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

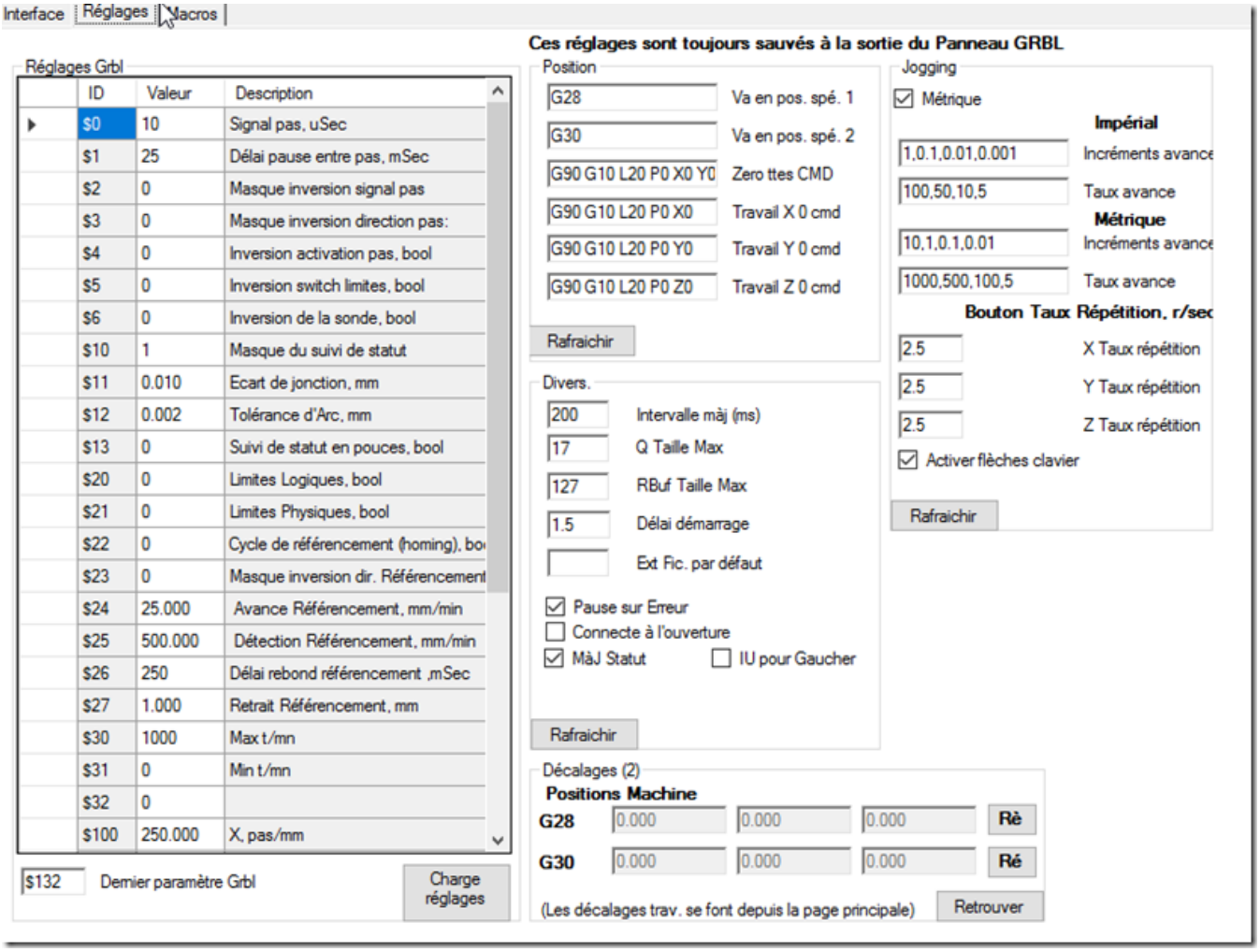
Il n'y a pas encore de procédure d'installation, ce qui va venir. Pour l'instant il faut donc télécharger

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



Les sous panneau des réglages comporte 2 cotés distincts. Sur la gauche, vous retrouvez les paramètres internes de Grbl, que vous pouvez 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 envoi 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) x 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….

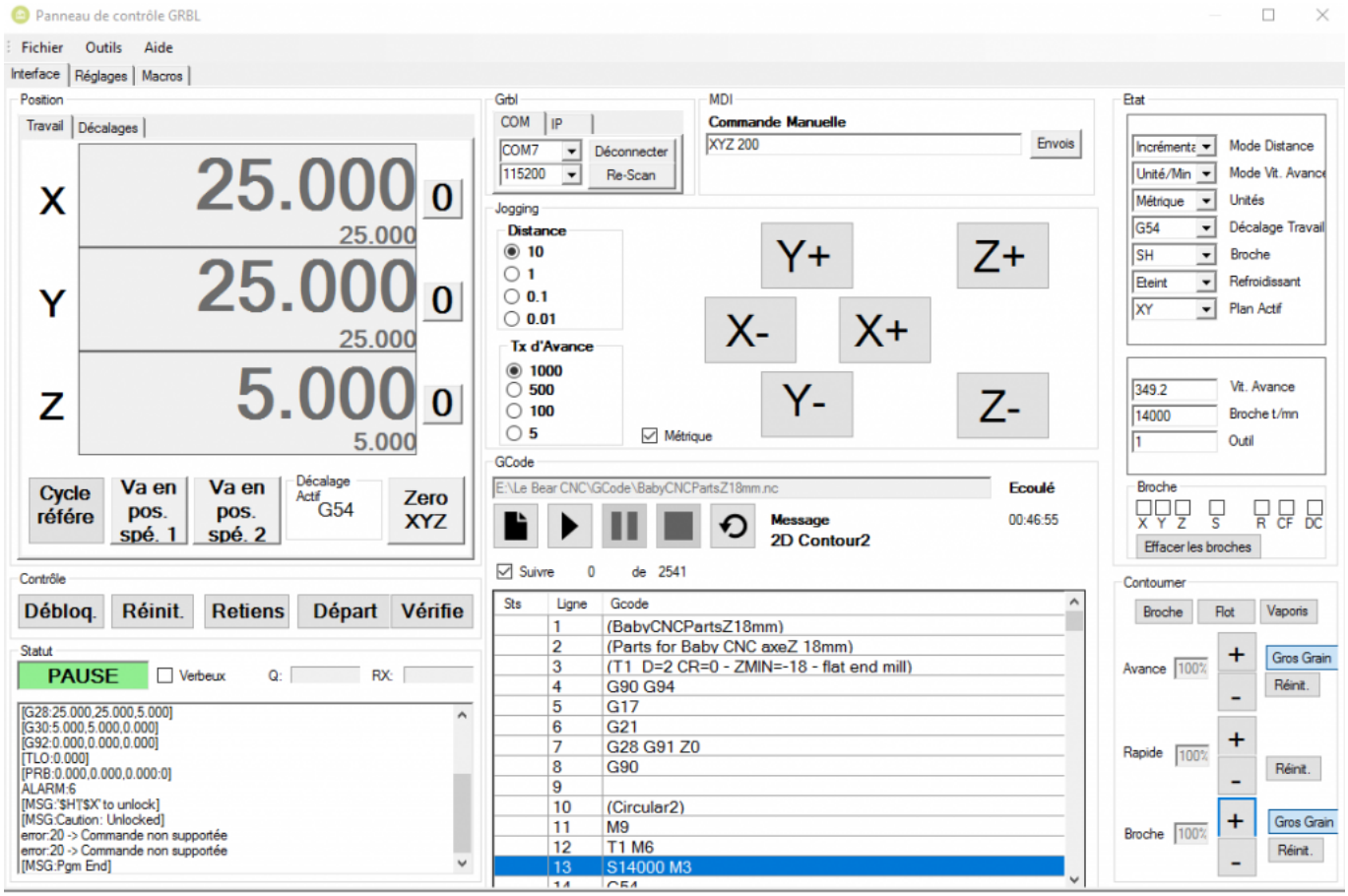
Sous Panneau Décalage. Nous venons de voir ci-dessus que les commandes GCode G28 et G30 permettent de dire à la machine d’aller se positionner sur une position préalablement définie, qui pourrait correspondre à celui d’une sonde d’outil, par exemple, où tout simplement au coin supérieur gauche de repos 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 à été envoyée à Grbl, à "OK" une fois exécutée, où 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 complétement 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…

[product id=5474]

Configurer et Paramètrer GRBL

C'est une question qui revient souvent, après avoir installé GRBL. Comment dois-je faire pour configurer et paramètrer 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 "//"

```
// If your machine has two limits switches wired in parallel to one axis, you
will need to enable
// this feature. Since the two switches are sharing a single pin, there is no
way for Grbl to tell
// which one is enabled. This option only effects homing, where if a limit is
engaged, Grbl will
// alarm out and force the user to manually disengage the limit switch.
Otherwise, if you have one
// limit switch for each axis, don't enable this option. By keeping it
disabled, you can perform a
// homing cycle while on the limit switch and not have to move the machine
off of it.
// #define LIMITS_TWO_SWITCHES_ON_AXES
```

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:

```
// Inverts pin logic of the control command pins. This essentially means when
this option is enabled
// you can use normally-closed switches, rather than the default normally-
open switches.
// NOTE: If you require individual control pins inverted, keep this macro
disabled and simply alter
// the CONTROL_INVERT_MASK definition in cpu_map.h files.
// #define INVERT_ALL_CONTROL_PINS // Default disabled. Uncomment to enable.
```

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

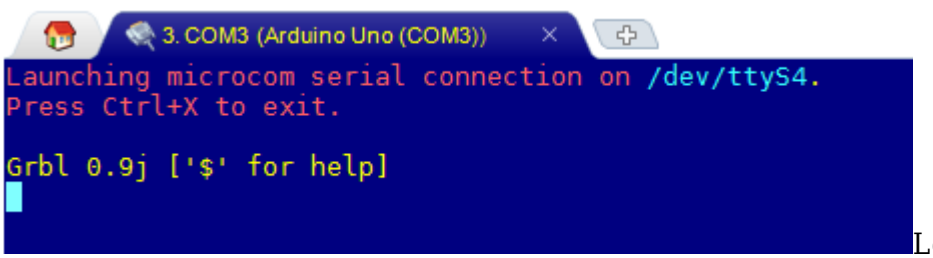
vitesses dont votre broche est capable:

```
// Enables variable spindle output voltage for different RPM values. On the
Arduino Uno, the spindle
// enable pin will output 5V for maximum RPM with 256 intermediate levels and
0V when disabled.
// NOTE: IMPORTANT for Arduino Unos! When enabled, the Z-limit pin D11 and
spindle enable pin D12 switch!
// The hardware PWM output on pin D11 is required for variable spindle output
voltages.
#define VARIABLE_SPINDLE // Default enabled. Comment to disable.

// Used by the variable spindle output only. These parameters set the maximum
and minimum spindle speed
// "S" g-code values to correspond to the maximum and minimum pin voltages.
There are 256 discrete and
// equally divided voltage bins between the maximum and minimum spindle
speeds. So for a 5V pin, 1000
// max rpm, and 250 min rpm, the spindle output voltage would be set for the
following "S" commands:
// "S1000" @ 5V, "S250" @ 0.02V, and "S625" @ 2.5V (mid-range). The pin
outputs 0V when disabled.
// BG Note: As the spindle max speed is said to be 12K, that's what I set
here. Should never go that fast in the GCode, but I will test.

#define SPINDLE_MAX_RPM 18000.0 // Max spindle RPM. This value is equal to
100% duty cycle on the PWM.
#define SPINDLE_MIN_RPM 0.0 // Min spindle RPM. This value is equal to
(1/256) duty cycle on the PWM
```

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.



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 tache, 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.

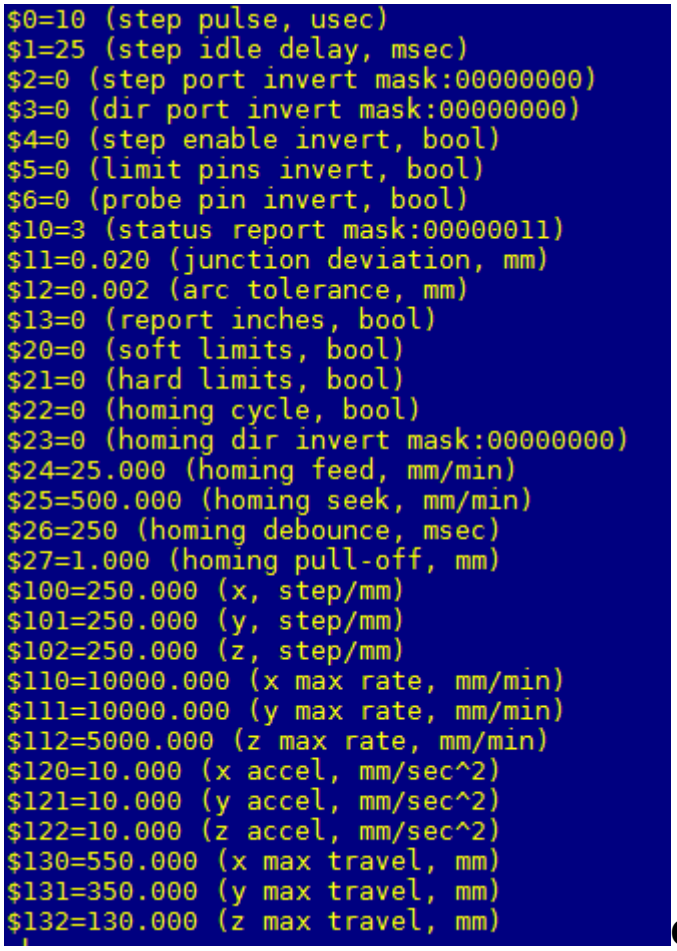
Paramètres 24, 25, 26 et 27. Homing. Comme dans E.T., « maison »!

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 à 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.



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 prochaine 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.

[CP_CALCULATED_FIELDS id= »6″]
Bien que tout ce qui précède soit relatif à GRBL, un certain nombre de principes et de calculs sont valables pour TOUTES les machine 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 écritoires 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 apparu 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 à 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 tache a 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 cordonnées 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 sauvée 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. Grace 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 complétement 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!