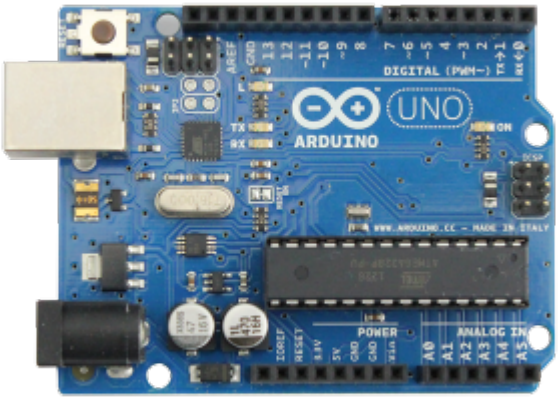
Visual Micro pour coder Arduino avec Visual Studio

Je ne sais pas pour vous, mais ces cartes microcontrôleur qui nous envahissent sous différentes formes sont extrêmement séduisantes pour ceux qui aiment “bidouiller”, d’autant qu’elles ouvrent la porte à tout un monde de capteurs divers et variés, circuits, montages, etc. Visual Micro permet de coder ces cartes depuis la Rolls des environnements de développement, Visual Studio.



Pour quelqu’un qui vient, comme moi, d’une informatique de gestion, principalement, pouvoir faire clignoter une LED avec un petit bout de code à quelque chose de magique. Il était donc logique de s’intéresser à la plus répandue de ces cartes, très bon marché, qui est maintenant toute une famille, l’Arduino. Plus spécifiquement, dans mon cas, l’Arduino Uno, puisque mes besoins ne nécessitent pas plus, mais ce que je vais décrire dans cet article est valable pour toute la famille, et bon nombre de clones.

Au début, je n’ai pas écrit grand chose, comme code, l’environnement de programmation, une application Java disponible sur le site officiel (https://www.arduino.cc/), me convenait très bien dans la mesure où il me permettait facilement de flasher la carte avec les exemples trouvés ici et là.

Au bout de quelque temps, lorsque des idées de projets personnels ont commencées à se bousculer, et donc qu’il me fallait actuellement coder, je n’ai pu m’empêcher de regretter les diverses fonctionnalités présentes dans l’éditeur de Visual Studio. (J’en profite pour indiquer à ceux qui ne le sauraient pas encore, qu’il existe une version gratuite, aussi puissante que la version pro, appelée Visual Studio Community. Disponible ici: https://www.visualstudio.com/fr-fr/products/visual-studio-community-vs.aspx). La version Community, par différence avec les “anciennes” versions Express, accepte les plugins, et ne vous limite pas à un seul type de développement, comme c’était le cas.

Je me suis donc mis à rêver une extension permettant d’écrire du code pour Arduino, directement depuis Visual Studio. Comme bien souvent, ce rêve correspondait à un réel besoin, je n’étais donc sûrement pas le seul à l’avoir fait. Après quelques recherches (Google is my friend!), j’ai découvert un produit appelé “Visual Micro” http://www.visualmicro.com/, le nom d’un plugin développé exactement dans ce but.

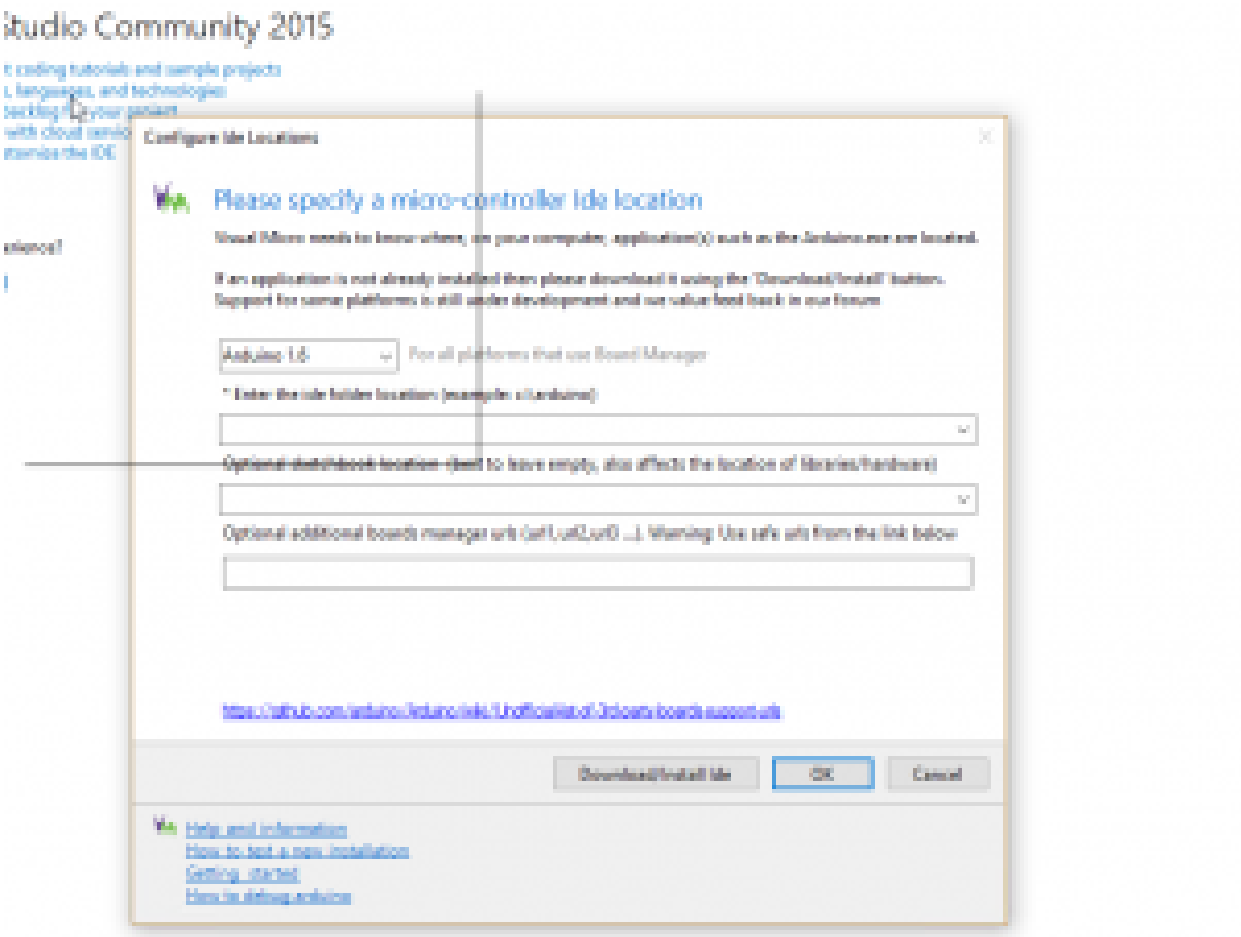
Plus qu'un éditeur dédié à l'Arduino, utilisant les capacité de l'éditeur de Visual Studio, il s'agit d'un véritable environnement pour Arduino et compatibles, intégré à Visual Studio. Ce plugin est gratuit, mais si vous vous en servez régulièrement, vous voudrez sans doute acquérir la version Pro (ce que j'ai fait), elle comporte quelques fonctionnalités supplémentaires, notamment coté debugger.

Voyons donc comment installer, régler, et utiliser cet outil dont je ne pourrais plus me passer, maintenant que je m'y suis habitué.

Pré-requis et Installation

Tout d'abord, il faut que l'environnement Arduino déjà cité soit installé sur votre machine, puisqu'avec lui sont installés un certain nombre d'utilitaires et de librairies qui vont être utilisés par Visual Micro. Les dernières versions de Visual Micro vous permettent de télécharger et d'installer l'IDE Arduino directement depuis l'écran de setup, au cas où vous n'auriez pas encore installé Arduino sur votre machine. Nous allons voir ci-dessous ces différentes étapes, en partant de Visual Studio Community avec Visual Micro gratuit, sur une machine Windows 10 en 64 bits (A ma connaissance, aucune différence avec Windows 7 pour cette installation).

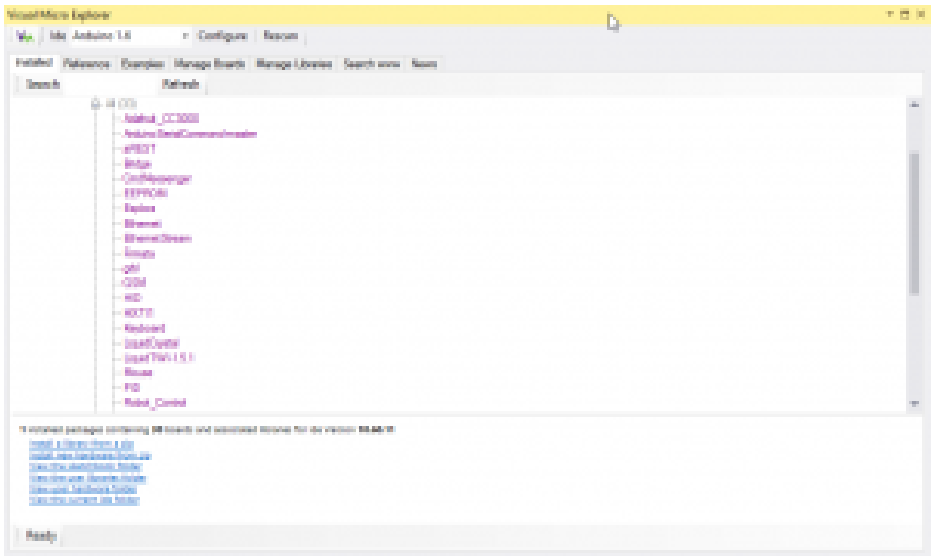
La première fois que vous lancez Visual Studio après avoir téléchargé et installé Visual Micro, l'écran de configuration va apparaître:



Il vous faut entrer les chemins correspondant à votre installation de l'IDE Arduino. Dans mon cas, l'Ide n'étant pas installé sur cette machine, je cliques sur le bouton "Download/Install Ide", ce qui

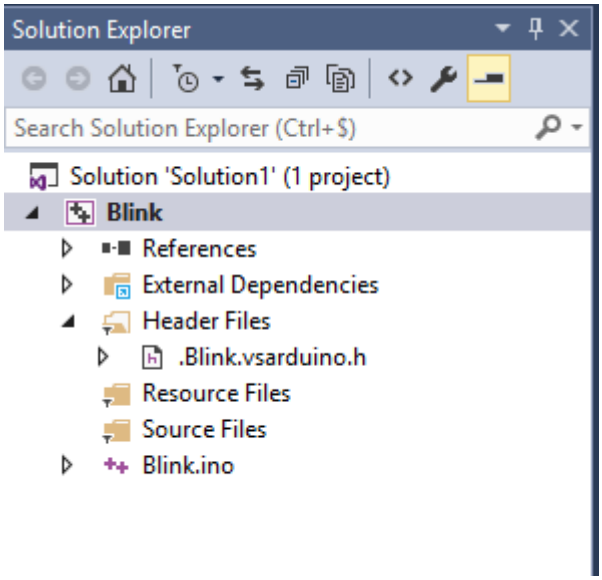
m’amène à la page de téléchargement du logiciel Arduino, où je choisis l’installateur pour Windows:

Une fois l’environnement Arduino installé, il vous faut indiquer les chemins. Par défaut, C:\Program Files (x86)\Arduino. Vous pouvez tester si votre environnement est correctement réglé dans Visual Micro en ouvrant Visual Micro Explorer (Il y a une nouvelle option dans le menu supérieur de Visual Studio, dénommée “vMicro”). Vous devriez voir la liste de toutes les cartes gérées, ainsi que l’ensemble des librairies installées:



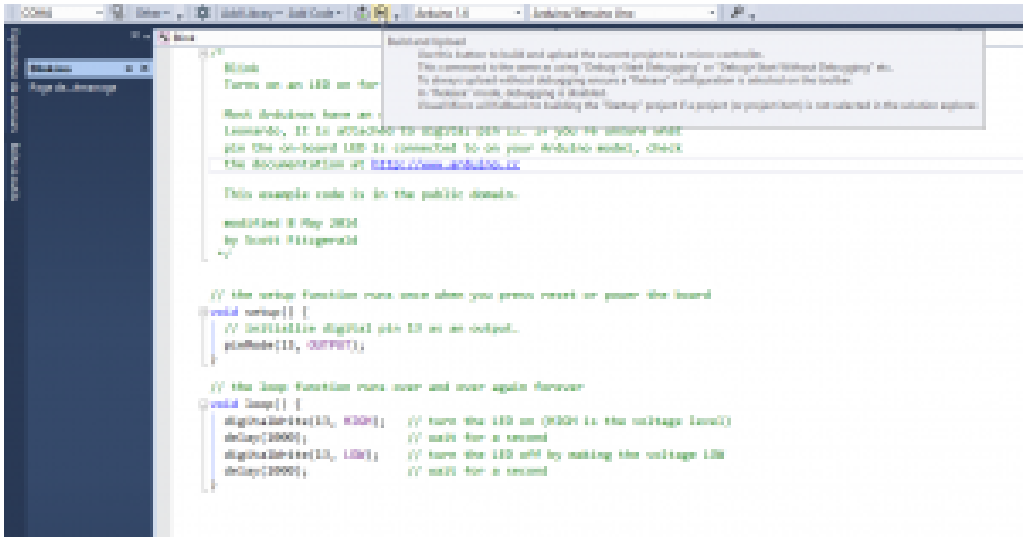
Selon les cas, vous pouvez avoir à localiser l’emplacement où Arduino installe les librairies et “sketches”. Sur ma machine, c’est dans “C:\Users\USERNAME\Documents\Arduino”. Si c’est le cas, il vous faut indiquer ce chemin sur l’onglet “Configure”.

Maintenant, vous pouvez tester pour vérifier si votre installation est correcte. Branchez votre Arduino avec un câble USB, Windows doit le reconnaître (vous avez installé le driver à la fin de l’installation de l’IDE). Chargez un des exemples (Onglet Exemples). Le plus classique est “Blink.ino”. Commencez par créer un nouveau projet: Fichier/Nouveau Projet/ Arduino Project:

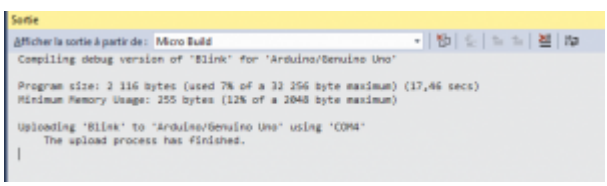


Le fichier Blink.ino est créé pour vous, vide. Il faut le supprimer, et faire ajout, pour aller chercher le fichier (déjà ouvert si vous avez cliqué sur son nom depuis Visual Micro Explorer) Si vous passez la souris sur le tab du nom du fichier, dans Visual Studio, le chemin est indiqué. Vous devez copier le fichier pour le mettre dans le répertoire du projet, à moins que vous n'exécutiez Visual Studio en mode Administrateur, ce qui n'est pas vraiment conseillé. Une autre solution est tout simplement de copier le contenu du fichier Blink exemple dans le fichier créé par Visual Studio.

Après avoir vérifié que le Port Com sélectionné est bien celui correspondant à votre Arduino, vous pouvez cliquez sur la petite flèche pour compiler et envoyer le code dans l'Arduino:



La fenêtre de sortie doit vous indiquer après quelque temps la réussite de l'opération, et vous devriez voir la led liée au Port 13 de l'Arduino clignoter toutes les secondes. Ce test est le plus classique parce qu'il ne nécessite aucun matériel en dehors de la carte elle-même.



Quels avantages? (Et éventuels inconvénients?)

Si vous êtes déjà familier avec Visual Studio, vous le savez déjà! Sinon, vous allez vite voir que c'est un excellent, si ce n'est le tout meilleur, environnement de développement actuellement disponible, et ce, complétement gratuitement! Voici une petite liste des fonctionnalités de Visual Micro avec Visual Studio:

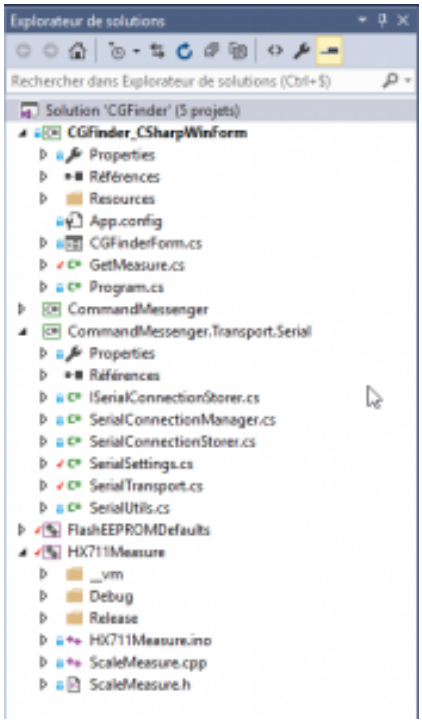
- **Complétement compatible avec l'IDE Arduino,** chaque projet de l'IDE Arduino est un projet Visual MIcro, et vice versa. Utilise les outils Arduino en arrière plan.
- **Interface utilisateur hautement configurable** (Disposition des fenêtres, couleurs et fontes, barres d'outils..)
- **Supporte IntelliSense:** Auto-suggestions, auto-complétions et nom des classes lorsque vous tapez
- **Vérification syntaxique "juste à temps":** Détecte, surligne et explique les erreurs de syntaxe immédiatement sans compilation
- **Support pour de multiples fichiers .ino, .cpp par projet**, sous-dossiers de projet pour de plus grand projets.
- **Support pour un grand nombre de cartes**: Arduino et compatibles, Teensy, Sparkfun, Adafruit, Intel,…
- **Débuggeur unique** (points d'arrêt, traçage, surveillance et changement de variables) *[Uniquement dans la version pro, gratuit pendant une période d'essai.]* Aucun matériel spécifique requis, fonctionne avec USB comme avec WiFi.
- **Puissant gestionnaire de librairies**
  - Télécharge des libraires tierce et les intègres à Visual Micro
  - Recherche de librairies en-ligne
- **Gestionnaire de cartes:** Téléchargez le support de nouvelles cartes directement depuis l'IDE, pas d'installation requise
- Travailles avec plusieurs cartes simultanément
- Création de nouvelles librairies Arduino en un clic
- **Support pour GIT et TFS**
- **Partage de code avec d'autres projets**
- Base installée de **plus de 10 000 utilisateurs**
- Support de **programmateurs et boot loaders**
- **Services Gratuits:**
  - **Support** au travers d'un forum modéré
  - **Mises à jour continues et gratuites** pour une compatibilité certaine avec les dernières version de l'IDE Arduino
  - Documentation complète
- Supporte Visual Studio 2012, 2013 et 2015

A cette liste déjà longue, je voudrais ajouter 2 éléments découverts à l'usage,

1. **Utilisation du système de "Solutions" de Visual Studio**

Ce qui peut sembler un inconvénient, si on est habitué à l'IDE Arduino, apportant une certaine lourdeur au regard de la simplicité de "télé verser" un simple fichier vers la carte micro-contrôleur, est en fait, à l'usage, un immense avantage. Il est en effet très facile de regrouper divers projets dans la même solution, que ces projets soient Arduino, où pas! Cela vous permet d'avoir, par exemple, plusieurs projets Arduino regroupés sous la même solution. Très facile alors de tester divers exemples, et de développer votre propre code à partir de ceux-ci. Très pratique aussi dans le cas où vous développez un programme Windows dialoguant par communication série avec votre

carte. Vous pouvez afficher simultanément les 2 codes l'un à coté de l'autre.



1. Sur cette copie d'écran, vous pouvez voir 5 projets dans la même solution. Deux sont des projets Arduino et trois des projets Windows, comprenant deux librairies. Visual Studio permettant de passer très facilement d'un projet à l'autre (Définir comme projet de démarrage), vous pouvez compiler à la fois pour Windows et pour Arduino depuis la même solution.
2. **2. Possibilité d'utiliser le contrôle de version, lié à Visual Studio**

Si vous n'utilisez pas de système de contrôle de version, vous devriez! Vous pouvez créer un compte Visual Studio Team Services gratuitement et gérer autant de projet que vous voulez. Ce projet peut être partagé avec 4 autres personnes. Le service devient payant au-delà. Ce serait vraiment dommage de se priver de se service, d'autant que vous pouvez choisir d'utiliser Gît.

En guise de conclusion sur Visual Micro

Je ne me vois pas revenir à l'utilisation exclusive de l'IDE Arduino pour mes développements. Le simple fait de bénéficier d'IntelliSense suffirait sans doute à faire pencher la balance, pour moi. Mais toutes les fonctions apportées sont utiles, le contrôle de version étant sans doute dans le tiercé de tête. Si vous codez souvent pour l'Arduino, si vous avez un projet qui vous tient à cœur, essayez Visual Micro, et dites-moi ce que vous en pensez.

Petite précision pour la transparence: je n'ai strictement aucun intérêt financier, ni avec Visual Micro Limited, la société éditrice de ce plugin, ni avec Microsoft.

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!