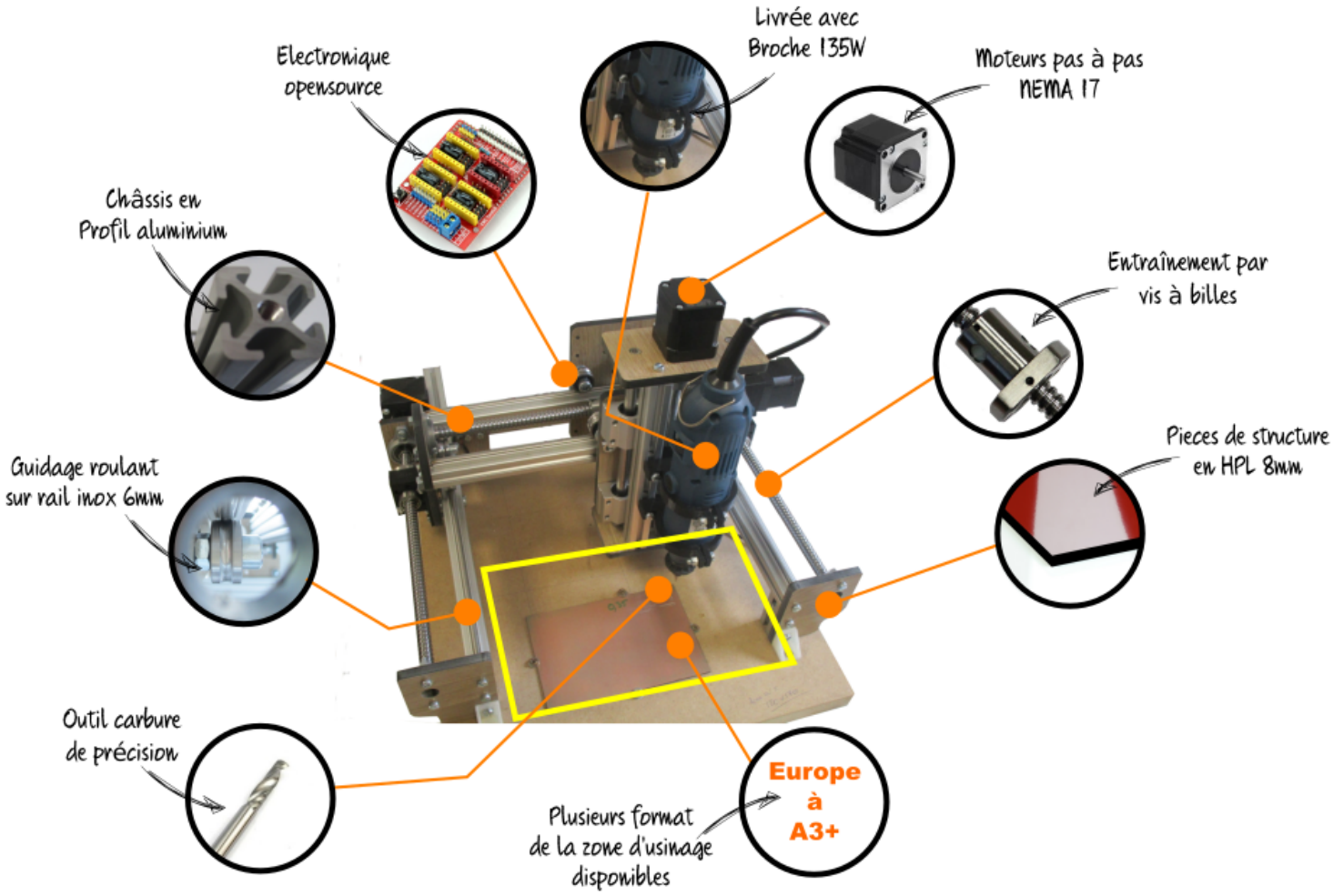
Open Maker Machine PLUS : installation du firmware GRBL

Par X. HINAULT – www.mon-club-elec.fr – Octobre 2016 - Tous droits réservés - Licence Creative Commons BY NC SA



NOTE : TOUTE LA PROCEDURE QUI SUIT EST A FAIRE « HORS TENSION » :
la carte électronique sera ici alimentée uniquement par le câble USB.

La mise sous tension et les tests de mise en route ne seront fait qu'une fois le firmware programmé (voir tuto « prise en main ») .

Pour comprendre : la chaîne logicielle de contrôle de l'Open Maker Machine PLUS

Principe général

Le principe général de contrôle de l'Open Maker Machine, commun aux imprimantes 3D opensource notamment, est le suivant :

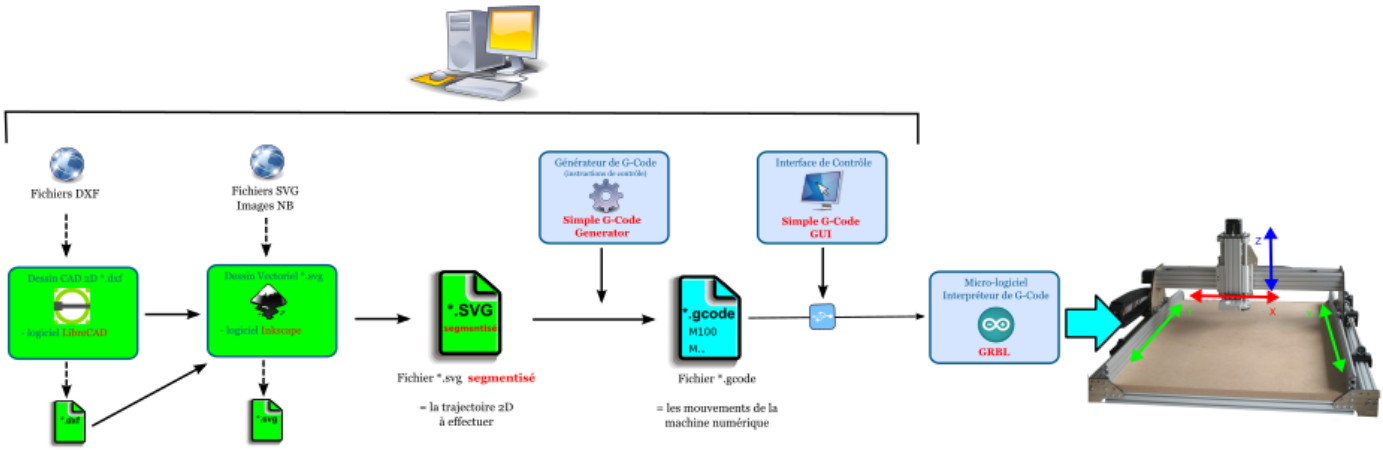
- à partir d'une pièce conçue dans un logiciel, en 2D (format SVG) ou en 3D (format STL) selon les cas...
- ... l'ensemble des mouvements nécessaires de la mécanique 3 axes pour créer la pièce vont être calculés et traduits en G-Code (retenez ce « gros-mot »), un langage simple de programmation d'automate
- les instructions ainsi obtenues vont ensuite être envoyés via le port série vers la machine qui va

les exécuter fidèlement.

Structure de la chaîne logicielle utilisée

Logiquement, l'Open Maker Machine PLUS , pour être mise en œuvre, nécessite une chaîne logicielle qui est open-source (libre en fait) incluant :

- un **décodeur de G-Code** : micro-logiciel (ou firmware) programmé dans la carte Arduino une fois pour toutes. Ce « décodeur de G-Code », comme son nom l'indique va décoder le G-Code reçu par la machine sur le port série. Autrement dit, il va transformer une chaîne reçue sur le port série (par exemple « G01 X10.0 Y10.0 ») en un mouvement de la machine (positionnement de l'outil en coordonnées X=10mm et Y=10mm).
- une **interface graphique de contrôle** (interface « homme-machine ») qui va permettre :
  - le contrôle manuel de la machine via le port série (USB),
  - l'ouverture d'un fichier de G-Code et l'envoi de son contenu vers l'Open Maker machines via le port série (USB)
- un **générateur de G-code** : logiciel graphique qui à partir soit d'un fichier STL (dessin 3D décrivant la surface d'un objet), soit d'un fichier SVG (dessin 2D vectoriel), va permettre :
  - de générer le G-Code (=les mouvements machine à exécuter)
  - en prenant en compte les paramètres voulus (diamètre d'outil, vitesse de déplacement, etc.)
- un logiciel de **conception graphique** :
  - soit **2D** : qui permettra de facilement générer un parcours outil 1 passe en créant un simple dessin vectoriel (fichier de type SVG)
  - soit **3D** (fichiers de type STL) : qui permettra de créer :
    - soit des objets 2D en épaisseur pour générer des parcours outils multi-passes,
    - soit même de véritables objets 3D pour générer des parcours outils complexes de sculpture 3D,



Synthèse des solutions concrètes retenues en pratique

Comme pour tout autre thématique, plusieurs solutions logicielles sont disponibles et je vous présente ici une synthèse résumant les choix conseillés ainsi que les alternatives possibles :

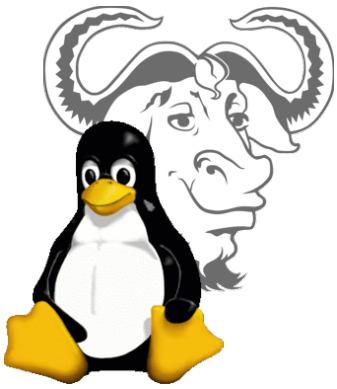
Fonction	Solution conseillée (décrite ici)	Solutions alternatives possibles
Décodeur de G-Code	GRBL	
Interface graphique de contrôle	Simple G-Code GUI pour GRBL	Interface GRBL

Générateur de G-Code	Simple G-Code Generator ou Pycam	Plugin Inkscape ± Slic3R adapté ?
Conception graphique 2D	Inkscape	LibreCAD Openscad (mode projection)
Conception graphique 3D	Freecad Openscad	Blender

Pré-requis

On présuppose ici :

- que vous disposez d'un ordinateur avec **un système opérationnel installé**
  - idéalement de type **Gnu/Linux** (=celui pour lequel l'ensemble des logiciels utilisés est fonctionnel), possiblement sous Windows ou Mac OsX (80 % des logiciels utilisés le seront, reste 20 % qu'il vous faudra adapter à votre système).
  - Je conseille idéalement un système Gnu/linux de type Debian ou Ubuntu :
    - une bonne solution « clés en main » est notre distribution « maison » (une Debian Testing) sous XFCE (voir :)
    - ou bien Xubuntu : http://xubuntu.org/getxubuntu/
- avec **le logiciel Arduino installé**



Installation du logiciel Arduino sous Gnu/Linux

Pour installer Arduino sous Gnu/Linux (Debian, Xubuntu) : dans un terminal :

```
sudo apt-get install arduino
```

une fois fait, brancher carte Arduino et ouvrir l'IDE Arduino : si le port série n'est reconnu, fermer l'IDE et à nouveau dans un terminal faire :

```
sudo usermod -a -G dialout  $USER
sudo chmod a+rw /dev/ttyACM0
```

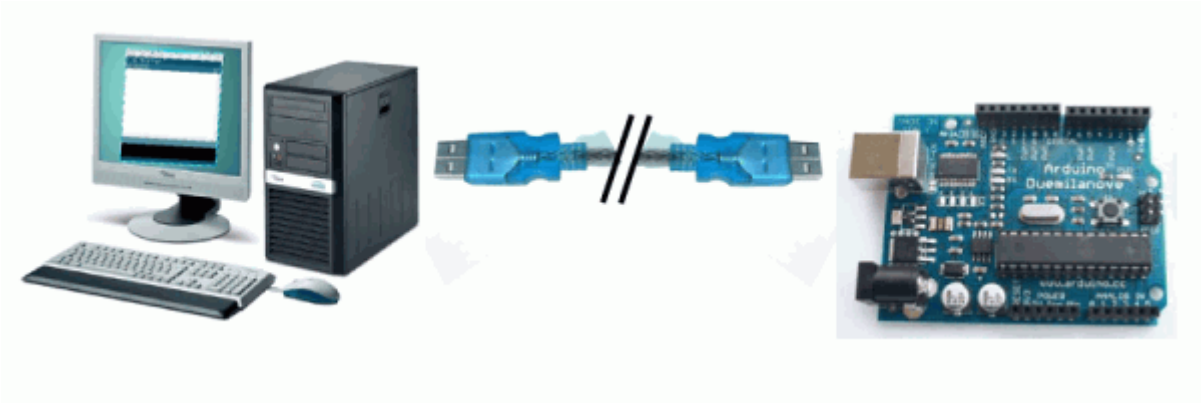
Ouvrir à nouveau l'IDE : le port série doit être détecté à présent.

Note : On pourra ensuite, si on préfère, télécharger la dernière version Arduino sur le site officiel et la dézipper pour l'exécuter de façon autonome.

Test de la carte Arduino

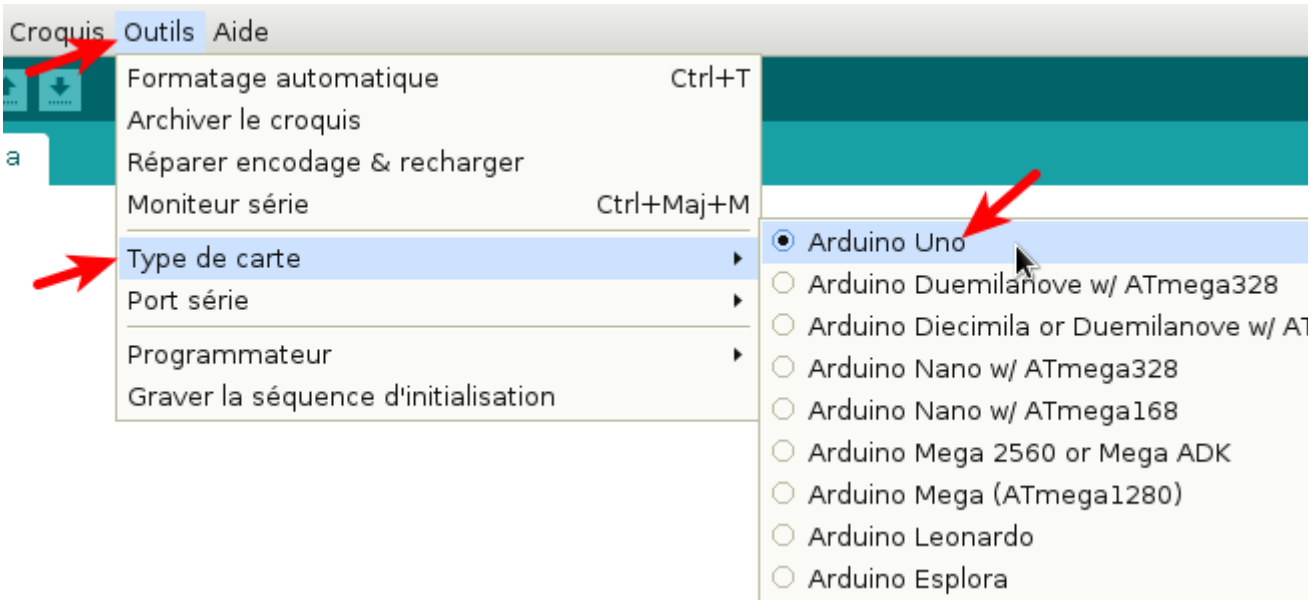
Avant de mettre l'Open Maker Machine PLUS sous tension de puissance (autrement dit, avant de mettre sous tension l'alimentation 220V – 12V/300W), on va programmer la carte Arduino alimentée simplement par le port USB.

Pour cela connecter le câble USB entre l'ordinateur et la carte Arduino comme on le ferait si on utilisait une carte Arduino seule :

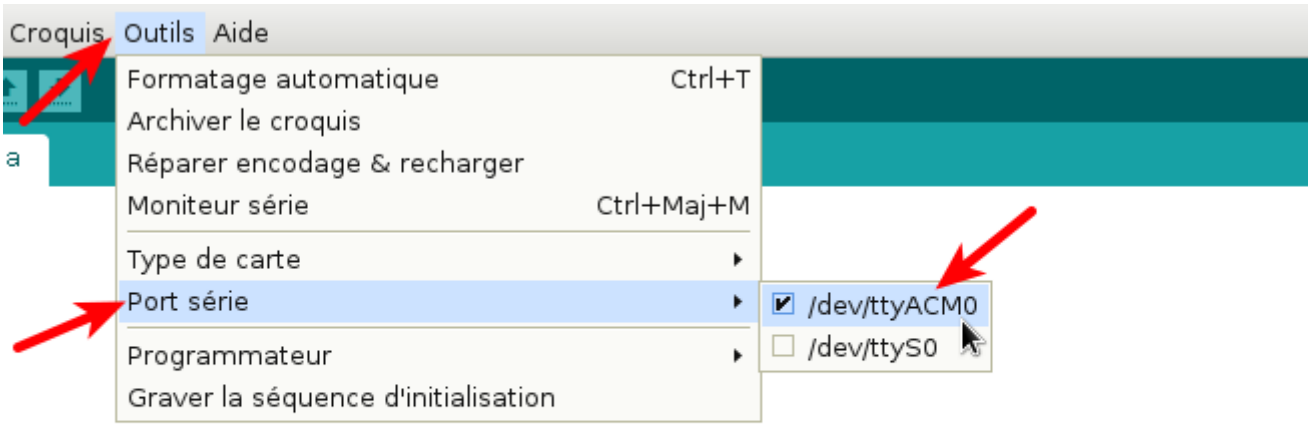


Ouvrir le logiciel Arduino :

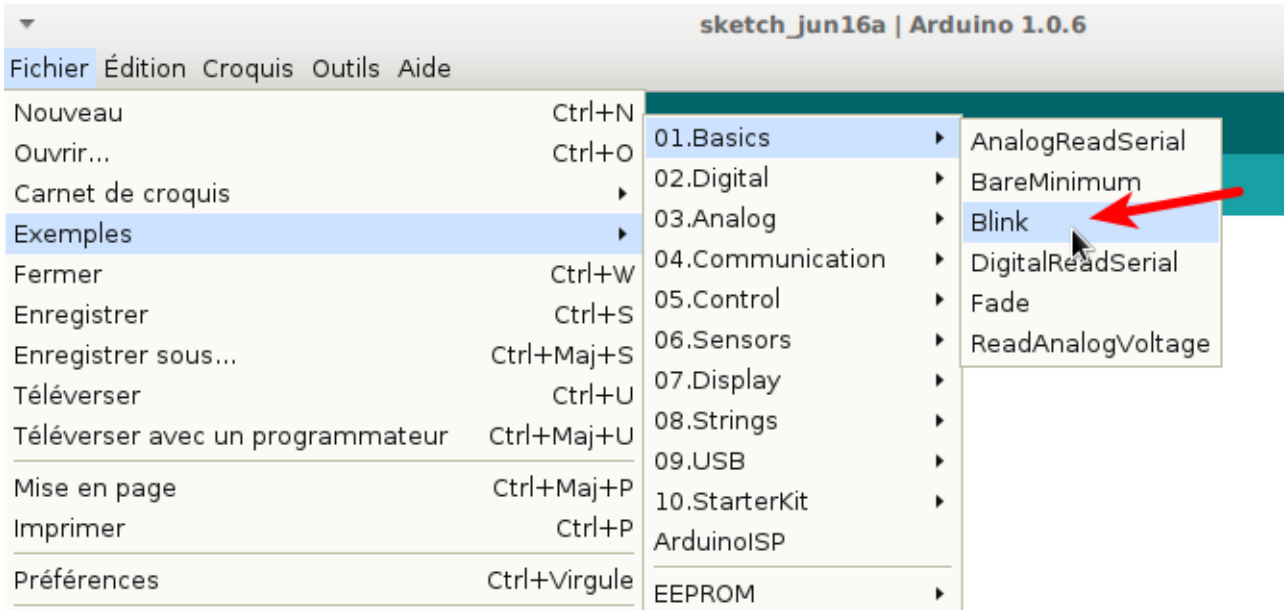
Sélectionner la bonne carte Arduino dans le menu **Outils > Type de carte > sélectionner la carte Arduino Uno** :



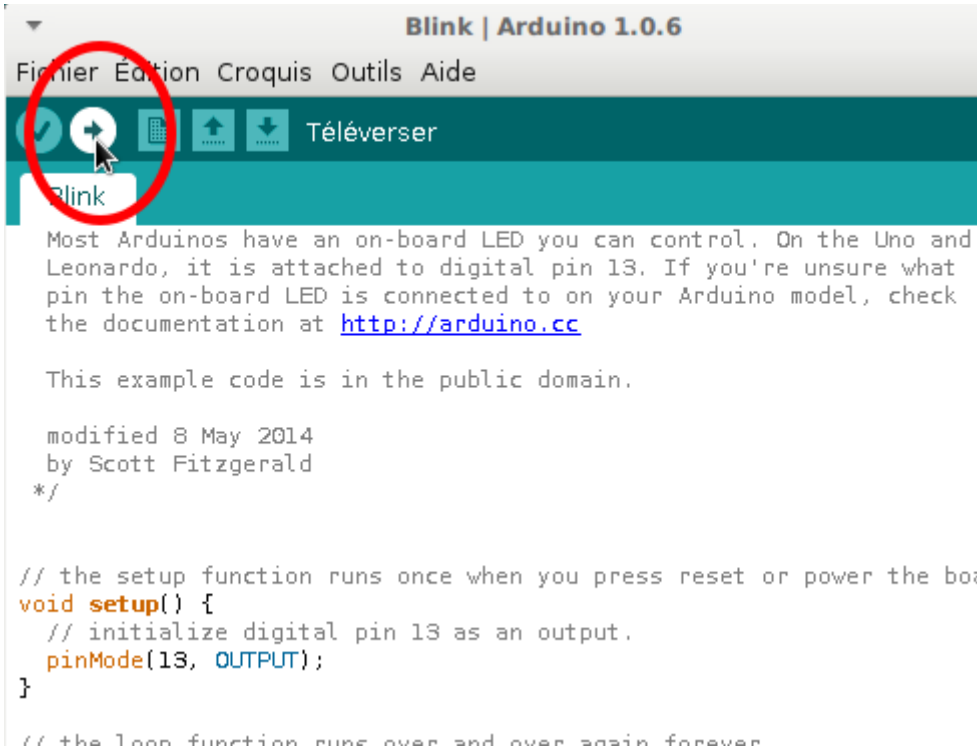
Vérifier également que la carte Arduino est correctement détectée et que le port série est bien sélectionné via le menu **Outils > Port série > sélectionner le port série de la forme /dev/ttyACM0** sous Gnu/Linux, sous la forme COM5 sous Windows, etc.



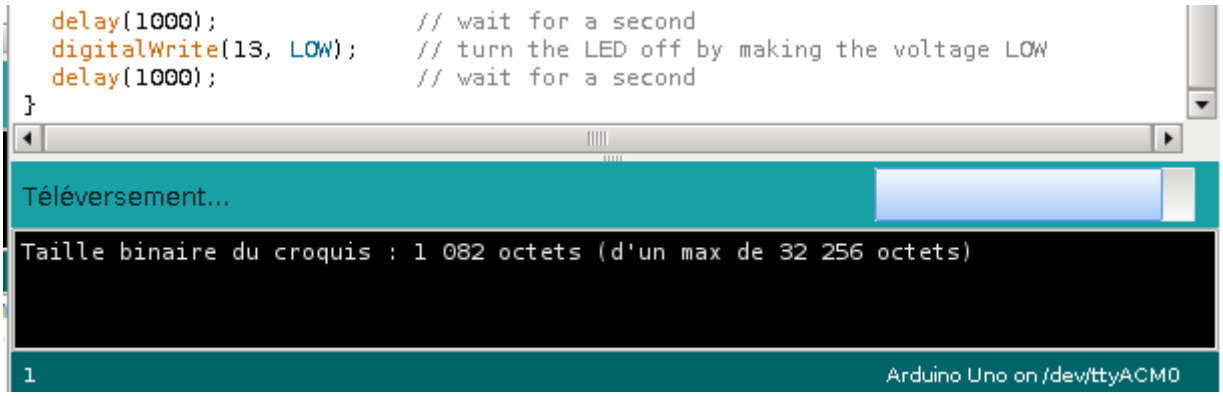
A présent, on peut tester la programmation de la carte avec un simple programme de test faisant clignoter la LED présente sur la carte, le programme d'exemple « Blink.ino » à ouvrir via le menu **Fichier > Exemples > Basics > Blink**



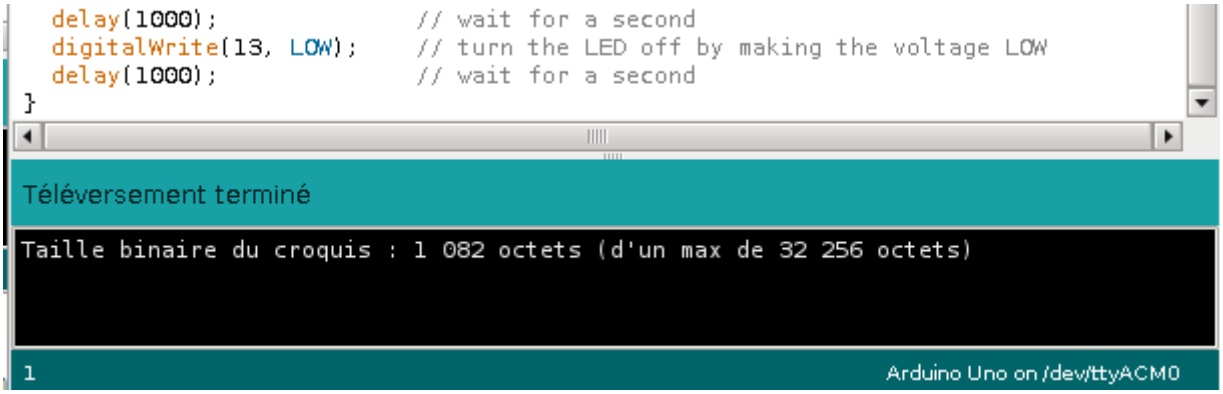
Le code s'ouvre dans la fenêtre d'édition et il suffit de cliquer sur le bouton **<Téléverser>** pour envoyer le code dans la carte Arduino :



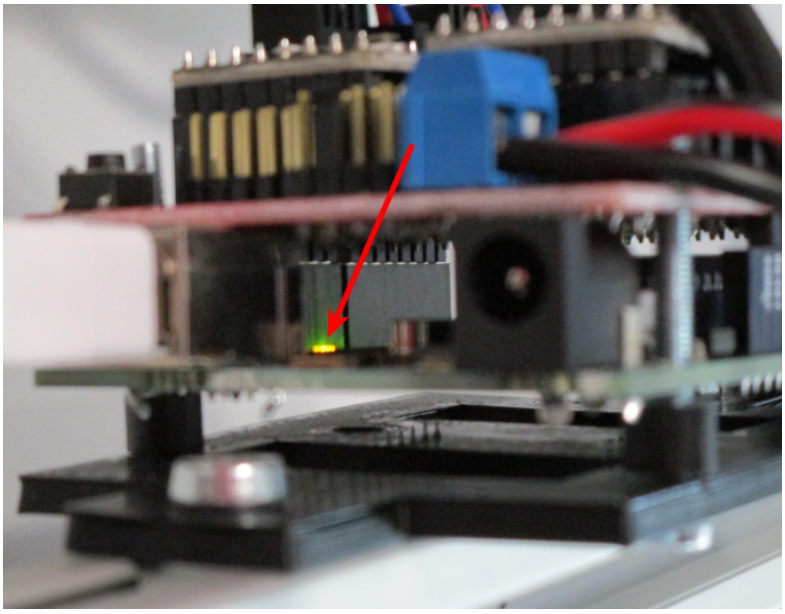
Pendant quelques secondes, on obtient une barre de progression dans le bas de la fenêtre attestant du bon transfert du programme dans la carte Arduino :



Puis suit un message attestant la fin du téléchargement :



La LED présente sur la carte Arduino UNO doit à présent clignoter :



Si tout s'est bien déroulé, vous êtes sûr que votre carte Arduino fonctionne correctement et vous pouvez passer à la suite. Si ce n'est pas le cas, reprenez la procédure en apportant les corrections nécessaires.

Programmation du décodeur de G-Code dans la carte Arduino

Introduction

L'électronique de contrôle de l'Open Maker Machine a pour rôle essentiel d'**assurer le décodage des instructions de G-Code reçues sur le port série** et de réaliser les mouvements machine correspondants. Autrement dit, l'électronique de contrôle de l'Open Maker Machine est un « décodeur de G-Code ».

La meilleure solution et de loin est d'utiliser un micro-logiciel plus abouti, tel que GRBL : https://github.com/grbl/grbl GRBL est le décodeur de Gcode qui a servi de base à Marlin par exemple.

Installation de GRBL

Sources utiles :

http://www.civade.com/post/2014/01/02/Arduino-et-GRBL-l-incontournable-solution-pour-piloter-une-petite-CNC

http://www.civade.com/post/2011/06/15/piloter-une-CNC-avec-Arduino-GRBL-moteurs-pas-a-pas (plus ancien)

Téléchargement

Récupérer le code ici : https://github.com/grbl/grbl

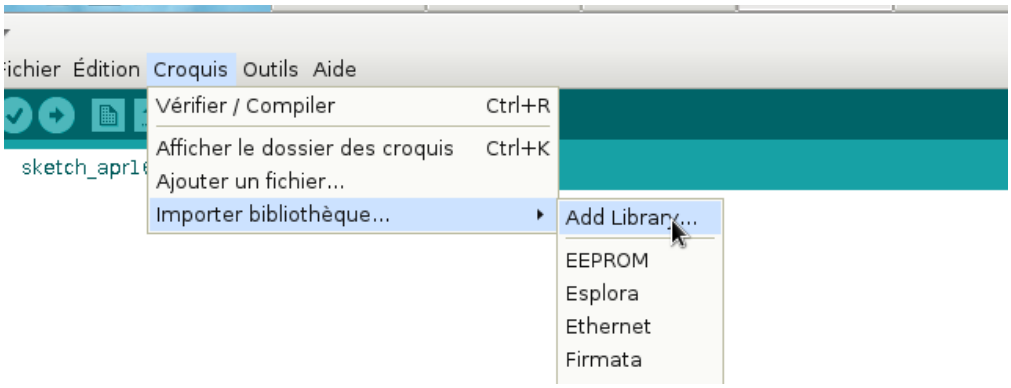
NOTE : la version de GRBL « prête à l'emploi » pour l'Open Maker Machine PLUS est fournie sur notre site du projet : http://cloud-mon-club-elec.fr:8080/files_openmakermachineplus/firmware/

Ouvrir le firmware dans le logiciel Arduino

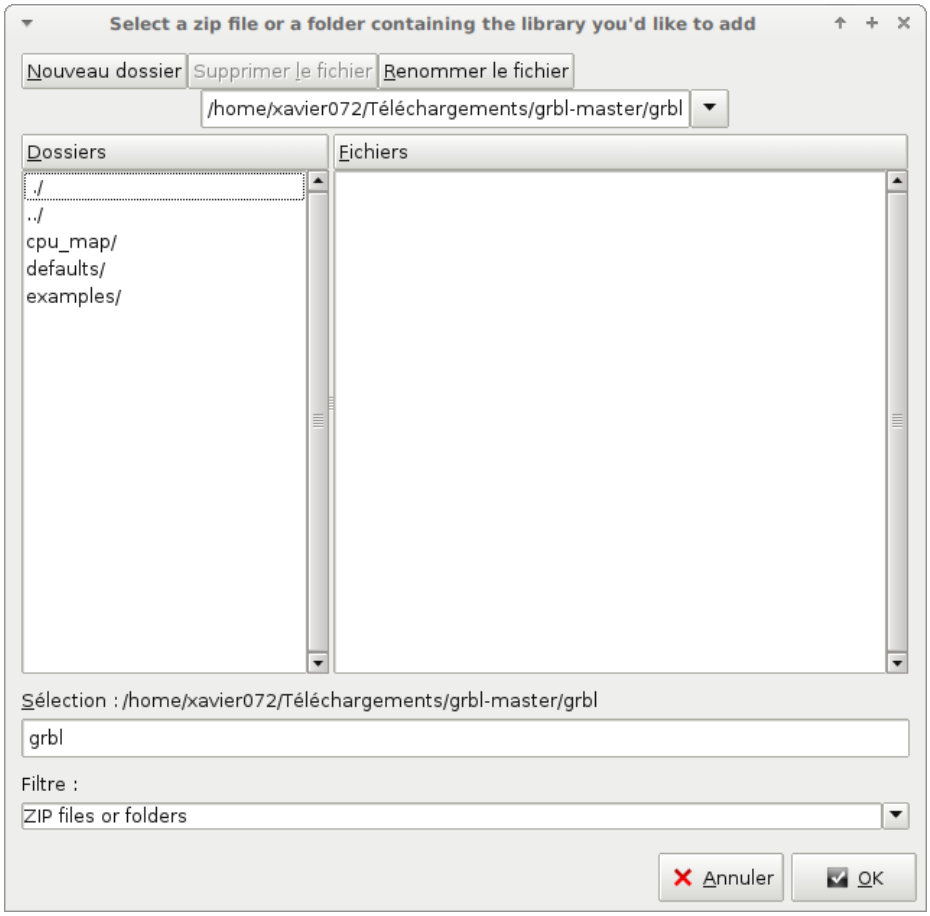
Le code GRBL n'est pas un pur code Arduino mais un code C. Ceci étant, il est possible de l'ouvrir directement dans l'IDE Arduino :

https://github.com/grbl/grbl/wiki/Compiling-Grbl

Cliquer sur le menu déroulant « Croquis » et cliquer sur « Ajouter une librairie » :



Dans la fenêtre qui s'ouvre, sélectionner le répertoire <grbl> dans le répertoire <grbl-master> obtenu précédemment. Ce répertoire contient uniquement les sources et un répertoire d'exemple :



On peut également simplement copier/coller le répertoire grbl dans le répertoire <libraries> de l'IDE Arduino.

NOTE : si vous utilisez notre version prête pour l'Open Maker Machine PLUS, vous n'avez pas besoin d'éditer les fichiers de configuration tel que décrit ci-dessous. Vous pouvez passer directement à la programmation de la carte Arduino (voir ci-dessous)

Configuration

Aller dans le répertoire Grbl placé dans le rép libraries / Arduino

Modifier le fichier defaults.h

Editer le fichier defaults.h et ajouter le bloc :

```
//--- ajout
#ifdef DEFAULTS_MAMACHINE
  // Grbl  default settings for My machine
  #include "defaults/defaults_mamachine.h"
#endif
//---

#ifdef DEFAULTS_GENERIC
```

Créer fichier defaults_mamachine.h

Aller dans le sous-rép grbl/default et le renommer en defaults_mamachine.h

Et adapter le contenu selon :

```
#ifndef defaults_h
#define defaults_h

// Grbl for open maker machine pro


  #define MOTOR_STP 200 // number of step per turn
  #define MICRO_STP 16  // Micro stepping 1/16 step
  #define X_SCREW_PITCH_MM 5 // Screw pitch in mm - 1605
  #define Y_SCREW_PITCH_MM 5 // Screw pitch in mm - 1605
  #define Z_SCREW_PITCH_MM 4 // Screw pitch in mm - 1204

  //
  #define DEFAULT_X_STEPS_PER_MM (MICRO_STP*MOTOR_STP/X_SCREW_PITCH_MM)
  #define DEFAULT_Y_STEPS_PER_MM (MICRO_STP*MOTOR_STP/Y_SCREW_PITCH_MM)
  #define DEFAULT_Z_STEPS_PER_MM (MICRO_STP*MOTOR_STP/Z_SCREW_PITCH_MM)

  // Grbl generic default settings. Should work across different machines.
  //#define DEFAULT_X_STEPS_PER_MM 250.0
  //#define DEFAULT_Y_STEPS_PER_MM 250.0
  //#define DEFAULT_Z_STEPS_PER_MM 250.0


  #define DEFAULT_X_MAX_RATE 500.0 // mm/min
  #define DEFAULT_Y_MAX_RATE 500.0 // mm/min
  #define DEFAULT_Z_MAX_RATE 500.0 // mm/min

  #define DEFAULT_X_ACCELERATION (10.0*60*60) // 10*60*60 mm/min^2 = 10 mm/sec^2
  #define DEFAULT_Y_ACCELERATION (10.0*60*60) // 10*60*60 mm/min^2 = 10 mm/sec^2
  #define DEFAULT_Z_ACCELERATION (10.0*60*60) // 10*60*60 mm/min^2 = 10 mm/sec^2

  #define DEFAULT_X_MAX_TRAVEL 450.0 // mm
  #define DEFAULT_Y_MAX_TRAVEL 450.0 // mm
  #define DEFAULT_Z_MAX_TRAVEL 80.0 // mm

// doc ici : https://github.com/grbl/grbl/wiki/Configuring-Grbl-v0.9
  #define DEFAULT_STEP_PULSE_MICROSECONDS 10
  #define DEFAULT_STEPPING_INVERT_MASK 0
  #define DEFAULT_DIRECTION_INVERT_MASK 7  // inversion +1 pour le X +2 pour le Y et +4 pour le Z - 7 pour les 3
```

```
  #define DEFAULT_STEPPER_IDLE_LOCK_TIME 255 // msec (0-254, 255 keeps steppers enabled)
  #define DEFAULT_STATUS_REPORT_MASK ((BITFLAG_RT_STATUS_MACHINE_POSITION)|
(BITFLAG_RT_STATUS_WORK_POSITION))
  #define DEFAULT_JUNCTION_DEVIATION 0.01 // mm
  #define DEFAULT_ARC_TOLERANCE 0.002 // mm
  #define DEFAULT_REPORT_INCHES 0 // false
  #define DEFAULT_INVERT_ST_ENABLE 0 // false
  #define DEFAULT_INVERT_LIMIT_PINS 0 // false
  #define DEFAULT_SOFT_LIMIT_ENABLE 0 // false
  #define DEFAULT_HARD_LIMIT_ENABLE 1  // True pour activer endstop
  #define DEFAULT_HOMING_ENABLE 0  // false
  #define DEFAULT_HOMING_DIR_MASK 0 // move positive dir
  #define DEFAULT_HOMING_FEED_RATE 25.0 // mm/min
  #define DEFAULT_HOMING_SEEK_RATE 500.0 // mm/min
  #define DEFAULT_HOMING_DEBOUNCE_DELAY 250 // msec (0-65k)
  #define DEFAULT_HOMING_PULLOFF 1.0 // mm

#endif
```

Le descriptif détaillé des différentes options est décrit ici :
https://github.com/grbl/grbl/wiki/Configuring-Grbl-v0.9

Modifier le fichier config.h

Ensuite, ouvrir le fichier config.h et modifier :

```
// Default settings. Used when resetting EEPROM. Change to desired name in defaults.h
//#define DEFAULTS_GENERIC
#define DEFAULTS_MAMACHINE
```

Dans ce fichier on peut aussi configurer la façon dont se fait le home :
Pour configurer le « homing » façon « Imprimante 3D » :

çà se passe dans le fichier config.h
commenter les autres pour le cycle et faire :

```
#define HOMING_CYCLE_0 ((1<<X_AXIS)|(1<<Y_AXIS)) // home X-Y seulement

#define HOMING_FORCE_SET_ORIGIN // Uncomment to enable. - set home en tant qu'origine
```

Dans la config dans le terminal :
$23=7 pour inversion dans toutes les directions si besoin
activer le homing avec :
$22=1
et désactiver le hardware limit

$21=0

et homing dir invert

$23=7

Dès lors, au lancement, on a :

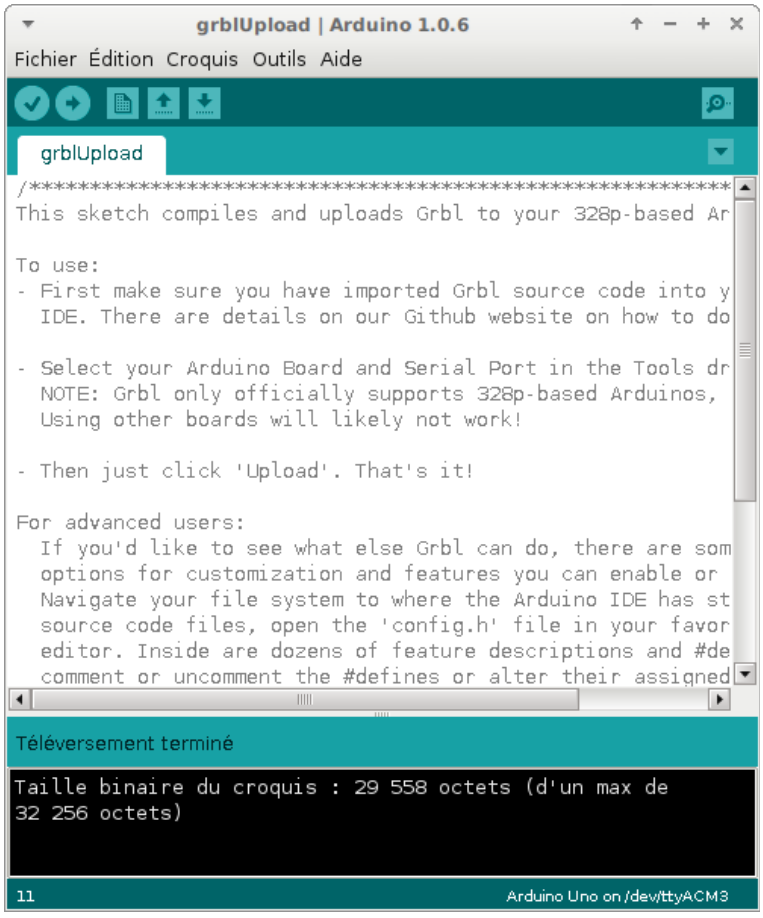
['$H'|'$X' to unlock]

Lancer le homing avec $H

Désactiver avec $X

Programmer la carte Arduino

Ensuite, aller dans exemples > grbl et choisir grbl upload



Note : si on a déjà programmé la carte Arduino, des valeurs ont été stockées en EEPROM et une nouvelle reprogrammation des valeurs n'est pas prise en compte. Pour cela, il faut reseter l'eeprom ce qui se fait avec l'ordre :

$RST=*

Puis reprogrammer.

Voir ci-dessous.

Voir : https://github.com/grbl/grbl/wiki/Frequently-Asked-Questions#my-grbl-settings-and-parameters-are-all-funky-after-flashing-grbl-how-do-i-clear-my-eeprom-to-start-from-a-clean-slate

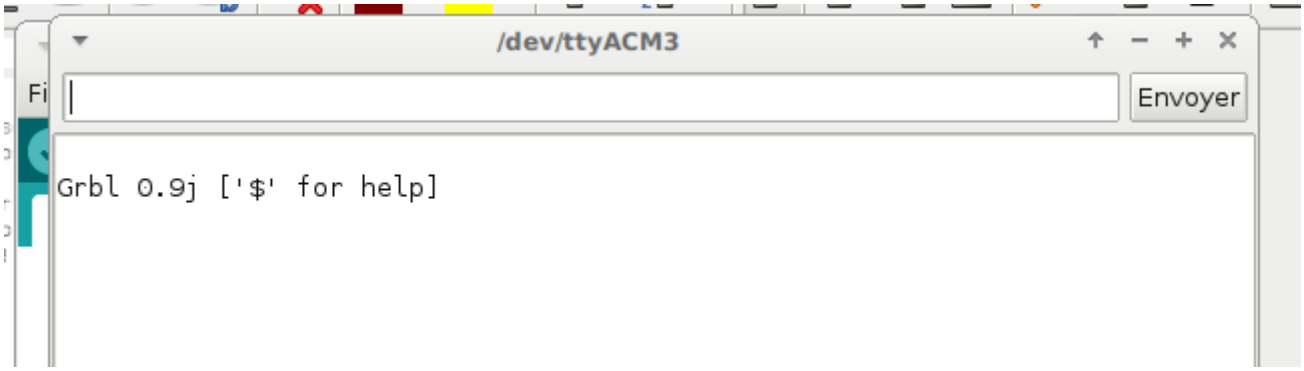
Note : il est aussi possible de programmer grbl avec les paramètres par défaut et secondairement

d'envoyer les paramètres souhaités avec un soft Python. Voir :

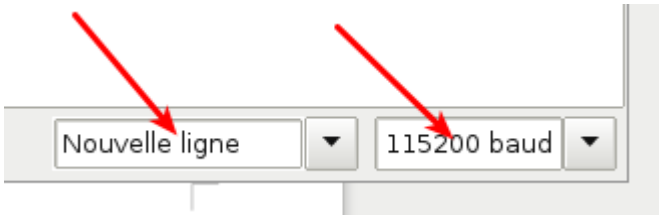
https://github.com/grbl/grbl/wiki/Frequently-Asked-Questions#writing-individual-settings-is-tedious-is-there-a-way-to-speed-this-up

Tester GRBL

Une fois fait, se connecter avec le terminal Arduino : on obtient l'invite :



Vérifier qu'on a bien configuré le terminal :



Et ensuite saisir $$ pour obtenir les information courante :

L'envoi de $ donne l'aide :

Voici la sortie que vous devez avoir avec $$ si tout est bien paramétré pour l'Open Maker Machine Pro :

$0=10 (step pulse, usec)

$1=255 (step idle delay, msec)
$2=0 (step port invert mask:00000000)
$3=7 (dir port invert mask:00000111)
$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=0.010 (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=1 (homing cycle, bool)
$23=7 (homing dir invert mask:00000111)
$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=640.000 (x, step/mm)
$101=640.000 (y, step/mm)
$102=800.000 (z, step/mm)
$110=1200.000 (x max rate, mm/min)
$111=1200.000 (y max rate, mm/min)
$112=500.000 (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=450.000 (x max travel, mm)
$131=450.000 (y max travel, mm)
$132=80.000 (z max travel, mm)

ON peut ensuite tester des Gcode :

```
List of Supported G-Codes in Grbl v0.9 Master:
  - Non-Modal Commands: G4, G10L2, G10L20, G28, G30, G28.1, G30.1, G53, G92, G92.1
  - Motion Modes: G0, G1, G2, G3, G38.2, G38.3, G38.4, G38.5, G80
  - Feed Rate Modes: G93, G94
  - Unit Modes: G20, G21
  - Distance Modes: G90, G91
  - Arc IJK Distance Modes: G91.1
  - Plane Select Modes: G17, G18, G19
  - Tool Length Offset Modes: G43.1, G49
  - Cutter Compensation Modes: G40
  - Coordinate System Modes: G54, G55, G56, G57, G58, G59
  - Control Modes: G61
  - Program Flow: M0, M1, M2, M30*
  - Coolant Control: M7*, M8, M9
  - Spindle Control: M3, M4, M5
  - Valid Non-Command Words: F, I, J, K, L, N, P, R, S, T, X, Y, Z
```

Les instructions de G-Code sont détaillées ici : http://reprap.org/wiki/G-code

Pour configurer le « homing » façon « Imprimante 3D » :

çà se passe dans le fichier config.h
commenter les autres pour le cycle et faire :
#define HOMING_CYCLE_0 ((1<<X_AXIS)|(1<<Y_AXIS)) // home X-Y seulement

#define HOMING_FORCE_SET_ORIGIN // Uncomment to enable. - set home en tant qu'origine

Dans la config dans le terminal :
$23=7 pour inversion dans toutes les directions si besoin
activer le homing avec :
$22=1

Dès lors, au lancement, on a :
['$H'|'$X' to unlock]

Lancer le homing avec $H
Désactiver avec $X

FAQ

Comment garder les moteurs engagés entre 2 ordres…. ?

Z qui redescend car moteur désengagé après ordre… Comment le garder engagé un certain temps ?

Ceci se fait avec $1=255

« $1 - Step idle delay, msec

Every time your steppers complete a motion and come to a stop, Grbl will delay disabling the steppers by this value. **OR**, you can always keep your axes enabled (powered so as to hold position) by setting this value to the maximum 255 milliseconds. Again, just to repeat, you can keep all axes always enabled by setting `$1=255`. »

Tout fonctionne normalement ?

Bravo ! Votre Open Maker Machine est prête à vous obéir au doigt et à l'oeil !