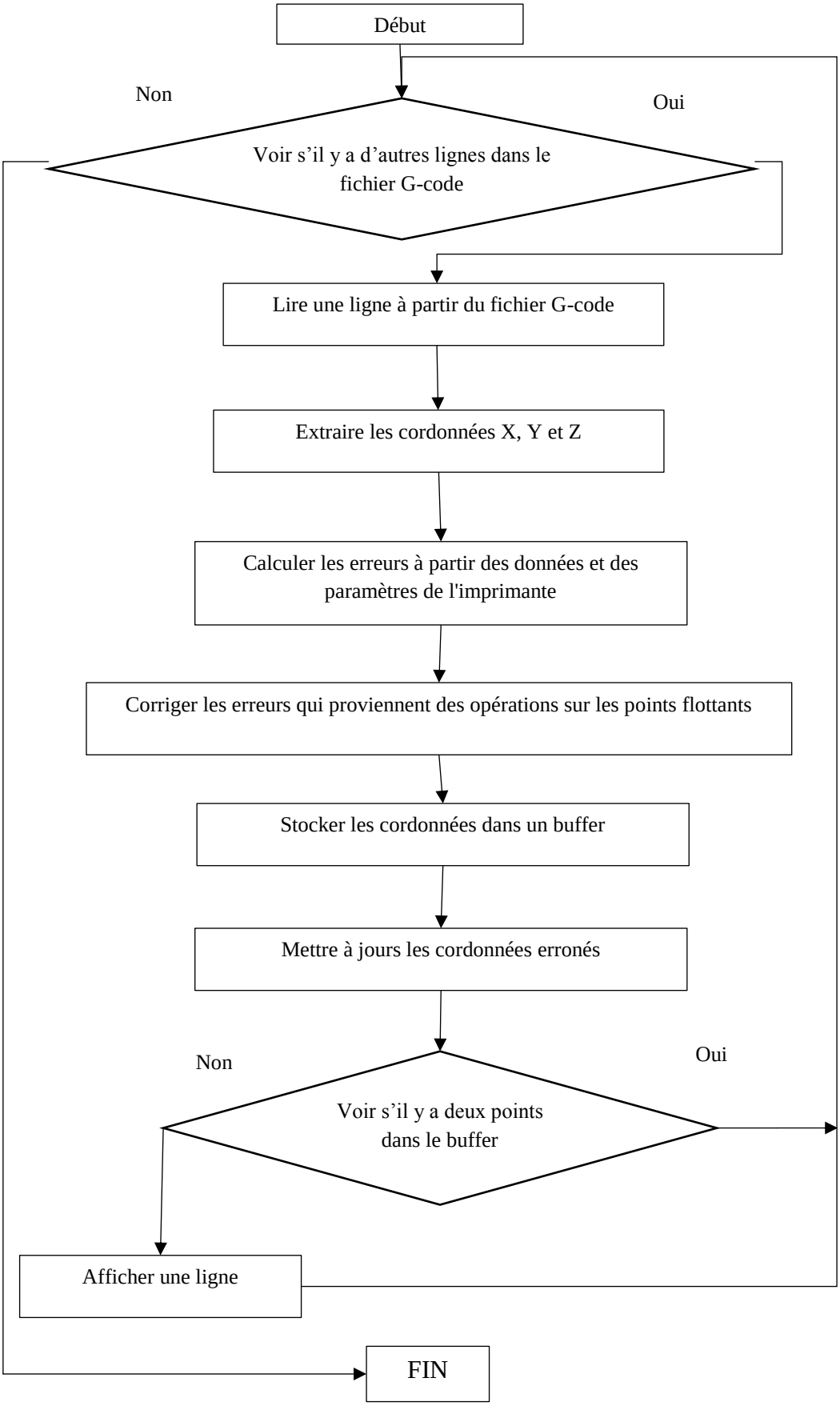

Figure 3.12: Organigramme du programme G-Code Simulator Preprocessor S

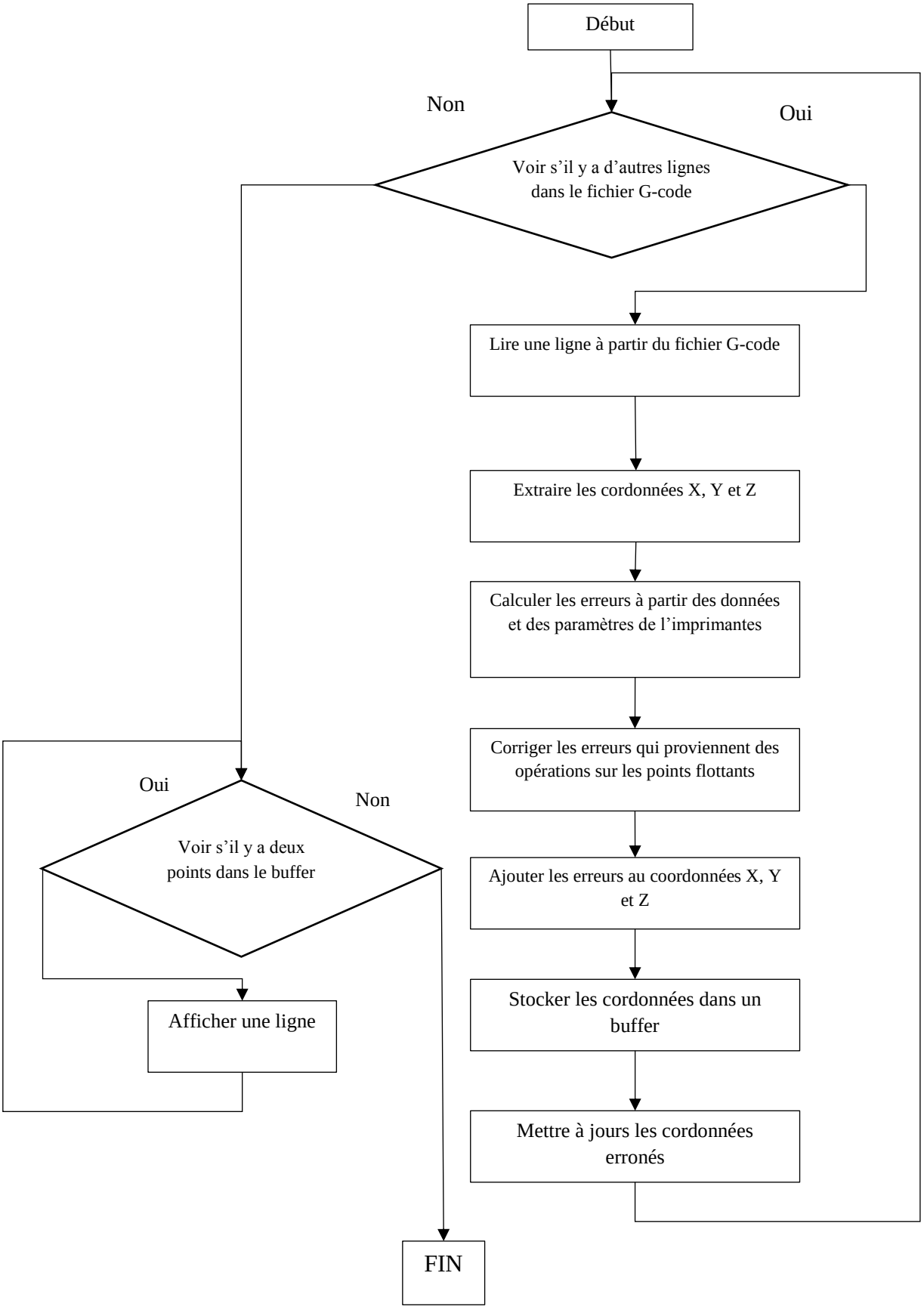

Figure 3.13: Organigramme du programme G-Code Simulator Preprocessor D

3.4.4. G-code Preprocessor

C'est un programme écrit en Processing (mode Java). Il fait des prétraitements sur le fichier G-code avant d'être envoyé à l'imprimante.

Ses principales fonctions sont la correction du cumul d'erreurs afin de prévenir les déplacements erronés des moteurs, la correction les erreurs qui proviennent des opérations sur les nombres représentés en virgule flottante, l'exécution de prétraitements pour optimiser l'impression ou la simulation.

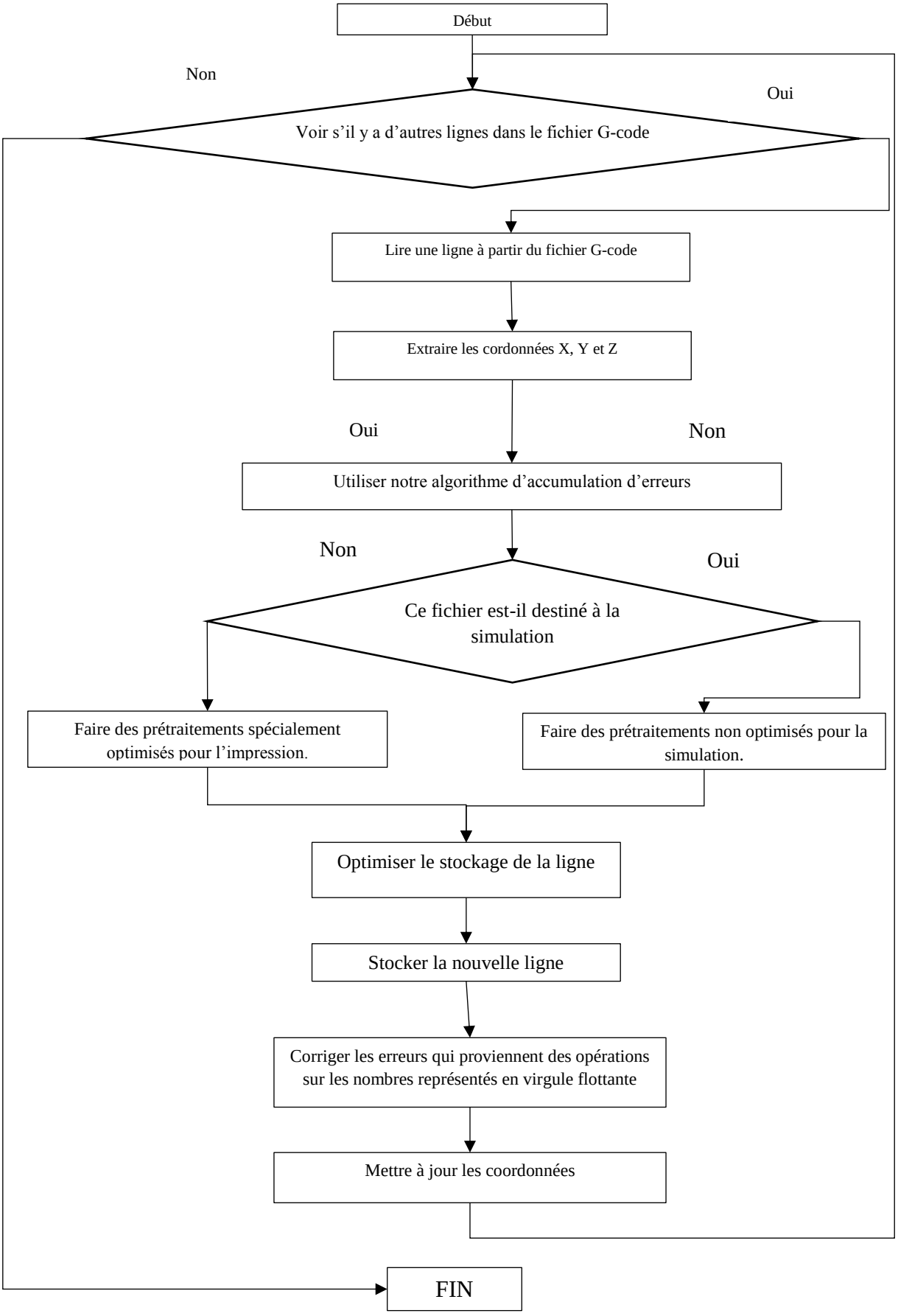

Figure 3.14: Organigramme du programme G-Code Preprocessor

3.4.5. G-code Futurs Mouvements

C'est aussi un programme qu'on a écrit en Processing (mode Java), il a le même concept que le G-code Preprocessor sauf que celui-là applique seulement l'algorithme du « Futurs mouvements ».

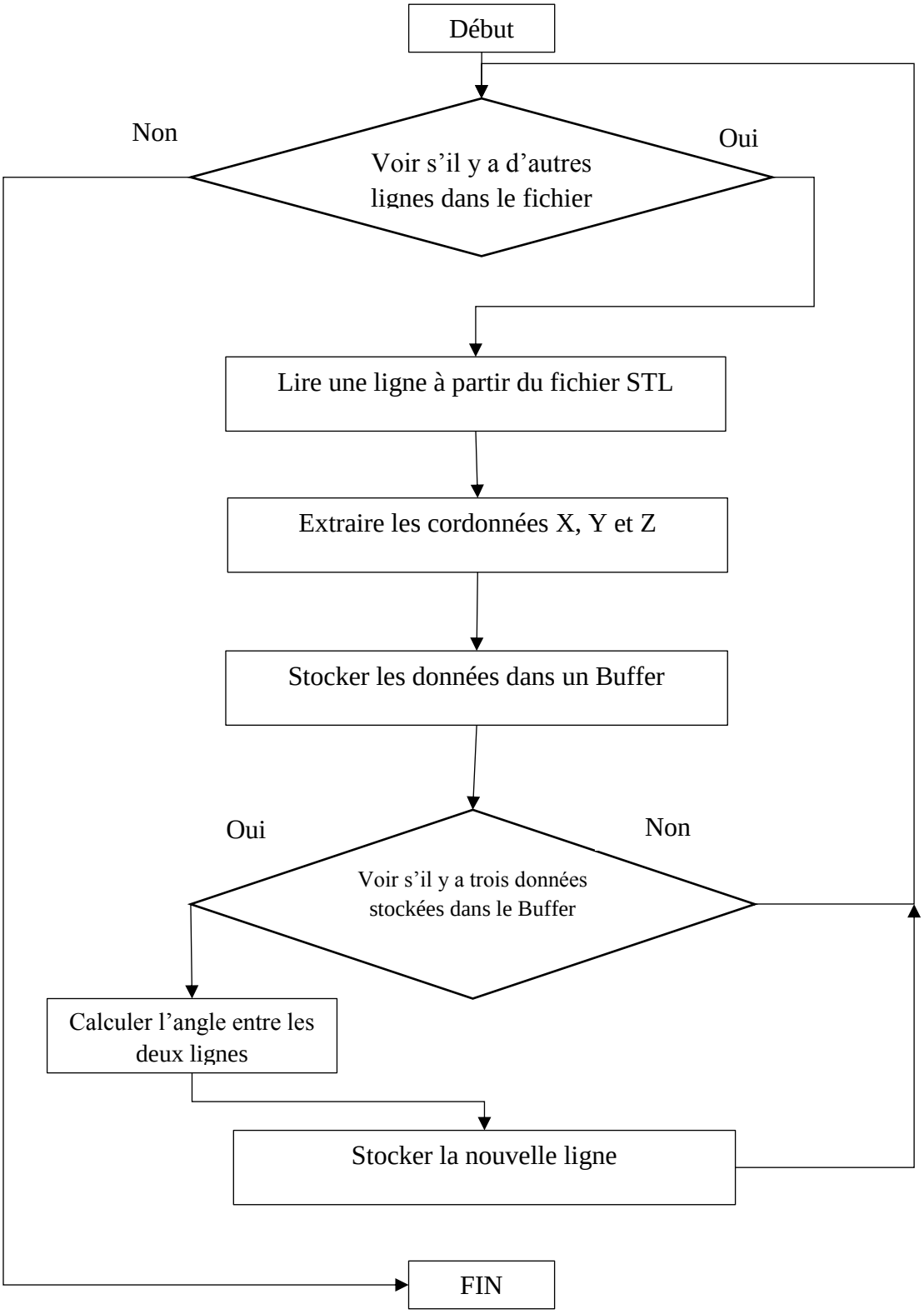

Figure 3.15: Organigramme du programme G-Code Futurs Mouvements

3.4.6. G-code Bézier

On a écrit un programme en Processing (mode Java) sur lequel on a implémenté les courbes de Béziers. Ce dernier modifie le fichier G-code en ajoutant de nouvelles lignes.

Le « **G-Code Bézier** », modifie le fichier G-code afin de diminuer les vibrations. Cependant l'utilisation de cette méthode présente un inconvénient

On remarque que plus la distance est courte, plus il est difficile de respecter les conditions du temps réel, alors la solution est d'implémenter une file de données.

Cependant, l'utilisation de l'algorithme des courbes de Bézier, ne garantit pas toujours un système en temps réel, car il induit une création de nouvelles lignes très petites. Ces dernières peuvent avoir un temps d'exécution inferieur au temps de réception en plus du temps de traitement.

La solution est d'implémenter l'algorithme au niveau du microcontrôleur, mais cela nécessite aussi l'implémentation d'une file de données. La taille de la file doit être au minimum égale ou supérieure aux degrés de la courbe de Bézier.

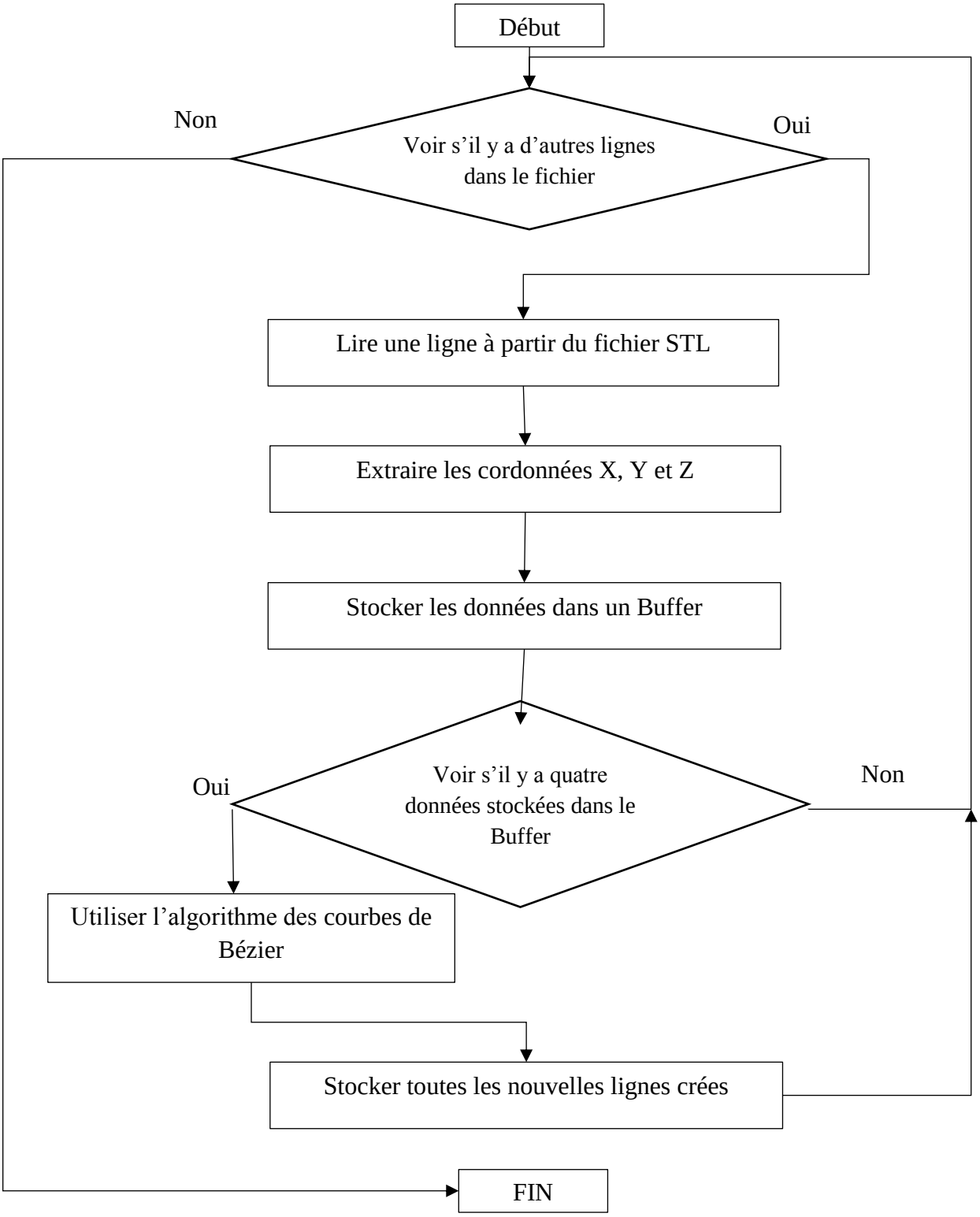

Figure 3.16: Organigramme du programme G-Code Bezier

3.5. Programmes qui tournent sur le microcontrôleur

3.5.1. Firmware

On a écrit un Firmware en C++ pour notre imprimante 3D basé sur un ATmega 2560 pour contrôler toute la machine grâce à des algorithmes que nous avons créés. C'est un mélange entre la programmation baremetal et les systèmes d'exploitation RTOS (real time operating system).

Le Firmware possède un ordonnanceur qui s'assure de l'exécution des tâches, ces dernières sont codées le plus efficacement possible. Pour cela de nombreuses caractéristiques des systèmes d'exploitation ont été abandonnées au profit de la vitesse d'exécution.

Ce Firmware a été conçu spécialement pour notre imprimante 3D à fin d'être le plus performant possible. Il a pour but de commander les différents moteurs, extrudeuse et sa température. Voici quelques-unes de ces caractéristiques:

- Affichage LCD.
- Contrôle de température avec PID.
- Impulsions des moteurs très rapide allant bien plus que 40KHz.
- Contacteurs fin de course.
- Algorithme de futurs mouvements.
- Accélération non linéaire.
- Réglage du niveau du plateau d'impression possible avec un capteur de proximité.

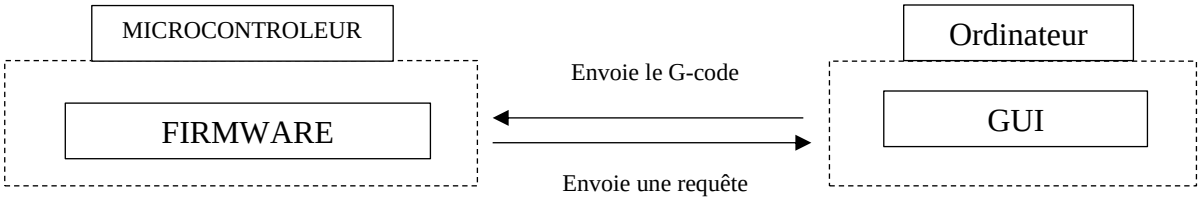

Figure 3.17: Schéma synoptique de la communication série

3.5.2. Optimisation du code et philosophie du Firmware

L'optimisation du code est la pratique qui consiste à modifier un code correct pour le rendre plus efficace. Il peut s'agir de la taille ou de la vitesse d'un programme. Pour cela on a posé quelques règles qu'il faut respecter lors de la conception du Firmware :

- Eviter les nombres en virgule flottante quand c'est possible.
- Passer en premier lieu la vitesse du code sur la taille du code, car on a beaucoup d'espace mémoire et ça n'a jamais été un problème dans ce projet.
- Passer par référence au lieu de passer par valeur.
- Utiliser les calculs à la compilation.
- Ne pas refaire plusieurs fois la même opération, créer une variable qui stocke le résultat de la première opération pour l'utiliser ensuite.
- Look up tables.
- Pas d'utilisation de librairie externe, car elles sont trop lentes en générale (du point de vu de la vitesse d'exécution). Leur utilisation saccagerait tous les autres efforts fournis.
- Eviter les variables plus de 8-bits quand c'est possible.

- Pas de fonctions de la libraire Arduino.

En ce qui concerne la philosophie du Firmware elle se résume par les points suivants:

- Réduire au maximum les traitements dans le microcontrôleur d'où l'utilisation des techniques de prétraitement qu'on a expliqué plus tôt dans ce chapitre d'où la création de programme tel que G-code Preprocessor.
- Implémenter dans Firmware seulement les fonctionnalités qui sont utiles et faisable pour notre l'imprimante 3D.
- Implémenter une partie des algorithmes sur ordinateur afin d'alléger au maximum la charge sur le microcontrôleur.
- Modifier le G-code, ajouter ce qui est utile et supprimer ce qui le n'est pas.
- Utiliser une combinaison de la programmation baremetal et du RTOS. L'utilisation du RTOS tout seul est une perte de performance inutile surtout dans un microcontrôleur 8-bits. L'utilisation de la programmation baremetal, n'est pas très pratique si on veut ajouter de nouvelle tâche ou fonctionnalités sur la machine, mais elle est très rapide. Et si une fonction ne marche pas sur le baremetal à cause du temps réel, il sera alors impossible de l'implémenter sur un RTOS surtout avec tout le « switching context » qui va avec. Une combinaison entre les deux est donc la meilleure décision.
- Utiliser la programmation modulaire.

3.5.3. Principe de base d'un Firmware d'imprimante 3D

Les imprimantes 3D, sont des machines à commandes numériques (CNC). Bien que le fonctionnement de l'imprimante 3D soit bien connu, il n'existe pas vraiment de documents expliquant le fonctionnement du Firmware mis à part les commentaires peu éducatifs qu'on trouve sur les sources codes. D'ailleurs c'est un peu le cas aussi du logiciel embarqué, Il n'y a pas beaucoup de livres sur le Firmware car ce n'est pas une discipline standard dont on peut parler de façon générique [30].

L'imprimante reçoit les commandes, elle les traite et en ensuite les exécute. Elle doit en général pouvoir :

- Recevoir une commande à partir d'un ordinateur, ou d'une carte SD ou de n'importe quelles autres interfaces.
- Pouvoir traiter et exécuter la commande à temps, la machine ne doit pas attendre trop longtemps. Cela impacterait drastiquement sur la qualité des produits.
- Contrôler la vitesse et la direction des moteurs des axes X, Y, Z, E afin de suivre une trajectoire bien précise.
- Contrôler la température d'une manière assez précise et sure.
- Afficher quelques données importantes telles que la température.
- Pouvoir contrôler les paramètres principaux de l'imprimante comme la vitesse, l'accélération, la température, etc...
- Avoir et utiliser quelques capteurs de fins de courses / capteurs de proximités pour la sécurité et le bon fonctionnement de l'imprimante.

Donc on peut déduire à partir de la liste qu'on vient de présenter qu'il y a certains points très importants qui doivent être pris en considérations lors de la conception du Firmware :

- Contrôle de la température.
- Précision de la machine.
- Réduction des vibrations de la machine.
- Respect du temps réel.
- Programmation compréhensible et facile à gérer.

3.5.4. Algorithme de Bresenham : mouvements des moteurs

Avant d'entrer dans plus détails dans ce chapitre, on doit tout d'abord expliquer comment les coordonnées X, Y et Z sont transformées en mouvements de moteurs.

L'algorithme de Bresenham (Figure 3.18) est l'une des méthodes les plus efficaces pour tracer une ligne dans un environnement basé sur des pixels, mais ce n'est pas la seule raison pour laquelle il est utile pour les calculs de recherche de chemin. L'algorithme de Bresenham est également attrayant car, contrairement à certains autres algorithmes de dessin au trait, il ne dessinera jamais deux pixels adjacents le long de l'axe le plus court d'une ligne [31].

La plupart des algorithmes de ligne nécessitent des opérations de division en virgule flottante ayant de faibles performances. Cependant, la pixellisation qui a été proposée par Bresenham était un algorithme à très grande vitesse qui pixellise la ligne droite ou le cercle ou l'ellipse uniquement avec des opérations d'addition / soustraction d'entiers purs [32].

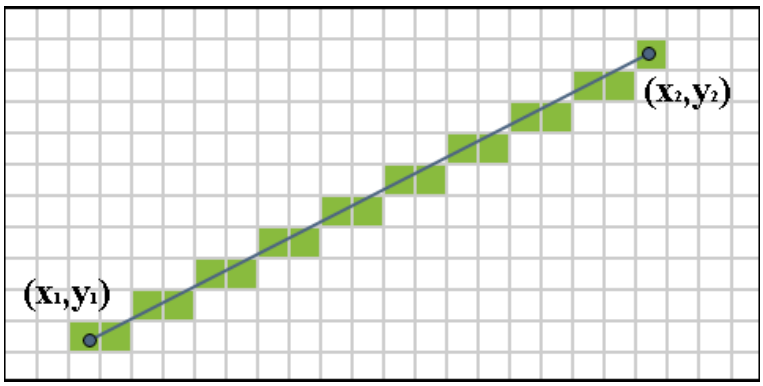

Figure 3.18: Illustration de l'algorithme de Bresenham.

Les principes fondamentaux de l'algorithme de Bresenham sont des « termes d'erreur » accumulés. C'est-à-dire que l'écran d'ordinateur correspond à un grand tableau bidimensionnel de carrés colorés. Un acte consistant à tracer une ligne droite (ou une autre figure) à ce stade c'est un processus d'approximation d'un système de coordonnées continu de nombres réels en un système de coordonnées discontinues. Par conséquent, un point fournit généralement une erreur entre les systèmes de coordonnées discontinus imprimés dans le système de coordonnées d'origine de la ligne et de l'écran, et le concept clé de l'algorithme de Bresenham est de sélectionner un point qui minimise le mieux cette erreur en traçant les lignes droites (ou une autre figure) [32].

Notre imprimante 3D utilise une version modifiée de l'algorithme de Bresenham, car comme on a cité, l'algorithme de Bresenham est fait pour les pixels et non pour les moteurs pas à pas.

3.5.5. Architecture du Firmware

Le Firmware est inspiré des systèmes d'exploitation en temps réel, il possède un ordonnanceur qui doit assurer l'exécution de certaines tâches. Les autres tâches sont entraînées par

des interruptions. Toutes les tâches utilisent différents modules du Firmware. Ces derniers utilisent les ressources disponibles sur le microcontrôleur ou des ressources externes.

Notre choix d'aller vers une architecture qui associe la programmation baremetal et les RTOS a été encouragé par les points suivants :

- Mis en point d'une architecture sur mesure pour notre imprimante.
- Avoir des performances très satisfaisantes.
- Facilité de gestions des tâches et des modules.
- Possibilité de restreindre certaines fonctionnalités a des tâches spécifiques, ce qui permet un débogage plus facile.
- Détecter et éviter des bugs avants même la mise en marche de la machine, pendant le développement du code.
- Gestion du Firmware plus facile et plus agréable si le Firmware est correctement mis en place et proprement programmé.

3.5.5.1. Programmation multitâche

L'exécution de plusieurs processus en parallèle est rendue possible de nos jours par la programmation multitâche au sein du système d'exploitation. La programmation multitâche que l'on appelle communément le multithreading, c'est-à-dire la programmation multithread pour gérer des threads en parallèle, est un domaine passionnant mais très complexe à mettre au point. Plusieurs exécutions parallèles au cœur de votre application devront partager des informations, s'attendre, et échanger des données [33].

Un thread est une unité d'exécution, un morceau de programme ou même la totalité du programme si l'application ne propose pas le multithread. Le multithreading signifie qu'à l'intérieur d'une même application, plusieurs tâches peuvent s'effectuer en pseudo parallèle [34].

Le Firmware qu'on a conçu utilise le principe du multithreading. Il y a plusieurs tâches qui tournent dans le programme, elles ont toutes un rôle et une priorité. Certaines des tâches peuvent s'exécuter en pseudo parallèle.

3.5.5.2. Ressources hardware interne du Firmware

Nous allons maintenant présenter les ressources hardware interne qu'utilise le Firmware, c'est-à-dire les ressources du microcontrôleur utilisées.

3.5.5.2.1. Timers

Un timer est un périphérique matériel permettant de mesurer des durées. Son rôle est de permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer. Le Firmware utilise deux timers, le premier c'est pour contrôler la génération des signaux pour les drivers. Le deuxième timer est utilisé pour le contrôle des tâches « d'arrière-plan ».

3.5.5.2.2. UART

Un UART, pour Universal Asynchronous Receiver Transmitter, est un émetteur-récepteur asynchrone universel. L'UART est utilisé pour recevoir et envoyer des données à l'ordinateur.

3.5.5.2.3. ADC

Un convertisseur analogique-numérique est un montage électronique dont la fonction est de traduire une grandeur analogique en une valeur numérique (codée sur plusieurs bits), proportionnelle au rapport entre la grandeur analogique d'entrée et la valeur maximale du signal. L'ADC est utilisé pour coder numériquement la valeur du signal de la résistance thermique.

3.5.5.3. Ressources hardware externe du Firmware

Les ressources hardware externe qu'utilise le Firmware, sont les ressources qui ne sont pas à l'intérieur du microcontrôleur.

3.5.5.3.1. Moteurs

Le Firmware doit contrôler les moteurs pas à pas, afin de suivre une trajectoire bien précise grâce à l'algorithme de Bresenham.

3.5.5.3.2. Capteurs fin de courses

Le Firmware doit prendre en compte les capteurs de fin de courses afin de détecter que la machine est arrivée à la limite (working area), évitant ainsi un dommage matériel.

3.5.5.3.3. LCD

Le Firmware affiche certaines données importantes pour l'utilisateur sur un LCD, telle que la température de l'extrudeuse par exemple.

3.5.5.3.4. Résistance thermiques

Afin de connaitre et de réguler la température de l'extrudeuse, il faut tout d'abord utiliser le circuit de la résistance thermique pour acquérir un signal. Ce dernier est transformé par l'ADC et traduit en température dans le Firmware. Par la suite le Firmware utilise le PID pour la régulation de la température.

3.5.5.3.5. Extrudeuse

Pour l'extrusion, le filament doit être en état de fusion et en même temps poussé par une pression. La pression est exercée par le moteur pas à pas de l'extrudeuse. Le Firmware doit s'assurer que la température de l'extrudeuse soit constante.

3.5.5.3.6. Capteur de proximité

Le capteur de proximité est utilisé pour la régulation du niveau de la plaque d'impression. Le Firmware possède une fonction pour corriger ce problème.

3.5.5.4. Ressources software du Firmware

Les ressources software du Firmware sont celles qui ont été virtuellement implémentées dans le code, donc dans la mémoire du microcontrôleur (mémoire RAM et Flash).

3.5.5.4.1. Drapeau d'états

La principale fonction du drapeau d'états est de contenir la tâche qui doit être exécutée. La tâche contenue est en cours d'exécution ou sur le point d'être exécutée. L'ordonnanceur utilise le drapeau d'états pour savoir quelle tâche doit être exécutée. L'accès au drapeau d'états n'est pas restreint à l'ordonnanceur. Contrairement à l'ordonnanceur qui ne peut que lire, certains modules peuvent eux lire et modifier le drapeau d'états. Les tâches quant à elles peuvent le modifier sans pouvoir lire son contenu.

3.5.5.4.2. Buffer de réception

Le buffer de réception est un buffer circulaire de tableau. Le buffer circulaire est une structure de données utilisant un buffer de taille fixe dont le début et la fin sont considérés comme connectés.

Le buffer de réception a un accès exclusif à l'UART (la partie réception) du microcontrôleur. Il peut alors lire la donnée reçue par le port série et la stocker dans l'une des cases d'un des tableaux du buffer circulaire.

L'écriture vers le buffer de réception est sur une case de tableau, contrairement à la lecture qui est sur la totalité du tableau. Autrement dit, la lecture du buffer de réception est sur toutes les cases du tableau.

Le buffer de réception doit gérer la taille du buffer circulaire, pour éviter un over-flow ou une lecture d'un tableau vide. Il doit aussi savoir détecter le début est la fin des commandes, un tableau ne doit pas contenir plus d'une commande.

Cette ressource est seulement accessible par le module récepteur série.

3.5.5.4.3. File de mouvements

La file de mouvements est aussi un buffer circulaire de mouvement. Nous parlerons plus tard de ce que c'est qu'un mouvement.

La file de mouvements, continent des mouvements qui sont prêts pour être exécutés. La file peut être remplie uniquement par la tâche « traitement ». Elle peut être lue par la tâche « démarrage des moteurs ». Certains modules comme le module moteur et le module capteurs de fin de courses peuvent aussi accéder à la file de mouvement.

Cependant l'accès de la tâche « démarrage des moteurs » diffère de l'accès du module capteurs de fin de courses à la file. Le premier cas accède à la file de mouvements en elle-même, quant au second cas, il accède à l'un des mouvements de la file, plus précisément le mouvement qui est en train d'être exécuté par la machine.

La tâche « traitement » et le module moteur peuvent quant à eux accéder à la file de mouvement elle-même et les mouvements en questions. Une fois encore, la tâche « traitement » ne peut pas accéder à un mouvement qui est en train de s'exécuter. Le module moteur quant à lui, peut accéder uniquement à un mouvement qui fut remplis par la tâche « traitement » et donc prêt à être exécuter.

Un accès simultané d'un mouvement de la file par la tâche « traitement » et le module moteur ou le module capteurs de fin de courses est interdit. Cela est mis en place afin d'éviter une corruption de donnés.

Ces règles qu'on a émises, ont pour but la synchronisation et l'assurance d'un bon fonctionnement de la machine. C'est pour cela qu'on a créé la file de mouvement, une sorte de vigile ou de garde qui contrôle l'accès aux mouvements. Lire et écrire en même temps par deux sources différentes est donc logiquement interdit.

Les mouvements de la file peuvent être accédés de deux manières différentes. La première c'est d'accéder au mouvement qui est en train de s'exécuter. La deuxième d'accéder au prochain mouvement vide, pour le remplir.

La file facilite l'utilisation des mouvements pour les modules et les tâches, on ne se souciera donc pas quel mouvement est vide pour écrire, ou quel mouvement a été rempli et disponible pour la lecture. La file de mouvements renferme certaines fonctions qui ont été optimisées. On sait déjà lors de la conception du Firmware, quel module a besoin de quelles ressources à tel moment. On peut donc fixer quelques fonctions de telle manière qu'à chaque fois qu'on demande une certaine ressource, quel index dans la file de mouvement on aurait a utilisé afin de remplir la requête. Malheureusement sur certaines ressources on ne peut pas appliquer ce concept, donc il est du devoir du programmeur de spécifier l'un des deux index de la file de mouvement. Le premier index sert à spécifier un mouvement vide, le deuxième index à spécifier le mouvement qui est en train de s'exécuter.

La file de mouvements peut aussi transmettre des informations entre les mouvements, cela permet une sécurité et une encapsulation totale. On reparlera de cette file de mouvement tout au long de ce mémoire.

3.5.5.4.4. Look up table

Il s'agit d'une structure de données stockée en mémoire et employée pour remplacer un calcul par une opération plus simple de consultation. Le gain de vitesse peut être significatif, car rechercher une valeur en mémoire est souvent plus rapide qu'effectuer un calcul important.

Le look up table est utilisé pour stocker les valeurs de la température et de la résistance équivalente. Il est uniquement accédé par le module de recherche binaire par intervalle.

3.5.5.4.5. Mouvement

Un mouvement est une combinaison de structure de données et de fonctions spéciales. Il possède plusieurs paramètres tels que la vitesse du mouvement, l'accélération, le nombre de cas de chaque axe pour n'en citer que quelques-uns.

3.5.6. Modules du Firmware

La programmation modulaire reprend l'idée de fabriquer un produit (le programme) à partir de composants (les modules). Elle décompose une grosse application en modules, groupes de fonctions, de méthodes et de traitement, pour pouvoir les développer et les améliorer indépendamment, puis les réutiliser dans d'autres applications.

Cette méthode de regroupement permet de réaliser une encapsulation comparable par certains aspects à celle de la programmation orienté objet, et permet l'organisation du code source en unités de travail logiques. Les modules définissent également des espaces de noms utiles lors de leur utilisation.

A présent on va vous présenter d'une manière générale les modules les plus importants du Firmware.

3.5.6.1. Moteurs

Le module Moteurs, a plusieurs fonctions :

- Contrôle des moteurs pas à pas de l'imprimante grâce à l'un des algorithmes de mouvements choisit en fonction du type de mouvement.
- Initialisation des moteurs et des timers.
- Implémentation de l'accélération et de la décélération.
- Implémentation de l'accélération non constante.
- Implémentation de l'algorithme du sélectionneur de mouvement.
- Le module moteur a un Mutex qui protège les ressources « moteurs ».

Un Mutex (en anglais : Mutual exclusion, *Exclusion mutuelle*) est une primitive de synchronisation utilisée pour éviter que des ressources partagées d'un système ne soient utilisées en même temps. Son implémentation varie selon les systèmes (masquage des interruptions, lecture/écriture en un cycle, etc.). Ces algorithmes permettent de réguler l'accès aux données, en autorisant par exemple qu'une seule routine y accède à la fois.

Le Mutex du module moteur permet d'assurer que les moteurs pas à pas ne soient pas utiliser par plusieurs modules ou tâches à la fois. Le Mutex est en mode occupé si une commande de type mouvement est en cours d'exécution. Le Mutex passe en mode attente (ou libre) si l'exécution de la commande est terminée.

Le module moteur, implémente aussi une tâche très importante, celle qui a la deuxième plus grande priorité des toute les tâches, c'est la « routine des moteurs ».

3.5.6.2. Récepteur série

Ce module contrôle le processus de la réception des données via le port UART du microcontrôleur. Il a un accès exclusif vers la ressource buffer de réception. Il a aussi un Mutex protégeant le buffer de réception.

Le Mutex permet de traquer l'état du buffer s'il est utilisé par quelqu'un ou non. Dans notre cas par exemple c'est seulement l'ordinateur qui peut envoyer des données vers le buffer de réception. Cela permet de savoir s'il y a des données qui sont en train d'arriver vers le buffer de réception, ou s'il est libre pour en recevoir. Cela permet d'éviter une corruption de données.

Le Mutex est en mode occupé si le récepteur série détecte un caractère qui est reçu par le port UART. Le Mutex est en mode attente (ou libre) si la réception des données du port UART est terminée.

Le récepteur série implémente une tâche très importante, elle a la troisième plus grande priorité, c'est la tâche « réception de données série ».

3.5.6.3. Emetteur série

La principale tâche de ce module est d'utiliser l'UART du microcontrôleur pour envoyer des données. Pour cela il doit faire en sorte de gérer les données qui sont destinées pour l'envoi, en les stockant et les envoyant une par une.

Pour cela il doit implémenter une tâche, elle aussi la troisième plus grande priorité, c'est la tâche « transfert de données série ».

Les données à envoyer peuvent être des données destinées au programmeur pour le debug ou à l'utilisateur. Elles peuvent être aussi des commandes destinées à l'ordinateur (interface graphique) pour en faire quelque chose.

3.5.6.4. Emetteur de requêtes

On sait que les microcontrôleurs ont une mémoire flash limitée et possèdent un nombre limité d'écritures. On ne peut pas donc flasher un G-code d'un objet à imprimer dans la mémoire flash d'un microcontrôleur. Le fichier G-code doit rester sur l'ordinateur, le microcontrôleur doit ensuite lire le fichier ligne par ligne. Chaque commande reçue doit être traitée puis exécutée.

Une requête dans notre imprimante 3D est une demande faite de la part du microcontrôleur à l'ordinateur pour demander une commande. C'est un code envoyé à l'ordinateur. Quand l'ordinateur reçoit ce code, il envoie une nouvelle commande si disponible. En cas de problème, le microcontrôleur envoie un message d'erreur à l'ordinateur qui sera affiché.

Bien sûr quand on parle de l'ordinateur, on parle de l'interface graphique qu'on a créée et qu'on a expliquée dans le chapitre 3.

Le module Emetteur de requêtes est responsable de cette communication, il possède aussi un Mutex. Ce dernier permet d'assurer que si une requête a été déjà faite, une autre est interdite. Le Mutex est en mode occupé lors d'envoi d'une requête, il sera ensuite libéré dès que la commande est compétemment arrivée vers le buffer de réception.

Tout cela permet une communication viable entre le microcontrôleur et l'ordinateur en évitant ainsi une corruption de données dues à la réception de deux commandes consécutives.

3.5.6.5. Régulateur de température

Le module régulateur de température a un objectif très direct, réguler la température de l'extrudeuse. Pour cela il doit :

- Lire la tension de la résistance thermique en utilisant l'ADC du microcontrôleur.
- Utiliser un contrôleur PID pour générer un signal PWM grâce à un Timer.

Le régulateur de température utilise la tension pour réguler l'extrudeuse et non la température, car dans le programme de l'interface graphique, on a déjà fait la conversion tension température. Il sait alors à quelle tension doit être la résistance thermique. Cela permet un gain de temps énorme.

Le module implémente deux tâches qui peuvent être exécutées une après une dans un ordre bien précis :

- La première tâche s'occupe de la lecture de l'ADC, de la régulation PID et de la génération du signal PWM.
- La deuxième tâche doit convertir la tension de la résistance thermique en résistance à partir de l'équation du diviseur de tension.

Nous reparlerons plus en détails de ces dernières quand on abordera les tâches du Firmware.

3.5.6.6. Recherche binaire par intervalle

Ce module implémente une fonction qui reçoit en entrée une valeur de résistance puis à l'aide du look up table donne en sortie la température correspondante, pour cela il doit appliquer un algorithme qu'on a conçu, la « Recherche Binaire Par Intervalle » (RBPI).

L'algorithme RBPI est inspiré du « Binary search algorithm » (ou en français la méthode dichotomique) d'où son nom. Pour comprendre la RBPI on doit d'abord expliquer la méthode dichotomique.

La recherche par dichotomie ne s'applique que sur les tableaux déjà triés. Pour rechercher une valeur dans un tableau avec une recherche séquentielle, il est inutile de balayer tout le tableau : il suffit de s'arrêter dès que la valeur de l'élément du tableau devient supérieure à ce qu'on recherche, d'où une probable complexité moyenne plus basse. Mais il reste une solution plus efficace. La dichotomie consiste à diviser par deux l'intervalle de recherche tant que l'élément recherché n'est pas trouvé [35]. A chaque fois qu'on divise l'intervalle, une de ces bornes (inférieure ou supérieure) change. Par la suite on regarde :

- Si la valeur recherchée est inférieure au milieu de l'intervalle, alors le milieu de l'intervalle devient la borne supérieure.
- Si la valeur recherchée est supérieure au milieu de l'intervalle, alors le milieu de l'intervalle devient la borne inférieure.

L'algorithme RBPI ne recherche pas spécifiquement une valeur, s'il la trouve tant mieux, si la valeur en question n'existe pas dans le tableau et ce qui est majoritairement le cas pour le look up table de la température. Il nous faut alors avoir un intervalle, on peut ensuite faire une interpolation pour avoir des résultats plus précis (On n'a pas implémenté l'interpolation car ça ne sert à rien d'avoir la température exacte). La différence entre la recherche par dichotomie et le RBPI est que l'une recherche une valeur dans un tableau seulement, et que l'autre recherche une valeur et s'il ne la retrouve pas donne son intervalle. Bien évidement on a soigneusement mis des conditions que si la valeur est hors des bornes du look up table, elle ne sera pas prise en compte.

3.5.6.7. Contrôleur du LCD

Au début nous avons essayé d'utiliser la librairie Liquid Crystal pour gagner du temps. Le problème avec cette librairie c'est le temps qu'elle prend pour envoyer une donnée au LCD 3.6 milliseconde pour trois caractères, temps jugé inacceptable pour notre machine.

Pour parer à ce problème on a créé une librairie optimisé, grâce à l'aide fiche technique du HD44780. Les instructions du HD44780 sont résumées dans l'annexe C. Les informations du timing d'écriture vers le circuit intégré sont présentées dans l'annexe C.

On remarque que les instructions prennent énormément de temps, 37 us minimum pour être exécuter, 1.52 ms minimum dans le cas du return home et le clear.

On remarque que le signal enable soit d'au moins 450 ns de largeur (HIGH) pour être pris en compte par le circuit intégré.

Donc pour la marge de sécurité de l'opération, on a mis le temps d'attente entre chaque commande égale à 50 us et le temps HIGH du signal enable égale à 1 us. Quand on envoie trois caractères il faut s'attendre à au moins à 153 us pour la fin de l'opération.

En ce qui concerne l'initialisation du HD44780 il faut suivre les instructions suivantes :

1. Attendre au moins 15ms depuis le passage à +5V de Vcc,
2. Envoyer l'instruction Function set avec la valeur 0011XXXX,
3. Attendre au moins 4.1ms,
4. Envoyer à nouveau l'instruction Function set avec la valeur 0011XXXX,
5. Attendre au moins 100µs,
6. Envoyer à nouveau l'instruction Function set avec la valeur 0011XXXX,
7. Si on veut activer le mode 4 bits, envoyer les 4 bits de poids fort de l'instruction Function set avec la valeur 0010,
8. Configuration du nombre de lignes et de la matrice, en envoyant l'Instruction Function set avec par exemple la valeur 00111000 (8 bits, 2 lignes, 5x8pixels),
9. Configuration du contrôle d'affichage, en envoyant l'instruction Display on/off control avec par exemple la valeur 00001110 (Affichage visible, curseur visible, curseur fixe),
10. Effacement de l'écran, en envoyant l'instruction Clear display, avec pour valeur 00000001,
11. Configuration du curseur, en envoyant l'instruction Entry set mode, avec par exemple pour valeur 00000110 (déplacement du curseur vers la droite, pas de décalage du compteur d'adresse).
12. Fin de l'initialisation, l'écran est prêt à recevoir les autres instructions permettant l'affichage.

A présent il reste à optimiser le code de la librairie pour qu'il s'exécute le plus rapidement possible.

On aurait pu utiliser le Read busy flag et utiliser les interruptions, au lieu d'attendre 50 us mais le résultat obtenu à la fin était satisfaisant. De plus la tâche d'affichage LCD a la plus basse priorité, elle ne s'exécute seulement que s'il n'y pas d'autres tâches à faire. En plus de ça, elle peut être interrompue par les tâches qui ont un temps critique à respecter.

3.5.7. Schéma de l'architecture du Firmware

Le schéma de l'architecture du Firmware (comment sont connectés les modules, qui peuvent accéder aux ressources) est composé de trois parties :

- La première partie (Figure 3.19) est la relation entre l'ordonnanceur et les tâches, comment l'ordonnanceur passe d'une tâche à une autre.
- Le deuxième partie (Figure 3.20) représente la routine d'interruption et ses liaisons ainsi que son influence sur les autres modules ou tâches.
- La troisième partie (Figure 3.21) illustre le reste de l'architecture avec tous les modules, les ressources software du Firmware et leurs connexions.

Légende :

Autorisation à sens unique d'un accès vers un module ou une ressource.
Autorisation à double sens d'un accès.
Tâches principales de la machine.
Modules.
Modules pouvant être accessibles de n'importe où.
Modules ne pouvant être accessibles qu'à partir d'un seul module.
Autorisation d'une opération de passage d'une tâche à une autre.

- DDM : démarrage des moteurs.

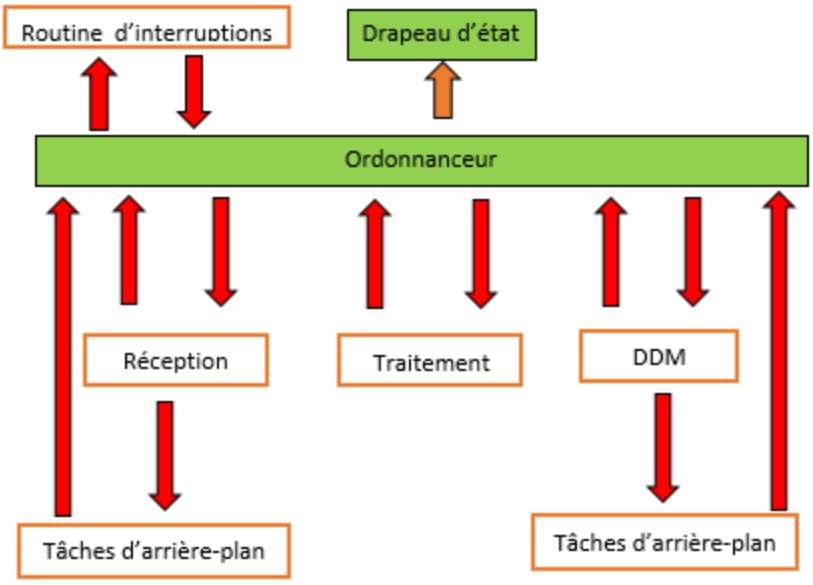

Figure 3.19: Schéma de la relation entre l'odonanaceur et les tâches.

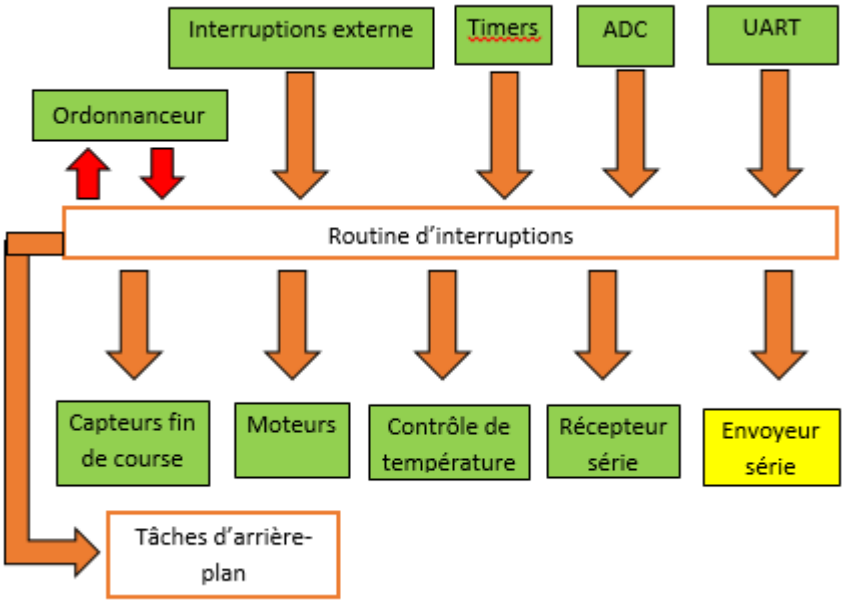

Figure 3.20: Schéma du rôle de la routine d'interruptions.

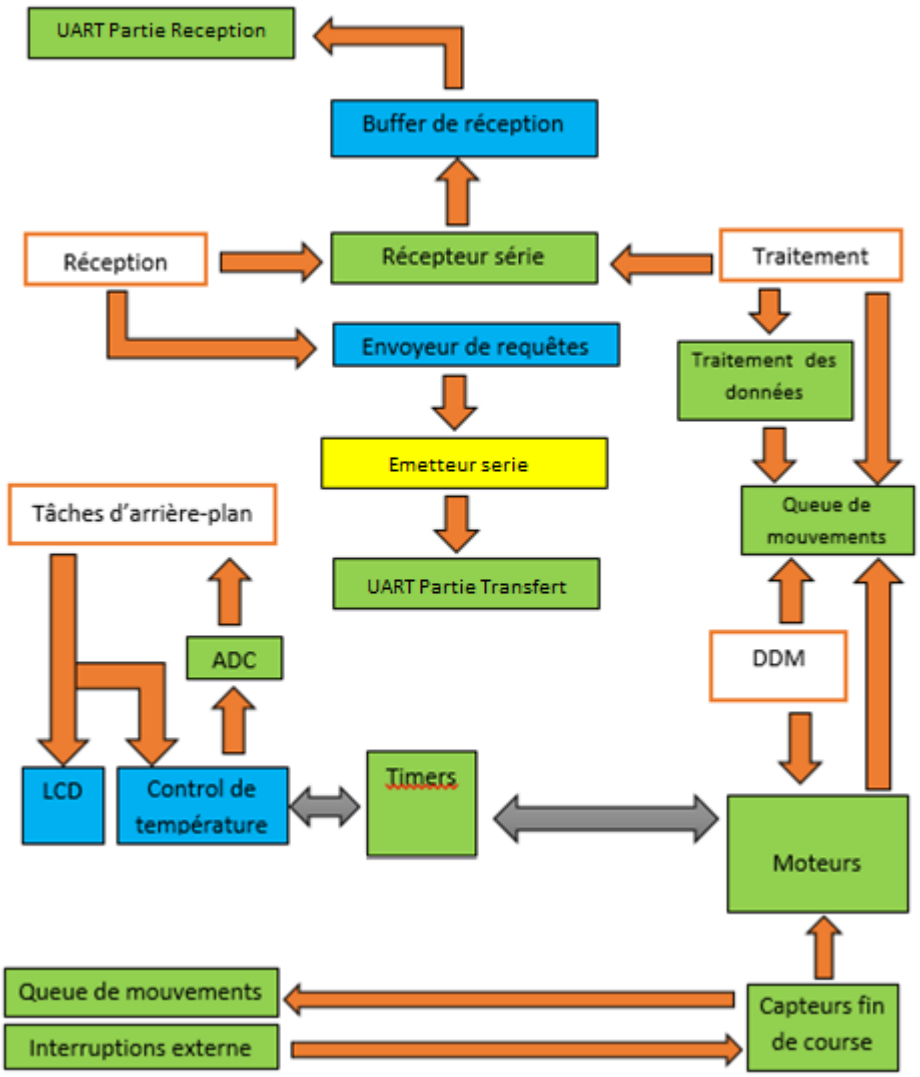

Figure 3.21: Schéma de l'architecture du Firmware.

3.5.8. L'ordonnanceur

L'ordonnanceur doit assurer l'exécution de quatre tâches principales. Les trois premières tâches sont : la tâche « réception », la tâche « traitement » et la tâche « démarrage des moteurs ». Ces tâches-là sont les plus prioritaires, elles doivent être exécutées l'une après l'autre dans un ordre bien précis : la tâche « réception » puis la tâche « traitement » et enfin la tâche « démarrage des moteurs ». Elles ne sont pas périodiques et dépendent uniquement des interruptions hardware du système, pour cela l'organisation du Firmware doit aussi prendre en compte les interruptions.

La quatrième tâche ou « tâches d'arrière-plan » a la plus basse priorité. Elle se compose de plusieurs petites tâches. Elle n'est exécutée que s'il n'y a aucune autre tâche principale qui doit être exécutée.

Les tâches vont être présentées brièvement en insistant sur leur rôle (ce qu'elle peut ou doit faire pendant son temps d'exécution).

3.5.8.1. Les quatre tâches de l'ordonnanceur

3.5.8.1.1. Tâche Réception

La tâche « réception » est la première tâche qui doit être exécutée. Son rôle principal est de s'assurer qu'il y a toujours de nouvelles données disponibles prêtes à être traitées, cela afin de préserver un flow continu de données vers la machine. Par conséquent la tâche peut :

- Accéder au module de réception pour voir s'il y a des données qui ont été réceptionnées et qui ne sont pas encore traitées, si c'est le cas le drapeau d'état devient en mode « traitement ». Dans le cas contraire, l'ordonnanceur exécutera les tâches d'arrière plans.
- Accéder au module de réception pour voir son état c'est-à-dire si une donnée est en train d'arriver cela permettra d'éviter une éventuelle corruption de données.
- Accéder au module de l'Emetteur de requête pour voir si une requête a été envoyée.
- Contrôler le nombre de données disponibles dans le module de réception afin que ce dernier n'excède pas la taille maximale du buffer de réception pour éviter un l'over-flow.
- Accéder exclusivement à l'Emetteur de requêtes, afin d'envoyer une requête par le port UART vers l'ordinateur pour recevoir une nouvelle donnée.

Si aucune donnée n'est prête pour être traitée l'ordonnanceur exécutera les taches d'arrière plans.

3.5.8.1.2. Tâche Traitement

La tâche « traitement » est la seconde tâche qui doit être exécutée. Son rôle principal est de traiter les données entrantes. Par conséquent la tâche peut :

- Lire les données qui arrivent à partir du Buffer de réception. Un premier traitement est fait sur la donnée pour permettre de comprendre la nature de la commande.
  Si la commande est du type mouvement, un autre traitement appelé traitement avancé est nécessaire pour que le drapeau d'états devient en mode « DDM ». D'autre part si la commande n'est pas de type mouvement, elle est exécutée immédiatement et le drapeau d'état devient en mode « réception » pour éventuellement recevoir une nouvelle donnée.
- Contrôler la taille de la file de mouvements pour ne pas arriver à un over-flow. En cas de saturation c'est-à-dire que la file de mouvements est pleine, le drapeau d'état devient en mode « DDM » prévenant ainsi une possibilité de deadlock.
- Remplir la file de mouvement s'il y a une commande de type mouvement.

3.5.8.1.3. Tâche démarrages des moteurs

La tâche « démarrage des moteurs » est la troisième tâche qui doit être exécutée. Son rôle principal est d'initialiser une tâche nommée « routines des moteurs ». Par conséquent la tâche peut :

- Exécuter le trieur de mouvement si aucun moteur n'est en marche et qu'il y a des commandes à exécuter dans la file de mouvement. A la fin de l'exécution de la tâche le drapeau d'état devient en mode « réception » éliminant ainsi toute possibilité de deadlock.

S'il y'a un moteur qui est déjà en marche alors l'ordonnanceur exécute les taches d'arrière plans.

3.5.8.1.4. Tâches d'arrière-plan

Il y a quatre taches périodiques dans l'arrière-plan, elles ont une période de 1 seconde. La première tâche est la plus importante. Elle est nécessaire et doit impérativement se terminer avant son échéance. Les autres tâches sont aussi importantes mais non nécessaires.

- La première tâche: dure environ 6 us, elle permet de réguler la température de l'extrudeuse. Tout d'abord elle accède à l'ADC du microcontrôleur pour lire sa valeur. L'ADC du microcontrôleur est relié à l'entrée de la résistance thermique, ensuite en utilisant un contrôleur PID elle permet de calculer le voltage nécessaire pour la régulation de la température. Finalement le voltage calculé est transformé en signal PWM grâce au Timer et envoyé à une broche du microcontrôleur directement relié au circuit de contrôle de la température.
- La deuxième tâche : qui dure environ 100 us permet de déduire la valeur de la résistance à partir du voltage lu.
- La troisième dure environ 200 us. Elle utilise l'algorithme de recherche binaire par intervalle. Cet algorithme permet de rechercher la valeur de la résistance dans une Look Up Table afin de trouver la température équivalente à la résistance.
- La dernière tâche est la moins prioritaire, elle dure 300 us et permet d'afficher sur le LCD la valeur de la température lue.

3.5.9. Les tâches qui sont déclenchées par les interruptions

Nous avons tout juste présenté quatre tâches : réception, traitement, démarrage des moteurs et les tâches d'arrière-plan. Elles sont toutes ordonnancées par l'ordonnanceur. A présent nous parlerons des tâches qui ont une plus grande priorité et qui dépendent uniquement des interruptions.

Au cours des dernières décennies, les interruptions ont été la principale méthode par laquelle les périphériques matériels peuvent envoyer des événements asynchrones au système d'exploitation [36] comme cité dans [37]. Le principal avantage de l'utilisation des interruptions pour recevoir des notifications des appareils sur leur interrogation est que le processeur est libre d'effectuer d'autres tâches en attendant une interruption. Cet avantage s'applique lorsque les interruptions se produisent relativement rarement [38] comme cité dans [37].

Dans notre Firmware, les tâches qui sont déclenchées par les interruptions (interrupt-driven) peuvent interrompre les tâches de l'ordonnanceur car elles ont des échéances très courtes. Elles doivent absolument s'exécuter avant leurs délais et ce pour le bon fonctionnement de la machine. C'est pour cette raison qu'elles ont la possibilité d'interrompre les autres tâches qui ne sont pas interrupt-driven.

Elles sont au nombre de quatre : la tâche « routine des moteurs », la tâche « réception de données série », la tâche « transfert de données série » et enfin les « tâches des capteurs de fin de course ».

Ces tâches ne peuvent pas s'interrompre entre elles, car leurs exécutions nécessitent quelques microsecondes (en dessous de 10 microsecondes au maximum). Etant donné que l'échéance maximum (dans le cas extrême) permise à la tâche « routine des moteurs » est de 53 us, alors même si on a à exécuter toutes les tâches, il restera encore énormément de temps. Par contre si on met en place un système de préemption, ça engendrerait un overhead supplémentaire, cela pourrait affecter les performances dans certain cas.

3.5.9.1. Tâche routine des moteurs

Cette tâche est sollicitée uniquement lors de l'exécution d'une commande de type mouvement (à chaque fois qu'un pas de moteur doit être fait).

Elle a pour but principal d'envoyer des impulsions au driver des moteurs afin de faire bouger les moteurs. Pour cela, elle doit utiliser l'algorithme du sélectionneur de mouvement afin de choisir quel algorithme de mouvement utiliser.

La tâche « routine des moteurs » a aussi la possibilité d'accéder à la file de mouvements pour voir s'il y a des mouvements qui sont prêts à être exécutés. Cela permet un gain en temps, réduisant ainsi le temps d'arrêt entre les commandes.

3.5.9.2. Tâche réception de données série

La tâche « réception de donnée série » est appelée lors de la réception d'un caractère à partir du port UART, le caractère est ensuite stocké dans le buffer de réception.

3.5.9.3. Tâche transfert de données série

Lorsqu'il y a des données prêtes à être envoyées du port UART vers l'ordinateur, la tâche « transfert de données série » est sollicitée. Pour cela, elle doit s'assurer que le port UART (la partie transmission) n'est pas occupé, afin d'éviter une corruption de données.

3.5.9.4. Tâches des capteurs de fin de course

Il y a en tout quarte tâches, elles suivent toutes le même principe, c'est pour cela qu'on les a regroupées.

Une fois qu'une interruption est déclenchée sur l'un des contacteurs de fin de course, l'une des quatre tâches est alors appelée. La tâche sollicitée dépend du contacteur activé.

La tâche appelée doit en premier lieu s'assurer que le contacteur de fin de course a été activé par la machine et non par un humain, ou autre chose. Pour cela elle doit accéder à la ressource file de mouvement et au module moteur pour voir vraiment si la machine aurait pu déclencher le capteur de fin de course (signifiant que la machine est arrivée à la limite est qu'il faudrait l'arrêter sinon un dommage matériel surviendrait).

Si c'est le cas la tâche ordonne un arrêt immédiat des moteurs et la fin de l'exécution du mouvement.

3.5.10. Régions critiques

Dans un environnement multitâche préemptif, il peut y avoir les périodes pendant lesquelles une tâche ne doit pas être interrompue ou préemptée, c'est la région critique. On doit interdire le changement de tâche et/ou désactiver les interruptions. Cela permet une exécution ininterrompue de la tâche en cours. Une fois la région critique passée, la tâche doit réactiver « la commutation de tâche (ou les interruptions) », sinon le système échouera. Il existe deux circonstances principales dans lesquelles une tâche ne doit pas être interrompue ou préemptée. Dans le premier cas, elle peut accéder à une ressource qui n'est pas sûr qu'elle soit accessible en toute sécurité par plusieurs clients simultanément. Quant au deuxième cas, la tâche doit exécuter des actions dans une période de temps

courte ou dans un ordre bien spécifique. Ce modèle permet à la tâche active une exécution sans aucune interférence potentielle des autres tâches [39].

Il y existe deux régions critiques dans notre Firmware. La première c'est l'initialisation du programme. La seconde est dans l'exécution de la tâche « réception », plus précisément, quand cette tâche demande d'envoyer une requête.

En entrant dans la région critique, la tâche « réception » désactive les interruptions, recevant ainsi automatiquement (virtuellement) la plus grande priorité, car cette dernière devient impossible à interrompre.

3.6. Paramètres du Firmware

Les paramètres les plus importants seront présentés brièvement :

- Debug : permet d'afficher certaines données supplémentaires.
- Fin de course : permet d'activer ou de désactiver les capteurs de fin de course.
- Extrusion absolue : permet de traiter les données concernant le moteur de l'extrudeuse comme des mouvements absolus.
- Positions absolues : permet de traiter les données concernant les moteurs des axes X, Y, Z comme des mouvements absolus.
- Simulation : permet de modifier certaines parties du programme et fonctions afin de pouvoir simuler le Firmware dans Proteus.
- Calcul des pas par millimètres : si vrai, alors le Firmware fera tout seul la conversion millimètres en pas pour chaque mouvement. Cependant, le G-code Preprocesseur devra prendre ça en compte.
- Taille du buffer de réception : la taille du buffer de réception impacte la taille de la file de mouvement car elles sont égales.
- Taille de la machine (working area).
- Pas par millimètres des axes X, Y, Z.
- Vitesse de communication UART.

D'autres paramètres techniques utilisés ne sont pas cités à cause de leurs complexités.

3.7. Initialisation du Firmware

Au début le programme fait appel à certains modules afin d'initialiser : Les timers, UART, ADC, les entrées/sorties etc… Nous n'avons pas parlé de ces modules-là car il faudrait qu'on parle de l'architecture interne du microcontrôleur pour les expliquer, par faute de temps on n'abordera pas ces points. Concernant l'initialisation du LCD, nous avons expliqué précédemment comment initialiser le LCD par le biais du microcontrôleur.

Cependant, nous allons faire le point sur une fonctionnalité très importante de notre Firmware d'imprimante 3D. On avait dit auparavant que l'ordinateur et donc l'interface graphique, pourrait envoyer des commandes à la machine si cette dernière le demande ou si l'utilisateur le fait par lui-même. On peut aussi envoyer à partir de la machine des commandes pour l'interface graphique, c'est ce que fait le module config qu'on va présenter toute de suite.

C'est un module d'initialisation, il permet d'afficher certains paramètres pour utilisateur sur l'interface graphique grâce au port UART. Il permet aussi un envoi de commandes vers la machine pour spécifier les paramètres utilisés. Ces paramètres reçus par l'interface graphique seront utilisés

pour le calcul et l'affichage. Par exemple le calcul des pas par millimètres et l'affichage de la vitesse des moteurs.

3.8. Corps du Firmware

L'ordonnanceur démarre juste après la phase d'initialisation du programme.

La première tâche à être exécutée est la tâche « réception ». La réception de données sur le port UART est entrainée par des interruptions (interrupt-driven) et gérée par la tâche « réception de données série ». Dès lors qu'il y a des données disponibles prêtes à être traitées dans le buffer de réception, la tâche « réception » a la possibilité de fait appel à l'Emetteur de requêtes pour émettre une demande. Cependant, l'appel doit suivre quelques règles bien strictes :

- Le buffer de réception ne doit pas être plein.
- Aucune donnée n'arrive dans le port UART.
- Il ne doit pas y avoir deux requêtes successives sans réponse de la part de l'ordinateur.

La tâche « réception » s'assure alors de ces points-là avant d'émettre une demande de requête. Ensuite, l'ordonnanceur exécute la tâche « traitement ». Par contre si aucune donnée n'est disponible dans le buffer réception alors l'ordonnanceur fait appel aux tâches « d'arrière-plan ».

Les tâches d'arrière-plan s'exécutent périodiquement une après l'autre. Elles peuvent être interrompues par les tâches « interrupt-driven » à n'importe quel moment lors de leurs exécutions. Elles peuvent aussi être interrompues mais pas à n'importe quel moment, par les autres tâches qui ne sont pas « interrupt-driven ». La préemption se fait uniquement lorsqu'une des quatre tâches « d'arrière-plan » aura fini de s'exécuter, alors on pourra l'interrompre pour qu'elle ne passe pas vers la prochaine.

Après avoir reçu une donnée sur le récepteur série, on passe alors vers la tâche « traitement », cette dernière doit tout d'abord s'assurer que la file de mouvement n'est pas pleine. Si c'est le cas, on ne pourra pas traiter une donnée faute de place. L'ordonnanceur retourne alors à la tâche « démarrages des moteurs », cela permet d'éviter d'avoir un « deadlock », car la tâche « traitement » attend qu'une place se libère dans la file. La tâche responsable de la libération de la file de mouvement est la tâche « routine des moteurs ». Pour que l'initialisation de cette dernière soit activée, la tâche « démarrage des moteurs » doit s'exécuter. Tout cela permet de vérifier et d'éviter un « deadlock ».

Dans le cas où la file de mouvement n'est pas pleine, alors la tâche « traitement » fait appel à la fonction traitement.

La fonction traitement doit lire les données arrivées à partir du Buffer de réception. Un premier traitement est alors fait sur la donnée pour permettre de comprendre la nature de la commande.

Si la commande est de type mouvement, un autre traitement appelé « traitement avancé » est nécessaire. Dans le cas contraire, elle est exécutée immédiatement et l'ordonnanceur passe vers la tâche « réception » pour éventuellement recevoir une nouvelle donnée.

Dans le cas où la commande qui vient d'être traitée est une commande de type mouvement, la tâche fait appel à la fonction « traitement avancé ». Cette fonction est appelée uniquement par l'ordonnanceur en mode « traitement ».

Une commande de type mouvement est une commande qui va faire marcher un ou plusieurs moteurs de la machine. Pour cela il faut calculer plusieurs paramètres afin de générer des signaux rapides et précis pour le contrôle des drivers des moteurs. Par exemples, transformer la distance à parcourir en pas de moteurs et calculer la valeur du compteur du Timer pour ne citer que ces deux-là. Cela permet certainement de faire marcher les moteurs. Mais sans l'utilisation d'autres algorithmes avancés, la machine ne fonctionnera pas correctement.

La fonction traitement avancé permet d'appeler deux algorithmes : le sélectionneur de mouvement et l'algorithme des mouvements futurs. Ces deux vous ont été présentés dans le chapitre 3. Enfin, la fonction traitement avancé rempli la file de mouvement avec un nouveau mouvement qui possède tous les paramètres nécessaires pour son exécution, tels que le nombre de pas à exécuter, le sens de la direction, quels moteurs vont marcher etc.…

Suite à l'appel de la fonction « traitement avancé », l'ordonnanceur passe vers la tâche « démarrage des moteurs ». Cette dernière doit initialiser la tâche « routine des moteurs ». Pour cela, elle doit tout d'abord vérifier s'il y a un mouvement disponible dans la file de mouvement et que la tâche n'est pas déjà initialisée. Dans le cas où ces conditions ne sont pas remplies, l'ordonnanceur exécute une des tâches « d'arrière-plan » si elles sont disponibles.

Dans les deux cas l'ordonnanceur retourne à la fin vers la tâche réception pour voir s'il peut encore recevoir une autre donnée entre temps. Cela permet d'éviter de perdre du temps CPU à attendre que la tâche « routine des moteurs » se termine. En plus, cette dernière a la possibilité de s'auto-réinitialiser, car elle a accès à la file de mouvement. Donc à la fin de chaque mouvement, elle va voir s'il y a de nouveaux mouvements prêts à être exécutés.

Enfin pour résumer :

- **La tâche « réception de donnée série »** : s'exécute dès qu'une nouvelle donnée arrive au port UART. (Priorité 3)
- **La tâche « réception »**: s'exécute s'il y a des données non traitées dans le buffer de réception. (Priorité 4)
- **La tâche « transfert de donnée série »**: s'exécute qu'on en envoie des donnés par le port UART. (Priorité 3)
- **La tâche « traitement »:** s'exécute si on peut ajouter un nouveau mouvement dans la file de mouvement. (Priorité 4)
- **La tâche « démarrage des moteurs »:** s'exécute s'il y a un nouveau mouvement et que la tâche « routine des moteurs » n'est pas initialisée. (Priorité 4)
- **Les tâches « d'arrière-plan »:** s'exécutent tous les secondes si aucune autre tâche principale n'est en train d'être traitée. (Priorité 5)
- **La tâche « routine des moteurs »:** s'exécute après son initialisation avec une période variable allant de 53 à 285 us. (Priorité 2)
- **Les tâches « capteur de fin de courses »:** s'exécutent si la machine touche l'un des contacteurs de fin de course. (Priorité 1)

La tâche disposant de la plus grande priorité est celle qui a le plus petit numéro.

3.9. Algorithmes des différents programmes

3.9.1. Algorithme du régulateur

Cet algorithme fonctionne bien avec un relais, donc avec deux modes (Tout ou rien). Alors le mode 1 est tout, et le mode 2 est rien. Un changement soudain de température n'est pas possible en réalité donc ça doit surement être un bruit. Pour cela avons conçus un filtre anti bruit.

Constant du filtre : Différence maximale entre la température précédente et la température actuelle admise pour que l'information ne soit pas considérée comme bruit.

Température d'activation du régulateur : Température minimale pour activer le régulateur.

Seuil1 : Seuil d'activation du mode 1.

Seuil2 : Seuil d'activation du mode 2.

Coef1 : coefficient du mode 1.

Coef2 : coefficient du mode 2.

En cas où le régulateur ne fonctionne pas correctement et pour des raisons de sécurité, il faut ajouter une condition pour limiter la température. C'est ce que la valeur de sécurité fait, cette dernière fait passer le régulateur en mode 2 (rien) si elle est dépassée.

Temps de Fermeture : Temps sur lequel le régulateur est en mode 2 (rien).

Temps d'ouverture : Temps sur lequel le régulateur est en mode 1 (tout).

```
Début algorithme
Si (|Ancienne température - Température actuelle | < Constant du filtre) alors :
        Si (Température actuelle < Température d'activation du régulateur) alors :
                Sortie = 1 ;
        Sinon :
                Différence = Température actuelle - Température désirée ;
                Si (Différence > Seuil1) alors :
                Régulateur = Différence x Coef1 ;
                Sinon si : (Différence < Seuil2) alors :
                Régulateur = Différence x Coef2 ;
                        Si Régulateur < Valeur de sécurité alors :
                        Sortie = 0 ;
Attendre (Temps de fermeture) ;
                        Fin si
                Fin si
        Fin si
Sortie = 1 ;
Attendre (Temps d'ouverture + Régulateur) ;
Sortie = 0 ;
Attendre (Temps de fermeture) ;
Fin si
Ancienne température = Température actuelle ;
Fin algorithme
```

3.9.2. Régulateur PID

Le régulateur PID (Figure 3.22), appelé aussi correcteur PID (proportionnel, intégral, dérivé) est un système de contrôle permettant d'améliorer les performances d'un asservissement, c'est-à-dire un système ou procédé en boucle fermée. C'est le régulateur le plus utilisé dans l'industrie dont les qualités de correction s'appliquent à de multiples grandeurs physiques.

Un correcteur est un algorithme de calcul qui délivre un signal de commande à partir de la différence entre la consigne et la mesure.

Le correcteur PID agit de trois manières :

- Action proportionnelle : l'erreur est multipliée par un gain *G* ;
- Action intégrale : l'erreur est intégrée et divisée par un gain *Ti* ;
- Action dérivée : l'erreur est dérivée et multipliée par un gain *Td*.

Il existe plusieurs architectures possibles pour combiner les trois effets (série, parallèle ou mixte), on présente ici la plus classique : une structure PID parallèle qui agit sur l'Erreur.

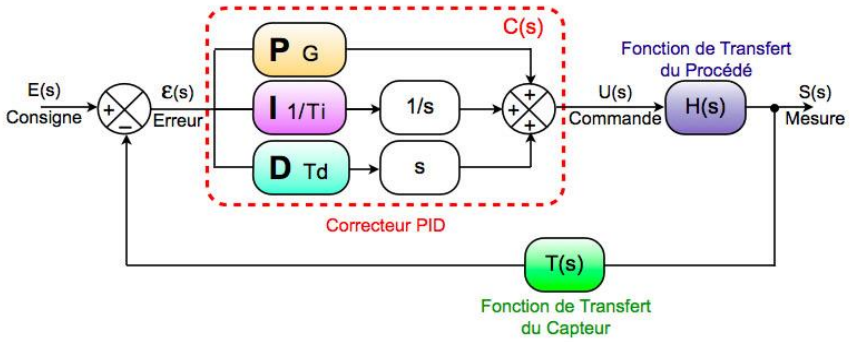

Figure 3.22: Schéma du PID.

3.9.3. L'algorithme d'accumulateur d'erreurs

L'algorithme qui va suivre a été implémenté dans le programme écrit en Java « G-code Préprocesseur ».

Position théorique précédente : C'est la position qui est censé être celle de la machine.

Position précédente : C'est la position réelle de la machine.

Nouvelle position : C'est la position vers laquelle la machine doit se déplacer.

Pas par millimètre : Nombre de pas par millimètre.

Erreur : La partie décimale de la différence entre la position actuelle et la position précédente.

Début :
Si (Position théorique précédente ≠ Nouvelle position) alors :
    Déplacement = Nouvelle position – Position précédente ;
    Déplacement = Déplacement * Pas par millimètre ;
    Erreur = Erreur + Différence – [Différence] ;
Déplacement = [Déplacement] ;
    Si (Erreur ≥ 10) alors :
        Déplacement = Déplacement + 1 ;
        Erreur = Erreur – 10 ;

Sinon si (Erreur ≤ 10) alors :
Déplacement = Déplacement - 1 ;
Erreur = Erreur + 10 ;
Fin si
Différence = Différence / Pas par millimètre ;
Position précédente = Position précédente + Déplacement ;
Fin si
Fin

3.9.4. Algorithme d'accélération et de décélération

Déplacement : Est le déplacement le plus grand entre le déplacement de l'axe X et le déplacement l'axe Y, doit être non nul.
Distance d'accélération : Est la distance sur la quel la vitesse continue d'augmenter.
Distance de décélération : Est la distance à partir duquel la vitesse commence à diminuer.

Début :
Distance d'accélération = | Vitesse initiale– Vitesse maximale | / accélération ;
Si (Déplacement <Distance d'accélération x 2) alors :
Distance d'accélération = Déplacement / 2 ;
Fin si
Distance de décélération = | Déplacement - Distance d'accélération | ;
Fin

3.9.5. Les courbes de Bézier

On peut s'approfondir plus dans la recherche sur la réduction des vibrations et trouver un algorithme intéressant utilisé principalement en informatique.

Les courbes de Bézier sont des courbes polynomiales paramétriques décrites pour la première fois en 1962 par Pierre Bézier qui les utilisait pour concevoir des pièces d'automobiles.

Une courbe de Bézier est définie par un ensemble de points de contrôle P0 à Pn, où n est appelé son ordre (n = 1 pour linéaire, 2 pour quadratique, etc.). Le premier et le dernier point de contrôle sont toujours les points d'extrémités de la courbe ; cependant, les points de contrôle intermédiaires ne se trouvent généralement pas sur la courbe. Voici la formule :

$$\binom{n}{k} = C_n^k = \frac{n!}{k!(n-k)!}.$$

Les coefficients binomiaux :

$$\begin{aligned}\mathbf{B}(t) &= \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i \\ &= (1-t)^n \mathbf{P}_0 + \binom{n}{1}(1-t)^{n-1} t\mathbf{P}_1 + \cdots + \binom{n}{n-1}(1-t)t^{n-1}\mathbf{P}_{n-1} + t^n \mathbf{P}_n \qquad 0 \leqslant t \leqslant 1\end{aligned}$$

3.9.6. Algorithmes des Futurs Mouvements

L'algorithme des futurs mouvements est sans doute l'un des algorithmes les plus performants que nous avons pu créer dans ce Firmware. Cet algorithme peut se résumer dans cette question. Si par

exemple vous conduisiez une voiture et qu'il y a un virage à droite juste devant vous, est ce que vous allez accélérer ou décélérer ? Dans un autre cas, vous êtes dans une autoroute parfaitement droite, est ce que vous allez accélérer ou décélérer ?

Vous l'avez certainement compris, l'algorithme des futurs mouvements permet à la machine d'ajuster son accélération, sa « vitesse maximale » et sa « vitesse finale » en fonction des futurs mouvements qu'elle va faire. Mais pas seulement, elle lui permet de savoir si elle a suffisamment de temps pour accélérer afin d'arriver à sa « vitesse maximale » puis décélérer jusqu'à sa « vitesse finale ». En se basant sur ces calculs, la machine s'ajuste et choisit la meilleure option réalisable.

De calculs complexes doivent être effectués pour l'application de cet algorithme surtout pour un microcontrôleur 8-bits cadencé à 16 Mhz. Alors pour remédier à ce problème une partie des calculs a été transférée vers l'ordinateur. L'algorithme des futurs mouvements est l'algorithme responsable de cette partie.

L'algorithme des futurs mouvements est un programme écrit en Processing (mode Java). Avant le démarrage de l'impression, il modifie le G-code de l'objet à imprimer en calculant et en rajoutant une composante clé, c'est l'angle.

L'angle du prochain mouvement est la clé pour réussir l'algorithme des futurs mouvements. En effet, si la machine sait quel angle elle va prendre durant son prochain mouvement elle peut ainsi s'ajuster au mieux comme le montre la (Figure 3.23).

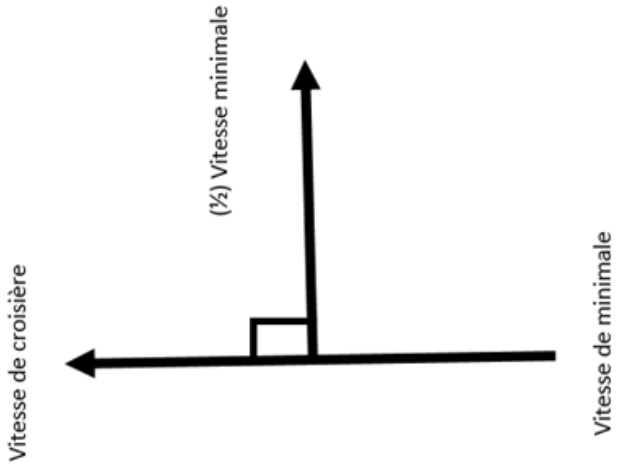

Figure 3.23: Changement de la vitesse de fin en fonction de l'angle.

Dès qu'un mouvement arrive à la « fonction traitement avancé » et après quelques calculs nécessaires, elle fait appel à l'algorithme des futurs mouvements. Cette dernière commence par transformer l'angle reçu en vitesse en utilisant une formule que nous avons développée à partir de la (Figure 3.24).

$$vitesse\ finale = \ angle \times \frac{(vitesse\ maximale - vitesse\ minimale)}{180} + vitesse\ minimale$$

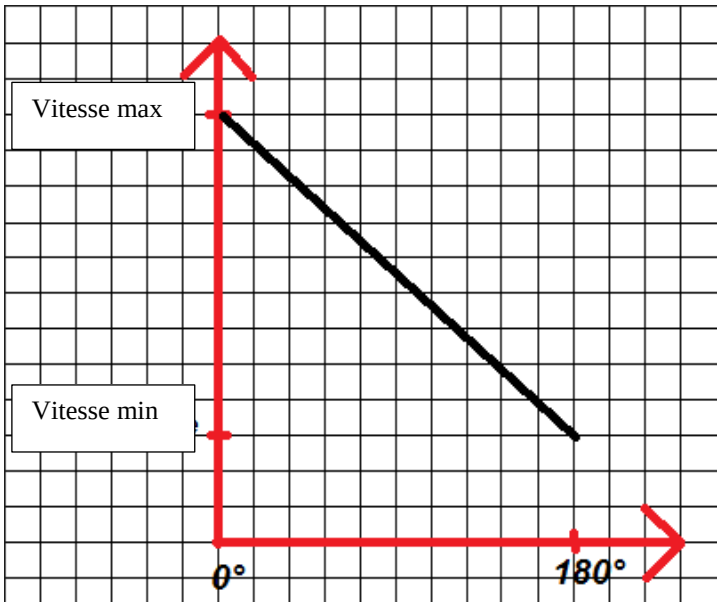

Figure 3.24: Graphe représentant la vitesse en fonction des dégrées d'un angle.

La vitesse calculée (vitesse finale) est celle que doit prendre la machine quand elle finit un mouvement. Cependant, comme nous l'avons cité précédemment dans certain cas la machine n'a pas suffisamment de temps pour accélérer afin d'arriver à sa « vitesse maximale » puis déceler jusqu'à sa « vitesse finale ». Alors l'algorithme doit s'assurer de la faisabilité. Dans le cas où ce n'est pas faisable il utilisera d'autre méthodes pour pouvoir produire la meilleure solution possible et réalisable.

3.9.7. L'algorithme du sélectionneur de mouvements et le trieur de mouvements

En quête de performance, d'efficacité et de rapidité on a développé un algorithme qui est divisé en deux parties : le codage (avec le sélectionneur de mouvements) et le décryptage (avec le trieur de mouvements).

Le sélectionneur de mouvement est appelé par la fonction traitement avancé, il permet de générer un code spécial pour chaque type de mouvement. Il existe 15 types de mouvements différents utilisés dans notre machine :

- Mouvement de l'axe X.
- Mouvement de l'axe Y.
- Mouvement simultané de l'axe X de l'axe Y.
- Ejection du filament sans mouvement des autres axes.
- Mouvement de l'axe X en éjectant du filament.
- Mouvement de l'axe Y en éjectant du filament.
- Mouvement simultané de l'axe X de l'axe Y en éjectant du filament.
- Mouvement de l'axe Z.
- Mouvement de l'axe X et de l'axe Z.
- Mouvement de l'axe Y et de l'axe Z.
- Mouvement de l'axe Z puis de l'axe X, de l'axe Y simultanément.
- Mouvement simultané de l'axe Z et éjection du filament.
- Mouvement de l'axe Z puis de l'axe X en éjectant du filament.
- Mouvement de l'axe Z puis de l'axe Y en éjectant du filament.

- Mouvement de l'axe Z puis de l'axe X et de l'axe Y simultanément en éjectant du filament.

Ce code sera enregistré avec le nouveau mouvement dans la « file de mouvements ».

Le trieur de mouvement comme son nom l'indique est un algorithme qui tri les mouvements à partir du code généré par le sélectionneur de mouvement. Ainsi le trieur de mouvement fera appel à un algorithme spécialisé pour chaque type de mouvement.

Cela permet de gagner en rapidité et en efficacité, car un mouvement de l'axe X tout seule est très simple par rapport à un mouvement simultané de l'axe X et Y, ce dernier requière l'utilisation de l'algorithme de Bresenham.

3.10. Conclusion

Dans ce qui précède, nous avons parlé des programmes qu'on a créés et conçus spécialement pour notre machine. Nous avons vu en détail l'architecture du Firmware avec tous ses composants. On a présenté l'ordonnanceur et les différentes tâches. Dans le chapitre qui suit nous procédons à l'implémentation et mise en œuvre de notre imprimante 3D, et nous montrons les résultats d'expérimentations des différentes parties mécaniques, électroniques, et logicielles.

Chapitre 4. Implémentation et résultats

4.1. Introduction

Dans ce chapitre nous montrons les solutions que nous avons mise en place, l'implémentation des programmes que nous avons développés pour notre imprimante 3D, et leur mise en œuvre. Nous mettons en évidences l'efficacité des solutions que nous avions proposées aux problèmes que nous avions soulevés dans le chapitre 3.

Enfin nous discutons des résultats obtenus.

4.2. Outils de programmation et environnement de développement

Nous avons développé nos programmes en utilisant différents logiciels et langages de programmation.

4.2.1. Visual studio code

Visual Studio Code (Figure 4.1) est un éditeur de texte générique multiplateforme open source, qui est fourni par Microsoft. Il s'exécute de manière native sur le système d'exploitation de votre choix (Mac OS X, Linux et Windows). Il prend en charge JavaScript, C#, C++, PHP, Java, HTML, R, CSS, SQL, Markdown, TypeScript, LESS, SASS, JSON, XML et Python ainsi que de nombreux autres formats de fichiers courants. L'assistant visuel (IntelliSense) complète automatiquement les descriptions des API lorsque vous les saisissez pour une plus grande rapidité et une plus grande précision. Vous diagnostiquez les problèmes de votre application avec les outils intégrés de débogage pour Node.js, TypeScript et JavaScript. Vous définissez des points d'arrêt dans votre code, des arrêts en cas d'exception et des variables espions, vous examinez votre code ou vous naviguez en remontant la pile des appels, ou rattachez-vous aux processus d'exécution locaux. Adoptez les flux de travail modernes grâce à la puissance et à la flexibilité de Git [40].

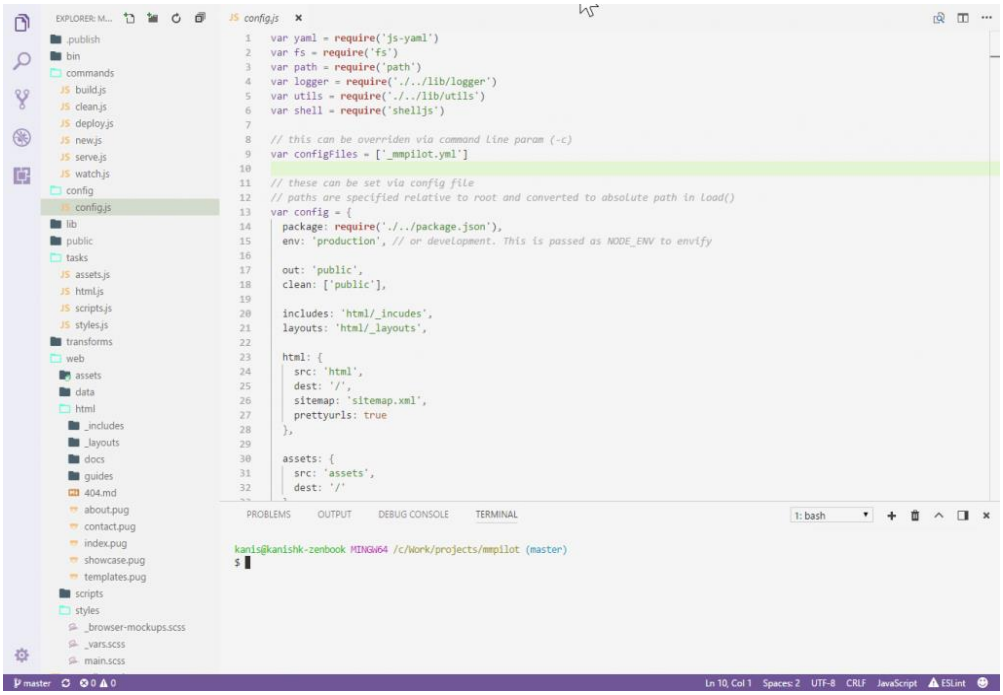

Figure 4.1: Visual studio code

4.2.2. Processing

Processing (Figure 4.2) est un environnement de développement libre (sous licence GNU GPL), créé par Benjamin Fry et Casey Reas, Il est basé sur la plate-forme Java et permet de programmer directement en langage Java.

Depuis 2001, Processing promeut la maîtrise des logiciels dans les arts visuels et la maîtrise des visuels dans les technologies. Il y a des dizaines de milliers d'étudiants, artistes, designers, chercheurs et amateurs qui utilisent Processing pour l'apprentissage et le prototypage [41].

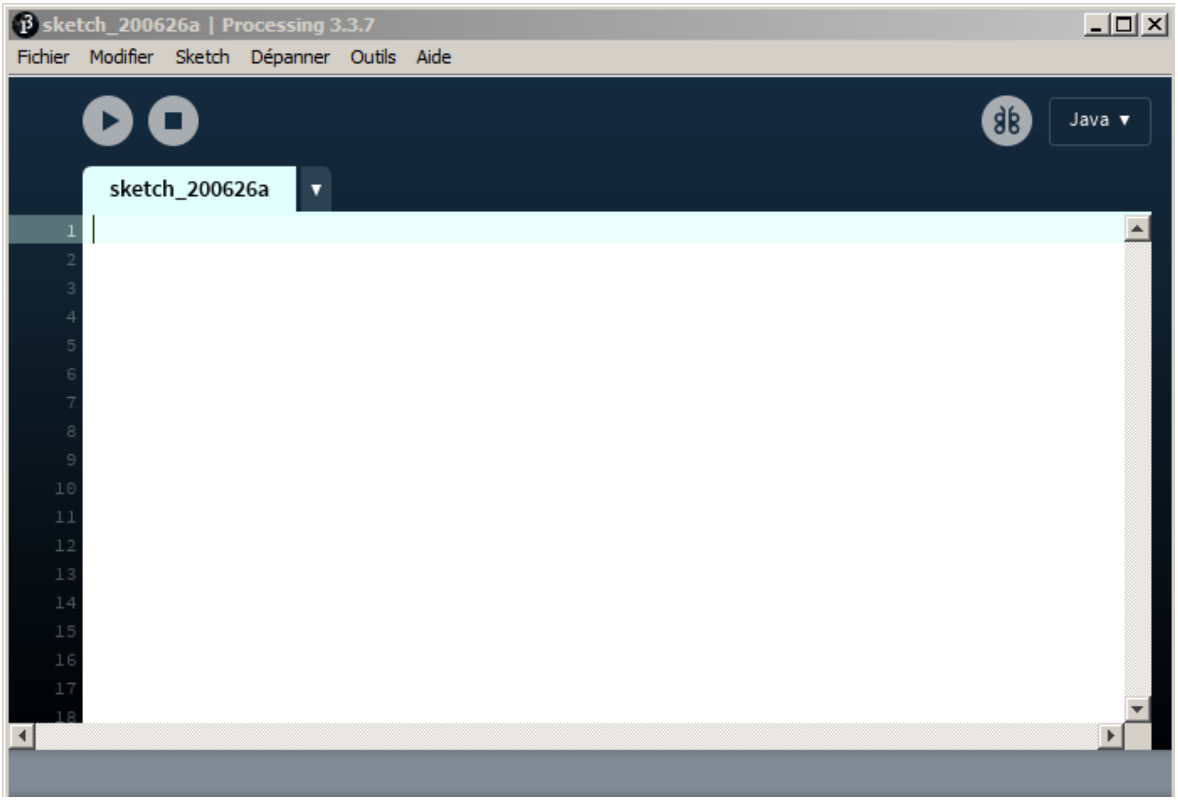

Figure 4.2: Processing

4.2.3. Cura

Pour imprimer en 3D, il faut un logiciel de tranchage, aussi appelé slicer, afin de convertir le modèle 3D en un fichier G-code qui fournira à l'imprimante 3D toutes les instructions nécessaires. L'un de ces slicers est Cura (Figure 4.3), qui a été développé par David Braam en 2014, puis adopté par Ultimaker. Il s'agit d'un logiciel open-source, surement le plus répandu sur le marché mondial de la fabrication additive. En 2019, Cura comptait déjà 600 000 utilisateurs et serait utilisé pour plus de 2 millions de pièces imprimées en 3D chaque semaine. L'un des principaux avantages de Cura est sa facilité d'utilisation, la prise en charge de différents formats de fichiers et sa compatibilité avec de nombreuses imprimantes 3D. Les formats de fichiers pris en charge sont STL, OBJ, X3D et 3MF. Autre caractéristique qui contribue à sa popularité est le fait qu'il soit compatible avec les systèmes d'exploitation les plus courants, Windows, Mac et Linux [42].

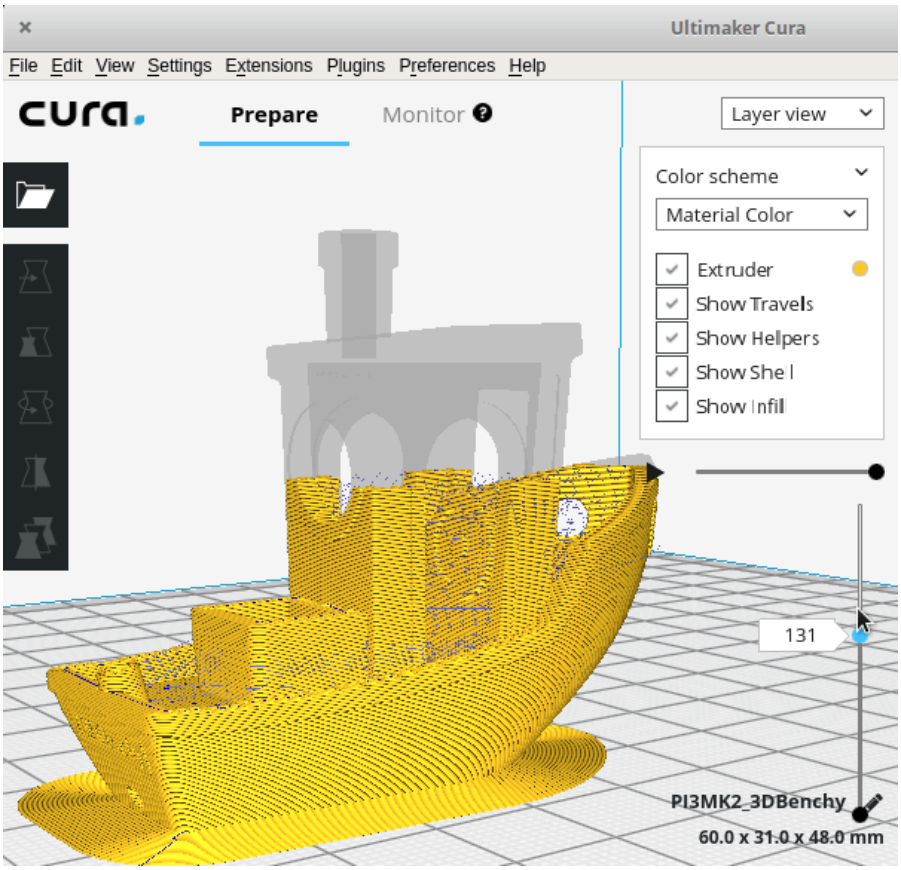

Figure 4.3: Interface du logiciel Cura

4.2.4. Netbeans

NetBeans (Figure 4.4) est l'environnement de développement Java gratuit et Open Source fourni par Sun Microsystems. NetBeans est également un Framework servant à la construction des autres outils de développement de Sun. La puissance de l'IDE (Integrated Development Environment) a été employée pour s'ouvrir à d'autres langages grâce à un système de plug-ins (modules) [43]. La plate-forme NetBeans est entièrement basée sur l'API Java (donc multiplateforme) avec AWT et Swing et intègre les concepts de Java Standard Edition (SE) [44].

Avant les autres IDE, NetBeans IDE fournit un support complet de première classe pour la dernière technologie Java et les dernières améliorations des spécifications Java. C'est la première fois que l'IDE prend en charge gratuitement JDK 8, JDK 7, Java EE 7. L'IDE n'est pas seulement un éditeur de texte. L'éditeur NetBeans peut mettre en retrait, faire correspondre les mots et les crochets et mettre en évidence le code source selon la syntaxe et la sémantique. Il fournit également des modèles de code, des conseils de codage et des outils de refactoring [45].

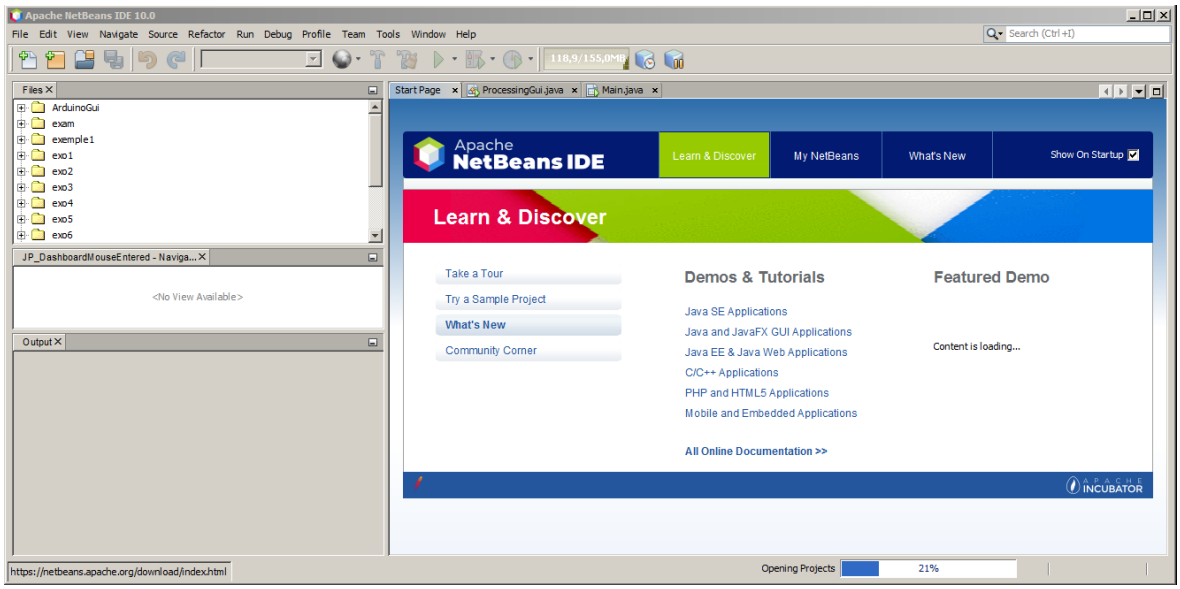

Figure 4.4: NetBeans

4.2.5. Atmel studio

Atmel Studio 7 (Figure 4.5) est la plate-forme de développement intégrée (IDP) pour développer et déboguer toutes les applications de microcontrôleur AVR® et SAM. L'IDP Atmel Studio 7 vous offre un environnement transparent et facile à utiliser pour écrire, construire et déboguer vos applications écrites en C/C++ ou en code assembleur. Il se connecte également de manière transparente aux débogueurs, programmeurs et kits de développement qui prennent en charge les périphériques AVR et SAM [46].

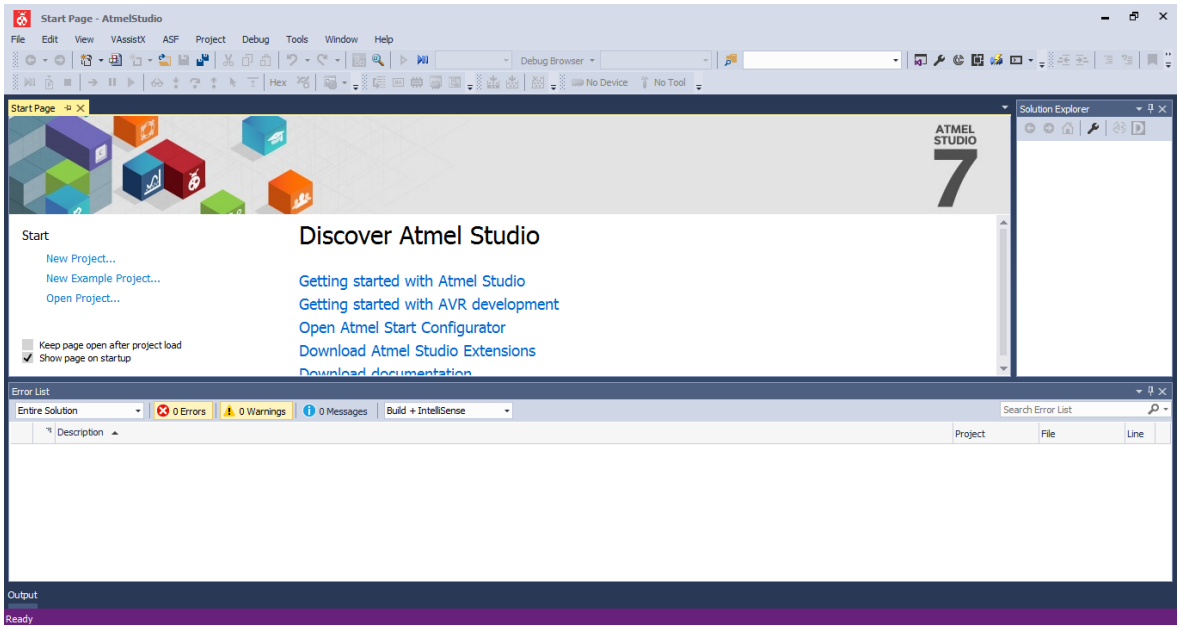

Figure 4.5: Atmel Studio

4.3. Implémentation des solutions utilisées pour la régulation de la température

La température est un élément clé de l'impression 3D, une mauvaise température (trop grande ou trop petite) risque de donner un résultat négatif voire même un échec de l'impression. Une fluctuation de température, donne une impression hétérogène.

Nous avons testé le régulateur PID avec un relais (seul disponible sur le marché) afin de contrôler la température de l'extrudeuse. Cependant, ça n'a pas bien marché, alors nous avons créé un nouvel algorithme spécialement pour la régulation de la température avec le relais.

Le problème avec le relais c'est que sa vitesse de switch (fermeture et ouverture) est très petite par rapport au Mosfet, ce qui engendre une fluctuation de température de quelques degrés Celsius.

A cet effet, on a créé un régulateur que nous avons implémenté sur un Arduino Mega. Les temps d'activation et de désactivation varient en fonction de la différence entre la température réelle et la température souhaitée (Figure 4.6). Le résultat obtenu (±2 degrés) est acceptable.

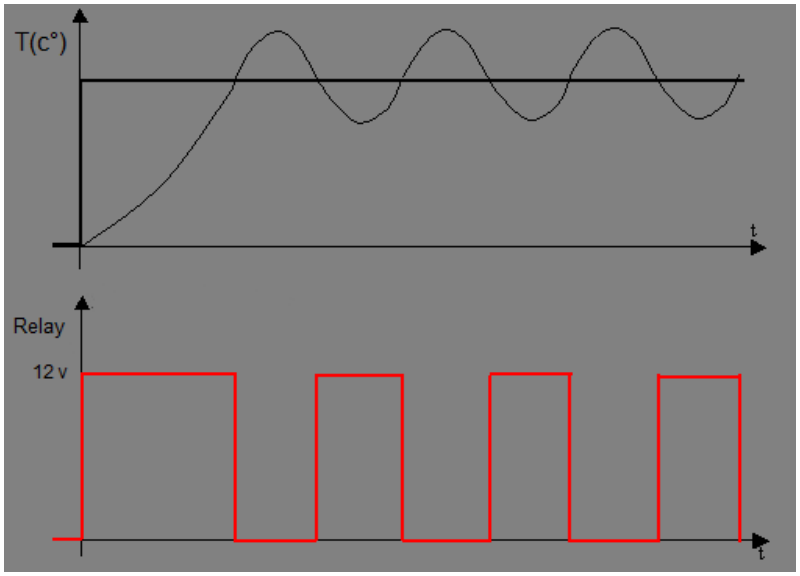

Figure 4.6: Graphe du régulateur de température.

4.3.1. Implémentation des programmes

```
11
12  // PID values
13  int16_t TempControl::PID_p = 0;
14  int16_t TempControl::PID_i = 0;
15  int16_t TempControl::PID_d = 0;
16
17  volatile uint8_t TempControl::Flag_ADC = 0;
18
19  volatile uint16_t TempControl::timer2_counter = 0;
20
21  const uint32_t PROGMEM TempControl::temperature_array[291][2] = {
313
314
315 void  TempControl::regulateTemp()
316 {
317
318   temperature_read = ADC;
319   PID_error = set_temperature - temperature_read;
320   PID_p = PID_error * KP;
321   PID_d = ((PID_error - previous_error) / elapsedTime) * KD;
322   PID_value = PID_p + PID_i + PID_d;
```

Figure 4.7: Extrait du code du PID

Un extrait d'un code qui permet de réguler la température de l'extrudeuse grâce au régulateur PID.

```
22    } else {
23      if ( abs( r - rOld ) < 20 ) {
24        if (r > 360) {
25          High();
26        }
27        else {
28          int coef = r - 330;
29          if (coef > 3) {
30            regulator = coef * 35;
31          }
32          else if (coef <  -1) {
33            regulator = coef * 95;
34          }
```

Figure 4.8: Extrait du code de l'algorithme du régulateur

Un extrait d'un code qui permet de réguler la température de l'extrudeuse grâce au régulateur que nous avons conçu.

4.3.2. Comparaison entre le PID et l'algorithme du régulateur

Avec l'algorithme du régulateur qu'on a mis en place, on prend plus de temps pour toutes les valeurs les plus optimales des coefficients, car il y a tout simplement plus de coefficients que dans un régulateur PID.

D'après nos tests, l'algorithme du régulateur convient mieux quand on a un switch tout ou rien comme un relais. D'une autre part, le régulateur PID doit être modifié pour être utilisé dans ce cas (le cas où on a un switch). Le PID doit utiliser un seuil lequel qui une fois dépassé permet au switch de s'ouvrir ou de se fermer.

A la fin dès qu'on a pu obtenir un Mosfet dont les caractéristiques convenaient à notre projet on a alors implémenté le régulateur PID dans le Firmware.

4.4. Implémentation des solutions pour l'amélioration de la précision

La précision est le critère essentiel dans l'impression 3D, et c'est aussi la plus difficile à mettre au point. Elle nécessite une équipe d'ingénieurs et de chercheurs spécialisés dans les différents domaines qui touchent à l'impression 3D (Mécanique, Chimie, Electronique, Informatique etc.…).

Pour améliorer la précision de notre imprimante 3D, on a réparti la solution de ce problème sur quatre parties distinctes, la partie informatique qui regroupe tous les programmes utilisés afin d'améliorer la précision d'impression, la partie électronique, la partie mécanique et la partie chimique.

La partie mécanique joue un rôle très important dans l'obtention d'une bonne impression. Les pièces nécessaires à notre machine on était conçus à l'aide du logiciel Adobe Illustrator et découpé avec une machine à commande numérique afin d'obtenir le meilleur résultat possible.

La qualité des composants utilisés est très importante. Dans la plupart du temps nous avons constaté un jeu dans l'assemblage des différentes pièces achetées. A titre d'exemple a un roulement linéaire de 8 mm de diamètre correspond une tige inférieure. L'accumulation de ce genre d'erreurs influe sur le bon fonctionnement de la machine. Le choix des composants de bonnes qualités est impératif.

Nous avons utilisé le PLA ou Poly lactic acid (Acide poly lactique) qui est une matière plastique d'origine végétale obtenu à partir d'amidon de maïs. Le PLA a une température de transition vitreuse (l'intervalle de température à travers lequel la matière passe d'un état caoutchouteux à un état vitreux, solide) d'environ 60 °C et une température de fusion de 175 °C.

Donc par exemple si on règle notre extrudeuse a une température de 180 °C, le PLA va passer de l'état solide à l'état liquide, alors il faudrait ensuite le faire refroidir le plus rapidement possible en dessous des 60 °C (la température de transition vitreuse). Pour cela il nous a suffi d'utiliser un ventilateur qu'on a attaché à l'extrudeuse (Figure 4.9).

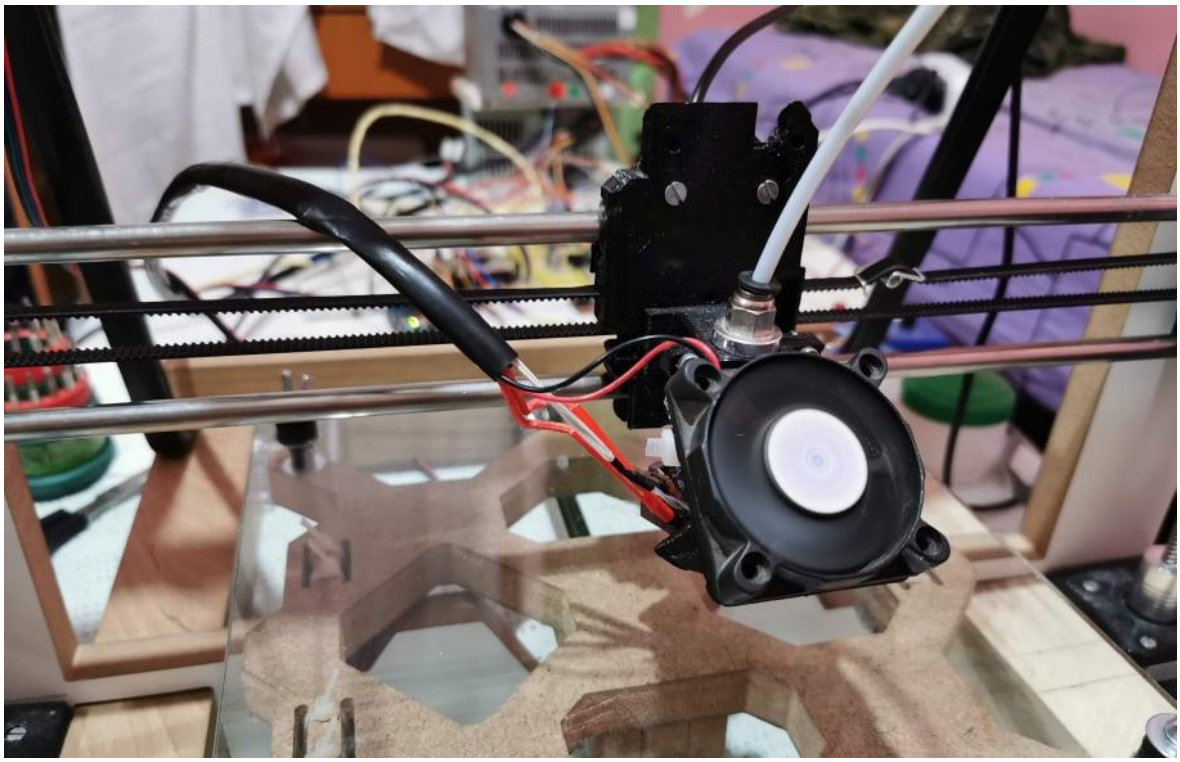

Figure 4.9: Ventilateur ajouté à l'extrudeuse

On peut jouer avec la précision de la machine en augmentant le nombre de pas par millimètre, pour cela il faut changer le nombre de pas par tour du moteur pas à pas, mais comme ce dernier est fixé lors de sa construction, il n'est donc pas possible de le modifier.

Cependant on peut utiliser la technique du « Microstepping », cette dernière n'envoie pas une impulsion de courant complète au moteur pas à pas pour le faire bouger. Elle envoie uniquement des impulsions partielles au moteur. En conséquence, le moteur ne tourne que d'une fraction de pas. Les valeurs habituelles de Microstepping sont de 16 à 64 micro pas par étape complète. Pour un moteur pas à pas de 1.8°, cela fait des pas de 0,1124° à 0,028° [47].

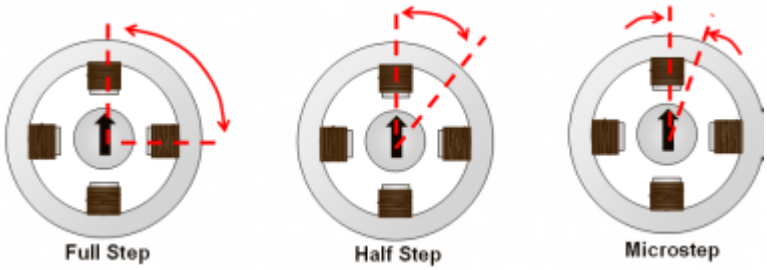

Figure 4.10: Exemple de Microstepping

Chaque moteur pas à pas aura un angle de pas défini associé pour un tour complet, le Microstepping permet de positionner la tige entre cet angle. La (Figure 4.10), represente un moteur pas à pas biphasé possédant un angle de pas égale a 90°. Si vous appliquez les techniques de Microstepping, vous pouvez positionner l'arbre entre les 90° du pas [48]. Microstepping offre les avantages suivants :

- Augmente la résolution en divisant une étape complète en sous-étapes.

- Offre des transitions plus douces entre les étapes.
- Réduit le bruit et les problèmes d'antirésonance.
- Maximise la sortie de couple à des taux de pas faibles et élevés.

Du coté électronique, on a utilisé un driver qui possède la technique du Microstepping. Le DRV8825 possède un maximum de 32 micro pas, il n'est pas la meilleure option mais il est certes le moins cher.

Malgré toutes les méthodes citées précédemment, nous n'avons pas atteint la précision souhaitée. A titre d'exemple, si on a une précision de 0.0025 mm / pas et qu'on demande à la machine de se déplacé de 0.003 mm, elle n'est capable de bouger que de 0.0025 mm, ce qui engendre une erreur 0.0005 mm. Un fichier G-code peut avoir des centaines de millier de lignes, alors l'accumulation de toutes ces erreurs fait que le résultat ne soit pas satisfaisant.

Il faut donc trouver une solution sans changer de matériel. L'idée d'un algorithme qui puisse modifier le fichier G-code, pour que les lignes transmises à l'imprimante puissent toutes être exécuté en minimisant l'erreur au maximum possible.

4.4.1. STL Viewer

```
44  ============================ ADD POINTS FUNCTION ============================
45     */
46
47    void AddPoints() {
48      points.add(new PVector(x, y, z));
49    }
50
51    /*
52  ============================ GET POINTS FUNCTION ============================
53     */
54
55    ArrayList<PVector> GetPoints() {
56      return points;
57    }
58
59    /*
60  ========================== POINT DESTRUCTOR FUNCTION ==========================
61     */
62
63    void PointsDestructor() {
64      for (int i = points.size() - 1; i >= 0; i--) {
65        points.remove(i);
66      }
67    }
68  }
```

Figure 4.11: Extrait du code du STL Viewer

Un extrait d'un code qui permet d'afficher des fichiers STL.

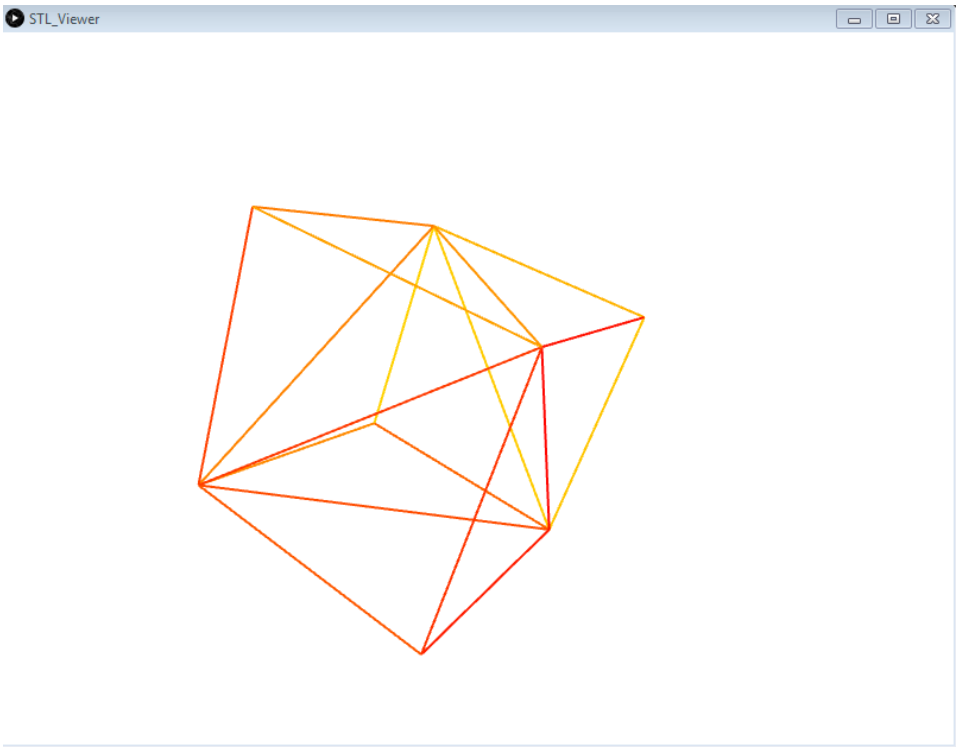

Figure 4.12: Affichage d'un objet 3D (Un cube) dans le STL Viewer

4.4.2. G-code Simulator

```
52          GCODE.AddPoints();
53
54          float hu = 0;
55          int IndexG0 = 0;
56          for (PVector v : GCODE.GetPoints()) {
57            if (GCODE.GetG0(IndexG0)==false) {
58              strokeWeight(0.2);
59              stroke(hu, 255, 255);
60              line(x1, y1, z1, v.x, v.y, v.z);
61              hu += 0.25;
62              if (hu > 255) {
63                hu = 0;
64              }
65            } else {
66            }
67            IndexG0++;
68            x1=v.x;
69            y1=v.y;
70            z1=v.z;
71          }
72        }
```

Figure 4.13: Extrait du code du G-Code Simulator

Un extrait d'un code qui permet de simuler des fichiers G-Code.

Figure 4.14: Simulation 3D dans le G-code Simulator en mode ligne par ligne

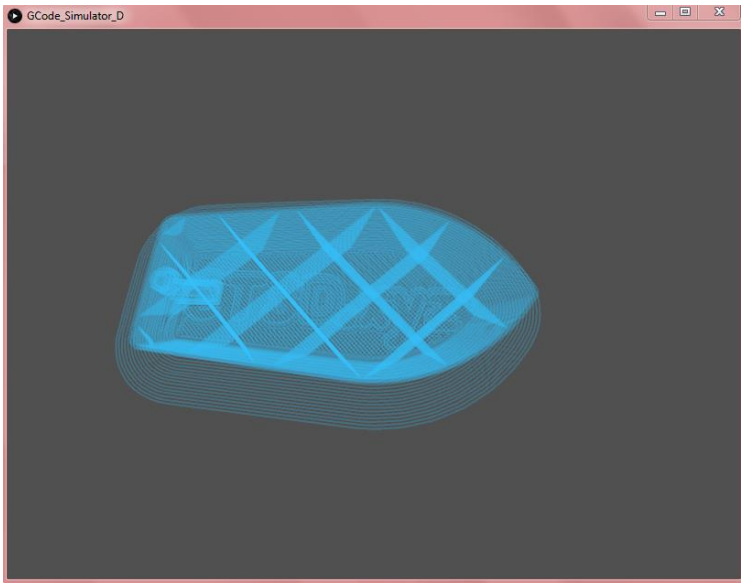

Figure 4.15: Simulation d'un objet 3D dans le G-code Simulator

4.4.3. Algorithme d'accumulation d'erreurs

```
229          dz = dz/steps_per_milimeter_Z;
230          flag_Z = true;
231        }
232
233        if ( space) {
234          if (is_there_G0_function) { //***** G0
235            processedGcode += "G0";
236            if (dx != 0.0 && flag_X) {
237              processedGcode += " X"+dx;
238            }
239            if (dy != 0.0 && flag_Y) {
240              processedGcode += " Y"+dy;
241            }
242            if (dz != 0.0 && flag_Z) {
243              processedGcode += " Z"+dz;
244            }
```

Figure 4.16: Extrait du code d'accumulation d'erreurs

Un extrait d'un code qui permet d'utiliser l'algorithme d'accumulation d'erreurs.

4.4.4. Résultat de simulation

Les résultats (Figure 4.17 et Figure 4.18) obtenus ont été satisfaisants.

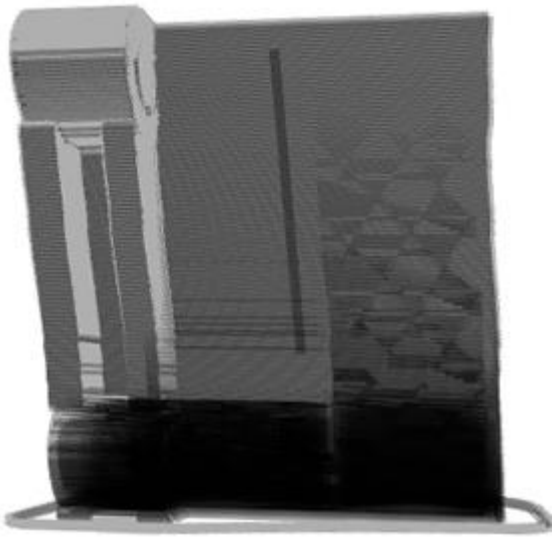

Figure 4.17: Simulation d'un objet sans l'algorithme d'accumulation d'erreurs

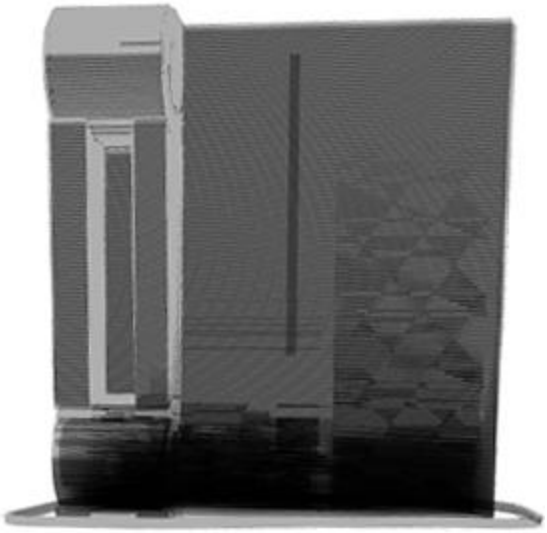

Figure 4.18: Simulation d'un objet avec l'algorithme d'accumulation d'erreurs

4.4.5. Résultat d'impression

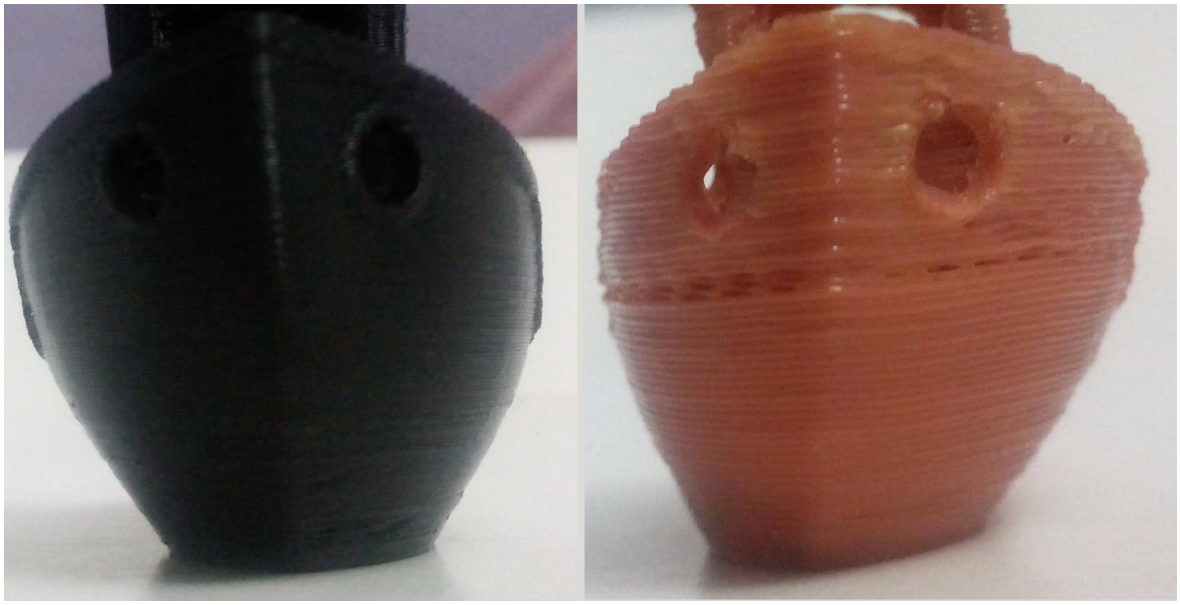

Figure 4.19: Avant (droite) et après (gauche) l'algorithme d'accumulation d'erreurs

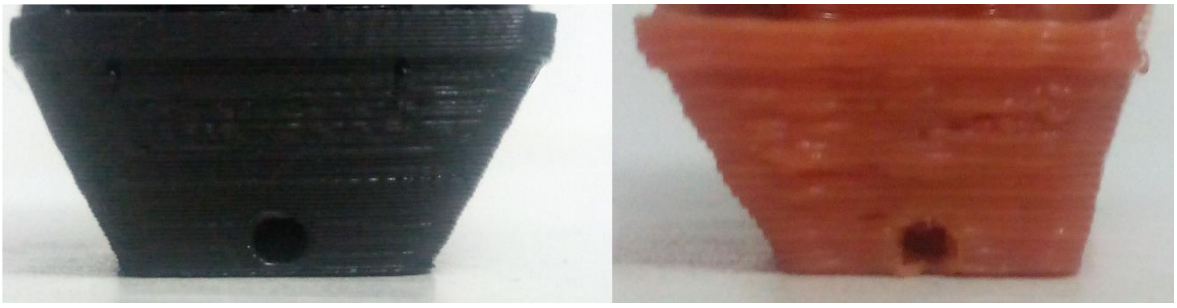

Figure 4.20: Avant et après l'utilisation l'algorithme d'accumulation d'erreurs

On remarque dans les Figure 4.19 et Figure 4.20 que l'objet de gauche imprimé avec l'algorithme d'accumulation d'erreurs qu'il est plus symétrique que celui de droite qui a été imprimé sans l'algorithme d'accumulation d'erreurs.

4.5. Implémentation des solutions pour stabiliser l'imprimante face aux vibrations

Nous avons déjà évoqué les problèmes de vibrations dans les imprimantes 3D. On avait dit qu'il faudrait réduire au maximum le poids de la machine et qu'il faudrait ajouter une sorte d'éponge sous la machine afin d'absorber les vibrations.

Dans cette partie nous allons parler des programmes et algorithmes qu'on a créés afin d'améliorer la qualité de l'impression.

4.5.1. Algorithme d'accélération et de décélération

Au début lors de la toute première conception du Firmware, quand la machine recevait un point (X, Y, Z), elle se déplaçait du début jusqu'à la fin avec une même vitesse. Un changement de direction faisait introduire les indésirables vibrations. La qualité des impressions prenait un sévère coup. Alors nous avions besoin d'une nouvelle solution, c'est là que l'idée d'un algorithme de décélération et d'accélération lors de chaque déplacement apparu. Cet algorithme a été implémenté dans Firmware sur un ATmega 2560.

Le principe de cet algorithme est d'accéléré à partir d'une certaine vitesse appelé « vitesse initiale », jusqu'à atteindre une certaine vitesse appelé « vitesse maximale », de continuer avec cette vitesse un certain temps et puis de décélérer jusqu'à la vitesse initiale, réduisant ainsi énormément les vibrations.

La décélération jusqu'à la vitesse initiale fait que l'imprimante a toujours une certaine vitesse à la fin de chaque mouvement, réduisant ainsi l'accélération du prochain mouvement (Figure 4.21).

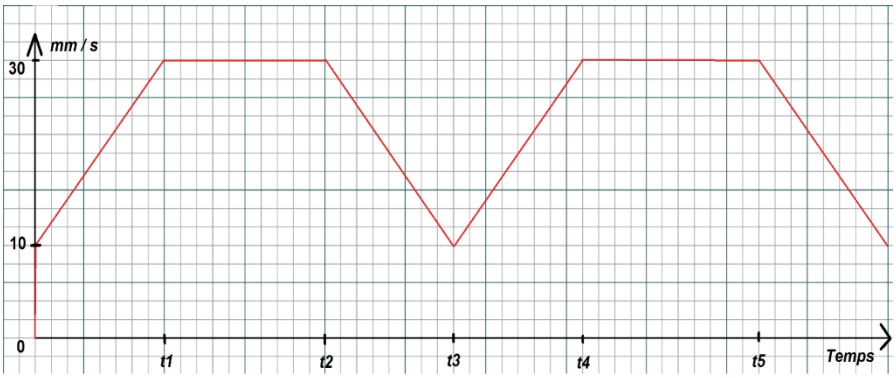

Figure 4.21: Graphe de la vitesse en fonction du temps.

Parfois la machine est appelée à faire de petits déplacements. Le problème avec la solution proposée, c'est que l'imprimante ne peut pas atteindre la « vitesse maximale » dans ce cas-là. Même dans le cas où elle pourrait, il serait impossible de décélérer jusqu'à la « vitesse initiale », provoquant dans tous les cas possibles des vibrations.

Alors pour éviter ce problème-là, il a fallu améliorer l'algorithme précédent. L'algorithme doit prévenir la machine si elle peut accélérer jusqu'à la « vitesse maximale ».

Dans le cas où elle ne peut pas accélérer jusqu'à la « vitesse maximale », alors elle ne cherchera pas à atteindre cette dernière, mais s'assurer un retour au point « vitesse initiale » (Figure 4.22), afin d'éviter le problème précèdent.

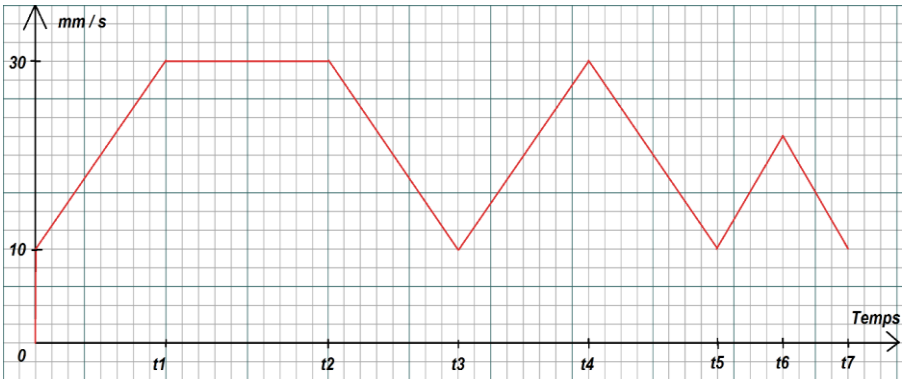

Figure 4.22: Graphe de la vitesse/temps lorsque on a de petits déplacements.

4.5.2. Accélération non constante

L'algorithme d'accélération et de décélération présenté précédemment a une certaine caractéristique et vous l'aurez certainement devinée, l'accélération/décélération est constante, par conséquence on a une vitesse qui est linaire comme le montre la (Figure 4.21 et Figure 4.22).

Cependant on à opter pour une accélération non constate qui permet d'avoir une vitesse non linéaire. La figure (Figure 4.23) présente deux graphes, le premier celui d'une vitesse linéaire et le deuxième d'une vitesse non linaire pour la même distance parcourue.

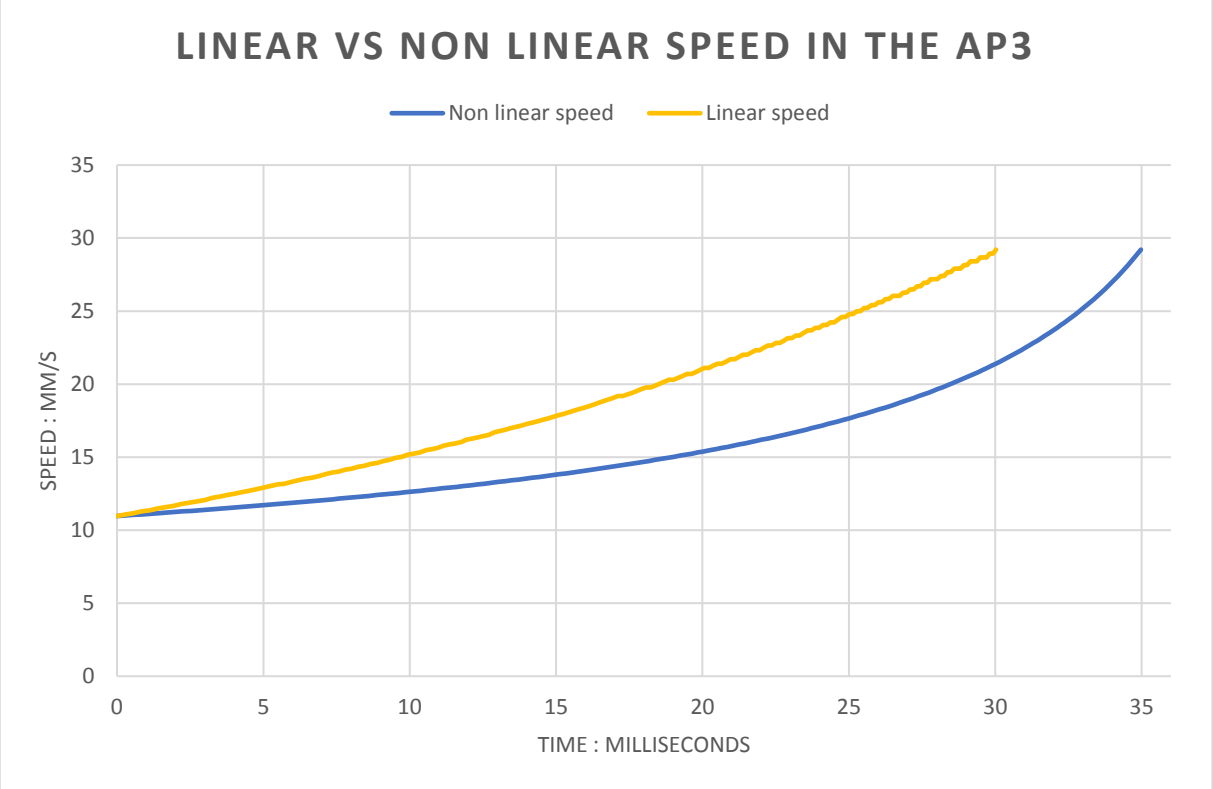

Figure 4.23: Graphes d'une accélération linéaire vs non linéaire.

Dans cet exemple on a choisi une « vitesse initiale » et une « vitesse maximale » respectivement égales à environ 11.228 mm/s, 29.901 mm/s.

La vitesse non linéaire (dans l'intervalle d'accélération) permet un démarrage doux du mouvement, et plus le temps passe plus la vitesse augmente jusqu'à la fin de l'intervalle d'accélération. Ceci aide à réduire les vibrations, ce qui donne l'impression que la machine flotte.

La (Figure 4.24) montre la vitesse de la machine en fonction du temps en appliquant une accélération non linéaire.

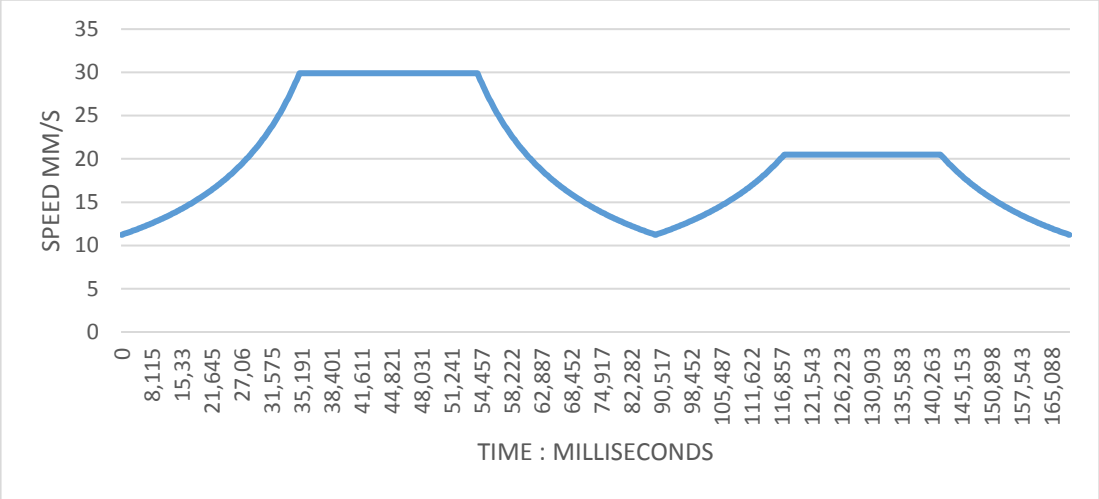

Figure 4.24: Graphe vitesse/temps de deux mouvements successifs.

Avec une vitesse linaire on a plus de jerk (dérivé du vecteur d'accélération) c'est-à-dire l'accélération change plus vite dans un court laps de temps, de ce fait, on a une distance qui est parcourue plus rapidement. Si on veut que la machine parcoure avec une vitesse linéaire la même distance parcourue avec une vitesse non linéaire, on recalcule l'accélération du graphe 1 (Figure 4.23) de façon à ce qu'elle soit égale à l'accélération moyenne du graphe 2.

Il faut alors que la « vitesse initiale » du graphe 1 soit inférieure à la « vitesse initiale » du graphe 2 (Figure 4.23), c'est-à-dire en dessous de 11.228 mm/s.

Ce n'est pas très pratique comme solution, car plus la « vitesse initiale » est faible plus la machine risque d'être lente. Comme on l'a dit précédemment, il y a certains mouvements si petits qu'on ne peut accélérer jusqu'à la « vitesse maximale », alors la machine devra ce contentée de se déplacer avec une vitesse avoisinant la « vitesse initiale ».

De plus même si on pouvait contourner ce problème descendre à moins de 10 mm/s n'est pas vraiment très recommandé. Car le mouvement de l'extrudeuse que l'on considère très souvent comme un mouvement linéaire, ne l'est pas.

Au début d'un mouvement, l'extrudeuse a besoin d'une certaine pression pour pouvoir extruder le filament, bien que la machine ait déjà commencée à se déplacer il n'est pas très rare de remarquer que le filament n'est toujours pas sorti de la tête de l'extrudeuse. Vise versa, à la fin du mouvement, les moteurs s'arrêtent mais le filament continue de sortie de la tête de l'extrudeuse.

Hormis ce problème, plus le mouvement de la tête de l'extrudeuse est lent, plus elle reste longtemps au-dessus de l'objet à imprimer, ce qui augmente fortement sa température provoquant ainsi l'altération de ce dernier. Pour cela une faible « vitesse initiale » est à éviter.

Bien sûr qu'il existe des solutions pour les problèmes qu'on a cités, mais pour quoi s'embêter à vouloir corriger des problèmes qu'on peut plus ou moins éviter. C'est pour ces raisons qu'on a choisi la vitesse non linéaire.

```
710             else
711             {
712               PORTB &= ~_BV(PIN_X_STEP);
713             }
714 #if _ACCELERATION_REDUCTOR == 1 && ACCELERATION_FACTOR > 1
715             accelerationFactor++;
716             if (accelerationFactor == ACCELERATION_FACTOR)
717             {
718               accelerateXY(mq);
719               accelerationFactor = 0;
720             }
721 #else
722             accelerateXY(mq);
723 #endif
724
725             fStopExtruder(mq);
726           }
727         }
728         else
```

Figure 4.25: Extrait du code d'accélération non linéaire

Les résultats obtenus après avoir implémenter cet algorithme était remarquablement efficace dans la réduction des vibrations.

La technique utilisée pour générer cette vitesse non linéaire ne dépend pas d'une équation. C'est grâce l'utilisation d'un timer du microcontrôleur seulement, cela permet un gain de temps considérable. On a essayé d'approximer les données du graphe de la (Figure 4.23) avec une fonction exponentielle. Le résultat est l'équation suivante :

$$y = 1.25 \times \left(e^{\frac{x}{11.5}} - e^{0} \right) + 11.228$$

Dans la (Figure 4.26) on remarque que les deux graphes sont similaires mais manquent un peu d'exactitude.

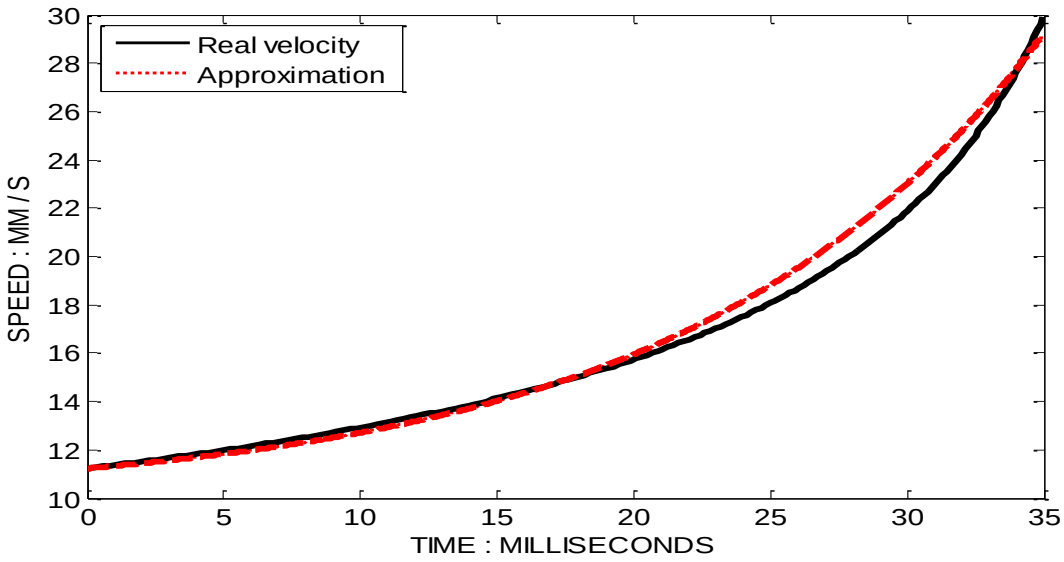

Figure 4.26: Vitesse réelle vs approximation exponentielle

Nous avons ensuite refait le test en utilisant cette fois une fonction polynomiale de cinquième ordre. Voici l'équation trouvée:

$$y = 2 \times 10^{-6}x^5 - 0.0001x^4 + 0.003x^3 - 0.0291x^2 + 0.2668x + 11.112$$

Comme le montre la (Figure 4.27), cette équation est bien plus exacte que la précédente.

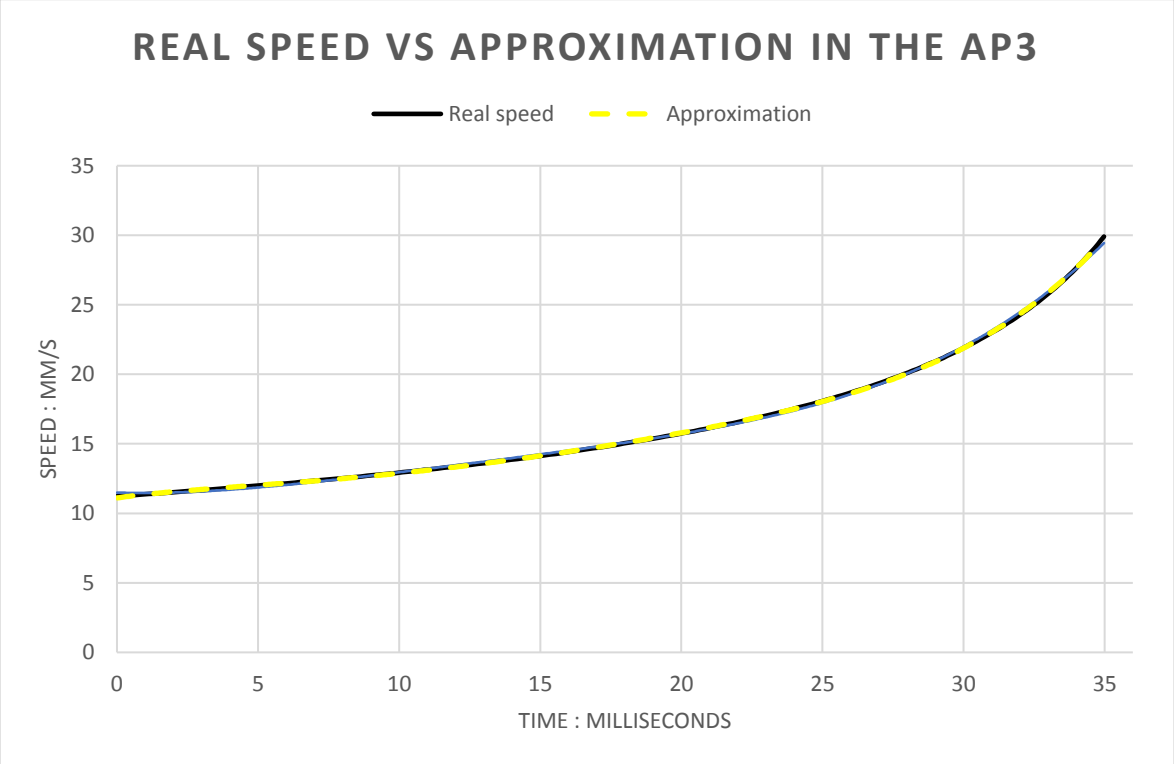

Figure 4.27: Vitesse réelle vs approximation polynomiale de 5éme ordre

4.5.3. Implémentation des programmes d'acceleration

```
#define mq_head mq.mouvements_array[mq.head]

inline void calculateXYAcceleration(MouvementQueue &mq)
{
  if (mq.compareDxDy() && mq.getDx_state(mq.head))
  {
    mq.updateImp_Start();
    mq.updateImp_Finish();
    mq.AccelerationCalculationX();
    mq.setAcceleration_x16(mq.getAcceleration(mq.head) * 16);
    mq.setImp_value_x16(mq.getImp_value(mq.head) * 16);
    mq.setImp_start_x16(mq.getImp_start(mq.head) * 16);
    mq.setImp_size(mq.getImp_start_x16(mq.head), mq.head);
  }
  else if (mq.getDy_state(mq.head))
```

Figure 4.28: Extrait du code d'accélération

Un extrait d'un code qui permet d'utiliser l'algorithme d'accélération et de décélération.

4.5.4. G-Code Bézier

On a implémenté les courbes de Béziers dans un programme écrit en Java « **G-Code Bézier** », ce dernier modifie le fichier G-code afin de diminuer les vibrations (Figure 4.30).

```
54 ////////////////////////////// BEGIN FUNCTION ////////////////////////////////
55 void GcodeBegin() {
56   Accu="";
57   index = 0;
58   lines = null;
59 }
60
61 ////////////////////////////// SELECTER FUNCTION ////////////////////////////////
62
63 void GcodeSelecter() {
64   GcodeBegin();
65   File file = null;
66   println("Loading the file please wait...");
67   selectInput("Select a file to simulate", "GcodeSender", file);
68 }
69
70 ////////////////////////////// SENDER FUNCTION ////////////////////////////////
71
72 void GcodeSender(File selection) {
```

Figure 4.29: Extrait du code du G-Code Bézier

4.5.5. Résultat des de la simulation

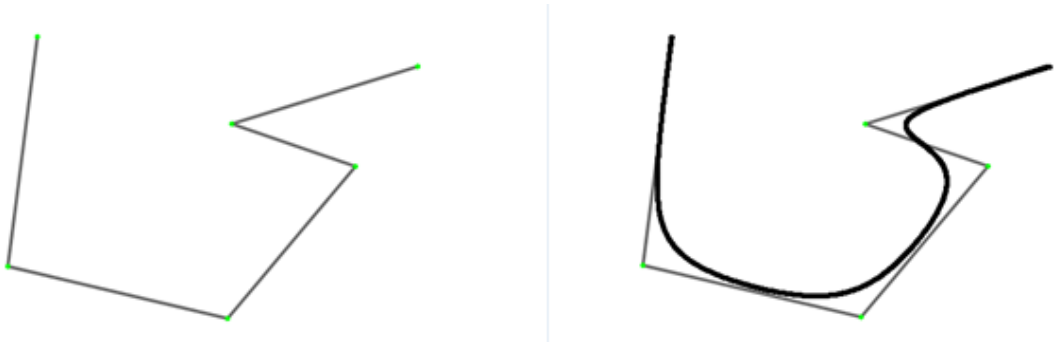

Figure 4.30: Exemple d'application des courbes de Bézier dans le G-code Bézier.

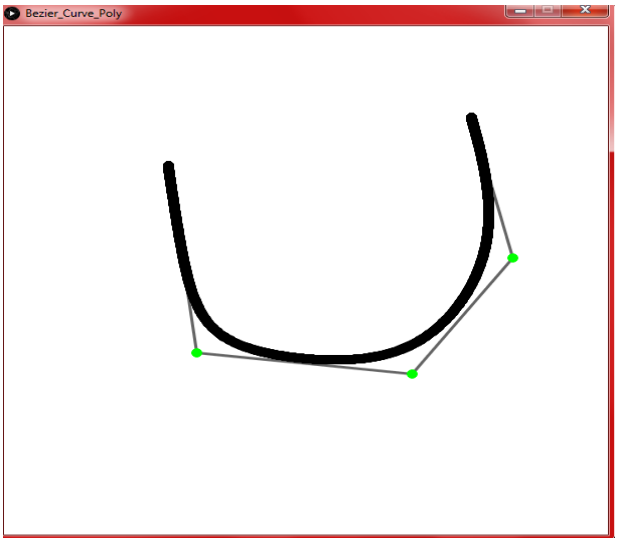

Figure 4.31: Exemple G-Code Bézier.

4.5.6. G-Code Futurs mouvements

```
10  Point() {
11    X_array[0] = 0;
12    Y_array[0] = 0;
13    Z_array[0] = 0;
14    XYZ_position_index = 0;
15  }
16
17  void PointsShift() {
18    for (int i = 0; i<2; i++) {
19      X_array[i] = X_array[(i+1)];
20      Y_array[i] = Y_array[(i+1)];
21      Z_array[i] = Z_array[(i+1)];
22    }
23  }
24
25  void AddPoints(float a, float b, float c) {
```

Figure 4.32: Extrait du code du G-Code Futurs Mouvements

Un extrait d'un code qui permet d'utiliser l'algorithme des futurs mouvements.

Les résultats obtenus après avoir implémenter cet algorithme était remarquablement efficace dans la réduction des vibrations.

4.6. Implémentation des solutions pour respecter le temps réel

En informatique, on parle d'un système temps réel lorsque ce système est capable de contrôler (ou piloter) un procédé physique à une vitesse adaptée à l'évolution du procédé contrôlé.

4.6.1. Définition du temps réel

Les systèmes informatiques temps réel se différencient des autres systèmes informatiques par la prise en compte de contraintes temporelles dont le respect est aussi important que l'exactitude du résultat, autrement dit le système ne doit pas simplement délivrer des résultats exacts, il doit les délivrer dans des délais imposés.

Pour tout système de *n* tâches, la condition suivante est nécessaire mais pas suffisante à sa faisabilité :

- *Ci* : Le temps de calcul de la tache *n° i.*
- *Ti* : Sa période.

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Une valeur supérieure à 1 signifie que le système nécessite plus de temps d'exécution que le processeur ne peut en fournir.

Dans le cas de notre imprimante 3D, à la fin de chaque exécution de mouvement la machine demande un nouveau mouvement. Dans ce cas, le processeur doit être capable de fournir la nouvelle donnée avant que la précédente donnée ne soit complément exécutée.

En d'autre mot, le processeur doit avoir reçu et traité la nouvelle donnée avant la fin d'exécution de la précédente, c'est qu'on appelle un système en temps réel.

Pour arriver à ce résultat, il a fallu optimiser chaque ligne de code, utiliser minutieusement la mémoire, les timers et les interruptions, ainsi une bonne architecture du Firmware est indispensable.

4.6.2. Flow continu

On désigne par flow de données, une disponibilité de données continues. D'autre part, une coupure de données (un flow discontinu) est très nocive pour la machine, car si elle vient à demander une nouvelle commande pour l'exécuter et qu'elle ne la reçoit pas immédiatement, cette dernière s'arrête. Par conséquent les moteurs de la machine seront aussi à l'arrêt pendant un certain temps (Temps d'arrêt entre les commandes). Ce dernier est égal à l'accumulation des temps suivant :

- Le temps que la tâche responsable de la réception de données demande une nouvelle commande.

- Le temps que celle-ci soit traité par une tâche afin de comprendre la nature de la donnée reçue.
- Le temps pour que la commande soit préparé pour l'exécution.

Généralement ce temps va être compris entre quelques centaines de microsecondes jusqu'à une centaine de millisecondes, il dépend de plusieurs paramètres.

4.6.3. Influence de la vitesse d'un microcontrôleur

La vitesse du microcontrôleur dépend principalement du type d'architecture RISC ou CISC, de la taille du bus de données, de la vitesse d'horloge, du MIPS, etc… Tous ces paramètres influent sur la vitesse d'exécution des instructions du CPU et par conséquent de la vitesse d'exécution du Firmware de la machine.

Cela est d'une importance capitale, par exemple si on a une instruction qui prend 2 cycles d'horloge; que l'horloge est cadencée à 1 Mhz sur un microcontrôleur A, alors son temps d'exécution est égal à 1 us. La même instruction est exécutée sur un autre microcontrôleur B mais ce dernier est cadencé à 2 MHZ, alors son temps d'exécution est égal à 0.5 us. A priori une différence de 0.5 us n'est pas très gênante. Dans la plupart des cas, si cette instruction n'est exécutée qu'une seule fois elle n'influe pas sur le bon fonctionnement du système.

Le problème survient par exemple lorsqu'on exécute cette instruction 100 fois dans une routine d'interruption ayant une période de 80 us. Le microcontrôleur A prendra 100 us à chaque fois que la routine d'interruption est appelée, le second prendra 50 us. On remarque qu'avec le premier cas cette opération devient impossible en temps réel car le temps d'exécution 100 us est largement supérieure à la période 80 us, mais dans le deuxième cas il est toute à fait possible car le temps d'exécution 50 us (sans prise en compte de l'overhead de l'appel de la ISR) est inférieure à la période 50 us.

Le choix du microcontrôleur est alors une décision très cruciale dans le succès d'un projet. Malheureusement on n'a pas toujours le choix ni les moyens de se permettre un « meilleur » microcontrôleur soit pour des problèmes techniques (manque d'expérience envers une architecture d'un microcontrôleur précis), des problèmes financiers ou de disponibilité des composants.

Dans notre cas pour résoudre ce problème sans avoir recours à changer de matériels on a mis au point des techniques et des méthodes spécialement conçues pour notre machine. On a aussi opté pour des techniques d'optimisation du code Firmware, nous n'allons pas parler de cette méthode dans ce mémoire car c'est tout un univers. Cependant nous allons juste citer quelques points pour vous orienter dans vos recherches :

- Utilisations des variables 8 bits dans un microcontrôleur 8 bits quand c'est possible est une sorte d'optimisation.
- Comprendre les Compilateurs aide beaucoup le développeur à optimiser le programme.
- Connaitre l'architecture interne peut servir au développeur d'optimiser le du code.
- Cependant l'optimisation du code diffère du contexte et dépend de l'ingéniosité et de l'expérience du développeur).

4.6.4. Temps d'arrêt entre les commandes

On avait dit que ce temps était variable et compris entre un certain intervalle. Plus on réduit ce temps, plus il sera bénéfique pour la machine. Afin d'optimiser notre programme et pour le bon

fonctionnement optimal de notre machine il faut identifier les principaux facteurs qui influents sur ce temps.

La vitesse de transmission série de données est le premier facteur. Elle s'exprime en baud c'est-à-dire bits par seconde, plus la vitesse de transmissions est grande plus les données arrivent plus vite vers la machine, plus vite elles sont traitées et exécutées. A priori on n'a qu'à choisir la plus grande valeur possible, mais malheureusement ce n'est pas le cas, car à chaque fois qu'une donnée arrive au hardware il faudrait la lire en software pour l'utiliser. Donc on en déduit que la vitesse de transmission série de données dépend de la vitesse du microcontrôleur et de l'algorithme qui est responsable de la réception. Nous parlerons plus en détail de ces points-là plus loin dans ce mémoire dans la partie « temps de réceptions des données ».

Le deuxième facteur, c'est la vitesse du microcontrôleur. Cet aspect est déjà traité plus haut dans la partie « Influence de la vitesse d'un microcontrôleur ».

Le troisième facteur est la taille de la donnée. Plus la donnée est longue plus il y a de caractères a envoyé à partir de l'interface série, plus ça prendra du temps

s. La technique utilisée pour optimiser sera présentée et expliquée plus tard dans la partie « G-code Preprocessor ».

Le dernier facteur c'est le temps de traitement de la donnée une fois arrivé. Plus il y a de traitements à faire, plus ça sera long. Connaitre les bonnes pratiques pour écrire un code optimal est donc fortement recommandé. Regarder le code assembleur produit par le compiler est toujours bénéfique, et parfois nécessaire.

4.6.5. Pipeline dans le Firmware

Trois étapes sont cruciales dans les machines à commandes numériques :

- La réception de la donnée.
- Le traitement de la donnée.
- L'exécution de la donnée.

L'objectif du pipeline c'est de pouvoir exécuter plusieurs taches simultanément comme le montre la (Figure 4.33).

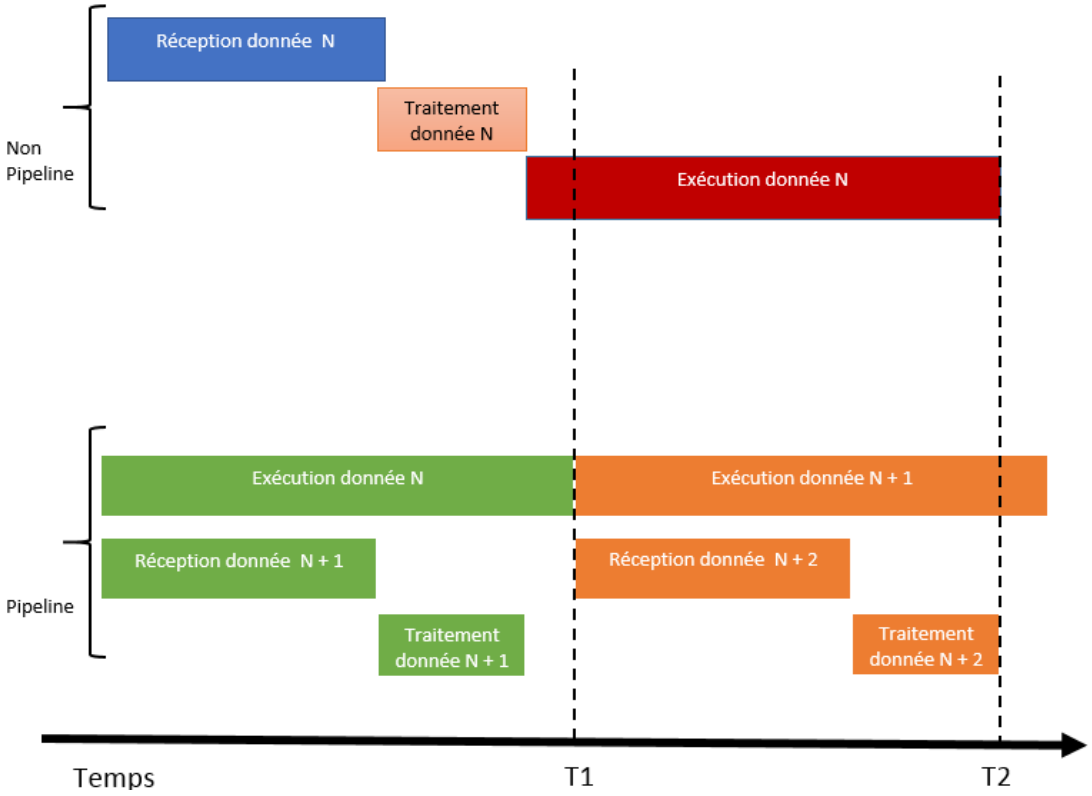

Figure 4.33: Principe de fonctionnement du pipeline du Firmware.

L'exécution avec le pipeline, permet de gagner beaucoup de temps. En général le gain est égal au temps de réception plus le temps de traitement.

Ce système de pipeline à deux étages est implémenté sur un Atmega2560, en combinant l'architecture interne du microcontrôleur, avec les techniques d'interruptions et d'ordonnancement implémentés dans la programmation.

L'architecture de l'ATmega 2560 permet la réception et le transfert de données à partir du port UART d'être quasiment indépendantes du software.

Cependant, ll est toujours nécessaire d'initialiser le UART dans le software. Ensuite le microcontrôleur se charge de la génération du clock de synchronisation, du contrôle des erreurs et du décalage des bits pour la réception ou l'envoie des données. Quand la transmission ou la réception est finie, le microcontrôleur met un bit flag a 1. Ce dernier peut générer une interruption s'il est configuré dans le software.

Grace à cette méthode le temps d'attente entre l'exécution de deux données est devenu négligeable, réduisant considérablement les vibrations.

Une autre méthode a été ajoutée pour réduire le temps de réception et le temps de traitement et consiste à faire des prétraitements sur le ficher G-code avant de l'envoyer à la machine. Cette technique est implémentée dans le G-code Preprocessor.

4.6.6. Le G-code Preprocessor

Pour réduire le temps de traitement et le temps de réception de données on a créé un programme en Processing (mode Java) qui peut:

- Supprimer les données inutiles du fichier G-code (par exemple les commentaires).

- Réduire et compresser les données à envoyer.
- Transformer toutes les positions/déplacement en entiers, pour ne pas avoir à utiliser les points virgules dans le microcontrôleur.
- Algorithme d'accumulation d'erreurs (voir précédemment).

Ces derniers permettent de réduire considérablement le temps de traitements et de réception.

```
148     GcodeSelecter();
149     break;
150   case 's':
151     {
152       String btsvariable = (String) JOptionPane.showInputDialog(frame,
153         "Name the file",
154         "Save file",
155         JOptionPane.WARNING_MESSAGE
156         );
157       if ( btsvariable != null ) {
158         background(240, 120, 60);
159         text("Saving file, please wait...", 180, 150);
160         println("Saving file, please wait...");
161         String[] list = GCODE.ShowPCode().toArray(new String[0]);
162         saveStrings(btsvariable+".txt", list);
```

Figure 4.34: Extrait du code G-Code Preprocessor

4.6.7. Resultat du G-Code Preprocessor

Si on compare la méthode classique et la méthode avec le G-code Preprocessor on trouve qu'on a un gagne trois plus de temps.

En prend comme exemple un pas par millimètre égale à 400, on souhaite déplacer les axes X et Y de la machine de 1 mm.

- Dans la méthode classique, il faut envoyer à la machine la ligne G1 X1.000 Y1.000, le traitement se fait en 202 µs.
- Dans la méthode G-code Preprocessor, il faut envoyer à la machine la ligne G1 X400 Y400, le traitement prend 67 µs.

4.6.8. Temps de réception de données

Le temps de réception est un élément clé, plus il est court plus en a de chance de respecter les conditions des systèmes en temps réel. Nous avons utilisé la communication UART pour communiquer avec le PC. L'ATmega 2560 16 Mhz peut avoir une vitesse de communication UART arrivant jusqu'à 2M bps (Tableau 1 dans l'annexe B).

Cependant, on n'a pas réussi à avoir plus que 500K bps.

Connaissant la vitesse de communication on peut facilement déduire la condition du système en temps réel :

Temps de traitement + Temps de réception < Temps d'exécution.

- Temps de traitement = 64 µs.
- Temps de réception = Nombre de caractères x Taille du caractère en bits / Vitesse de communication.

- Temps d'exécution = Distance x Vitesse de la machine.

On remarque que plus la distance est courte, plus il est difficile de respecter les conditions du temps réel, alors la solution est d'implémenter une file de données.

Cependant, l'utilisation de l'algorithme des courbes de Bézier, ne garantit pas toujours un système en temps réel, car il induit une création de nouvelles lignes très petites. Ces dernières peuvent avoir un temps d'exécution inferieure au temps de réception plus le temps de traitement.

La solution est d'implémenter l'algorithme au niveau du microcontrôleur, mais cela nécessite aussi l'implémentation d'une file de données. La taille de la file doit être au minimum égale ou supérieure aux degrés de la courbe de Bézier.

4.6.9. Algorithme de sélectionneur et de trieur de mouvements

```
876
877  void Motors::mouvementSorter(MouvementQueue &mq)
878  {
879
880    switch (mq_tail.mouvement_state_selection)
881    {
882    case ms_null:
883      mq.removeAvailable_mouvement();
884      mq.incrementTail();
885      if (mq.getAvailable_mouvement())
886      {
887        mouvementSorter(mq);
888      }
889      if (Flags::getFlag_machine() == 2)
890      {
891        Flags::setFlag_machine(RECEIVING);
892      }
893      break;
```

Figure 4.35: Extrait du code de l'algorithme du sélectionneur

Extrait d'un code qui permet l'algorithme du sélectionneur et du trieur de mouvement implémenter dans le Firmware qui permet de gagner en temps d'exécution et de précision.

4.7. Options de l'interface graphique

L'interface graphique a été divisée en quatre parties ou plus précisément quatre « panel » : le Tableau de bord, le Terminal, les Paramètres et le Contrôle manuel. On peut choisir à quel panel on veut accéder en cliquant sur l'un deux sur la barre de gauche, en plus de ces quatre panels il y a aussi sept options en dessous : Ouvrir un port, Démarrer l'impression, Choisir un fichier, Voir un fichier, Préprocesseur du G-code, Simulateur du G-code, Fermer le port. A présent nous allons vous expliquer plus en détails les options, et les fonctionnalités de l'interface graphique.

```
private void JP_DashboardMouseClicked(java.awt.event.MouseEvent evt) {
    JP_CardLayout.removeAll();
    JP_CardLayout.repaint();
    JP_CardLayout.revalidate();

    JP_CardLayout.add(JP_DashboardFrame);
    JP_CardLayout.repaint();
    JP_CardLayout.revalidate();
    if (portState) {
        LB_MotorSpeed.setText("              " + pubg.motorSpeed);
        LB_ExtrusionSpeed.setText("              " + pubg.extruderSpeed);
        LB_Temperature.setText("              " + pubg.temperature);
        LB_Bluetooth.setText("       " + pubg.bluetoothState);
        LB_LimitSwitch.setText("       " + pubg.limitSwitchState);
        if (pubg.flag_bluetooth) {
            jPanel15.setBackground(Color.green);
        }
        if (pubg.flag_limitSwitch) {
            jPanel16.setBackground(Color.green);
        }
    }
```

Figure 4.36: Extrait du code de l'interface graphique

Extrait d'un code de l'interface graphique qui permet de contrôler plusieurs aspects de la machine.

4.7.1. Tableau de bord

Dans le tableau de bord (Figure 4.37) on peut voir l'état du port série, s'il est ouvert ou fermé. On peut aussi voir la vitesse des moteurs X et Y, la température et la vitesse de l'extrudeuse, l'état du Bluetooth, les capteurs de fins de courses en plus de quelques informations sur l'imprimante.

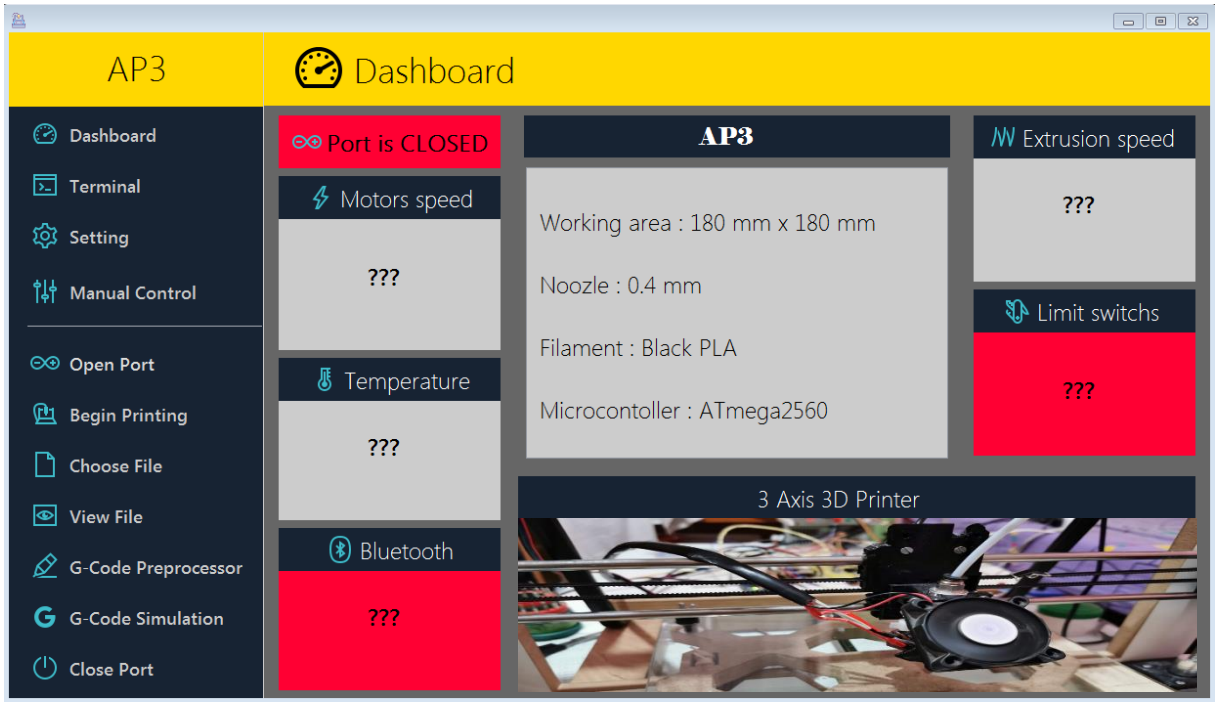

Figure 4.37: Tableau de bord du GUI (imprimante non connectée)

Ces valeurs sont par défaut, quand la machine est déconnecté égale à « ??? », ils sont mis à jour dès que le port série devient ouvert (Figure 4.38).

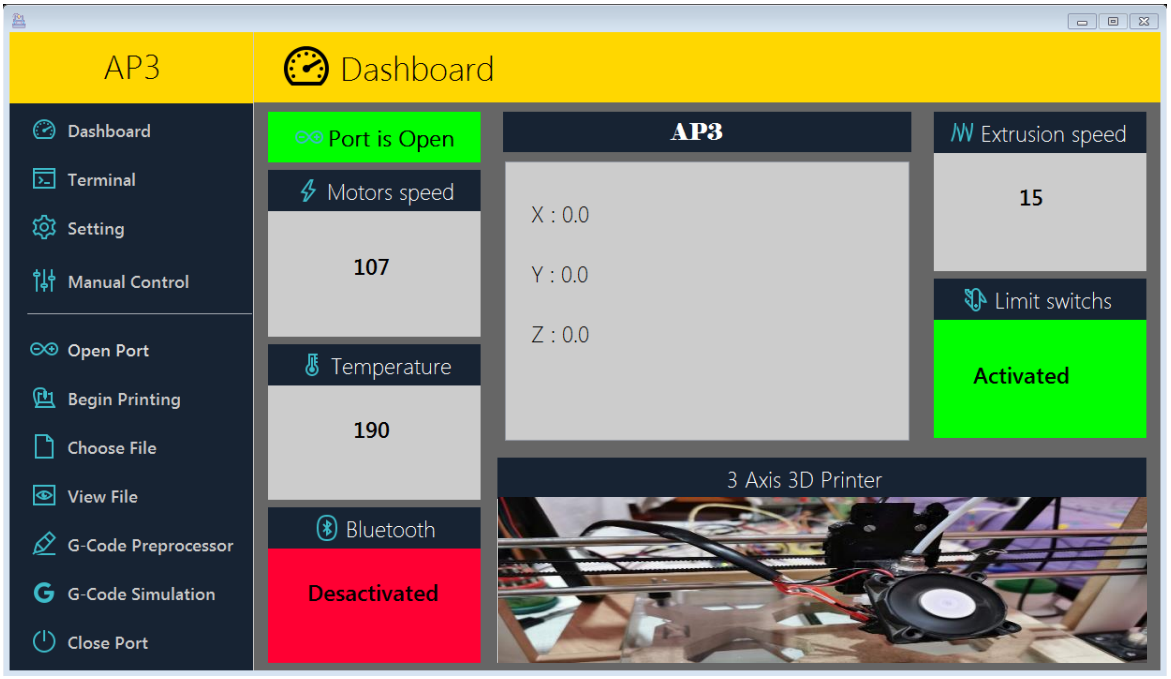

Figure 4.38: Tableau de bord du GUI (l'imprimante connectée)

En bas à gauche il y'a sept options :

- Ouvrir port : elle permet de choisir un port pour établir une communication série.
- Commencer l'impression : elle permet d'imprimer un objet.
- Choisir un fichier : permet de choisir un fichier G-code à imprimer.
- Voir fichier : permet de visualiser le fichier à imprimer.
- G-code Preprocessor : permet d'ouvrir le G-code Preprocessor.
- G-code Simulation : permet d'ouvrir le simulateur.
- Fermer port : permet de mettre fin à la communication série.

4.7.2. Terminal

Le terminal (Figure 4.39) est un point d'accès de communication entre l'utilisateur de l'imprimante et le Firmware de l'imprimante. On peut voir les commandes qui sont envoyées à ou reçues à partir de l'imprimante.

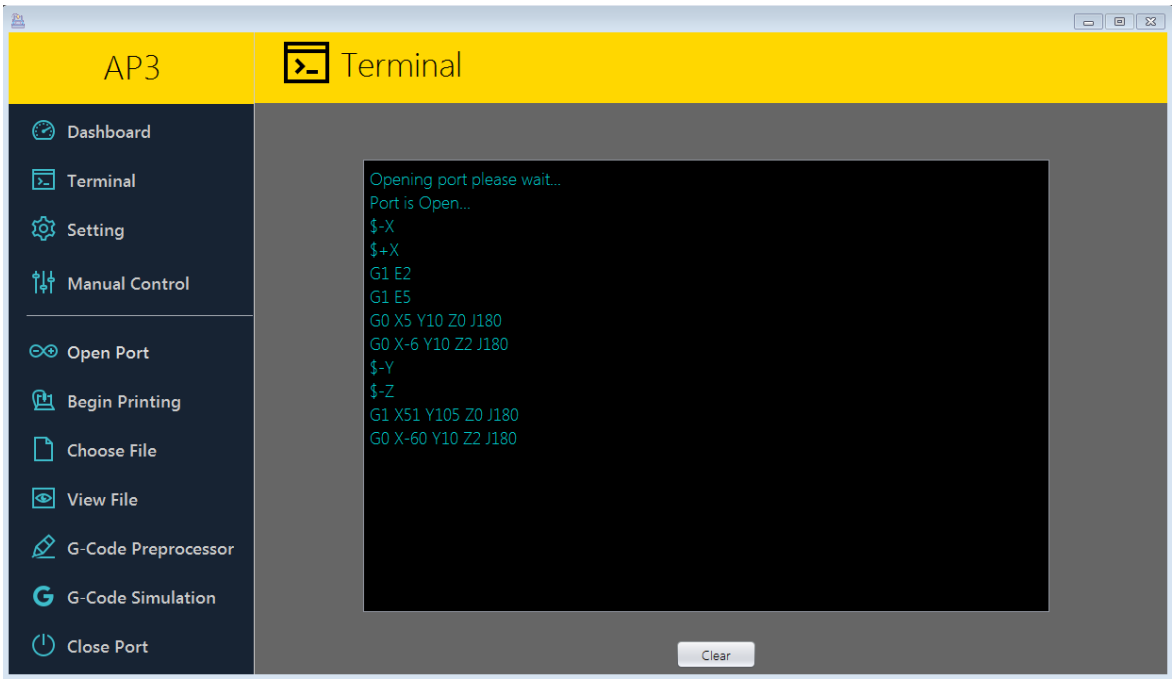

Figure 4.39: Terminal de l'interface graphique

4.7.3. Paramètres

Concernant les paramètres (Figure 4.40) il y a :

- Les pas par millimètres : permet de modifier les pas par millimètre des axes X, Y, Z.
- Vitesse moteur : permet de modifier la vitesse des moteurs des axes X et Y.
- Hotend : permet d'allumer ou éteindre l'extrudeuse.
- Température : permet d'ajuster la température de l'extrudeuse.
- Vitesse d'extrusion : permet de modifier la vitesse d'extrusion filament.
- Auto Bed Leveling : une fonction qui permet de régler la hauteur du plateau d'impression afin qu'il soit calibré et bien droite et perpendiculaire avec la tête d'extrusion.
- Nouvelle origine : permet de modifier l'origine x, y, z de l'imprimante 3D.
- Accélération : permet de modifier l'accélération des moteurs des axes X et Y.
- Jerk : permet de modifier la vitesse minimale de l'imprimante.
- G Mode : permet de changer entre deux modes, un mode ou les pas sont calculés à l'avance, l'autre mode est le mode normal ou les pas seront calculés dans le Firmware.
- Bluetooth : permet d'activer ou de désactiver le Bluetooth.
- Fin de course : permet d'activer ou de désactiver les capteurs de fin de courses.
- Ventilateur : permet d'éteindre ou d'allumer le ventilateur.
- Calcule : permet de calculer la vitesse et l'accélération de la machine en se basant sur les pas par millimètres et sur la période du signal d'impulsion du microcontrôleur.

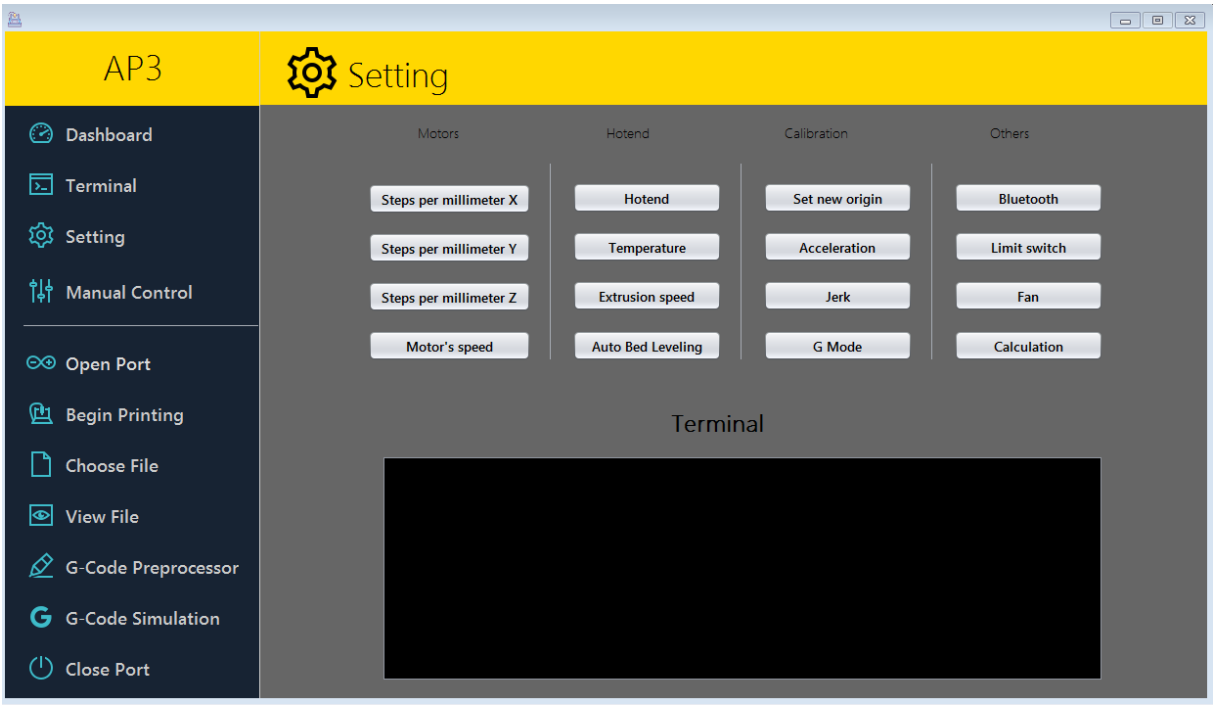

Figure 4.40: Paramètres de l'interface graphique.

4.7.4. Contrôle manuel

Il est impératif pour toute machine à commande numérique de posséder un système de contrôle manuel. Pour cela on a incorporé cette fonctionnalité dans l'interface graphique (Figure 4.41), en créant plusieurs boutons qui peuvent:

- Avancer ou reculer les axes X, Y, Z.
- Extruder ou rétracter du filament.
- Origine : permet de repositionner l'origine x, y, z.
- Maison : permet d'envoyer la machine vers son l'origine.
- Pause : suspendre l'impression.
- Start : reprendre l'impression.
- Restart : redémarrer l'impression.
- G0 : envoyer la machine vers une position x, y, z.
- G1 : envoyer la machine vers une position x, y, z en extrudant du filament.
- AU : arrêt d'urgence.

Dans le panel Contrôle manuel, Il y a aussi un petit terminal.

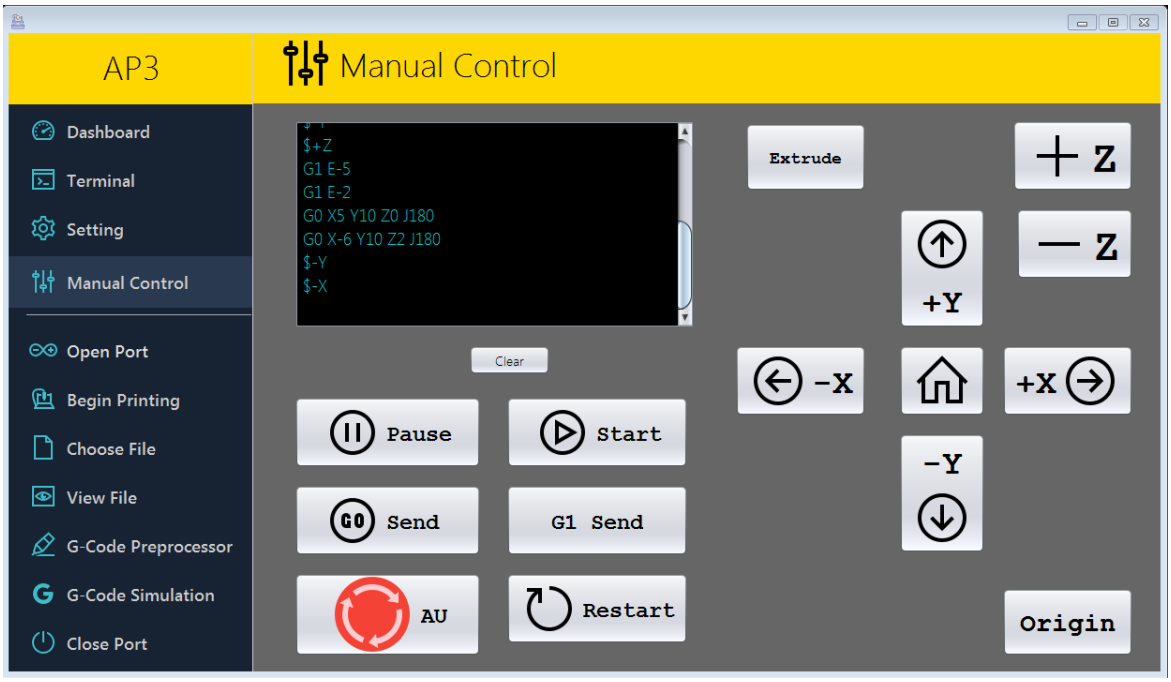

Figure 4.41: Contrôle manuel de l'interface graphique

4.8. Exemple d'implémentation de Firmware

4.8.1. Tâche de réception de donnée série

```
7   void SerialReceiver::fReceiving0()
8   {
9     SerialReceiver::setReceiving_state(true);
10  #if _PROTEUS == 1
11    RX_queue.buffer_write(UDR0, '\r');
12  #else
13    RX_queue.buffer_write(UDR0, '\n');
14  #endif
15  }
16  /*******************************************
17
18  void SerialReceiver::fReceiving1()
19  {
20  #if _PROTEUS == 1
```

Figure 4.42: Extrait du code de la tache réception de donnée

Un extrait d'un code qui permet de recevoir des données à partir du port UART.

4.8.2. Tâche d'arrière-plan

```
71      sendCommand(0x80);
72    }
73
74     void setCursor(const uint8_t &row, const uint8_t &col)
75    {
76      if(row == 1){
77            sendCommand(0x80 | col) ;
78      }
79      else{
80            sendCommand(0x80 | (0x40 + col)) ;
81      }
82    }
83
84    char lcdBuffer[32];
85    uint8_t data_pins[4] = {PIN_LCD_D4, PIN_LCD_D5, PIN_LCD_D6, PIN_LCD_D7};
86
87    void inline sendChar(const uint8_t &value)
88    {
```

Figure 4.43: Extrait du code d'une des tâches d'arrière-plan

Un extrait d'un code qui permet de recevoir des données à partir du port UART.

4.8.3. Tâche de transfert de donnée série

```
14  String SerialSender::TX_inputString1;
15
16  /*************************************************************************/
17
18  void SerialSender::transferData()
19  {
20    if (FLAG_bufferReady && TX_inputString[TX_stringPointer] != 0)
21    {
22      UDR0 = TX_inputString[TX_stringPointer++];
23    }
24    else
25    {
26      UCSR0B &= ~_BV(TXCIE0);
27      FLAG_bufferReady = 0;
```

Figure 4.44: Extrait du code de la tâche transfert de données

Un extrait d'un code qui permet de recevoir des données à partir du port UART.

4.9. Etapes d'impression

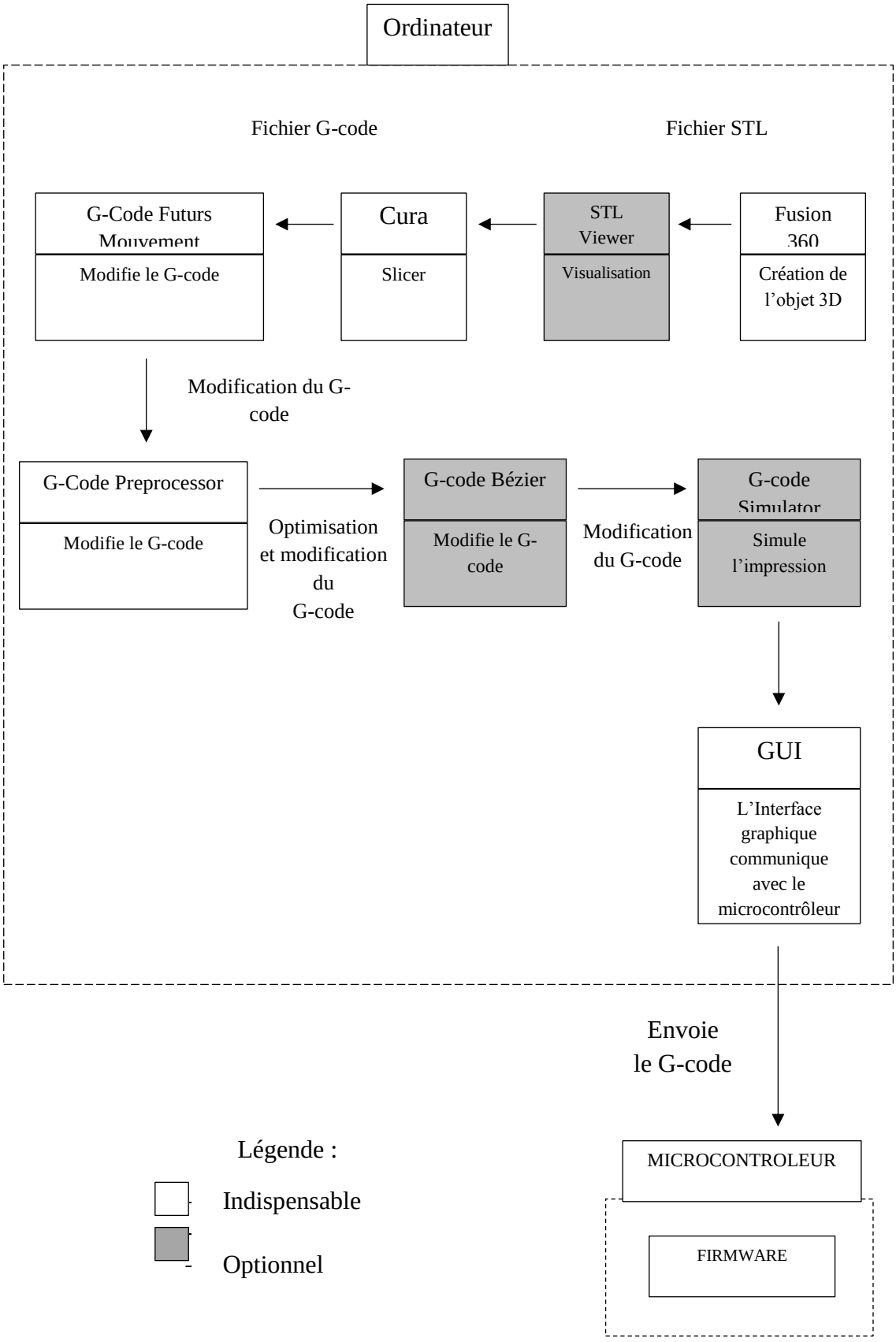

Figure 4.45: Schéma synoptique des étapes d'impression

Les étapes sont successivement:

1- Créer le modèle en 3D dans le logiciel Fusion 360.
2- Générer le fichier G-Code l'objet 3D avec le logiciel Cura.
3- Utiliser le programme qu'on a créé le G-Code Futurs Mouvements pour appliquer l'algorithme Futurs Mouvements au G-Code.
4- Utiliser le programme qu'on a créé le G-Code Preprocessor pour appliquer l'algorithme d'accumulation d'erreurs.
5- Optionnel : Utiliser le programme qu'on a créé le G-Code Bézier pour appliquer les courbes de Béziers.
6- Optionnel : Utiliser le programme qu'on a créé le G-Code Simulator pour simuler l'impression de l'objet.
7- Utiliser l'interface graphique qu'on a créé pour contrôler la machine et envoyer le ficher au microcontrôleur (l'imprimante 3D).
8- Le Firmware qu'on a créé est présent dans le microcontrôleur. Il permet de contrôler l'électromécanique de la machine pour imprimer l'objet.

4.10. Discussion des problèmes et des résultats

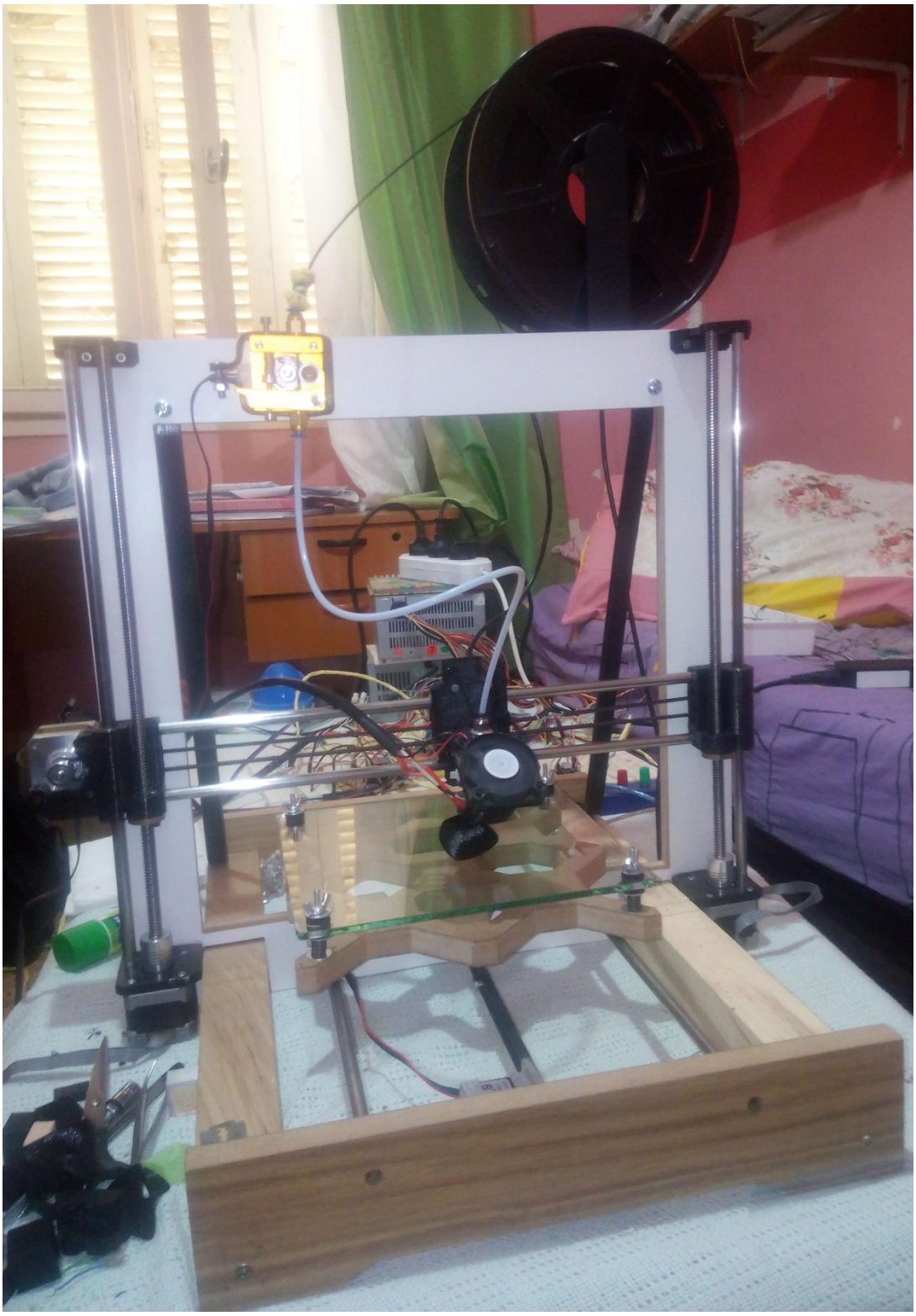

Figure 4.46: L'imprimante assemblée et fonctionnelle

On a eu beaucoup de difficulté dans la conception de l'architecture du Firmware parce qu'on n'a pas trouvé de modèles d'architecture disponible sur internet. Cependant, il y a des Firmware open-source sans documentation sur l'architecture en question ou les détails comme qu'on vous a

présenté. La littérature existante est très basique et ne correspond pas à notre objectif. Elle s'adresse surtout à ceux qui veulent utiliser le Firmware pour une imprimante 3D. On cite comme exemple le Firmware « Marlin » l'un des meilleurs pour ne pas dire le meilleur Firmware open-source pour imprimante 3D en ce moment. La conception de « Marlin » a commencé en 2011 en se basant sur le Firmware « Sprinter » et « GRBL ». Si on regarde les fichiers sources, on trouve beaucoup de « preprocessor directives », ça rend à notre avis le code très difficile à comprendre. Dans tous les cas cela prendrait énormément de temps juste pour lire les quelques 300,000 lignes de leur code. On a donc créé notre propre architecture.

On ne parlera pas assez de l'importance des bonnes pratiques de programmation surtout quand le projet devient complexe. Ces pratiques-là doivent être apprises et respectées, sinon le temps perdu pour rechercher un bug ou pour ajouter une nouvelle fonction sera colossal.

Un des grands problèmes rencontrés consiste en l'impossibilité de debugger, car il n'y a pas de méthode pour le faire dans un Arduino Mega. Notons que pour un système qui tourne en temps réel et à une vitesse rapide, il est très difficile d'afficher quelque chose pour la voir. Le LCD prend minimum 50 us pour afficher un caractère, alors il est impossible d'afficher des données concernant la tâche « routine des moteurs » (car elle s'exécute périodiquement toutes les 53 us).

Un autre problème consiste dans le fait qu'on ne dispose pas d'oscilloscope qui nous permet de visualiser les signaux générés par le microcontrôleur pour comprendre les problèmes qui surviennent. Afin de résoudre ce problème, on a simulé le Firmware dans « Proteus » et grâce aux outils que ce dernier fourni, on a pu simuler, debugger, prédire et éviter quelques problèmes.

Un autre problème majeur nous a pris beaucoup de temps. Parfois on a des problèmes dans la réalité mais dans la simulation ces problèmes ne ressortent pas. Pour cela il est impératif d'avoir une architecture solide et viable et de bonnes pratiques de programmation. Car si un problème survient du côté software, il est très difficile de l'identifier.

Une variable qui dépasse sa capacité, une autre qui fut oublié à mettre à volatile, un casting de variable qui ne passe pas etc... Ces problèmes peuvent avoir un impact majeur dans les performances de l'imprimante 3D et sont très difficiles à détecter, car le Compiler ne les signale pas toujours comme des erreurs.

Il y a aussi le problème de la vitesse du microcontrôleur, l'ATmega 2560 8-bits, cadencé à 16 MHz, cela limite ce qu'on peut faire avec. Pour augmenter les performances du Firmware il faut utiliser des techniques d'optimisations, comprendre la fonction des « Compilers » et regarder le code assembleur généré. Pour ce projet on est passé du standard C++ 11 vers le standard C++ 14, principalement à cause de la fonction « constexp », qui est bien plus flexible et utile pour avoir un code optimisé à la compilation.

Les problèmes qu'on a cités ne sont pas spécifiques à la conception d'un Firmware pour imprimante 3D, mais concerne l'ensemble des projets centrés sur un système embarqué.

En fin de compte, l'imprimante 3D marche très bien, les commandes sont bel et bien reçues, traitées et remplies dans la file de mouvements. Il n'y a pas d'over-flow du buffer de réception. Tous les algorithmes marchent bien, l'accélération et décélération de la machine sont parfaites. L'algorithme des futurs mouvements ajoute énormément de stabilité à la machine. Le seul problème concerne l'axe Y, qui présente un jeu dans les roulements linéaires. Malheureusement c'est un problème de fabrication non solvable. On a eu aussi des problèmes d'adhésion à la plaque

d’impression car elle faite de verre, ce dernier ne qui colle pas très bien avec le PLA, on a dû utiliser du ruban adhésif.

Figure 4.47: Un objet 3D imprimé grâce à l’imprimante 3D

La qualité d’impression de la machine est comparable à celles trouvée sur le marché.

4.11. Conclusion

Dans ce chapitre, on a montré l'implémentation soft et hard de l'imprimante 3D, ainsi que les résultats des étapes successives de son fonctionnement pour la création d'un objet 3D. On a ensuite discuté les résultats obtenus et les problèmes rencontrés, que ce soit du côté software ou du côté hardware.

Conclusion générale

Le bon fonctionnement de l'imprimante dépend de plusieurs paramètres tels que : le design, la stabilité de la température, le Firmware, les composants électroniques utilisés, etc…

Les problèmes rencontrés dans la réalisation du projet nous ont permis de développer nos connaissances dans les différents domaines touchant à l'imprimante 3D.

Une meilleure connaissance en mécanique et en robotique nous aurait fait gagner certainement beaucoup de temps. Nous regrettons de n'avoir pas eu une formation dans ce domaine pour nous orienter, cela dit, ça nous a poussé à trouver d'autres solutions pour augmenter la précision sans changer de design. Ce qui n'est pas très courant dans le domaine de l'industrie. Au lieu de trouver des solutions software aux problèmes rencontrés, la plupart des entreprises investissent beaucoup d'argent dans du matériel de haute qualité. Le progrès de l'électronique et de l'informatique, en particulier l'intelligence artificielle pourrait bien changer cela.

Le manque de connaissance et de matériel, nous a poussés à chercher et à tester des solutions, en électronique et plus particulièrement en informatique. Les méthodes que nous avons conçues et implémentées afin d'optimiser le G-code, et les algorithmes créés pour augmenter la précision et réduire les vibrations ont été d'une efficacité satisfaisante.

Le Firmware fut la chose la plus délicate à mettre en place sur un microcontrôleur 8-bit. Ça nous a poussé à étudier en détails l'architecture interne du microcontrôleur et le fonctionnement des Compiler. Pour écrire du code convivial pour le compilateur, vous devez avoir une compréhension pratique des compilateurs. Certaines modifications simples apportées à un programme, comme la modification du type de données d'une variable fréquemment utilisée, peuvent avoir un impact important sur la taille du code, tandis que d'autres modifications n'ont aucun effet. Avoir une idée de ce qu'un compilateur peut et ne peut pas faire facilite une telle optimisation [49]. Les méthodes d'optimisation du code, l'utilisation minutieuse des ressources et l'ordonnancement des taches sont des points à ne pas prendre à la légère, tout cela afin d'avoir un système qui peut tourner en temps réel avec son maximum de potentiel.

La réalisation de l'imprimante 3D, celle qu'on trouve de plus en plus chez les particuliers, fut pour nous un véritable trésor de connaissances. En ce qui concerne les futurs projets il y a beaucoup de choses intéressantes :

Premièrement comme on l'a cité précédemment, le moteur de l'extrudeuse est considéré comme un comportement linéaire, ce qui est faux. La première chose qui semble logique à faire est de trouver un algorithme qui puisse corriger ce problème.

Deuxièmement il y a un problème dont nous n'avons pas parlé, c'est le « z-wobble », sa source est purement mécanique. Généralement il survient lors de la fabrication, il serait très intéressant de trouver un algorithme qui le corrige.

Troisièmement, il existe des publications qui parlent de la vitesse en S ou « s curve velocity ». Son incorporation dans notre imprimante 3D (si c'est possible) apporterait un grand plus.

Quatrièmement, il existe de nouvelles publications qui ont traité le sujet des imprimantes 3D 5 axes, fournissant une méthode pour la conception d'un slicer 5 axes. Pour cela il faudrait d'abord qu'on conçoit un slicer pour imprimante 3D 3 axes.

Pour conclure on espère que ce travail sera un guide dans la conception d'une machine à commande numérique. En surlignant les contraintes et les problèmes que vous allez trouver lors de la réalisation de votre projet, ainsi que les solutions que nous avons proposées, aussi bien dans la partie hardware que la partie software. L'architecture du Firmware mis en place n'est pas seulement réservée à l'impression 3D, on peut imaginer de nombreux projets et problèmes pouvant être résolus grâce à ce mélange de la programmation baremetal et de RTOS. Par exemple des projets où vous devrez avoir plusieurs tâches à faire tourner dans un temps très réduit. On espère aussi que les solutions, les méthodes et les algorithmes que nous avons conçus vous seront utiles et vous inspireront.

Bibliographie

[1] A. Bahi et D. Khaled, Conception et réalisation d'une machine à commande numérique à 3 axes, Editions universitaires europeennes EUE, 2015.

[2] M. Kurma et H. Lipson, Impression 3D, la révolution en marche, edi8, 2014.

[3] B. Luyt et M. Berchon, L'impression 3D, Eyrolles, 2014.

[4] «Impression 3D - Son histoire,» [En ligne]. Disponible sur: http://www.fabulous.com.co/guide-impression-3d/en-bref/histoire/. [Accès le 16 mai 2020].

[5] T. Côme et G. Rouet, Innovations managériales, enjeux e perspectives, L'Harmattan, 2015.

[6] J.-l. Charvolin, Conception des pièces plastiques injectées, Lavoisier, 2013.

[7] «Prototypage rapide: tout ce qu'il faut s'avoir,» Formlabs, 2020. [En ligne]. [Accès le 16 mai 2020].

[8] «Quels objets peut-on fabriquer avec une imprimante 3D,» [En ligne]. Disponible sur: http://www.primante3d.com/objet-impression-3d/. [Accès le 13 MAI 2020].

[9] D. Rieutord, Les robots terrestres parmi les hommes, Paris: L'Harmattan, 2017.

[10] L. Gaëtan, «Le procédé d'impression 3D FDM ou FFF,» 24 février 2015. [En ligne]. Disponible sur: https://www.a3dm-magazine.fr/news/fabrication-additive-polymeres/procede-dimpression-3d-fdm-fff. [Accès le 17 mai 2020].

[11] A. R. Prajapati, H. K. Dave et S. R. Rajpurohit, «Investigation on Quality of In-house Fabricated PLA for 3D Printing Application,» *Advances in Additive Manufacturing and Joining,* pp. 277-285, 2020.

[12] B. T. Dan, D. R. Khodos, O. Khairallah, R. Ramlal et Y. Budhoo, «The Effect of 3-D Printing Process on the Mechanical Properties of Materials,» 2017.

[13] Imprimante 3D fonctionnement - Comment ça marche ?.

[14] «Les materiaux dimpression 3D : plastiques, metaux...,» [En ligne]. Disponible sur: http://www.primante3d.com/materiaux/. [Accès le 12 mai 2020].

[15] A. Charles, P. M. Bassan, T. Mueller, A. Elkasser et S. G. Scholz, «On the Assessment of Thermo-mechanical Degradability of Multi-recycled ABS Polymer For 3D Printing Applications,» chez *Sustainable Desgin and Manufacturing,* 2019.

[16] O. Le Frapper, AutoCAD 2010 Des fondamentaux à la presentation détaillée, Edition ENI, 2009.

[17] Freelabster, «Qu'est qu'un fichier STL ?,» 03 Septembre 2017. [En ligne]. Disponible sur: https://www.freelabster.com/fr/blog/quest-quun-fichier-stl/.

[18] D. C. Planchard, Engineering Design with Solidworks 2019, Florida: SDC Publications, 2019.

[19] M. Chetto, Ordonnancement dans les systèmes temps réel, iste editions, 2014.

[20] T. Groussard, Java 6 Les fondamentaux du langage Java, Editions ENI, 2009.

[21] Y. Altintas, Manufacturing Automation, Cambridge University Press, 2012.

[22] M. Cheng, «How to Take Vibration out of Stepmotors,» 22 Novembre 2008. [En ligne]. Disponible sur: https://www.machinedesign.com/motors-drives/article/21832522/how-to-take-vibration-out-of-stepmotors.

[23] J. Tang, «Minimizing Stepper Motor Vibration,» 30 Septembre 2019. [En ligne]. Disponible sur: https://blog.orientalmotor.com/minimizing-stepper-motor-vibration.

[24] Z. Li, «Solutions to Reduce Stepper Motor Resonance,» 20 Septembre 2019. [En ligne]. Disponible sur: https://www.motioncontrolonline.org/content-detail.cfm/Motion-Control-Videos/Solutions-to-Reduce-Stepper-Motor-Resonance/content_id/3209.

[25] B. Knight, «Understanding Inertia Ratio and Its Effect On Machine Performance,» Novembre 2015. [En ligne]. Disponible sur: https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwiViffok_rpAhWrAWMBHVnACcEQFjACegQICxAD&url=https%3A%2F%2Fus.mitsubishielectric.com%2Ffa%2Fen%2Fsupport%2Ftechnical-support%2Fknowledge-base%2Fgetdocument%2F%3Fdocid%3D3E26SJWH3ZZR-4.

[26] D. W. Jones, «Stepping Motor Physics,» 1998. [En ligne].

[27] Cummins Generator Technologies, «AGN 183 - Rotor Inertia,» [En ligne]. Disponible sur: https://www.stamford-avk.com/sites/stamfordavk/files/AGN183_B.pdf.

[28] D. Collins, «How do I calculate the inertia of a servo-driven system,» 29 Septembre 2016. [En ligne]. Disponible sur: https://www.motioncontroltips.com/faq-how-do-i-calculate-the-inertia-of-a-servo-driven-system/.

[29] B. Aumaille, J2SE Les fondamentaux de la programmation Java, Editions ENI, 2002.

[30] J. Sun, V. Zimmer, M. Jones et S. Reinauer, Embedded Firmware Solutions, Apress Open, 2015.

[31] D. M. Bourg et G. Seemann, AI for Game Developers, O'Reilly Media, Inc, 2004.

[32] S. Jung, J.-g. Song et S. Kim, «A Study on Marker Overlapping Conrol for M2M-Based Augmented Reality Object Loading Using Bresenham Algorithm,» chez *Signal Processing and Multimedia*, 2010.

[33] P. Rey, DUT informatique - programmation orientee objet en c# (tome 6), Books on Demand, 2016.

[34] C. Mandin, Windows Server 2003 Installation, configuration et administration, Editions ENI, 2003.

[35] S. Rohaut, Algorithmique Techniques fondamentales de programmation, Editions ENI, 2009.

[36] E. F. Cood, «Multiprogramming,» chez *Advances in Computers*, vol. 3, New York, Academic Press, 1962, pp. 77-153.

[37] A. Gordon, N. Amit, N. Har'El, M. Ben-Yehuda, A. Landau, A. Schuster et D. Tsafrir, «ELI: bare-metal performance for I/O virtualization,» *ACM SIGARCH Computer Architecture News*, p. 411–422, 2012.

[38] T. L. Ross, D. M. Washabaugh, P. J. Roman, W. Cheung, K. Tanaka et S. Mizuguchi, «Method and apparatus for performing interrupt frequency mitigation in a network node». United States of America Brevet 6,115,775, 2000.

[39] B. P. Douglass, Design Patters for Embedded Systems in C, Elsevier, 2011.

[40] P. Rey, jQuery 3 avec Visual Studio Code, Books on Demand, 2019.

[41] Processing, «processing.org,» [En ligne]. Disponible sur: https://processing.org. [Accès le mai 2020].

[42] M. Renard, «Cura : tout les caractéristique du slicer 3D,» 27 Mars 2020. [En ligne]. Disponible sur: https://www.3dnatives.com/cura-slicer-3d/. [Accès le 17 mai 2020].

[43] A. Brillant, Ruby: Les fondamentaux du langage - Mise en oeuvre avec Ruby on Rails, Editions ENI, 2008.

[44] H. Böck, The Defenitive Guide to NetBeans Platform 7, Apress, 2012.

[45] Apache Netbeans, «Présentation de NetBeans IDE,» [En ligne]. Disponible sur: https://netbeans.org/features. [Accès le 17 mai 2020].

[46] Microchip, «Atmel studio7 | Microchip Technology,» [En ligne]. Disponible sur: https://www.microchip.com/mplab/avr-support/atmel-studio-7. [Accès le mai 2020].

[47] D. Collins, «Microstepping for Stepper Motors,» 17 Novembre 2017. [En ligne]. Disponible sur: https://www.linearmotiontips.com/microstepping-basics/.

[48] Microchip, «Stepper Motor | Motor Type | Motor Control | Microchip Technology,» [En ligne]. Disponible sur: https://www.microchip.com/design-centers/motor-control-and-drive/motor-types/stepper.

[49] J. Ganssle, The Firmware Handbook, Elsevier, 2004.

[50] O. G. Popa, Learn Firmware And Software Design, Corollary Theorems Ltd, 2005.

[51] Y. Mergy, Arduino-uno en pratique, Books on Demand, 2016.

[52] M. Jufer, Traité d'électricité Volume IX électromécanique, Press polytechniques et universitaires romandes, 2004.

[53] T. W. Schultz, C and the 8051, PageFree Publishing, 2004.

[54] «NTC thermistor >> Resistor Guide,» [En ligne]. Disponible sur: http://www.resistorguide.com/ntc-thermistor/. [Accès le 14 mai 2020].

[55] W. S. Levine, The Control Handbook, CRC Press LLC, 1996.

[56] S. L. Herman, Intrustrial Motor Control, 7th Edition, Delmar Cengage Learning, 2014.

[57] V. D. Hunt, Industrial Robotics Handbook, New York: Industrial Press Inc, 1983.

[58] J. M. Hughes, Arduino: A Technical Reference, O'Reilly Media, Inc, 2016.

[59] A. Kurniawan, Arduino Mega 2560 A Hands-On Guide for Beginner, 2019.

[60] Arduino, «Arduino - ArduinoMega2560,» 11 Janvier 2017. [En ligne]. Disponible sur: https://www.arduino.cc/en/Guide/ArduinoMega2560. [Accès le 12 mai 2020].

[61] D. Dubins, Electronics and Microprocessing for Research, 2nd Edition, Cambridge Scholars Publishing, 2019.

[62] RepRap organisation, «DRV8825 chip - RepRap,» 5 Novembre 2017. [En ligne]. Disponible sur: https://reprap.org/wiki/DRV8825_chip. [Accès le 11 mai 2020].

[63] «The best microcontroller projects,» [En ligne]. Disponible sur: https://www.best-microcontroller-projects.com/hitachi-hd44780.html. [Accès le 08 mai 2020].

[64] Opencircuit, «LJ12A3-4-Z / BX Proximity sensor N / O NPN 4mm,» [En ligne]. Disponible sur: https://opencircuit.shop/Product/LJ12A3-4-Z-BX-Proximity-sensor-N-O-NPN-4mm.

[65] S. Mueller, Le PC: Architecture, maintenance et mise à niveau, Pearson Education France, 2008.

[66] «E3D V6 All- Metale Hotend bowden- 1.75mm - 3DJAKE France,» [En ligne]. Disponible sur: https://www.3djake.fr/e3d/v6-all-metal-hotend-bowden-175-mm. [Accès le 17 mai 2020].

[67] M. Renard, «Fusion 360, le logiciel pour l'impression 3D - 3D natives,» 2020 Avril 29. [En ligne]. Disponible sur: https://www.3dnatives.com/fusion-360-logiciel-impression-3d-29042020/. [Accès le 17 mai 2020].

[68] R. H. Shih, Parametric Modeling with Autodesk Fusion 360, Kansas: SDC Publications, 2020.

[69] Elektronique, «Proteus (ISIS et ARES) - logiciel Electronique,» [En ligne]. Disponible sur: http://www.elektronique.fr/logiciels/proteus.php. [Accès le 13 mai 2020].

[70] Y. Chen, Design and Development of Intelligent Robot, Lambert Academic Publishing, 2011.

[71] S.-K. K. D.-H. C. I. S. Suk-Hwan Suh, Theory and Design of CNC Systems, Springer, 2008.

[72] M.-P. Cani, Façonner l'imaginaire: De la création numérique 3D aux mondes virtuels animés, Fayard, 2015.

A. Annexe A

Microcontrôleurs

Le Microcontrôleur est un circuit programmable capable d'exécuter un programme et qui possède des circuits d'interface intégrés avec le monde extérieur. Ils sont fréquemment utilisés dans les systèmes embarqués, comme les contrôleurs des moteurs automobiles, les télécommandes, les appareils de bureau, l'électroménager, les jouets, la téléphonie mobile, etc. Toutes les solutions à base de composants programmables ont pour but de réduire le nombre de composants sur le circuit électronique et donc fiabiliser le circuit.

Les termes « microcontrôleur » et « microprocesseur » font référence à la même chose. La différence entre les deux est qu'un microcontrôleur est - généralement, et non nécessairement - un microprocesseur plus simple. Généralement construit avec des registres de données et un bus de données de largeur 8 ou 16 bits, avec une vitesse de CPU plus faible [50].

De nos jours pour de nombreux électroniciens, les microcontrôleurs et leur programmation restent un domaine obscur [51].

Moteurs pas à pas

Un moteur pas à pas est un transducteur permettant une conversion d'énergie et d'information de caractère électromécanique. Son alimentation est de type électrique digital ou impulsionnel. Son mouvement, rotatif ou de translation, est de type incrémental continu [52].

Les moteurs pas à pas sont principalement de trois types :

Les moteurs à reluctance variable

Ils ne possèdent que des électroaimants au stator, et le rotor est composé de plusieurs dents en fer doux. On le fait tourner en alimentant les bobines les unes après les autres (le rotor essai de s'aligner avec le champ magnétique), et on change de sens en changeant l'ordre d'excitation des bobines.

Les moteurs à aimant permanent

Le rotor est polarisé et on le contrôle en alimentant les bobines les unes après les autres dans un sens puis dans l'autre, son contrôle est donc plus compliqué car il faut inverser le courant dans les bobines, à l'exception des moteur unipolaire, chaque bobine possède une connexion en leur milieu pour faire des « demi-bobine >> et éviter cette inversion de courant (on reconnaît s'ils sont unipolaires quand ils ont 6 fils au lieu de 4, pour un moteur à 2 bobines par exemple).

Les moteurs hybrides

Ils sont le mélange des deux autres types : Un rotor avec des dents polarisées. Il se contrôle comme le moteur à aiment permanent et a pour avantage de pouvoir posséder un grand nombre de pas, un rendement plus élevé.

La structure fixe stator est généralement de type reluquant variable et sup porte les bobinages. La structure mobile rotor est également à réluctance variable (moteurs reluquant ou à reluctance différentielle) ou à caractéristique de magnétisation périodique (moteurs électromagnétiques).

L'analyse complète du comportement d'un moteur pas à pas nécessite la résolution numérique des équations dynamiques [52].

Avantages des moteurs pas à pas

- L'angle de rotation du moteur est proportionnel à l'impulsion d'entrée.
- Le moteur a un couple complet à l'arrêt (si les enroulements sont sous tension).
- Un positionnement précis et la répétabilité du mouvement puisque les bons moteurs pas à pas ont une précision de 3 à 5 % pour un pas et que cette erreur est non cumulative d'un pas à l'autre.
- Excellente réponse pour le démarrage/arrêt/fonctionnement inverse.

Pont H

Le pont en H (Figure A.1) est une structure électronique servant à contrôler la polarité aux bornes d'un moteur à courant continu ou pas à pas, c'est à dire à contrôler le sens de rotation du moteur et à faire varier la vitesse du moteur en modulant la tension à ses bornes. Il est composé de quatre commutateurs (transistors, ...) généralement disposés en forme de H sur les schémas, d'où le nom [53].

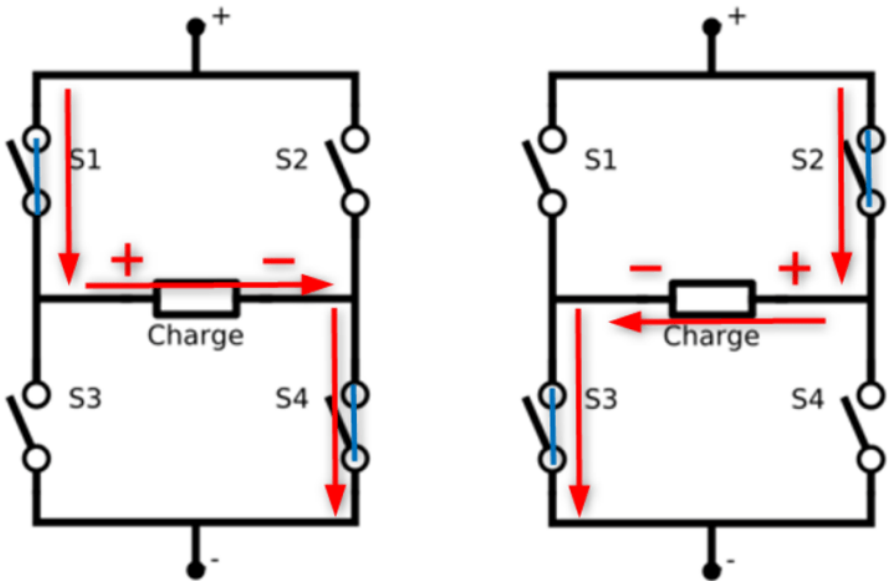

Figure A.1: Exemple de fonctionnement du pont H

Principe de fonctionnement du pont H

Dans son principe de base, le pont H est un assemblage de 4 transistors (S1 à S4) montés de telle façon que le courant puisse circuler dans les deux sens au travers du moteur, ce qui permettra de changer son sens de rotation.

La structure du pont permet de modifier le sens du courant dans le moteur en gérant correctement l'ouverture et la fermeture des commutateurs :

*Si S1 et S4 sont fermés alors que S2 et S3 sont ouverts, le courant circulera de S1 vers S4.

*Si S2 et S3 sont fermés alors que S1 et S4 sont ouverts, le courant circulera de S2 vers S3.

Nous vous rappelons que : S1 et S3 ne doivent jamais être commutés en même temps (tout comme S2 et S4) car cela provoquerait un court-circuit franc et destructeur.

Résistances thermiques

NTC signifie coefficient de température négatif Les thermistances NTC sont des résistances avec un coefficient de température négatif ce qui signifie que la résistance diminue avec l'augmentation de la température. Ils sont principalement utilisés comme capteurs de température résistifs et dispositifs limiteurs de courant. Le coefficient de sensibilité à la température est environ cinq fois supérieur à celui des capteurs de température au silicium (Silistors) et environ dix fois supérieur à ceux des détecteurs de température à résistance (RTD). Les capteurs NTC sont généralement utilisés dans une plage de -55 °C à 200°C [54].

Afficheur LCD

LCD (Figure A.2) signifie "Liquid Crystal Display" et se traduit, en français, par "Écran à Cristaux Liquides". Ces écrans sont partout, vous en trouverez dans plein d'appareils électroniques disposant d'afficheur : les montres, le tableau de bord de votre voiture, les calculatrices, etc. Cette utilisation intensive est due à leur faible consommation et coût. Mais ce n'est pas tout ! En effet, les écrans LCD sont aussi sous des formes plus complexes telles que la plupart des écrans d'ordinateur ainsi que les téléviseurs à écran plat. Cette technologie est bien maîtrisée et donc le coût de production est assez bas.

Figure A.2: Afficheur LCD 16x2

Beaucoup de projets, quels qu'ils soient, requièrent une visualisation et souvent en temps réel. L'afficheur LCD alphanumérique offre une solution à ce problème, à condition de ne pas traiter de graphiques [51].

Contacteurs fin de course

Les interrupteurs de fin de course (Figure A.3) sont utilisés depuis des décennies pour indiquer les positions. Ils sont constitués de contacts électriques actionnés mécaniquement. Un contact s'ouvre où se ferme lorsqu'une variable (position, niveau) a atteint une certaine valeur. Il existe des centaines de types de fins de course [55]. Les interrupteurs de fin de course contiennent un type de bras de pare-chocs impacté par un objet. Le type de bras de pare-chocs utilisé est déterminé par l'application de l'interrupteur de fin de course. Lorsque le bras de pare-chocs est impacté, il provoque un changement de position des contacts [56].

Figure A.3: Contacteurs de fin de courses

Capteur de proximité

Un capteur de proximité (Figure A.4) est un appareil qui détecte quand un objet est proche d'un autre objet, ça peut aller de centièmes de pouce à quelques pouces, selon le capteur utilisé. La plupart des capteurs de proximité n'indiquent que la présence ou l'absence d'un objet dans leur zone de détection, mais certains peuvent également fournir des informations sur la distance entre l'objet et le capteur [57].

Figure A.4: Capteurs de proximités

B. Annexe B

ATmega2560

L'ATmega 2560 (Figure B.1) est un microcontrôleur CMOS 8 bits basse consommation basé sur l'architecture RISC améliorée AVR. En exécutant des instructions puissantes en un seul cycle d'horloge, I'ATmega 2560 atteint des débits approchant 1 MIPS par MHz, ce qui permet au concepteur du système d'optimiser la consommation d'énergie par rapport à la vitesse de traitement [58].

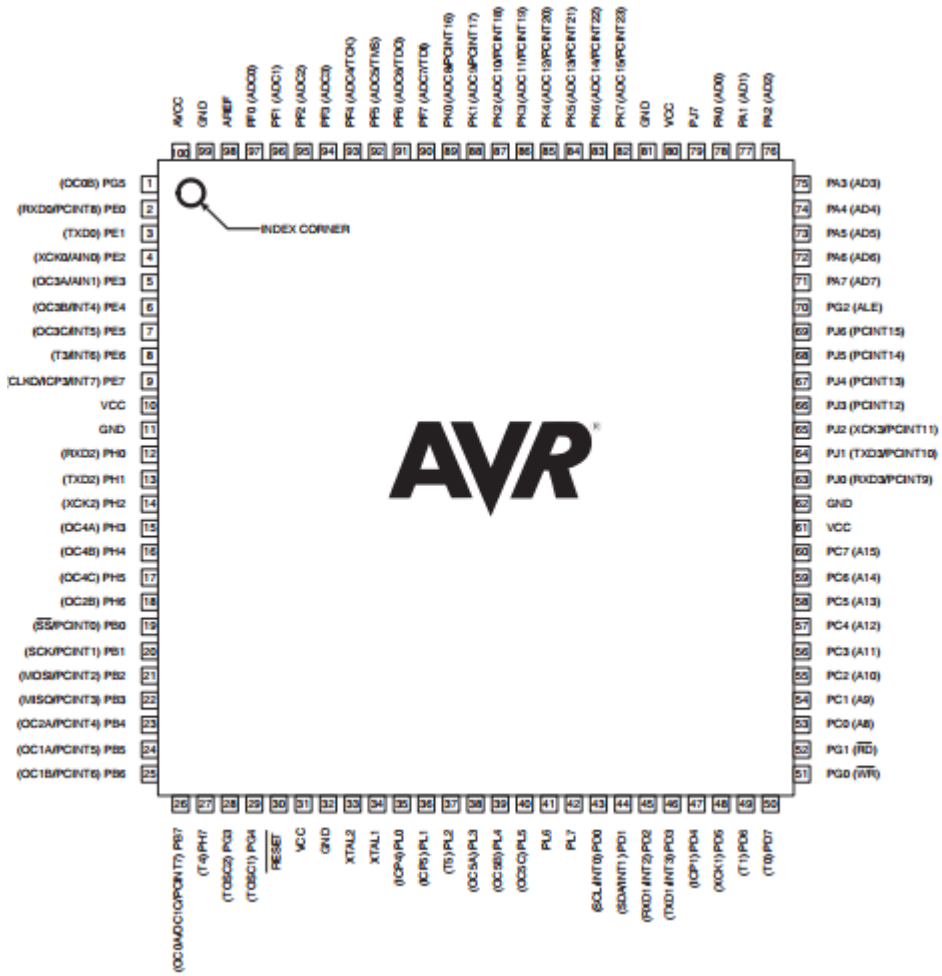

Figure B.1: Pinout de l'ATmega2560

Le noyau Atmel AVR® (Figure B.2) combine un riche ensemble d'instructions avec 32 registres de travail à usage général. Tous les 32 registres sont directement connectés à l'unité logique arithmétique (ALU), permettant d'accéder à deux registres indépendants en une seule instruction exécutée en un cycle d'horloge. L'architecture résultante est plus efficace en code tout en atteignant des débits jusqu'à dix fois plus rapides que les microcontrôleurs CISC conventionnels [58].

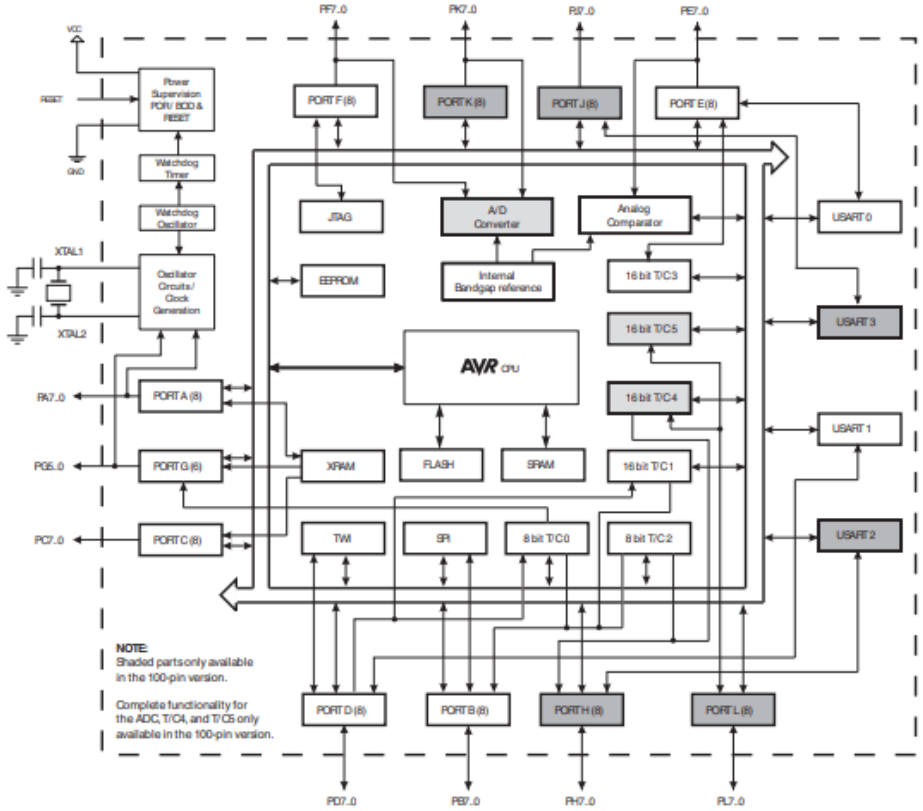

Figure B.2: Diagramme de l'ATmega 2560

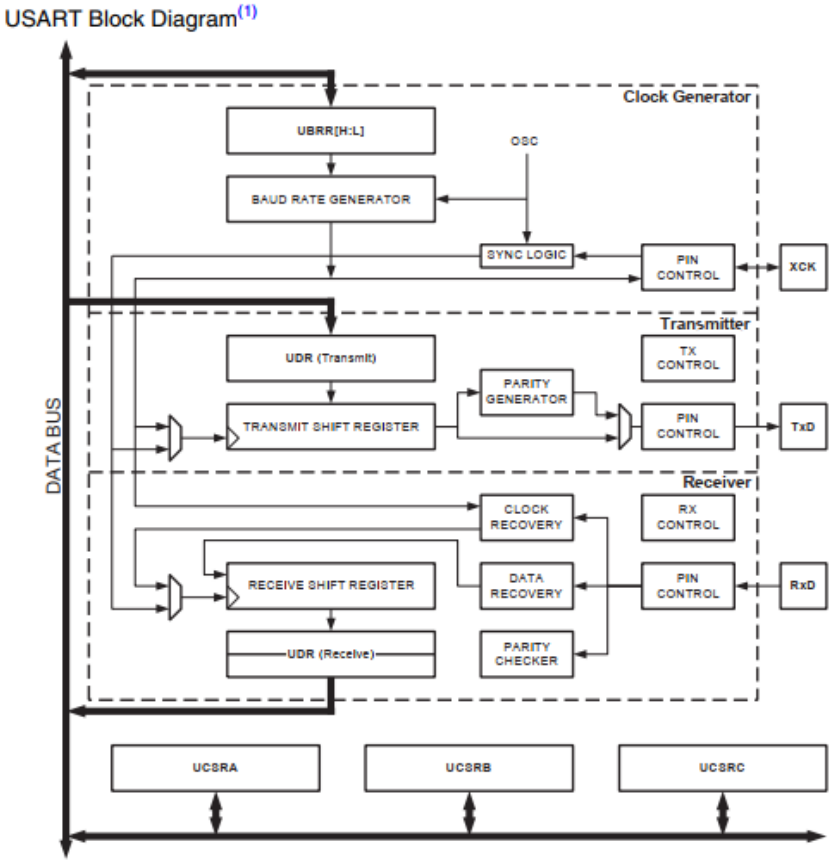

Figure B.3: Schéma du UART de l'ATmega2560

Tableau 1: Valeurs du registre UBRR et les Baud Rates correspondantes.

Table 22-12. Examples of UBRRn Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4K	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2K	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8K	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4K	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6K	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8K	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%
115.2K	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4K	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250K	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

Note: 1. UBRR = 0, Error = 0.0%

Arduino Mega

L'Arduino Mega 2560 (Figure B.4) est une carte microcontrôleur basée sur l'ATmega2560. II dispose de 54 broches d'entrée / sortie numériques (dont 15 peuvent être utilisées comme sorties PWM), 16 entrées analogiques, 4 UART (ports série matériels), un oscillateur à cristal de 16 MHz, une connexion USB, une prise d'alimentation, un en-tête ICSP, et un bouton de réinitialisation [59].

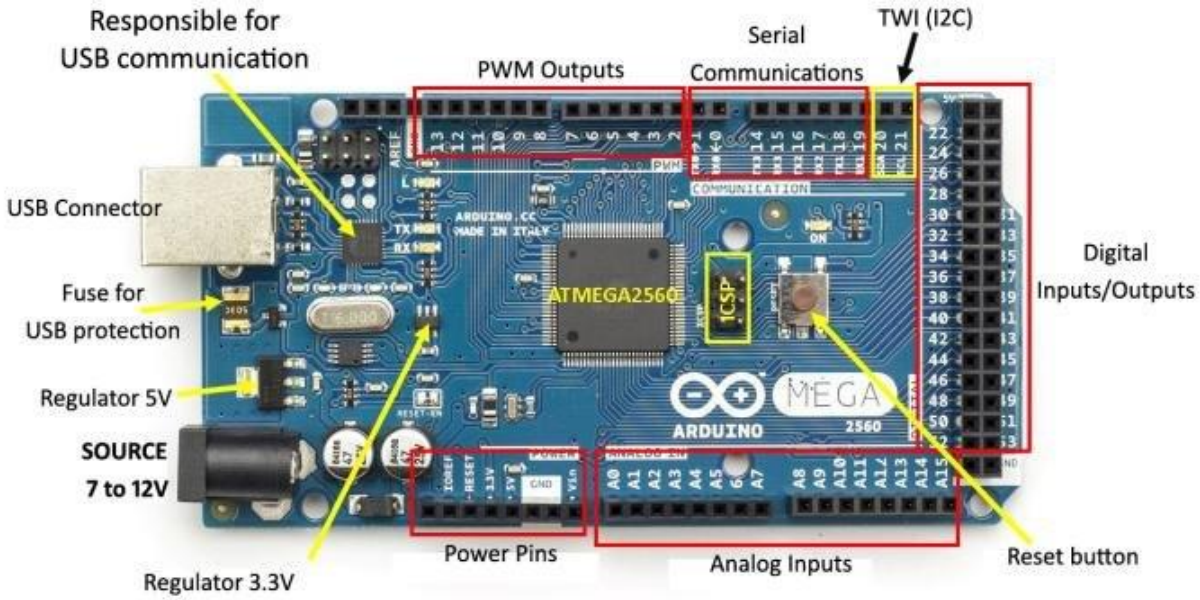

Figure B.4: Arduino Mega

L'Arduino MEGA 2560 est conçu pour les projets qui nécessitent plus de lignes d'E/S, plus de mémoire d'esquisse et plus de RAM. Un plus grand espace pour votre croquis, c'est la carte recommandée pour les imprimantes 3D et les projets de robotique. Cela donne à vos projets beaucoup d'espace et d'opportunités en maintenant la simplicité et l'efficacité de la plate-forme Arduino [60].

17HD34008-22B (Nema 17)

Le moteur pas à pas Nema-17 tire son nom des dimensions. Il existe de nombreux modèles Nema-17 différents, variant dans la tension et le courant de fonctionnement, le couple de maintien et l'angle de pas. Le moteur pas à pas Nema-17 est devenu extrêmement populaire dans l'impression 3D et les appareils CNC [61]. Le (Tableau 2) présente les caractéristiques du moteur pas à pas que nous avons utilisés.

Tableau 2: Caractéristiques du 17HD34008-22B

Angle du pas	1.8°
Nombre de phases	2
Résistance d'isolation	100 MOhm min (500V DC)
Class d'isolation	Class B
Inertie du rotor	38 g.cm²
Poids	0.23 kg
Voltage	4.08 V
Courant	1.2 A
Résistance par phase	3.4 Ohm ± 10%
Inductance par phase	5 mH ± 20%
Couple de maintien	320 mN.m
Coupe de détente	12 mN.m
Vitesse nominale	500 tr/min
Dimension	42 × 42 × 34

DRV8825

Au cœur du module (Figure B.5) se trouve un pilote de Texas Instruments DRV8825. Le pilote de moteur pas à pas DRV8825 a une capacité d'entraînement de sortie jusqu'à 45 V et permet de contrôler un moteur pas à pas bipolaire avec un courant de sortie jusqu'à 2,2 A par bobine. Avec seulement 2 broches (Step et Direction), on peut contrôler un moteur pas à pas [62].

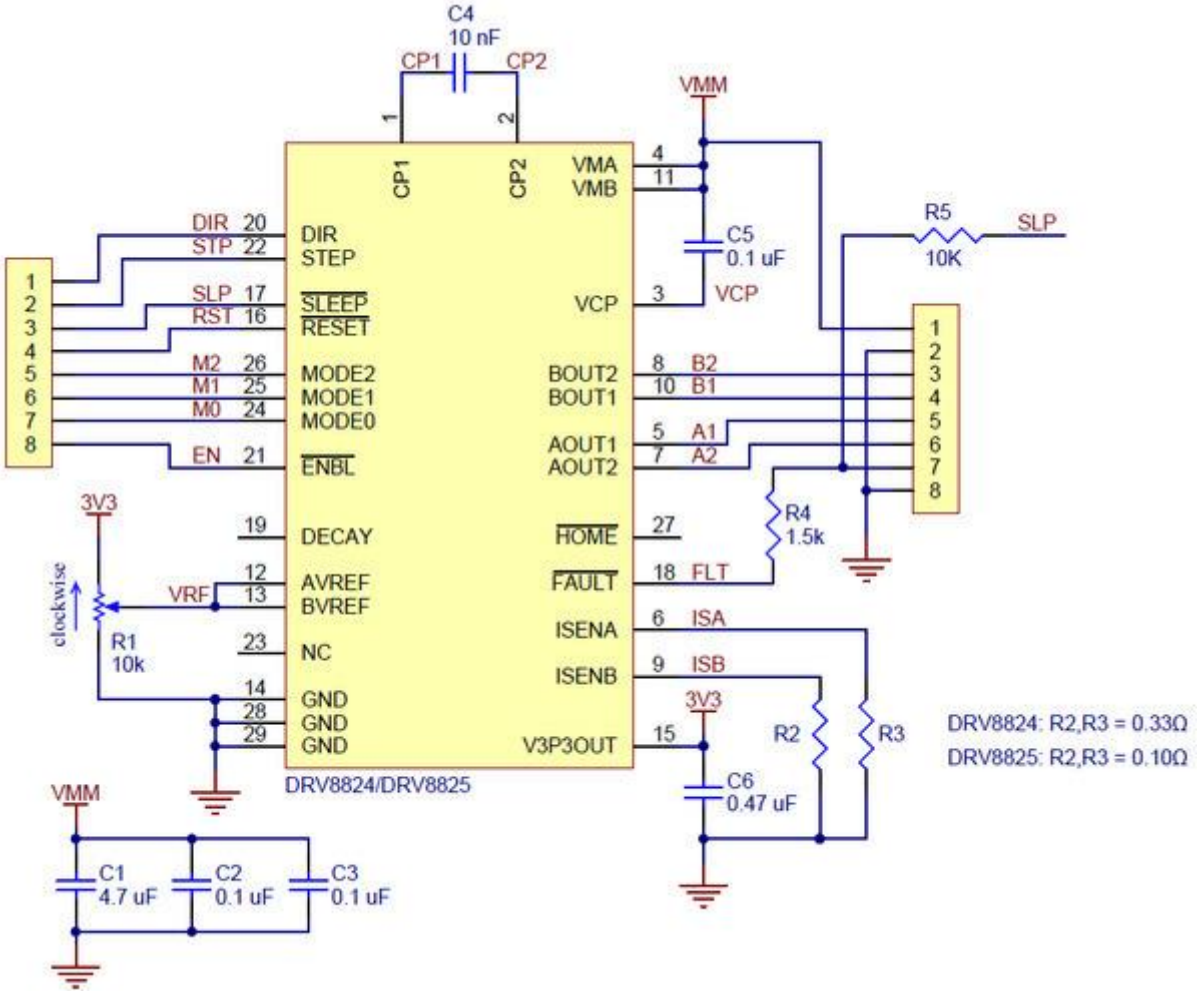

Figure B.5: Schéma du Pololu DRV8825

Les broches M0, M1, M2 permettent la configuration du moteur en mode pas à pas jusqu'à 1/32 pas. Des fonctions d'arrêt internes sont fournies pour la protection contre les surintensités, la protection contre les courts-circuits, sous-tension et la surchauffe. Les caractéristiques du circuit intégré DRV8825 vous sont présentées dans le (Tableau 3).

Tableau 3: Caractéristiques du DRV8825

Voltage maximum	45 V
Courant maximum par bobine	2.2 A
Voltage logique	3.3 V ou 5 V
Micro pas maximum	1 / 32
Taille minimale de l'impulsion Step	1.9 us

NTC B3950 1000K

Puissance de dissipation maximum : 45mW.

$R25°C = 100K\Omega \pm 1\%$

$B25°C/50°C = 3950 \pm 1\%$

STP55nf06l

C’est un transistor mosfet N channel avec ces caractéristiques là (Figure B.6) (Tableau 4) :

Tableau 4: Caractéristiques électriques du STP55nf06l

Symbol	Parameter	Value	Unit
V_{DS}	Drain-source voltage (V_{GS} = 0)	60	V
V_{GS}	Gate-source voltage	± 16	V
I_D	Drain current (continuous) at T_C = 25°C	55	A
I_D	Drain current (continuous) at T_C = 100°C	39	A
I_{DM}(1)	Drain current (pulsed)	220	A
P_{TOT}	Total dissipation at T_C = 25°C	95	W
	Derating factor	0.63	W/°C
dv/dt(2)	Peak diode recovery voltage slope	20	V/ns
E_{AS}(3)	Single pulse avalanche energy	300	mJ
T_J T_{stg}	Operating junction temperature Storage temperature	-55 to 175	°C

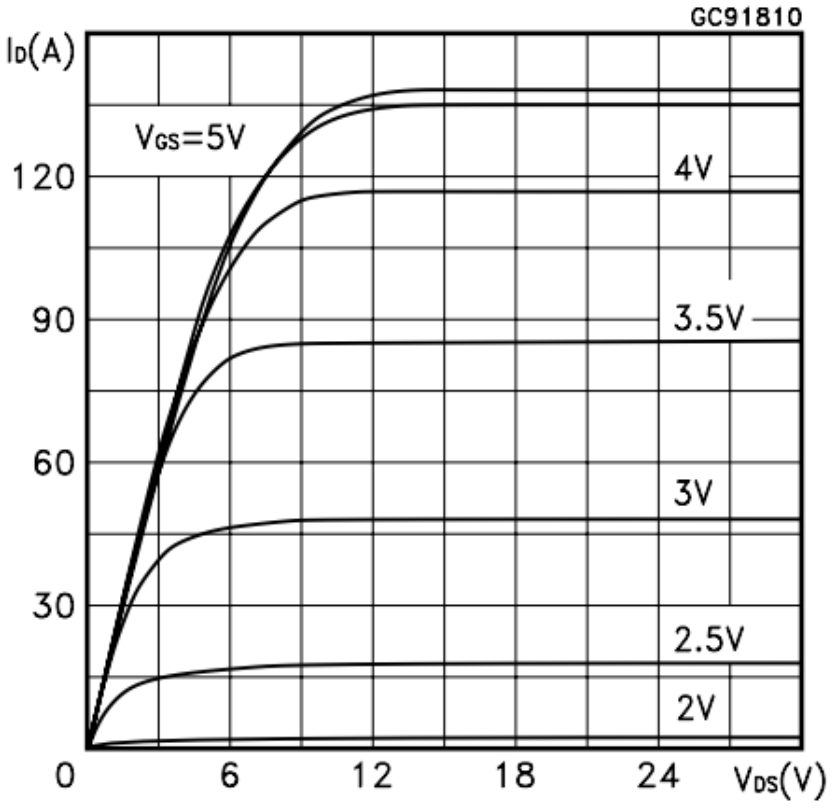

Figure B.6: Graphe du courant Id en fonction du voltage Vds

HD44780

L'Hitachi HD44780 est un chipset de pilotage LCD pilotant généralement un écran de 2 lignes par 16 caractères. Vous pouvez obtenir de nombreuses tailles d'écran différentes, par exemple 16x4, 20x4, 80x1. Ces types d'affichages sont très peu coûteux et peuvent afficher n'importe quel texte ASCII souhaité [63]. Les pins du HD44780 vous sont présentés dans le (Tableau 5).

Tableau 5: Explication des pins du HD47780

Borne	Symbole	Type	Fonction
1	Vss ou V0	Alim	Masse 0V
2	Vcc ou Vdd	Alim	Alimentation générale 5V
3	Vee	Alim	Alimentation du panneau LCD (Contraste des caractères) Vee = Vcc – Vss Vss = Vcc → Vee = 0 → Caractères invisibles Vss = 0 → Vee = Vcc → Contraste maximum
4	RS	Entrée	RS = 1 → Sélection du registre de données RS = 0 et R/W = 0 → Sélection du registre d'instruction RS = 0 et R/W = 1 → Sélection du drapeau BUSY et du compteur d'adresse
5	R/W	Entrée	R/W = 0 → Mode écriture R/W = 1 → Mode lecture
6	E	Entrée	Entrée de validation Les entrées RS et R/W sont lues sur le front montant, et le bus de données est lu sur le front descendant.
7	D0	Entrée/Sortie	Bus de données, bit n°0 (LSB)
8	D1	Entrée/Sortie	Bus de données, bit n°1
9	D2	Entrée/Sortie	Bus de données, bit n°2
10	D3	Entrée/Sortie	Bus de données, bit n°3
11	D4	Entrée/Sortie	Bus de données, bit n°4
12	D5	Entrée/Sortie	Bus de données, bit n°5
13	D6	Entrée/Sortie	Bus de données, bit n°6
14	D7	Entrée/Sortie	Bus de données, bit n°7 (MSB)
15	A	Alim	Anode du système de rétro-éclairage (à alimenter en 5V à travers une résistance de 50 à 100Ω pour limiter le courant à 100mA)
16	K	Alim	Cathode du système de rétro-éclairage (masse)

Il peut aussi fonctionner soit en 8-bits soit en 4 bits :

En mode 8-bits, l'octet contenant les données est envoyé à l'afficheur (ou lu par l'afficheur) directement sur les broches D0 à D7. En mode 4-bits, on n'utilise que les broches D4 à D7, les broches D0 à D3 doivent être connectées à la masse.

L'octet de données est envoyé (ou lu) en 2 fois, d'abord les 4 bits de poids fort, par une première validation sur la broche E, puis les 4 bits de poids faible, par une seconde validation sur la broche E.

Cependant il a un timing (Figure B.7) bien précis à respecter qu'on en envoie ou on reçoit des données, dans notre cas on va seulement écrire.

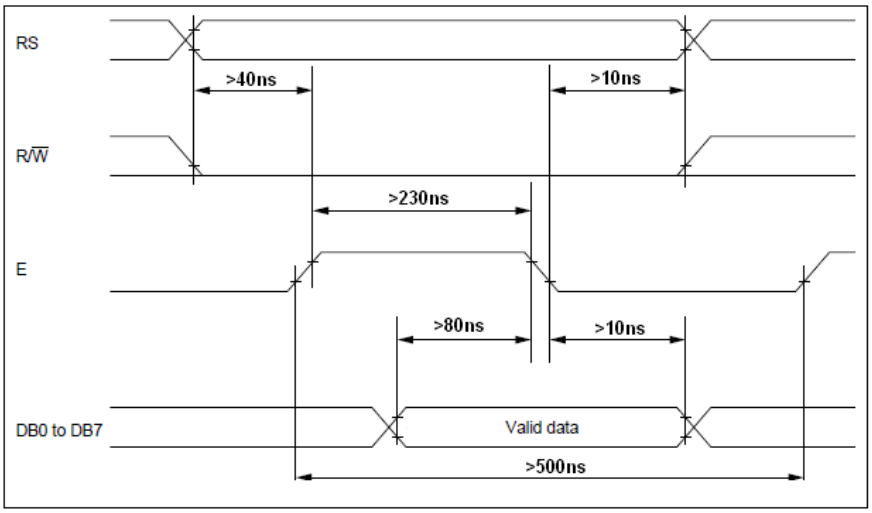

Figure B.7: Timing d'une opération d'écriture

Les instructions du HD44780 sont dans ces tableau-là :

Tableau 6: Tables des instructions du HD44780.

Table 6 Instructions

Instruction	Code RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{op} or f_{osc} is 270 kHz)
Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address 0 in address counter.	
Return home	0	0	0	0	0	0	0	0	1	—	Sets DDRAM address 0 in address counter. Also returns display from being shifted to original position. DDRAM contents remain unchanged.	1.52 ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Sets cursor move direction and specifies display shift. These operations are performed during data write and read.	37 µs
Display on/off control	0	0	0	0	0	0	1	D	C	B	Sets entire display (D) on/off, cursor on/off (C), and blinking of cursor position character (B).	37 µs
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	—	—	Moves cursor and shifts display without changing DDRAM contents.	37 µs
Function set	0	0	0	0	1	DL	N	F	—	—	Sets interface data length (DL), number of display lines (N), and character font (F).	37 µs
Set CGRAM address	0	0	0	1	ACG	ACG	ACG	ACG	ACG	ACG	Sets CGRAM address. CGRAM data is sent and received after this setting.	37 µs
Set DDRAM address	0	0	1	ADD	ADD	ADD	ADD	ADD	ADD	ADD	Sets DDRAM address. DDRAM data is sent and received after this setting.	37 µs
Read busy flag & address	0	1	BF	AC	AC	AC	AC	AC	AC	AC	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 µs

Tableau 7: Suite des instructions du HD44780.

Instruction	Code RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	Execution Time (max) (when f_{cp} or f_{OSC} is 270 kHz)
Write data to CG or DDRAM	1	0	Write data								Writes data into DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
Read data from CG or DDRAM	1	1	Read data								Reads data from DDRAM or CGRAM.	37 µs t_{ADD} = 4 µs*
	I/D = 1: Increment I/D = 0: Decrement S = 1: Accompanies display shift S/C = 1: Display shift S/C = 0: Cursor move R/L = 1: Shift to the right R/L = 0: Shift to the left DL = 1: 8 bits, DL = 0: 4 bits N = 1: 2 lines, N = 0: 1 line F = 1: 5 × 10 dots, F = 0: 5 × 8 dots BF = 1: Internally operating BF = 0: Instructions acceptable										DDRAM: Display data RAM CGRAM: Character generator RAM ACG: CGRAM address ADD: DDRAM address (corresponds to cursor address) AC: Address counter used for both DD and CGRAM addresses	Execution time changes when frequency changes Example: When f_{cp} or f_{OSC} is 250 kHz, $37\ \mu s \times \frac{270}{250} = 40\ \mu s$

Note: — indicates no effect.

* After execution of the CGRAM/DDRAM data write or read instruction, the RAM address counter is incremented or decremented by 1. The RAM address counter is updated after the busy flag turns off. In Figure 10, t_{ADD} is the time elapsed after the busy flag turns off until the address counter is updated.

Tableau 8: Timing dans une operation d'ecriture.

Write Operation

Item	Symbol	Min	Typ	Max	Unit	Test Condition
Enable cycle time	t_{cycE}	1000	—	—	ns	Figure 25
Enable pulse width (high level)	PW_{EH}	450	—	—		
Enable rise/fall time	t_{Er}, t_{Ef}	—	—	25		
Address set-up time (RS, R/$\overline{W}$ to E)	t_{AS}	60	—	—		
Address hold time	t_{AH}	20	—	—		
Data set-up time	t_{DSW}	195	—	—		
Data hold time	t_{H}	10	—	—		

Lj12a3-4-z/bx

Le Lj12a3-4-z/bx (Figure B.8) est un capteur de proximité qui détecte un objet métallique à une distance maximale de 4 mm du capteur. Le capteur fonctionne comme un transistor NPN (normalement ouvert), ce qui signifie que la tension est coupée lorsqu'un objet métallique est détecté à moins de 4 mm du capteur [64].

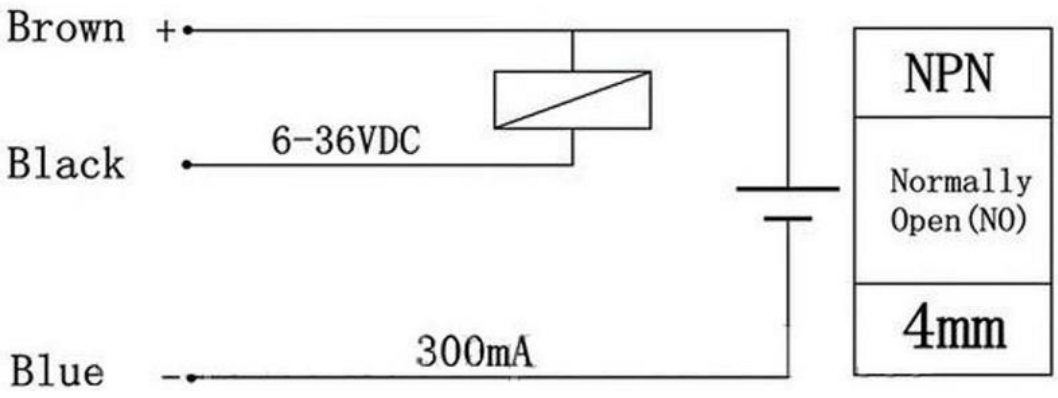

Figure B.8: Schéma du Lj12a3-4-z/bx

Le Lj12a3-4-z/bx possède les caractéristiques citées dans le (Tableau 9).

Tableau 9: Caractéristiques éléctriques du Lj12a3-4-z/bx

Courant max	300mA
Voltage	6V - 36V
Distance de détection	4 mm
Temps pour la détection	0.5 secondes
Production	NPN
Matériel de détection	Métal

Alimentation ATX

ATX c'est le format d'alimentation à découpage utilisé dans les ordinateurs PC de type Pentium II et postérieur. L'alimentation (Figure B.9) fournit les tensions de sorties suivantes : +5 V, -5 V, +12 V, -12 V et +3 3 V [65].

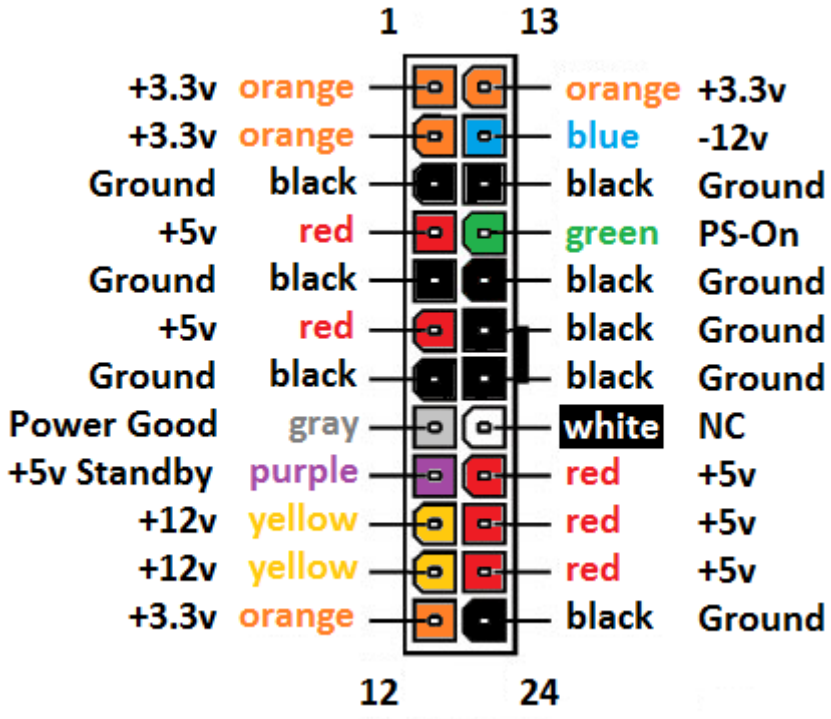

Figure B.9: Pin d'une alimentation ATX

E3D Hotend

L'extrudeuse E3D v6 (Figure B.10) a été créée par la société E3D. La transition thermique nette attendue par les utilisateurs de « HotEnds E3D » est préservée, ce qui permet une haute qualité d'impression. Une coupure thermique nette permet un meilleur contrôle de la sortie du filament, afin que vous puissiez démarrer et arrêter plus rapidement pendant l'extrusion, et rendre la rétraction plus efficace. Cela signifie des impressions plus nettes et plus précises [66].

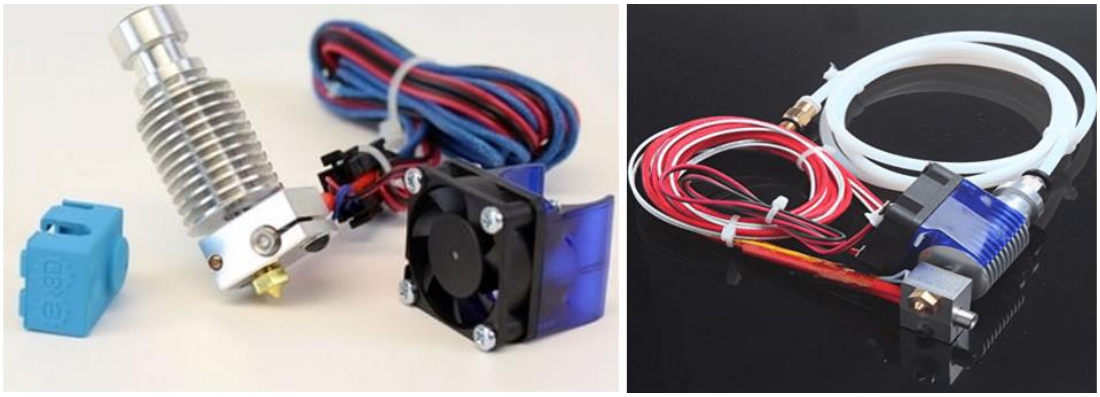

Figure B.10: L'extrudeuse E3D v6

Mécanique MK8

La mécanique d'extrusion MK8 est faite spécialement pour les Nema 17. Il existe trois modèles du MK8 (Figure B.11).

Figure B.11: Les trois modèle du MK8

LM8UU

Les roulements à billes linéaire sont utilisés car ils permettent un mouvement linéaire avec un faible coefficient de frottement. Ils sont constitués de bagues pourvues de pistes de recirculation des billes, ils subissent un coefficient de frottement très faible et une résistance accrue.

Les caractéristiques les plus importants de la LM8UU (Figure B.12):

- Diamètre intérieure : 8mm.

Mode	L1（mm)	L2(mm)	D1(mm)	D2(mm)
LM8UU	23.95	15.25	14.99	8.38

Figure B.12: Mensurations du LM8UU

Tige filetée T8

Une tige filetée est en mécanique le composant mâle d'un système vis/écrou destiné à l'assemblage de pièces ou à la transformation de mouvement. Son complément, pièce femelle est l'écrou. L'ensemble du système vis/écrou constitue un boulon.

Les caractéristiques les plus importants de la courroie tige filetée T8 (Figure B.13) :

- Pas : 8mm.
- Diamètre : 8mm.

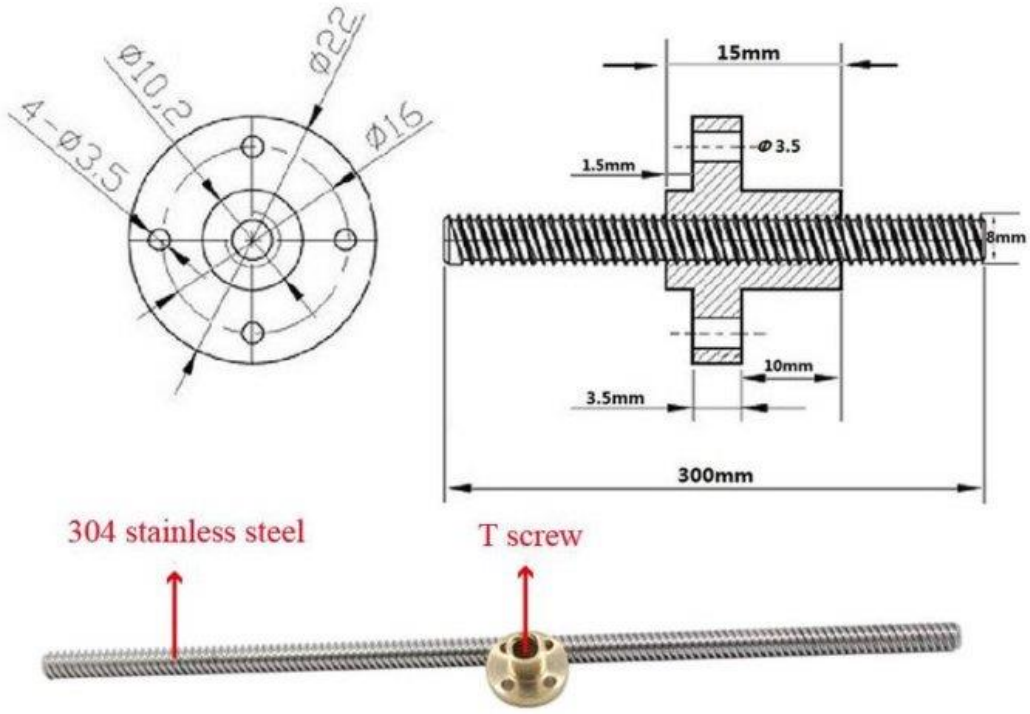

Figure B.13: Tige filetée T8

Courroie GT2

La courroie est utilisée pour la transmission des mouvements. C'est une pièce construite d'un matériau souple. La courroie est utilisée avec des poulies. Par rapport à d'autres systèmes, elle présente l'avantage d'une grande souplesse de conception, le concepteur à une grande liberté pour placer les organes moteurs et récepteur, elle est économique, silencieuse et amortie les vibrations et les chocs.

Les caractéristiques les plus importants de la courroie GT2 (Figure B.14) :

- Pas = 2mm.

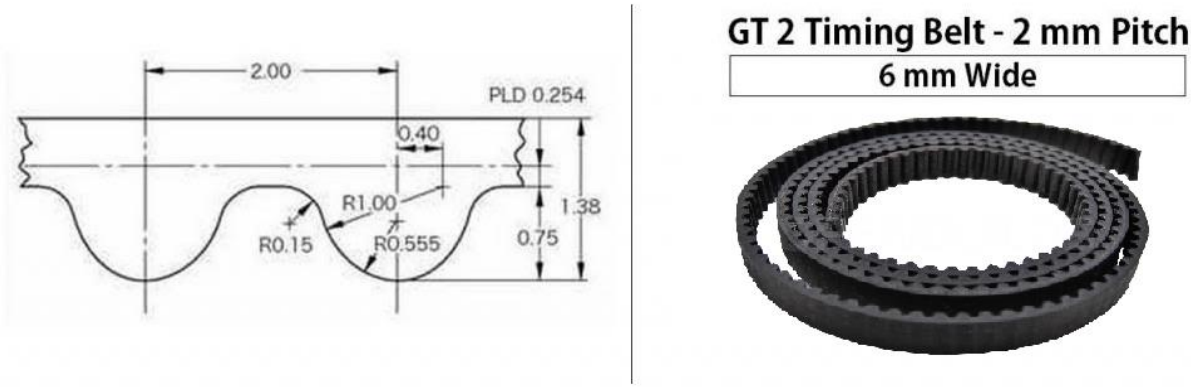

Figure B.14: Courroie GT2

Pignon et poulie GT2

La combinaison poulie, pignon et courroie permet la transformation d'un mouvement rotationnel en linéaire.

Les caractéristiques les plus importants du pignon GT2 (Figure B.15) :

- Pas : 2mm.
- Circonférence intérieure/extérieure : 5mm/12.22mm.
- Nombre de dents : 20.

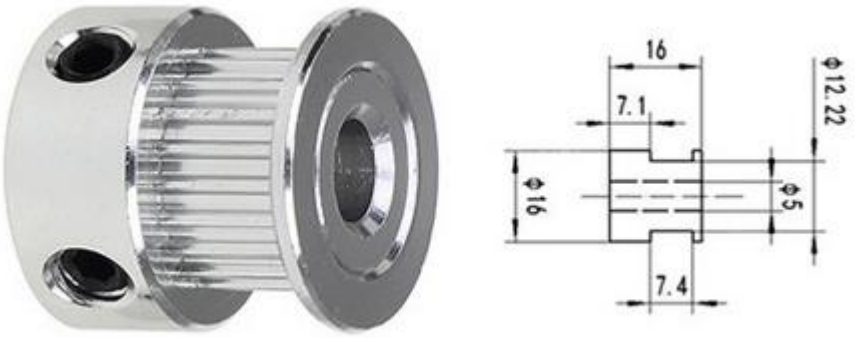

Figure B.15: Pignon GT2.

Les caractéristiques les plus importants de la poulie GT2 (Figure B.16) :

- Pas : 2mm.
- Circonférence intérieure/extérieure : 5mm/18mm.
- Nombre de dents : 20.

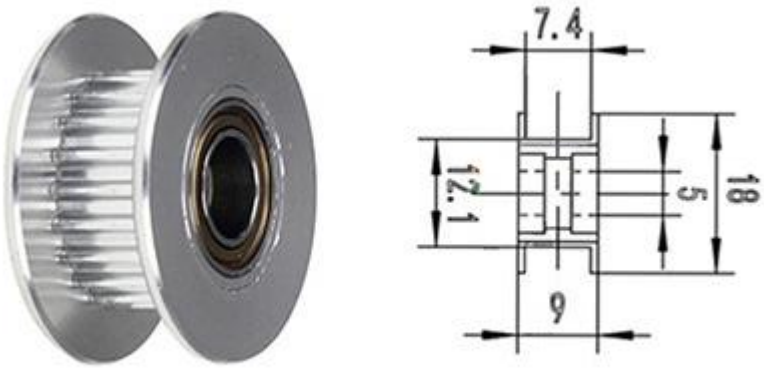

Figure B.16: Poulie GT2

Accouplement

Lors de nombreux montages mécaniques se présente le problème de la transmission du mouvement entre les axes ou les arbres des machines. L'union est la manière la plus simple d'obtenir

cette transmission car elle fonctionne en unissant les extrémités de ces arbres, et en transmettant ainsi la rotation d'un arbre à l'autre. La bonne résolution de cette transmission dépend non seulement du bon fonctionnement de l'appareil mais également de la durée de vie utile des codeurs ou des machines connectées.

Étant donné les erreurs dimensionnelles inhérentes à tout montage mécanique, les axes correspondant aux arbres à unir maintiendront entre eux des différences de positionnement ou « désalignements » qui compliquent la transmission du mouvement. Ces désalignements peuvent être axiaux, radiaux ou angulaires (Figure B.17).

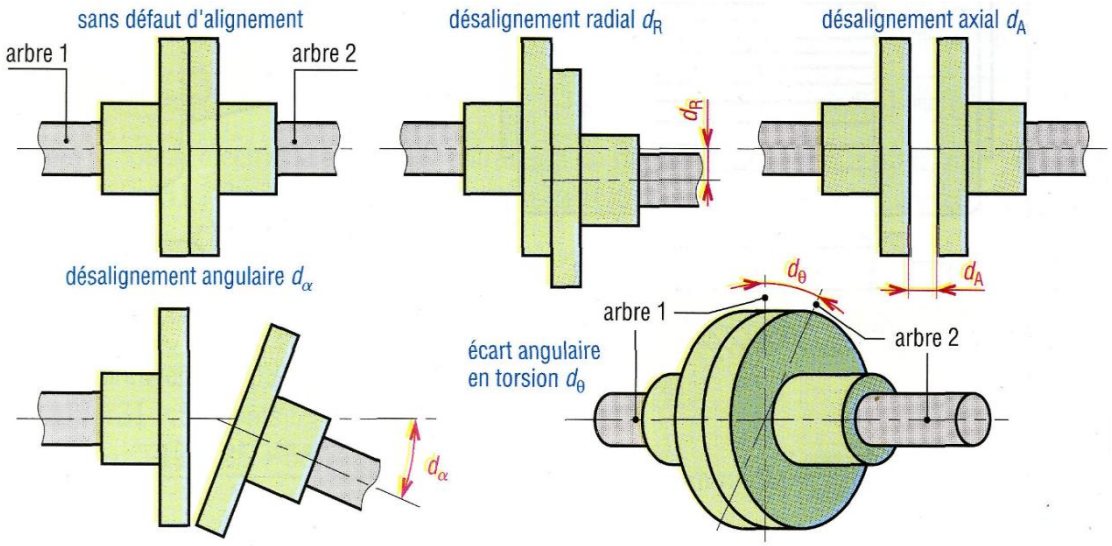

Figure B.17: Différents types d'alignements possibles.

Dans tous les cas, le système d'union utilisé pour la transmission devra pouvoir les absorber, en évitant les effets négatifs de charges sur les axes, les roulements, les appuis et les châssis. Les désalignements entraînent également une fatigue ou une usure de l'union, par conséquent, pour le choix de l'union, on devra tenir compte de la vitesse de rotation, en minorant les désalignements maximums admissibles. Pour cela il faudrait ajouter un accouplement flexible (Figure B.18) entre les deux systèmes.

Figure B.18: Accouplement flexible.

C. Annexe C

Fusion 360

Comme son nom l'indique, Fusion 360 se veut une solution complète, offrant des outils de CAO mais aussi de FAO (fabrication assistée par ordinateur) et d'IAO (ingénierie assistée par ordinateur). Pour les applications de fabrication additive, il s'agit d'un logiciel plus facile à utiliser qu'AutoCAD car il vise à faciliter le développement de produits [67].

Autodesk Fusion 360 est un package intégré basé sur le cloud, tous les fichiers sont stockés sur le serveur central et sont accessibles n'importe où. Les avantages d'utiliser un système basé sur le cloud sont la capacité de collaboration en temps réel et de travailler n'importe où, sur n'importe quel appareil, et il n'y a pas besoin de s'inquiéter du stockage de fichiers ou de la perte de fichiers [68].

Proteus

Proteus est une suite logicielle destinée à l'électronique. Développé par la société Labcenter Electronics, les logiciels incluent dans Proteus permettent la CAO dans le domaine électronique. Deux logiciels principaux composent cette suite logicielle: ISIS, ARES, PROSPICE et VSM [69].

Le logiciel ISIS de Proteus est principalement connu pour éditer des schémas électriques. Par ailleurs, le logiciel permet également de simuler ces schémas ce qui permet de déceler certaines erreurs dès l'étape de conception.

Le logiciel ARES est un outil d'édition et de routage qui complètement parfaitement ISIS. Un schéma électrique réalisé sur ISIS peut alors être importé facilement sur ARES pour réaliser le PCB de la carte électronique.

Le plus grand avantage est que le Proteus Design Suit offre la possibilité de co-simuler à la fois code de microcontrôleur bas niveau dans le cadre d'une simulation de circuit SPICE en mode mixte [70].

D. Annexe D

Ce modèle a été téléchargé à partir du site prusa3d (Figure D.1).

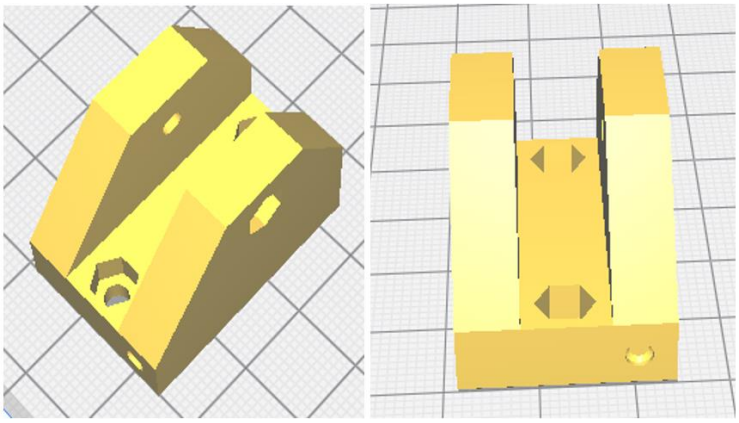

Figure D.1: Support de la poulie de l'axe Y.

Ces modèles ont été téléchargés à partir du site prusa3d (Figure D.2, Figure D.3, Figure D.4 et Figure D.5).

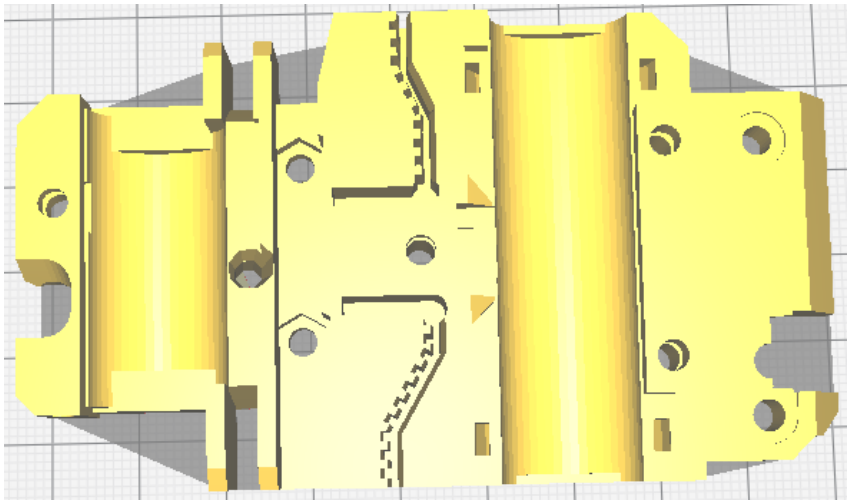

Figure D.2: Partie avant du transporteur X

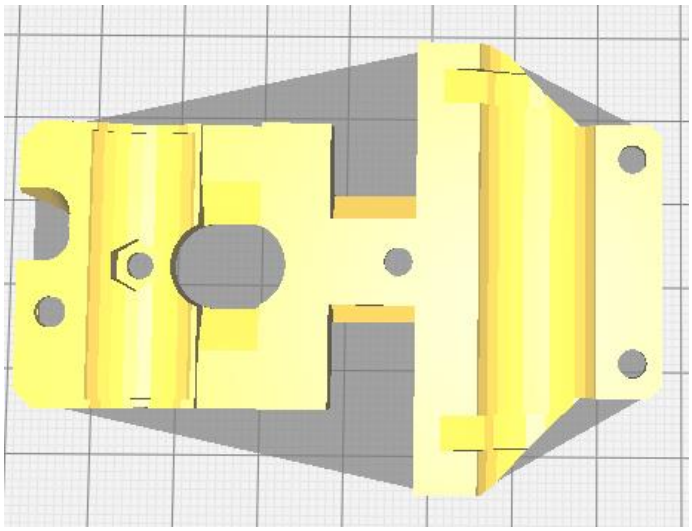

Figure D.3: Partie arrière du transporteur X.

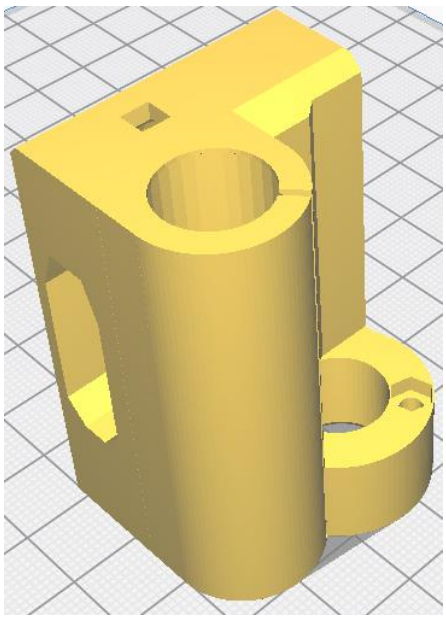

Figure D.4: Transporteur Z droit

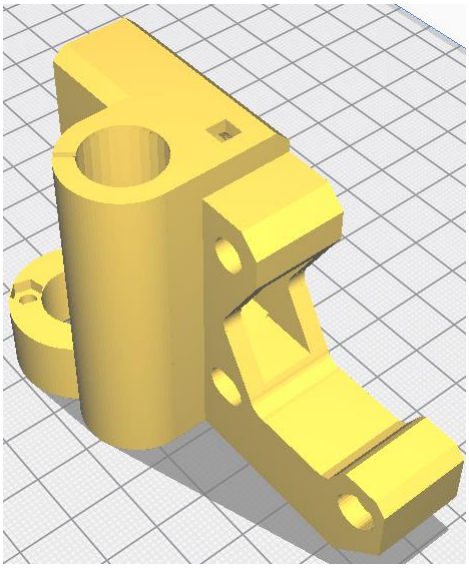

Figure D.5: Transporteur Z gauche

Ces modèles ont été téléchargés à partir du site prusa3d (Figure D.6).

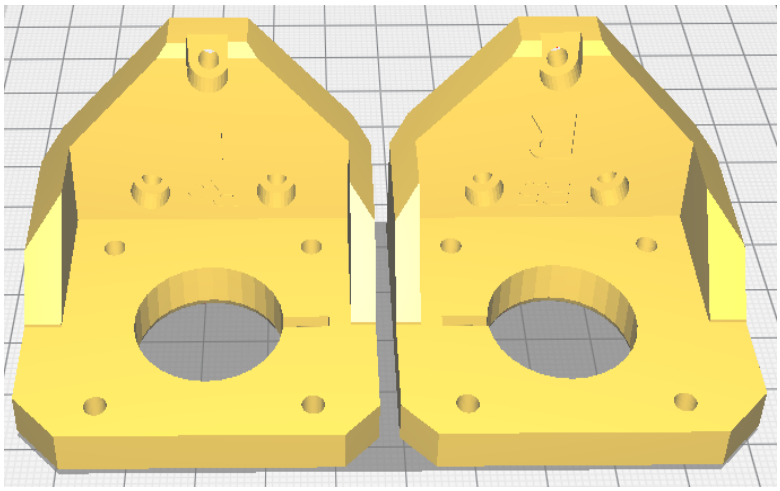

Figure D.6: Supports des moteurs de l'axe Z

Ces modèles ont été téléchargés à partir du site prusa3d (Figure D.7).

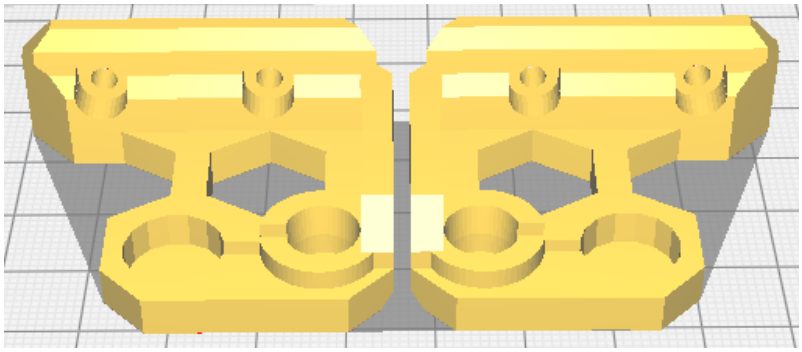

Figure D.7: Supports de la tige filetée et glissante de l'axe Z