



La Chaîne Logicielle

De l'idée de départ jusqu'à la fabrication de la pièce sur votre machine CNC, tout va se passer sur l'ordinateur, utilisant différents logiciels selon l'étape du processus à effectuer, constituant ainsi une chaîne logicielle.

LA Chaîne logicielle, dans l'absolu, n'existe pas. Il y a la vôtre, celle que vous allez petit à petit adopter, qui vous sera la plus familière. Même si, selon les cas particuliers, vous allez sans doute avoir besoin de l'adapter.

Dans cet article, La Chaîne Logicielle est générique. Nous allons voir les différentes étapes nécessaires pour transformer une idée en objet fini, et évoquer les différents outils logiciels qui peuvent être utilisés. Certains logiciels permettent d'effectuer TOUTES les étapes, où presque, alors que d'autres ne permettent qu'une seule opération. Nous reviendrons plus en détail sur certains de ces logiciels dans le futur.

Il existe aussi de nombreux utilitaires, permettant de passage d'une étape à une autre, comme la conversion d'un fichier DXF en GCode, mais je n'en parlerais pas dans cet article, dans la mesure où le but est d'avoir le MINIMUM d'étapes. Je vais donc me focaliser sur les outils plus complets, par rapport à des utilitaires ne remplissant qu'une seule fonction. [Le Forum est là pour en parler](#), une discussion me semble plus appropriée qu'un article, qui ne peut brosser qu'un tableau générique.

Étapes de Création

1. Conception/Dessin 2D.
2. Passage au 3D (ou Conception 3D directement).
3. Génération des chemins de l'outil (des outils), création du GCode.
4. Envoi du GCode à la machine.
5. Interprétation du GCode et transformation en mouvements sur la machine.

Nous allons surtout nous intéresser aux 3 premières étapes, indépendantes des étapes 4 et 5, nécessaires quelle que soit la machine et technique utilisée (Fraisage, découpe laser où même impression 3D). Mon but n'est pas d'être exhaustif et de citer tous les logiciels possible pour une étape particulière, mais de proposer une découverte de certains outils particulièrement adaptés à une utilisation amateur/semi-professionnelle, de par leur prix et (ou) leur simplicité d'utilisation.

C'est l'ensemble des logiciels et utilitaires permettant d'effectuer au mieux chacune de ces étapes qui vont constituer votre chaîne logicielle.

Étape 1 : Conception/Dessin 2D

C'est sans doute pour cette première étape que la palette de choix possibles est la plus vaste, puisque pratiquement n'importe quel logiciel de dessin peut-être utilisé. La difficulté risque d'être le passage aux étapes suivantes, il vaut donc mieux choisir d'entrée un outil connu pour son intégration aisée dans l'ensemble du processus.

Pour cette raison, il paraît nécessaire d'éliminer d'emblée les logiciels de dessins "matriciels" (raster) pour se pencher vers les outils de dessin vectoriels (vectors, lignes). Pendant qu'on y est, autant se focaliser sur les outils type dessins industriels, plutôt qu'artistiques, selon vos buts, bien sûr. Il existe de nombreuses solutions permettant de passer de l'un à l'autre, simplement cela rajoute une étape, donc de la complexité, du temps et une source d'erreurs supplémentaire.

◦ Inkscape

- Pourtant, le premier logiciel que je veux présenter est justement [un de ceux qui permettent cette conversion](#), mais il est capable de bien plus! Je veux parler de [Inkscape](#), qui est un logiciel de dessin vectoriel Open Source et gratuit. Attention, il ne s'agit pas de dessin paramétré, type dessin industriel, mais de dessin libre, "artistique", ce qui peut parfaitement vous convenir, pour réaliser des signaux, panneaux, logos, etc...
- Étant capable d'exporter les créations en fichier SVG et DXF, il pourra s'intégrer avec tous les outils dont nous allons parler par la suite.

◦ Draftsight

- Créé par Dassault Systèmes, [Draftsight](#) est un équivalent très proche des fameux Autocad et Autocad LT. On ne parle plus de dessin "artistique" ici, mais de toute la puissance nécessaire à la CAO, avec toutes les fonctionnalités habituellement trouvées dans ce genre d'outils. Il a l'avantage d'avoir une version gratuite. La seule vraie différence entre la version gratuite et la version professionnelle est LISP, un langage de programmation permettant d'automatiser de nombreuses tâches répétitive de dessin.

◦ TurboCAD

- Cet outil existe depuis déjà bien longtemps et continu à évoluer régulièrement. [La version de base, malgré son prix très bas, est une alternative crédible à Autocad](#). Il existe de nombreuses versions, comportant plus de fonctionnalités, allant jusqu'à la conception 3D.

Étape 2: Conception/Modélisation 3D

On aborde ici les outils les plus spectaculaires, longtemps réservés aux seules entreprises capables de s'offrir des outils dont la licence dépassait allégrement les 10 000 € par poste. Cela a bien changé récemment, mais pas pour tout le monde! Tout d'abord, une distinction importante: Conception et Modélisation 3D ne sont pas la même chose. La conception permet d'élaborer, dessiner, visualiser un objet en 3 dimensions. La modélisation va plus loin, permettant de modéliser des mouvements des objets en relation les uns avec les autres, de simuler des efforts, etc. Autrement dit, tous les outils de conception 3D ne sont pas nécessairement aussi des outils de modélisation.

◦ SketchUp

- Le plus connu des outils de conception 3D (mais PAS de modélisation) est certainement [SketchUp](#). La version "Make" est gratuite, et il existe beaucoup de tutoriels et d'exemples sur le Net. Son principal inconvénient, de mon point de vue, est qu'il s'agit plus d'un logiciel de "dessin" en 3D, que d'un logiciel de "Conception", ce qui le rend parfaitement adapté à l'architecture, mais pas autant à l'assemblage mécanique. Ce que je veux dire, c'est que SketchUp manipule des volumes (un cube est un assemblage de 6 faces carrées) et pas réellement des objets, où un cube est un solide dont toutes les dimensions sont égales. Cela peut sembler, à la lecture, de petites nuances, mais en pratique, dans l'utilisation quotidienne, c'est une vraie différence.
- L'autre inconvénient, c'est la difficulté à exporter sous des formats standards en vue de la fabrication et la génération de GCode.
- Personnellement, c'est avec SketchUp que j'ai commencé la conception 3D, mais je suis très heureux d'avoir découvert Fusion 360 dont je ne pourrais plus me passer.

◦ SolidWorks

- Passons maintenant au Français de la série, [SolidWorks](#), lui aussi produit par Dassault Systèmes. On a ici affaire à un véritable outil industriel, capable de modéliser des solides. Il est utilisé par un certain nombre de modélistes connus, comme [Laurent Buissyne](#), dont tous les plans récents publiés dans RC Pilot sont créés avec SolidWorks.
- Je n'ai personnellement aucune expérience avec cet outil, mais à voir les résultats produits, je n'ai aucun doute sur ces capacités. Les inconvénients sont, tout d'abord, le prix de la licence, qui dépend des modules choisis, mais tourne autour de 7 000€ (sauf si vous pouvez bénéficier d'une licence éducation). Ensuite, il n'y a pas de module de fabrication directement intégré, il va donc falloir passer par un export en fichier d'échange (DXF), puis l'utilisation d'un autre outil logiciel pour générer les passages d'outils et le GCode.

◦ Autodesk Inventor

- Concurrent direct de SolidWorks, [Inventor](#) est aussi parfaitement adapté à nos besoins, même si lui non plus ne comporte pas de module capable de générer le GCode. N'ayant pas non plus d'expérience personnelle avec cet outil, je ne peux que [vous renvoyer à cette comparaison](#). Au final, tout dépend de vos besoins particuliers, qu'il vaut mieux avoir identifiés correctement, puisqu'au final, sur 5 ans, vous aurez dépensé plus de 12 000 \$US

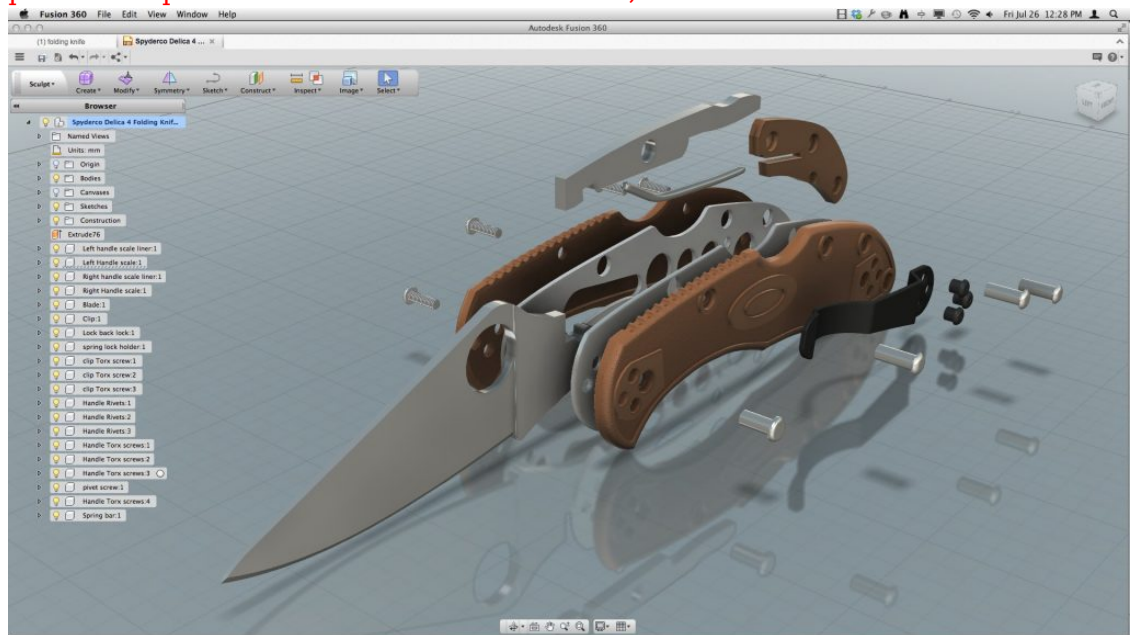
◦ Gamme Vectric (Cut2D, VCarve, Aspire, Cut3D)

- Spécialement conçus pour le fraisage en 2 et 3D, les divers logiciels de cette gamme n'ont pas toutes les possibilités d'outils comme SolidWorks ou Inventor, mais ils permettent d'effectuer toutes les étapes nécessaires, depuis le dessin de base, jusqu'à la génération du GCode. Ils comportent aussi (de base où en complément) de généreuses bibliothèques de pièces décoratives que vous pouvez

incorporer dans votre création, où juste fraiser pour elles même. Ils sont complètement orienté bois, et TRES faciles d'emploi. Pour des débuts avec une fraiseuse CNC de table, [Cut2D Desktop](#) (135€), vous permettra d'obtenir des résultats immédiatement, puisqu'à lui tout seul il est capable de l'ensemble des opérations.

◦ Fusion 360

- Le petit dernier parmi les grands, et de ce fait le moins connu, [Fusion 360 est le plus complet au regard de nos besoins](#), dans la mesure où c'est le seul à intégrer directement un module de génération de GCode, mais pas seulement! Il peut aussi piloter **directement** une imprimante 3D, générer du GCode adapté à la découpe laser où plasma. C'est donc non seulement un outil de Conception/Modélisation 3D évolué, avec simulation des joints/articulations entre pièces, mais aussi un outil de fabrication des pièces modélisées. Le Gode peut aussi être généré pour du VRAI 3D, pas juste du 2.5D (X, Y et Z) comme la plupart des autres outils le font.
- Il a aussi l'immense avantage, pour nous, d'être gratuit, pour autant que vous où votre entreprise ne fassiez pas plus de 100 000 \$US/An (ou que vous puissiez justifier d'un statut Étudiant/Professeur). J'aurais l'occasion d'expliquer cela en détail dans un futur article. **ATTENTION, vous ne pouvez bénéficier de cette possibilité que si vous utilisez le lien ci-dessus, PAS LE SITE FRANCAIS!**



◦ CamBam

- Selon vos besoins, [CamBam](#) peut s'avérer suffisant, c'est pour cette raison que je le cite ici. Il ne s'agit pas réellement de "conception 3D", mais plutôt de dessins 2D auxquels on affecte une épaisseur. Il peut aussi importer des fichiers DXF et est capable d'importer un certains nombre de formats de fichiers 3D, que vous ne pourrez cependant pas modifier dans CamBam. Très connu et très utilisé, CamBam n'est pas gratuit.

Étape 3: Génération des chemins de l'outil et du GCode

Tout d'un coup, il y a beaucoup moins de monde pour cette étape! Si vous tapez dans Google "[generate GCode from DXF](#)" vous allez avoir une foultitude de résultats (79 700 exactement!). Pour une recherche un peu moins restrictive ("GCode generator") vous obtenez 274 000 résultats! Pourquoi dire qu'il y a beaucoup moins de monde, au vu de ces résultats? Tout simplement parce qu'il s'agit, à 99%, d'utilitaires plus ou moins bien ficelés, relativement limités et manquant de souplesse. La encore, votre expérience peut différer, et je vous invite à utiliser le [Forum](#) pour partager votre expérience et poser des questions à ce sujet. Je me limiterais donc, dans cet article, aux 2 outils déjà cités.

CamBam

C'est le but premier de CamBam, parfaitement capable de générer des chemins complexes, tenant correctement compte des fraises utilisées et des machines auxquelles le [GCode](#) est destiné. La licence est abordable, et il existe maintenant en Français. Étant largement répandu, il est assez facile de trouver de l'aide à son sujet, notamment au travers de discussions sur [Usinages](#). Le seul véritable inconvénient de CamBam, par rapport à un module intégré comme dans Fusion 360, est que toute modification de dernière minute va nécessiter l'utilisation de plusieurs logiciels et la régénération de fichiers d'échange.

Exemple: j'ai prévu des percements de diamètre 5mm, et je me rends compte qu'il vaut mieux percer à 6mm. Avec CamBam, il faut

1. Modifier le dessin d'origine dans DraftSight (par exemple)
2. Exporter en fichier DXF
3. Aller dans CamBam
4. Importer le Fichier DXF
5. Refaire les opérations nécessaires pour générer le [GCode](#) depuis ce nouveau fichier.

Fusion 360

Le module CAM de Fusion 360 est pratiquement un rêve devenu réalité. Pour comparer avec les étapes de l'exemple précédent, voici les étapes nécessaires:

1. Modifier le diamètre des trous dans le module Model
2. Retourner dans le module CAM et régénérer les chemins d'outils.

C'est tout! N'oubliez pas non plus que vous pouvez directement imprimer (ou faire imprimer par des entreprises spécialisées) en 3D vos objets conçus avec Fusion 360, depuis le module Model. Selon vos besoins, vous pourrez trouver Fusion 360 un peu lourd, même si l'interface utilisateur me semble très réussie. Il y a beaucoup de possibilités, donc beaucoup à apprendre. Mais CamBam demande aussi un apprentissage non négligeable.

Gamme Vectric

Je ne veux pas répéter ce que j'ai écrit plus haut, mais [l'ensemble de la gamme Vectric](#) est à classer ici aussi.



Étape 4: Envoi du [GCode](#) à la machine CNC

Les étapes 4 et 5 ne sont distinctes, pratiquement, qu'avec [GRBL](#), qui ne comporte pas d'interface utilisateur. Pour tous les autres programmes couramment utilisés par les amateurs et artisans (par différence avec l'industrie), c'est le même logiciel qui va effectuer ces 2 étapes, bien que généralement la partie "envoi des impulsions aux moteurs" va être gérée par le firmware de la carte externe branchée sur le PC. L'étape 4 commence donc avec l'existence d'un fichier GCode (généralement avec une extension .NC ou .CNC), dont il nous faut maintenant "faire quelque chose avec"....

Si vous utilisez GRBL, du fait qu'il "tourne" sur un Arduino, vous allez avoir besoin d'un logiciel interface utilisateur, pour communiquer et contrôler GRBL.

- **Panneau de Contrôle GRBL**

- Ce logiciel est le seul à être francisé (à ma connaissance) . Vous [pouvez le télécharger ici](#). Il comporte tous les contrôles nécessaires au pilotage d'une fraiseuse CNC, tels qu'on peut les trouver sur des machines industrielles. Il évolue constamment. Bien qu'il ne comporte pas la visualisation de la pièce travaillée et de la position de l'outil, il reste mon préféré.

- **grblControl / Candle**

- Relativement récent parmi ces logiciels "envoyeurs de GCode", il a l'avantage de pouvoir fonctionner aussi sur Linux, en plus de Windows. Il permet aussi de visualiser

graphiquement le GCode, mais est un peu moins complet en ce qui concerne l'interaction avec GRBL. Vous pouvez le [télécharger directement depuis son répertoire GitHub](#), puisqu'il est Open Source, tout comme GRBLPanel / Panneau de Contrôle GRBL

Comme dit précédemment, il existe de nombreux autres outils remplissant cette fonction. Vous pouvez [accéder à la liste ici, complète](#) (ou à très peu près, je crois).

Étape 5: Interprétation du GCode et envoi des impulsions aux pilotes des Moteurs

Ne nous y trompons pas, dans pratiquement tous les cas de figure (pratiquement, parce qu'avec le Raspberry PI et équivalents, on voit commencer à apparaître des solutions complètement intégrées), c'est une carte reliée au PC qui va effectivement envoyer les commandes aux drivers. Et le code se trouve dans le firmware de la carte. Pas vraiment de différences donc avec GRBL, dans le principe. La vraie différence vient du type de liaison entre le PC et la carte électronique (généralement propriétaire), ainsi que du Firmware lui même, bien sûr.

- **Mach3, Mach 4**

- Plus aucun PC, et encore moins de portable/tablette ne comportent maintenant de port parallèle, qui fait un peu figure d'antiquité. Pourtant Mach3, comme [Mach4, bien plus récent](#), ont tous deux besoin d'un port parallèle, dans leurs versions de base. Il existe des plugins et différentes cartes permettant d'utiliser un port USB, mais il me semble préférable d'utiliser un logiciel conçu dès l'origine pour utiliser un port USB où un port Ethernet. Ces logiciels sont complets, même si un peu complexes à mettre en œuvre au départ. Ils peuvent piloter jusqu'à 6 axes, ce qui dépasse très largement les besoins amateurs et même professionnels en fraisage courant.

- **PlanetCNC**

- Alternative relativement peu connue, [PlanetCNC](#) conçoit et fabrique ses propres cartes contrôleurs USB, et diffuse son propre logiciel (gratuit) compatible avec ces cartes. L'ensemble est d'un fonctionnement robuste et sans soucis particuliers, après avoir proprement paramétré le logiciel en fonction de sa machine.

- **CNCdrive**

- Alternative intéressante et moderne à Mach3, avec une licence à 55€ en ce moment, [le logiciel de CNCdrive, UCCNC](#) peut importer les réglages faits pour Mach3. Certaines des cartes proposées sont compatibles avec Mach3, elles existent pour port USB où pour port Ethernet. L'utilisation d'un port Ethernet comporte plusieurs avantages (par rapport à un port USB), notamment la possibilité d'accroître la distance entre le PC et la carte sans risques, et une connexion plus stable, sans bruits électriques.

- **GRBL**

- Il n'est bien sûr [pas question d'oublier GRBL](#), qui est le seul à ne pas avoir besoin de carte propriétaire, mais de l'Open Source Arduino. Un Uno suffit, et le simple fait que toutes ces fonctionnalités "tiennent" dans cette mémoire ridiculement petite est complètement étonnant. La version actuelle, 1.1, est disponible sur [Github ici](#). GRBL va évoluer vers l'utilisation d'un Atmel (Arduino Due)

Tableau Récapitulatif de possibles composants de votre Chaîne Logicielle

Logiciels / Etapes	Conception 2D	Conception / Modélisation 3D	Génération GCode	Envoi GCode	Interprétation GCode - Impulsions Moteurs	Hardware
InkScape	X					
DraftSight	X					
TurboCAD Standard	X					
SketchUp		X (Conception)				
SolidWorks	X	X				
Inventor	X	X				
Fusion 360	X	X	X			
Vetric Cut 2 D	X		X			
Vetric VCarve	X		X			
Vetric Aspire	X	X (Pas de modélisation)	X			
Vetric Cut3D		Lecture uniquement fichiers 3D	X			
CAMBAM		X (Conception simple 2.5D)	X			
Panneau de Contrôle GRBL				X		
GrblControl				X		
GRBL					X	Arduino Uno - USB-Série
Mach3 et Mach 4				X	X	Cartes Contrôleurs, Port // ou USB avec Plugin
PlanetCNC				X	X	Carte USB propriétaire
CNCdrive				X	X	Cartes USB et Ethernet

J'espère que cet article aide à clarifier qui fait quoi, et comment. Surtout, n'hésitez pas à commenter et à poser des questions, ci-dessous, dans les commentaires, ou dans le Forum, qui est particulièrement adapté à ce genre de discussion.



Panneau de Contrôle Grbl

Comme je l'ai évoqué dans mon article précédent, [Grbl](#) est un logiciel "firmware", exécuté sur un [Arduino](#), qui analyse le [GCode](#) reçu, construit un modèle mathématique des actions à effectuer, et exécute ces mouvements en envoyant un flot continu de pulsions haute fréquences aux moteurs pas à pas. Le seul petit détail, c'est que Grbl n'a aucune interface utilisateur, par différence avec Mach3, PlanetCNC où encore LinuxCNC, qui tous répondent à la même définition, mais permettent de piloter la machine depuis des écrans appropriés. Il faut donc disposer d'un autre logiciel, exécuté sur une autre machine, en liaison avec Arduino, pour pouvoir interagir avec la machine. Il existe plusieurs de ces logiciels, mais aujourd'hui je veux vous présenter le "Panneau de Contrôle Grbl" (Grbl-Panel, en Anglais) logiciel [Open source](#) dont je viens de finir la traduction complète.

Présentation

Le Panneau de Contrôle Grbl n'est pas le plus "joli" des divers logiciels interfaces pour Grbl qui existent, par contre il est certainement le plus "professionnel", dans la mesure où il intègre de nombreuses fonctions habituellement trouvées sur des machines industrielles. Il "clone" plus où moins un panneau physique comme on pourrait le trouver sur une machine Fanuc où Haas. Tous les outils nécessaires au réglage et au fraisage sont organisés de manière claire, correspondant à l'expérience acquise par la pratique de dizaines d'années d'organisation d'un poste de machiniste. L'interface est épurée et ne comprend pas la partie graphique qui flatte l'œil mais n'apporte pas grand chose de concret à l'interface home machine.

Le Panneau de Contrôle Grbl est compatible avec Grbl 0.9g et suivants, et intègre notamment les nouvelles possibilités de Grbl v1.1, dont nous parlerons plus loin dans cet article. Il donne aussi la possibilité de switcher la disposition des contrôles de manière à être mieux adapté à un(e) gaucher(ère).

Survol Rapide

Une opération de fraisage typique requière le déplacement manuel par petits pas (jogging) au point origine 0 pour la pièce considérée. Souvent cela demande une mise au point sur chacun des axes. C'est pour cette raison qu'il y a un bouton zéro près de chaque axe. Une fois que vous exécutez un fichier GCode, vous ne devriez plus être capable de "jogger" où d'envoyer des commandes manuelles. ici les 2 sont liés, une opération bloque l'autre. Lorsque qu'une condition d'alarme est détectée par Grbl, l'exécution du GCode est stoppée. Les boutons de contrôle et la visualisation est groupée en sous-panneaux qui sont activés/désactivés en fonction de ce que le Panneau de Contrôle Grbl est en train de faire. Un onglet pour les réglages fourni un certain nombre de zones éditables qui modifient le fonctionnement du panneau, où ce qui est envoyé à Grbl. Bien sûr vous pouvez aussi modifier directement les réglages internes à Grbl depuis ce même onglet.

Dans le Détail

1- Installation

Il n'y a pas encore de procédure d'installation, ce qui va venir. Pour l'instant il faut donc [télécharger ici un fichier ZIP](#), et le décompacter, de préférence dans votre répertoire C:\Program Files (x86), ou tout simplement c:\Programmes sur une machine 32 bits. Par la suite, vous trouverez la version francisée sur GitHub, avec la version originale de Grbl-Panel.

2 - Connection

Pour effectuer la liaison série avec l'Arduino, il vous faut brancher un câble USB entre votre Ordi et Arduino. Le driver USB/Série est installé avec l'IDE Arduino, sinon, un Windows récent comme Windows 7 où Windows 10 devrait être capable de le trouver et l'installer tout seul comme un grand, lorsque vous branchez votre micro-contrôleur. Vous avez certainement remarqué un onglet IP, permettant d'effectuer la liaison série au travers d'un module WiFi pour Arduino. Nous reviendrons plus tard sur cette possibilité, et comment effectuer ce lien coté Arduino. A ma connaissance, Le Panneau de Contrôle Grbl est le seul à offrir cette option. Avant de cliquer sur "Connecter", tous les boutons et panneaux sont désactivés, aucune des diverses options n'est disponible.

3 - Réglages

Interface Réglages Macros

Réglages Grbl

ID	Valeur	Description
\$0	10	Signal pas, uSec
\$1	25	Délai pause entre pas, mSec
\$2	0	Masque inversion signal pas
\$3	0	Masque inversion direction pas:
\$4	0	Inversion activation pas, bool
\$5	0	Inversion switch limites, bool
\$6	0	Inversion de la sonde, bool
\$10	1	Masque du suivi de statut
\$11	0.010	Ecart de jonction, mm
\$12	0.002	Tolérance d'Arc, mm
\$13	0	Suivi de statut en pouces, bool
\$20	0	Limites Logiques, bool
\$21	0	Limites Physiques, bool
\$22	0	Cycle de référencement (homing), bo
\$23	0	Masque inversion dir. Référencement
\$24	25.000	Avance Référencement, mm/min
\$25	500.000	Détection Référencement, mm/min
\$26	250	Délai rebond référencement .mSec
\$27	1.000	Retrait Référencement, mm
\$30	1000	Max t/mn
\$31	0	Min t/mn
\$32	0	
\$100	250.000	X, pas/mm

\$132

Demier paramètre Grbl

Charge réglages

Ces réglages sont toujours sauvs à la sortie du Panneau GRBL

Position

G28

Va en pos. spé. 1

G30

Va en pos. spé. 2

G90 G10 L20 P0 X0 Y0

Zero ttes CMD

G90 G10 L20 P0 X0

Travail X 0 cmd

G90 G10 L20 P0 Y0

Travail Y 0 cmd

G90 G10 L20 P0 Z0

Travail Z 0 cmd

Rafraichir

Divers.

200

Intervalle mäj (ms)

17

Q Taille Max

127

RBuf Taille Max

1.5

Délai démarrage

Ext Fic. par défaut

☒

Pause sur Erreur

☐

Connecte à l'ouverture

☒

MàJ Statut

☐

IU pour Gaucher

Rafraichir

Décalages (2)

Positions Machine

G28

0.000

0.000

0.000

Rè

G30

0.000

0.000

0.000

Ré

(Les décalages trav. se font depuis la page principale)

Retrouver

Jogging

☒

Métrique

1.0.1.0.01.0.001

Incréments avance

100.50.10.5

Taux avance

10.1.0.1.0.01

Incréments avance

1000.500.100.5

Taux avance

Bouton Taux Répétition, r/sec

2.5

X Taux répétition

2.5

Y Taux répétition

2.5

Z Taux répétition

☒

Activer flèches clavier

Rafraichir

ouvez obtenir en tapant "\$\$" dans un terminal série, [comme expliqué dans l'article à ce sujet](#). Un certain nombre de paramètres, plus bas dans la liste, permettent d'indiquer des valeurs interactivement alors qu'auparavant il fallait aller modifier le fichier config.h, comme pour les courses maximum (Valable depuis Grbl 1.1). Pour modifier une valeur, cliquez sur la case valeur, pour la sélectionner, une 2ème fois pour l'éditer, puis un double-clic rapide pour aller écrire la nouvelle valeur dans Grbl. Beaucoup plus simple, vous pouvez simplement appuyer sur la touche entrée. Vous pouvez vérifier que la nouvelle valeur a bien été prise en compte et cliquant sur "charge réglages".

Sur la partie droite figurent un certain nombre de réglages qui vont modifier ce que le Panneau de Contrôle envoie à Grbl, et non directement Grbl lui même. Comme cet article n'est pas un manuel utilisateur, ni un manuel d'apprentissage de GCode, je ne vais pas entrer dans le détail de chacun des réglages. Je vais par contre évoquer rapidement leur utilisation:

Sous Panneau Position. Les commandes GCode G28 et G30 sont communément utilisées pour indiquer la position de repos, où de départ et de fin. Nous verrons par la suite comment indiquer à Grbl quelles sont ces positions sur votre machine. Les autres commandes font ce qui est indiqué à coté. Plongez-vous dans un manuel sur GCode pour comprendre quels sont les ordres envoyés à la machine, mais à priori je ne changerais rien ici, sauf cas particulier.

Sous Panneau Divers. La aussi, sauf à savoir vraiment ce que vous faites, je ne changerais rien ici, sauf pour la dernière ligne "Ext Fic. par Défaut". Cette entrée est destinée à faciliter la recherche des fichiers GCode, sur la page Interface. Suivant vos habitudes et préférences, entrez "nc", où "cnc", sans le point devant. Les autres paramètres concernent: la fréquence de rafraîchissement des données en provenance de Grbl, la taille de la Queue de commandes envoyées, et le buffer mémoire

dans lequel ces commandes sont stockées, et le délai au démarrage, en millisecondes.

Parmi les cases à cocher au bas du panneau, il est préférable de laisser celles cochées par défaut. Si vous cochez “connecte à l’ouverture”, le panneau de contrôle essayera de se connecter comme il l’est maintenant la prochaine fois que vous le lancez. Si vous cochez interface Utilisateur pour gaucher, sur la page Interface, les panneaux Position et Grbl Jogging s’intervertissent, pour avoir Jogging à gauche et Position à droite.

Sous Panneau Jogging. Tout d’abord la case à cocher métrique, qui signifie que tout ce que vous allez faire avec le jogging est en dimension métriques. Sinon, tout est en pouces. ATTENTION, cela ne concerne que le jogging. Les incréments d’avance sont les différents nombre de millimètres (pour les valeurs métriques) d’avance que vous voulez commander. Les taux d’avance, est la vitesse, en millimètres/minutes, à laquelle vous voulez que le déplacement s’effectue. Ces choix sont ceux proposés sur la page Interface, dans le sous panneau jogging. Par exemple, si vous sélectionnez 10 comme Incrément, et 100 comme Taux, un appui sur X+ provoquera un mouvement de 10 millimètres sur l’Axe des X à une vitesse de 100 millimètres/minutes. il faudra donc $(100/60) \times 10$, soit 16.66 secondes pour effectuer ce déplacement.

Le taux de répétition concerne le nombre de fois où l’ordre est envoyé, par seconde. La case à cocher dessous permet d’utiliser les flèches clavier pour ces déplacements; Pratique sur un ordi, pas vraiment sur une tablette....

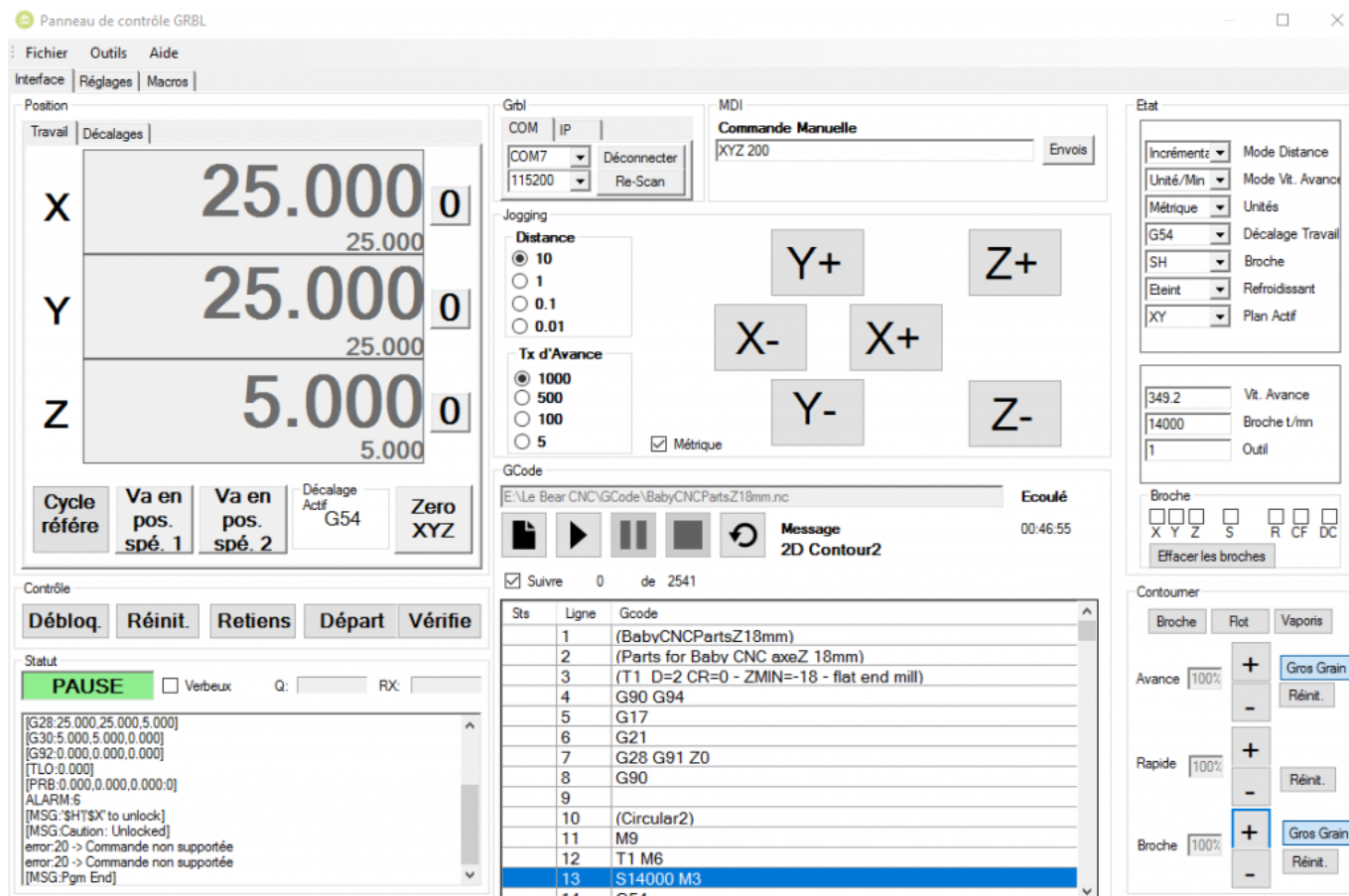
Sous Panneau Décalage. Nous venons de voir ci-dessus que les commandes GCode G28 et G30 permettent de dire à la machine d’aller se positionner sur une position préalablement définie, qui pourrait correspondre à celui d’une sonde d’outil, par exemple, où tout simplement au coin supérieur gauche de repos de la machine (sans déclencher les fins de course). Pour définir cette position, positionnez (par jogging) la broche exactement où vous le souhaitez, puis revenez sur cette page réglage, et appuyez sur le bouton “Ré”, pour entrer ce réglage. Par la suite, à la fin de votre programme GCode (si votre générateur ne le fait pas), ajoutez cette commande G28, et la machine retournera à cette position que vous venez de définir. Même chose pour G30.

Comme indiqué au-dessus de ces sous-panneaux, ces réglages sont sauvegardés à chaque fois que vous quittez le programme, vous n’avez donc rien à faire pour les conserver.

4 - Fonctionnement

Nous aurons d’autres occasions de revenir plus en détail sur les différentes manières de régler et faire fonctionner une machine, et donc de revenir sur décalages (offsets), référencements (homing), etc. Pour l’instant nous allons simplement expliquer les bases du fonctionnement du Panneau de Contrôle Grbl pour faire fonctionner votre machine. En gros, on a 2 manières principales de piloter sa machine au travers de Grbl depuis le panneau de contrôle: “manuellement”, en envoyant directement des commandes GCode, où par un fichier de programme GCode.

La partie “manuelle” comporte la partie “Jogging”, mais aussi la zone de saisie “Commande Manuelle”, où vous pouvez saisir vous-même une commande, où une suite de commandes, qui sont envoyées à la machine lorsque vous pressez le bouton Envoi. Il comporte aussi la partie “Macros”, qui peut s’avérer très pratique pour de petites opérations “semi manuelles”. Vous définissez une macro dans l’onglet du même nom, et vous verrez apparaître, sous la zone de saisie de commande manuelle, un bouton du nom de votre macro. Appuyer sur ce bouton envoi à la machine la suite de commandes GCode définie dans votre macro.



Le panneau de Contrôle prêt à lancer l'exécution d'un programme GCode

La partie "automatique", où Fichier GCode, commence par la sélection d'un fichier GCode, en cliquant l'icone ressemblant à un document. Une fois le fichier chargé, les éventuelles lignes de statut créées par l'envoi de commandes manuelles sont remplacées par les lignes du programme GCode. La colonne statut passe de "Sent", lorsqu'elle a été envoyée à Grbl, à "OK" une fois exécutée, ou à "Err" si GRBL n'a pas pu comprendre/exécuter la commande. Selon l'erreur, vous pouvez sélectionner la ligne de code suivante, et presse la flèche envoi à nouveau, pour poursuivre l'exécution du programme. J'ai personnellement expérimenté avec un programmes GCode dépassant allégrement le million de lignes sans rencontrer le moindre problème. Le dernier commentaire rencontré s'affiche dessous "Message".

Pendant l'exécution d'un programme GCode, tous les boutons, zones, etc. , sont désactivés sauf les boutons pause et arrêt, qui vont respectivement mettre en pause le déroulement de la lecture des lignes de GCode, où complètement arrêter le programme.

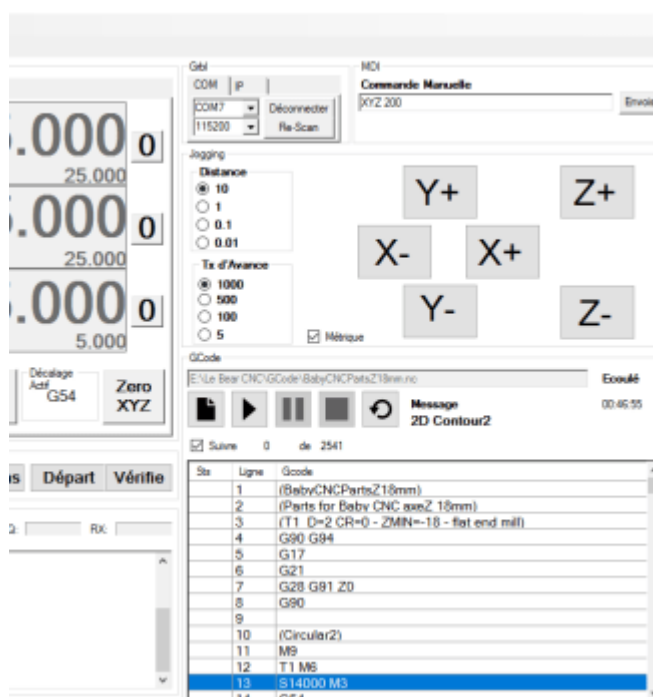
Conclusion

Ce contrôleur pour Grbl n'est certainement le plus connu des divers programmes existants ayant la même fonction, mais c'est à mon sens le plus complet, offrant des possibilités inconnues des autres logiciels du même type, telles que macros et plusieurs décalages pré-réglés. Il intègre déjà toutes les nouvelles fonctions de la version 1.1 de Grbl, et est entièrement en Français (si votre Windows est réglé pour être en Français).

Vous pouvez télécharger les sources de Grbl-Panel sur GitHub: <https://github.com/gerritv/Grbl-Panel>, et compiler/modifier avec Visual Studio. Peut-être plus pratique pour beaucoup, [vous pouvez télécharger les fichiers binaires ici, sur le site](#), mais aussi sur Github, en téléchargeant la dernière mise à jour, V1.0.9.8.

Un petit mot, pour finir, sur le futur du Panneau de Contrôle Grbl. Gerrit et moi avons pas mal discuté pendant mon travail de localisation/traduction, qui va permettre à ce logiciel d'être maintenant traduit dans d'autres langues que juste Anglais et Français. Il nous apparaît évident à tous deux qu'il est beaucoup plus pratique, et conforme aux pratiques "modernes", de piloter sa machine CNC depuis une tablette que depuis un vieux PC coincé à l'atelier, prenant poussière et toiles d'araignées.... C'est pourquoi le Grbl-Panel, en dehors d'autres améliorations ponctuelles, va être porté en C# avec Xamarin, de manière à pouvoir être exécuté depuis n'importe quel matériel fonctionnant sous Windows où sous Android, en communiquant avec l'Arduino par Bluetooth (donc sans fil), et peut-être aussi sous Apple/iOS.

Autrement dit, le Panneau de Contrôle GRBL est bien vivant, et va continuer à évoluer. Si vous avez des remarques, suggestions, souhaits, n'hésitez pas à en faire part, nous ferons de notre mieux, avec Gerrit, pour en tenir compte si cela apparaît faisable, et souhaitable...



Panneau de Contrôle Grbl

Note **3.86** sur 5

0,00 € (0,00 €) [Lire la suite](#)

[Configurer et Paramétrer GRBL](#)

C'est une question qui revient souvent, après avoir installé [GRBL](#). Comment dois-je faire pour configurer et paramétrer GRBL pour ma machine? Bien sûr, chaque machine est différente, dans ses courses, dimensions, moteurs, vitesse dont elle est capable, etc....

Il y a deux types différents de “réglages” possible: “En dur”, à la compilation, en modifiant des valeurs dans un fichier, et en interactif, alors que le programme s’exécute. Dans la suite de cet article, et pour mieux différencier les deux, on va appeler les réglages en dur “configuration”, et ceux en interactif “paramétrage”, même si cette différence de vocabulaire est un peu arbitraire. J’espère simplement que cela contribuera à la compréhension.

Configuration

On va commencer par décrire ce qui ne l’est pas très souvent, la modification de valeurs dans certains fichiers qui seront compilés avec GRBL lorsque vous le flashez sur l’[Arduino](#). Si vous avez déjà installé GRBL dans votre Arduino, vous l’avez certainement compilé et flashé tel que, sans faire de modifications. Très bien, cela fonctionne, et ce n’est pas un problème particulier. Il peut cependant être utile de savoir comment modifier la configuration de base, et en quoi cela peut vous aider.

Dans votre IDE (Visual Studio pour moi), ouvrez le fichier config.h qui doit se trouver dans votre répertoire GRBL. C’est un assez long fichier (+- 400 lignes) extrêmement bien commenté, en Anglais bien sûr. Nous allons isoler ici les paramètres de configuration les plus importants pour s’adapter au mieux à votre machine spécifique.

Tout d’abord, si vous liez en parallèle 2 interrupteurs de butée pour 1 même axe, au lieu d’avoir un switch câblé séparément pour chaque extrémité, il faut dé-commenter la ligne commençant par define, ci-dessous, en supprimant les “//”

```
[crayon-5c4743755e3ce281685047/]
```

Ensuite, et toujours en liaison avec vos interrupteurs de fin de course, il faut définir si vous câblez vos switches en “normalement ouvert”, où en “normalement fermé”. Par défaut, GRBL est défini pour utiliser des switches normalement ouverts. Dé-commentez la ligne pour un fonctionnement en normalement fermé:

```
[crayon-5c4743755e3dc235282067/]
```

Depuis peu, GRBL est capable de gérer (s’il reçoit les commandes [G-Code](#) correspondantes) la vitesse de rotation de la broche. Si vous vous servez de cette fonction, laissez là activée (elle l’est par défaut), sinon désactivez-là. Si vous la laissez activée, il vous faudra aussi définir la plage de vitesses dont votre broche est capable:

```
[crayon-5c4743755e3e0426135403/]
```

Jusqu’à présent, je n’ai pas eu besoin de m’intéresser de près aux autres paramètres du fichier config.h, mais sachez que vous avez peut-être là la solution à certaines difficultés que vous pourriez rencontrer en utilisant GRBL

Paramétrage

Le paramétrage se fait alors que l’Arduino est branché et que vous êtes en liaison série avec lui. Il existe beaucoup de logiciels différents permettant d’envoyer le G-Code et de contrôler GRBL, que ce soit depuis un PC, un Mac où une machine Linux. Pour éviter d’entrer de suite dans d’autres considérations de choix pour ces logiciels, nous allons paramétrer GRBL le plus simplement du monde avec un terminal série, en envoyant manuellement les commandes.


```
3. COM3 (Arduino Uno (COM3)) x +
Launching microcom serial connection on /dev/ttyS4.
Press Ctrl+X to exit.

Grbl 0.9j ['$' for help]
```

Lorsque vous vous connectez avec un terminal série sur votre Arduino, si le flashage s'est passé correctement, vous devriez obtenir une réponse Grbl 0.9j ['\$' for help], tel que vous pouvez le voir ici.

Pour lister l'ensemble des paramètres qui vous permettront d'adapter Grbl aux spécificités de votre machine, il vous suffit de taper \$\$. Grbl va répondre en renvoyant la liste visible ci-dessous.

Paramètres 100, 101 et 102, nombre de pas par millimètre.

Nous allons tout d'abord nous intéresser aux paramètres 100, 101 et 102. Ces paramètres, comme

```
Grbl 0.9j ['$' for help]
$0=10 (step pulse, usec)
$1=25 (step idle delay, msec)
$2=0 (step port invert mask:00000000)
$3=0 (dir port invert mask:00000000)
$4=0 (step enable invert, bool)
$5=0 (limit pins invert, bool)
$6=0 (probe pin invert, bool)
$10=3 (status report mask:00000011)
$11=0.020 (junction deviation, mm)
$12=0.002 (arc tolerance, mm)
$13=0 (report inches, bool)
$20=0 (soft limits, bool)
$21=0 (hard limits, bool)
$22=0 (homing cycle, bool)
$23=0 (homing dir invert mask:00000000)
$24=25.000 (homing feed, mm/min)
$25=500.000 (homing seek, mm/min)
$26=250 (homing debounce, msec)
$27=1.000 (homing pull-off, mm)
$100=250.000 (x, step/mm)
$101=250.000 (y, step/mm)
$102=250.000 (z, step/mm)
$110=600.000 (x max rate, mm/min)
$111=600.000 (y max rate, mm/min)
$112=600.000 (z max rate, mm/min)
$120=10.000 (x accel, mm/sec^2)
$121=10.000 (y accel, mm/sec^2)
$122=10.000 (z accel, mm/sec^2)
$130=550.000 (x max travel, mm)
$131=350.000 (y max travel, mm)
$132=130.000 (z max travel, mm)
ok
```

indiqué dans le commentaire entre parenthèses, définissent le nombre de pas (des moteurs pas à pas) nécessaires pour parcourir une distance de 1 millimètre sur l'axe indiqué. La plupart du temps, 100 et 101 (X et Y) seront identiques, alors que Z (102) aura une valeur différente. Pour vous faciliter la tâche, vous trouverez au bas de l'article un [calculateur pour trouver cette valeur](#), mais la formule est assez simple:

Valeur = Nombre de pas (du moteur) pour faire 360°/Pas de la Visse sans fin (où nombre de dents de la poulie * espace dents chaîne où courroie)

Résultat = Valeur x Micro Pas.

Exemple: Moteur Nema 17, 1.8° par pas, soit 200 pas pour 360°

Courroie dentée, 2mm d'espace entre dents.

Poulie 20 dents

Opération: $(200/40) * 4$ (par exemple) = 5

Dans cet exemple, il faut 5 pas des moteurs pour parcourir une distance de 1 millimètre. **Attention, dans GRBL, tout est en millimètres, donc, si vous avez des dimensions en pouces (inches), n'oubliez pas de multiplier par 25,4 avant de faire vos opérations.**

Paramètres 130, 131 et 132. Distance maximum à parcourir sur chacun des axes.

Ici, il n'y a pas de calcul à faire, il suffit d'entrer en millimètres la longueur de coupe sur cet axe. En théorie, et en l'absence de fausses manipulations, si ces distances sont entrées correctement, les détecteurs de fin de course devraient être inutiles.

Paramètres 110, 111 et 112. Vitesse maximum sur X, Y et Z.

Ici on entre plus dans le domaine de l'expérimentation personnelle que dans le purement rationnel. Il s'agit ici de vitesse de déplacement maximum, HORS TRAVAIL, c'est à dire pour se repositionner, et pas pour couper. Il y a de nombreuses discussions sur les forums à ce sujet, et nombreux sont ceux qui soutiennent que des valeurs de 12 000 à 14 000 sont parfaitement acceptables, alors que d'autres pensent que 5 000 est bien suffisant. Tout dépend de votre mécanique ET de votre électronique. Pour commencer, une valeur de 8 000 me semble correcte, et la moitié, 4 000, pour l'axe des Z. Expérimentez avec des commandes manuelles pour voir comment votre machine se comporte.

Paramètres 24, 25, 26 et 27. Homing. Comme dans E.T., « maison »!

La procédure de homing (retour à la maison, plus techniquement, point 0, où X= 0, Y = 0 et Z = 0), est importante, et pas le plus simple réglage à effectuer, mais il n'est pas obligatoire, simplement il faudra, s'il n'est pas utilisé, que tous vos fichiers G-Code soient générés pour exprimer des coordonnées relatives, et pas absolues, un concept sur lequel nous reviendrons plus tard. Tout d'abord, ces paramètres ne seront pris en compte que si le paramètre 22 est à 1 (activé), et il faut que vous ayez câblé vos fins de course.

La procédure de Homing va déplacer le portique sur X et Y, d'abord relativement vite, jusqu'à ce qu'il touche les fins de course, puis plus lentement pour détecter le point exact où l'interrupteur se déclenche, en effectuant des mouvements lents de va et vient.

- Le **paramètre 24** correspond à cette vitesse lente. 25 mm/min devrait fonctionner. Le but est d'obtenir une localisation exacte et répétable à chaque fois.
- Le **paramètre 25** correspond à la vitesse de déplacement jusqu'à déclencher les interrupteurs. Une vitesse trop élevée, et votre portique risque de heurter violemment vos fins de course. Une vitesse trop lente, et le processus va prendre trop de temps. Expérimentez, 500 à 600 mm/min sont de bonnes valeurs de départ.
- Le **paramètre 26** est la valeur en millisecondes pendant laquelle le signal électrique de l'interrupteur peut « rebondir » lorsqu'il est déclenché. Ce bruit peut (et devrait!) être filtré électroniquement, mais GRBL utilise ce court délai pour laisser le temps au signal de se

stabiliser.

- Le **paramètre 27** est la valeur de laquelle le processus de homing va se décaler des fins de course, de manière à éviter un déclenchement accidentel des fins de course, une fois que le processus a été achevé.

Il n'est normalement pas nécessaire de toucher aux autres paramètres, sauf cas très particuliers, sur lesquels nous reviendrons éventuellement à l'occasion.

```

$0=10 (step pulse, usec)
$1=25 (step idle delay, msec)
$2=0 (step port invert mask:00000000)
$3=0 (dir port invert mask:00000000)
$4=0 (step enable invert, bool)
$5=0 (limit pins invert, bool)
$6=0 (probe pin invert, bool)
$10=3 (status report mask:00000011)
$11=0.020 (junction deviation, mm)
$12=0.002 (arc tolerance, mm)
$13=0 (report inches, bool)
$20=0 (soft limits, bool)
$21=0 (hard limits, bool)
$22=0 (homing cycle, bool)
$23=0 (homing dir invert mask:00000000)
$24=25.000 (homing feed, mm/min)
$25=500.000 (homing seek, mm/min)
$26=250 (homing debounce, msec)
$27=1.000 (homing pull-off, mm)
$100=250.000 (x, step/mm)
$101=250.000 (y, step/mm)
$102=250.000 (z, step/mm)
$110=10000.000 (x max rate, mm/min)
$111=10000.000 (y max rate, mm/min)
$112=5000.000 (z max rate, mm/min)
$120=10.000 (x accel, mm/sec^2)
$121=10.000 (y accel, mm/sec^2)
$122=10.000 (z accel, mm/sec^2)
$130=550.000 (x max travel, mm)
$131=350.000 (y max travel, mm)
$132=130.000 (z max travel, mm)

```

Comment changer les paramètres?

Il suffit de taper \$nombre du paramètre = nouvelle valeur, par exemple \$110=10000. Faites à nouveau \$\$ pour vérifier les nouvelles valeurs. Comme vous pouvez le voir sur cette copie d'écran les paramètres 110, 111 et 112 ont été modifiés.

Conclusion

J'espère que cet article aura levé certaines difficultés à l'utilisation de Grbl. Bien que ce programme ne nécessite qu'un Arduino Uno, il est assez complet et permet de piloter des machines CNC « comme les grands ». Dans un prochain article, nous évoquerons différents programmes pilotant l'Arduino par le biais d'une interface graphique, et examinerons l'un deux plus en détails, sa version Française devant être disponible sous peu.

Bien que tout ce qui précède soit relatif à GRBL, un certain nombre de principes et de calculs sont valables pour TOUTES les machines CNC.



GRBL c'est quoi?

GRBL est un logiciel pour contrôler le mouvement de machines qui font “des trucs”. Si le mouvement [Maker](#) était une industrie, GRBL en serait le standard.

La plupart des imprimantes 3D Open-Source sont basées sur GRBL. Il a été adapté pour être utilisé dans des centaines de projets, comprenant des machines à découpe laser, des écritoirs manuels automatisés, perceuses, peintre de graffiti et machines à dessins bizarroïdes... En raison de ses performances, de sa simplicité et de sa frugalité en besoins matériels, GRBL a grossi en un vrai petit phénomène Open Source.

Origine

Vous avez peut-être déjà lu ce nom ici où là, si vous vous intéressez à la CNC amateur et à la manière de les contrôler. GRBL est un contrôleur logiciel libre, open-source, haute performance, écrit en C hautement optimisé pour tourner sur un [Arduino](#) “standard” (Uno).

Le créateur, [Simen Svale Skogsrud](#) explique sur son blog que lorsqu’il commanda sa première fraiseuse pilotée par ordinateur, en 2007, il a été perplexe quant à la manière de la contrôler. La pratique courante à l’époque (qui n’a que peu changée!) était de sauver une vieille boîte beige avec un port parallèle et d’utiliser cela pour envoyer les impulsions aux moteurs. Cela lui apparut juste dépassé. Il voulait un simple système embarqué sur la machine qui dialoguerait en USB avec un ordinateur portable. Ce sont exactement les mêmes raisons qui m’ont poussées à choisir Arduino + GRBL comme solution principale.

Choix du Contrôleur

Simen a choisi l’[Arduino](#) pour plusieurs raisons. La plus importante étant que c’était, et c’est toujours, le système embarqué le plus populaire dans la communauté du DIY (Do It Yourself, Faites-le Vous Même). Un ordinateur d’une simplicité touchante.

Pourtant, c’est une machine chétive au regard de la tâche à effectuer. Ses 2kb de mémoire sont risibles, même par rapport au standard des années 80.

Voici ce que GRBL doit faire avec ces 2Kb:

- - Analyser G-Code, un langage ordinateur cryptique né dans les années 50 pour décrire les

actions idéalisées d'une fraiseuse

- - Construire un modèle de ces actions et les traduire en une séquence de mouvements physiquement possible pour une machine donnée.
- - Exécuter ces mouvements en envoyant un flot continu de pulsions haute fréquence aux moteurs pas à pas qui déplacent effectivement l'outil.

Faire tenir tout ça dans si peu d'espace est un vrai tour de force. Les premières versions de GRBL ont été offertes en Open-Source par Simen en 2009. Depuis 2011, c'est **Sungeun K. Jeon Ph.D.** (@**chamnit**) qui est le leader du projet.

GRBL, pour qui?

Pour ceux qui font du fraisage et ont besoin d'un contrôleur simple pour leur système basé sur l'omniprésent Arduino. Ceux qui haïssent d'encombrer leur atelier avec un PC-Tour juste pour le port parallèle. Bricoleurs/Hackers qui veulent un contrôleur écrit en compact et modulaire C comme base de leur projet.

Fonctionnalités

GRBL est prêt pour la production non intensive. Nous l'utilisons pour tous nos fraisages, depuis Laptop où PCs, utilisant d'excellentes interfaces utilisateur. Il est écrit en C optimisé pour utiliser les fonctions intelligentes des puces Atmega328, pour obtenir un timing précis et des fonctions asynchrones. Il est capable de maintenir un taux de pas supérieur à 30kHz, et délivre un courant propre de pulsations de contrôle.

GRBL est pour l'usinage 3 axes. Pas d'axe de rotation (pas encore!) – Juste X, Y et Z.

L'interpréteur [G-Code](#) implémente un sous-ensemble du standard NIST rs274/ngc et est testé au travers d'un grand nombre d'outils sans problèmes. Mouvements linéaires, circulaires et hélicoïdaux sont supportés sans problèmes.

- G-Codes supporté avec **v0.9i**
 - **G38.3, G38.4, G38.5:** *Sondage*
 - **G40:** *Modes de compensation du rayon de coupe*
 - **G61:** *Modes de contrôle du chemin de l'outil*
 - **G91.1:** *Modes Distance d'Arc IJK*
- G-Codes supportés dans **v0.9h**
 - **G38.2:** *Sondage*
 - **G43.1, G49:** *Décalage dynamique de longueur d'outils*
- G-Codes supportés dans **v0.8** (and **v0.9**)
 - **G0, G1:** *Mouvements Linéaires*
 - **G2, G3:** *Mouvements en Arcs et Hélicoïdaux*
 - **G4:** *Creuser*

- **G10 L2, G10 L20:** Réglage des décalages du travail
- **G17, G18, G19:** Sélection de plan
- **G20, G21:** Unités
- **G28, G30:** Va en position pré-définie
- **G28.1, G30.1:** Réglage de position pré-définie
- **G53:** Mouvement en coordonnées absolues
- **G54, G55, G56, G57, G58, G59:** Système des coordonnées du travail
- **G80:** Abandonne le mode de Mouvement
- **G90, G91:** Modes de distance
- **G92:** Décalage des coordonnées
- **G92.1:** Efface le décalage du système de coordonnées
- **G93, G94:** Modes de taux de débit
- **M0, M2, M30:** Pause et termine le programme
- **M3, M4, M5:** Contrôle de la broche
- **M8, M9:** Contrôle de refroidissement

La plupart des options de configuration peuvent être réglées en fonctionnement et sauvées en mémoire (eeprom) entre les sessions et même conservées entre différentes versions de GRBL lorsque vous mettez à jour le firmware.

Gestion de l'accélération

Dans les débuts, les contrôleurs [Arduino](#) n'avaient pas de planification d'accélération, et ne pouvaient pas tourner à pleine vitesse sans difficultés. La gestion constante de l'accélération par GRBL avec son planificateur d'avance a résolu cette issue et a été répliquée partout dans le monde du micro-contrôleur CNC. GRBL utilise intentionnellement un planificateur d'accélération constante simplifié, largement suffisante pour une utilisation amateur. Grâce à cela, le temps de développement est utilisé sur les algorithmes de planification et sur l'obtention de la certitude que nos mouvements sont solides et fiables.

Si vous voulez en savoir plus sur GRBL et les fonctionnalités des dernières versions, je vous invite à [vous reporter au Wiki original](#), dont cet article est une traduction partielle.



Finalement!

Voilà, Le Bear CNC est maintenant “officiellement” en ligne. Oui, ça a mis longtemps, et non, c’est loin d’être complet et terminé. Terminé, le site ne le sera sans doute jamais. Il sera en évolution constante, en fonction de vos souhaits, de vos désirs, et.. de vos achats....

Notre but est de faire de ce site une plate-forme “incontournable”, comme on dit, en ce qui concerne la CNC amateur. Il existe plein de sites et de blogs où sont décrits soit des machines et des réalisations, soit des boutiques offrant divers produits “en vrac”, avec bien peu d’explications sur le pourquoi du comment, pourquoi des choix, comment utiliser. L’Idée derrière le Bear CNC est de ne proposer que des produits pour lesquels nous avons une expérience d’utilisation, que nous décrirons au travers d’articles techniques détaillés, et, selon les cas, appuyés par des vidéos explicatives. Certains de ces produits seront des réalisations “Made in Le Bear”, parce que nous aurons détecté un besoin spécifique.

Dans l’ensemble, la partie électronique et logicielle s’appuiera sur [GRBL](#), le firmware écrit spécifiquement pour l’Arduino Uno. Il existe de nombreux logiciels pour envoyer le [GCode](#) et contrôler le déroulement des opérations avec [GRBL](#). Nous aurons de nombreux articles à ce sujet, et vous proposerons un logiciel pour PC EN FRANCAIS (nous sommes sans doute les seuls à le faire) dans ce but. Cela ne veut pas dire que nous laisserons complètement tomber les utilisateurs de Mach3 où de PlanetCNC. Nous n’avons cependant aucune intention de “pousser” à l’utilisation de ces systèmes, à la fois pour des raisons de coût, et de flexibilité. En effet, même la plus moderne version de Mach3, Mach4, ne propose pas de version USB par défaut (il faut un module complémentaire), et encore moins de liaison sans fil.

Nous verrons au fur et à mesure les solutions possibles, ainsi que la chaîne logicielle complète, de l’idée de départ à la réalisation de la pièce par fraisage CNC.

Nous somme bien évidemment à l’écoute de vos remarques et commentaires. N’hésitez surtout pas à partager vos attentes et à faire part de vos critiques, constructives, bien sûr!