

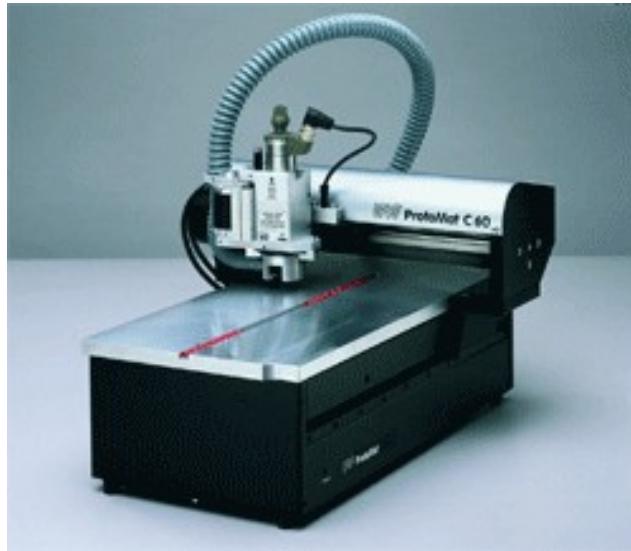
Refurbishing LPKF Protomat C20 PCB CNC Mill Machine

By : Naim Fuad

Last Update : 8 / 10 / 2020

Github Page : <https://github.com/NaimFuad/LPKF-C20-Refurbishing>

Video : <https://youtu.be/ZwDG8o3o-HE>



The LPKF C20 machine is a PCB milling machine that creates a PCB board by milling the contour based from the PCB design. We bought this machine in 2005 and last year this machine has stop functioning due to faulty at controller board.

Since the machine is a close-sourced machine, the only way to repurpose the machine is to make use of the existing hardware and replacing the board controller.

Overview of Machine

Front View



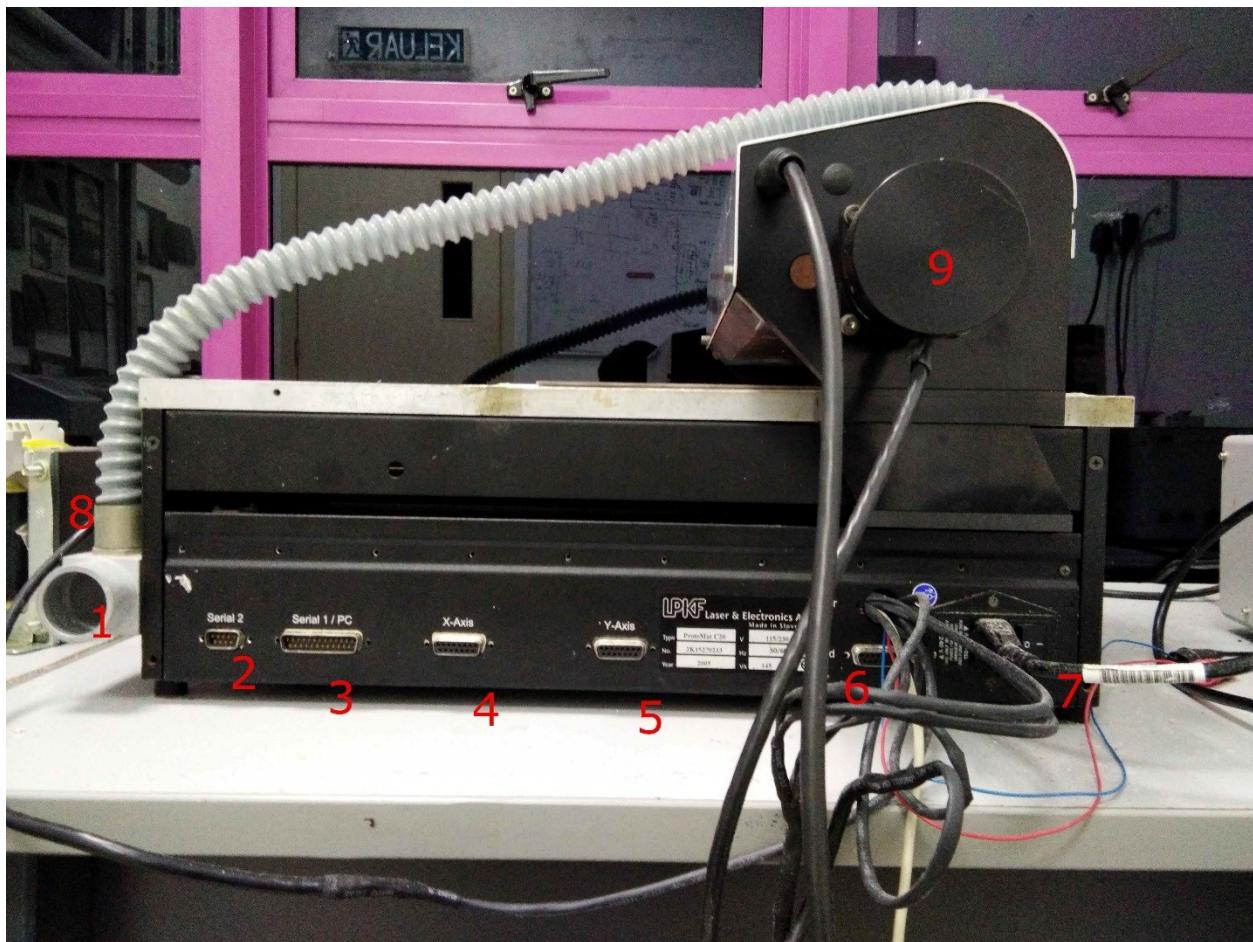
1 - Power Indicator

2 - X-axis motor

3 - DC Spindle Motor / Plotter

4 – Working Depth Limiter

Rear view



1 – Vacuum System Connection

2 – Serial port (9-pin) for expansion units

3 – Serial port (25-pin) for connection to PC

4 – x Motor Connection

5 – Y Motor Connection

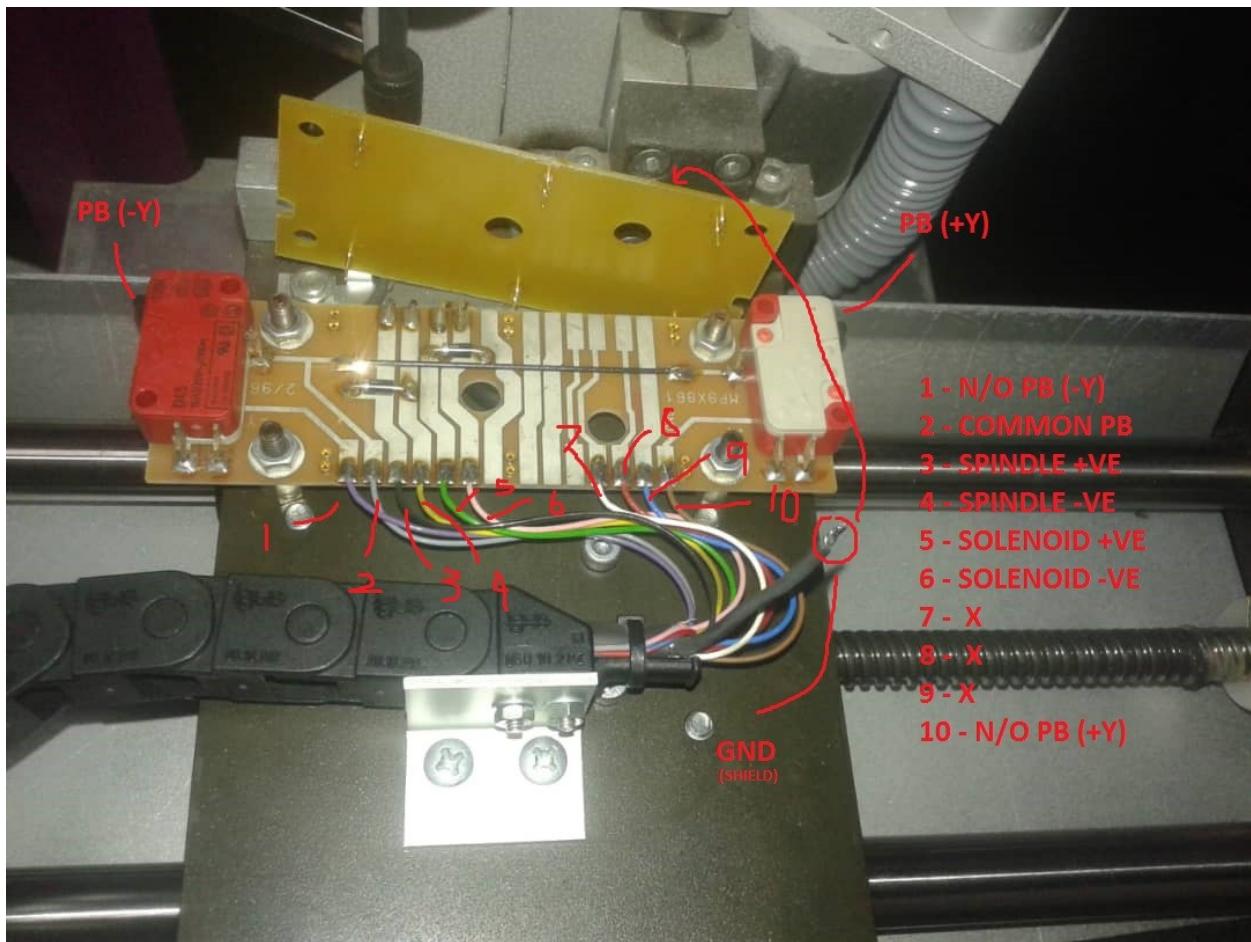
6 – Connection for plotter head

7 – ON/OFF switch with mains connector and fuses

8 – X Motor

9 – Y Motor

Plotter Board



1 – Normally Open Limit Switches for – Y axis

2 – Common terminal for Limit Switches

3 – Spindle / Plotter +DC Supply

4 – Spindle / Plotter –DC Supply

5 – Solenoid Actuator +DC Supply

6 – Solenoid Actuator –DC Supply

7 – NC

8 – NC

9 – NC

10 – Normally Open Limit Switches for +Y Axis

Controller Board



1 – Mains EMI Filter

2 – Transformer

3 – Rectifier (GBU4D) 34V DC Output

4 – Rectifier (GBU4D) 78V DC Output

5 – Rectifier (GBU4D) 11V DC Output

6 – Processing Unit

7 – Stepper Motor Controller (L6203) DMOS Full Bridge Driver

8 – Stepper Motor Controller (L6203) DMOS Full Bridge Driver

9 – Plotter Head Controller

Plotter / Spindle Head



1 – Shock Absorber

2 – Solenoid

3 – Working Depth Limiter

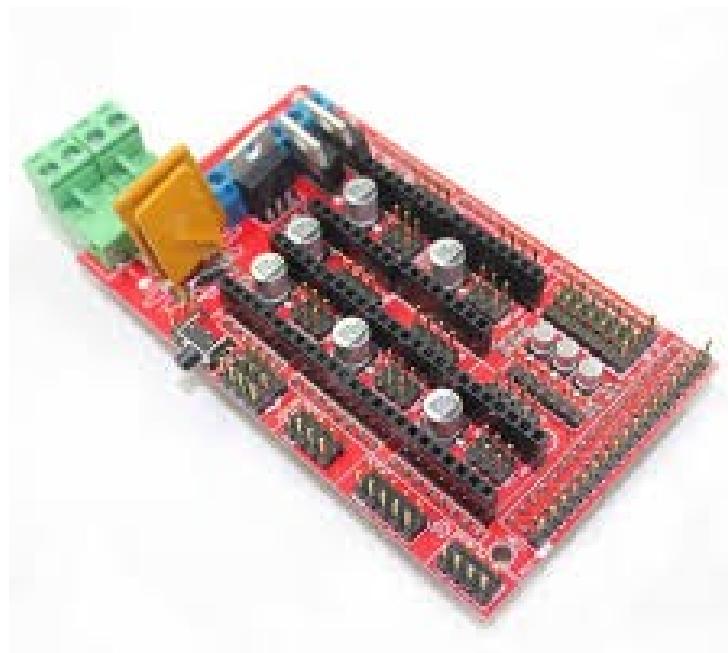
4 – Tool Changer Handle

5 – Mounting for Spindle

6 – DC Spindle Motor

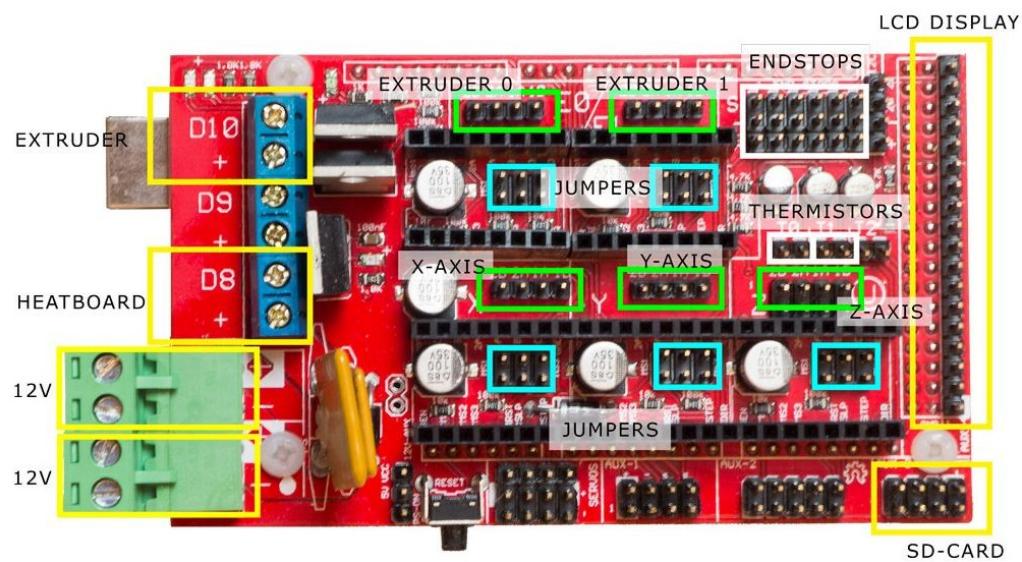
7 – Vacuum Connector

Replacing Controller Board



The controller board was replaced with Open Sourced CNC Machine controller board called RAMPS board or in this case RAMPS version 1.4. This board mainly consist of header for microcontroller, limit switches, stepper motor driver , input supply and with polyfuse to protect against overcurrent, mosfet to drive spindle output.

RAMPS 1.4 Overview



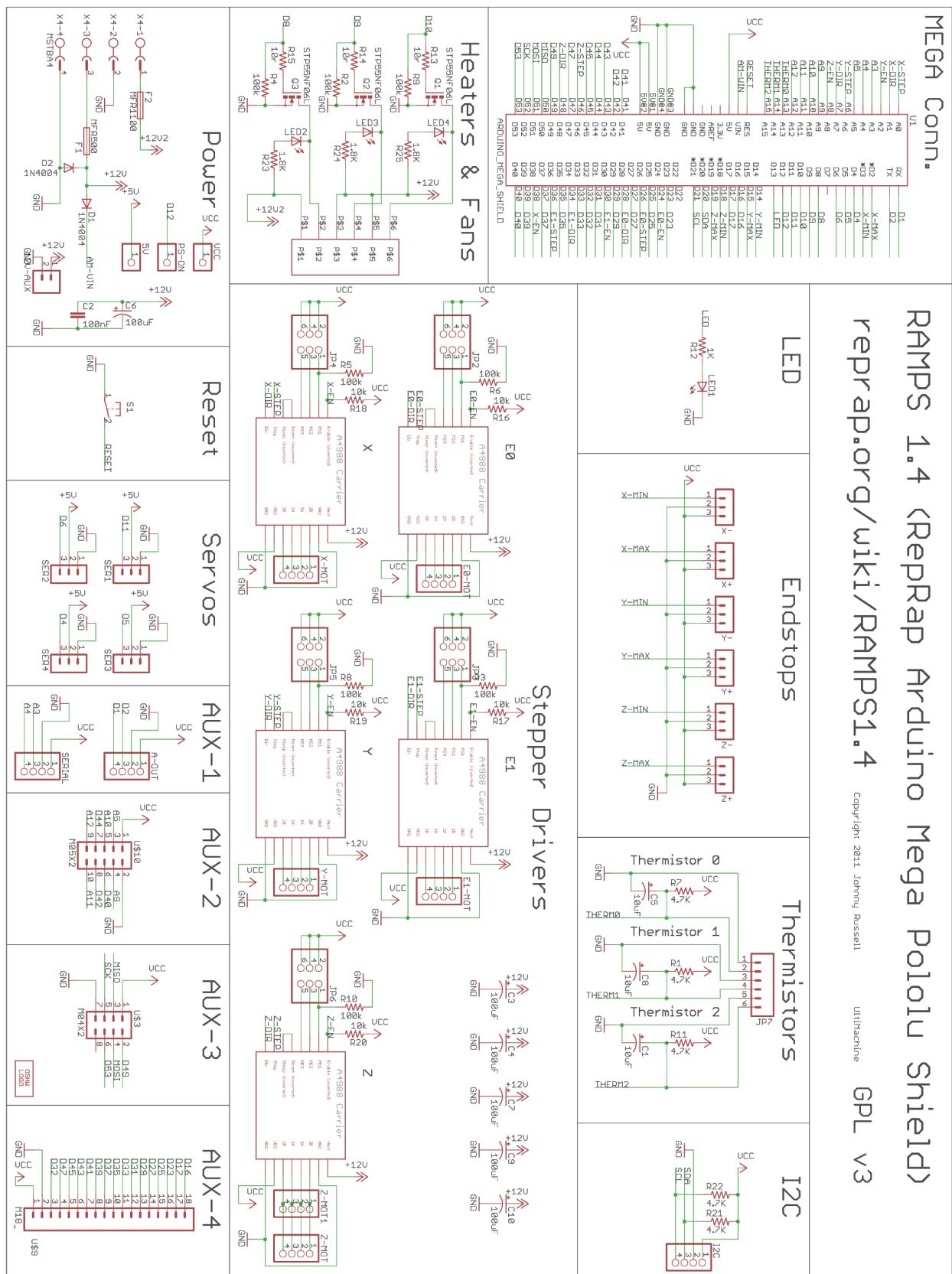
MEGA Conn.

RAMPS 1.4 <RepRap Arduino Mega Pololu Shield>
reprap.org/wiki/RAMPS1.4

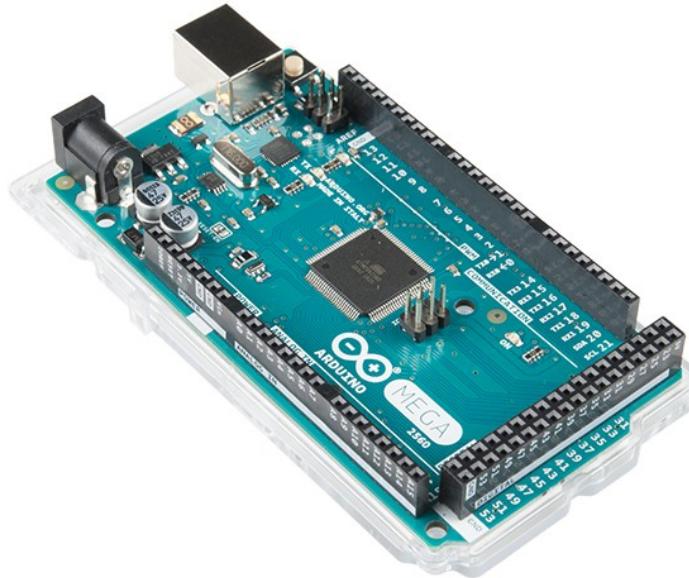
Copyright 2011 Johnny Russell

Ultimachine

GPL v3

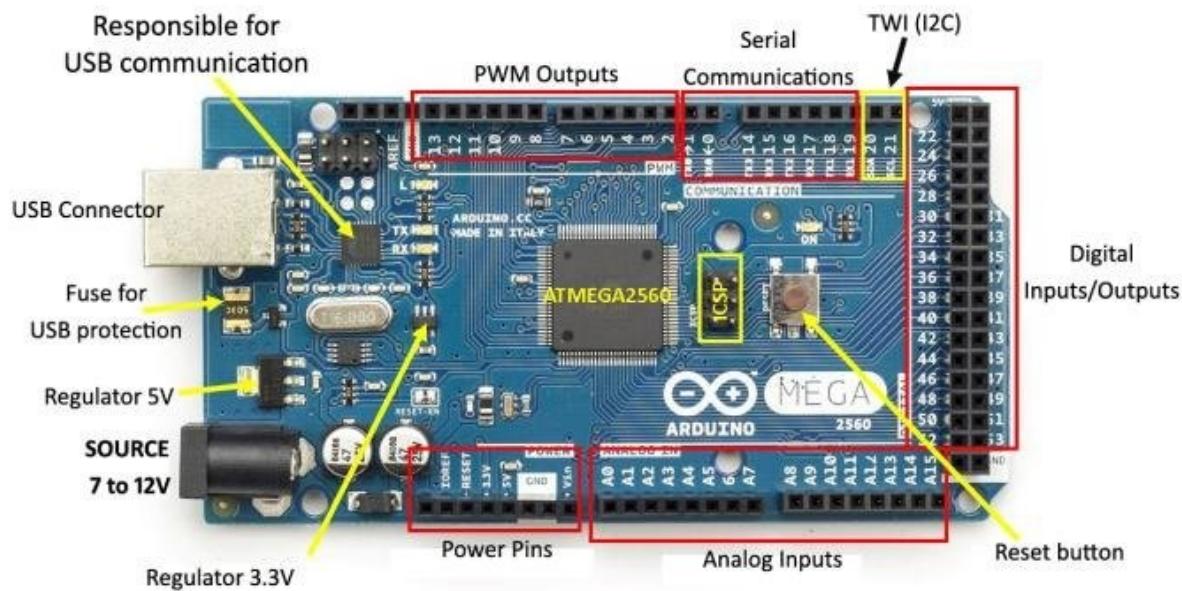


Replacing Processing Unit

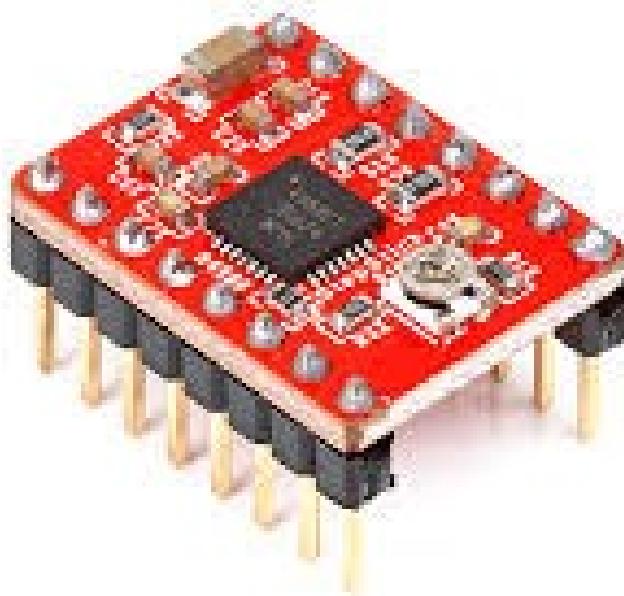


The processing unit was replaced with an Open Sourced microcontroller board based on ATmega 2560 chip. This microcontroller unit consists of power regulator, UART for USB interfaces, serial communications, and multiple GPIO for controller output.

Arduino Mega Overview

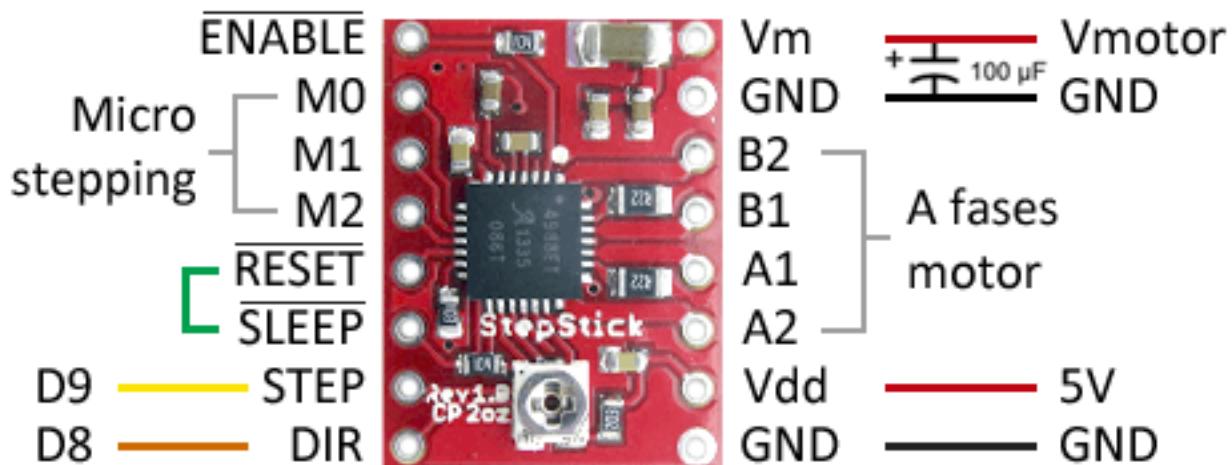


Replacing Stepper Motor Driver

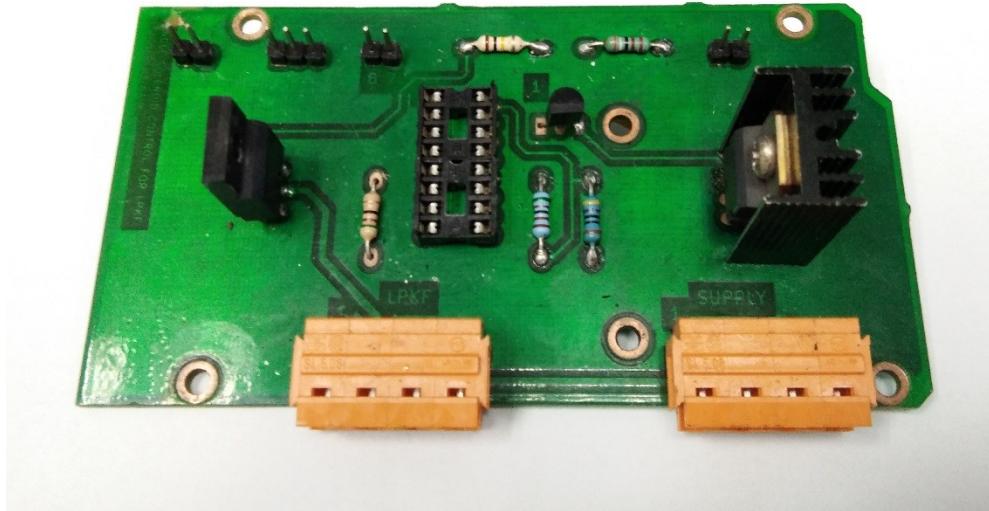


A Breakout board for Allegro's A4988 was used to control the stepper motors. This board was specifically used to control bipolar stepper motor and features adjustable current limiting, overcurrent and over temperature protection and five different microstep resolutions. It is able to deliver up to 1A per phase without heat sink

Allegro A4988 Overview

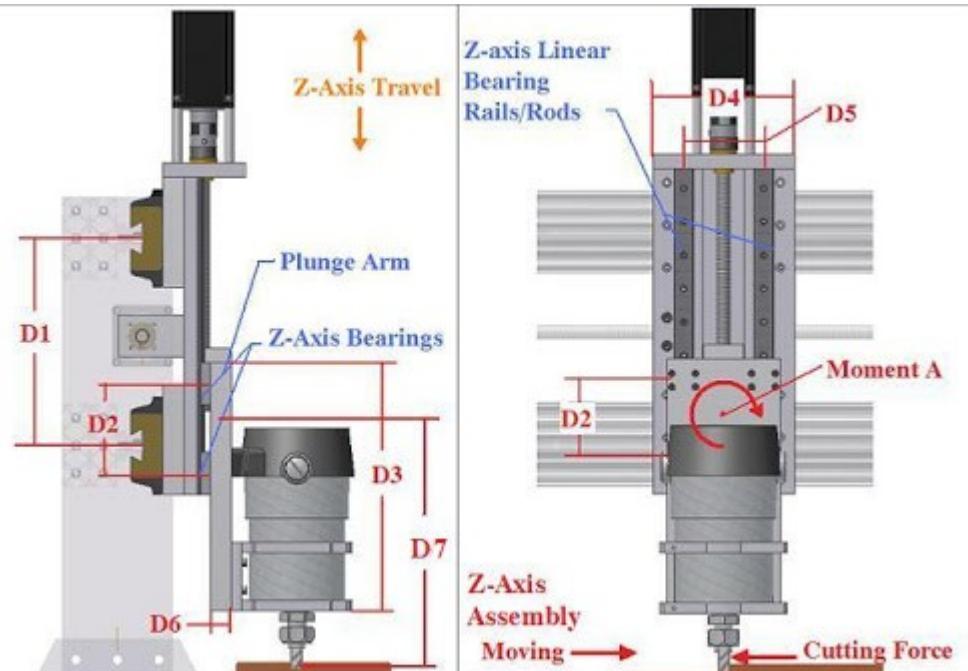


Adding Plotter / Spindle Control

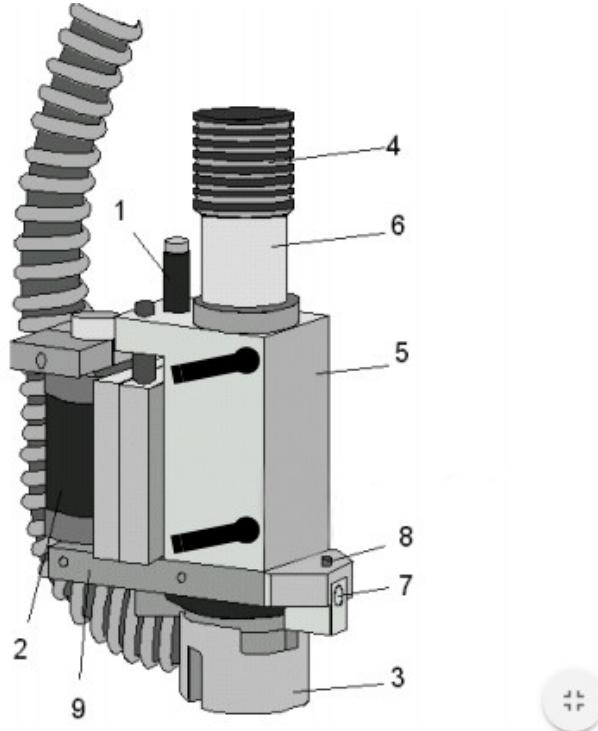


A spindle control board was designed and fabricate in order to control the solenoid actuator that hold the spindle motor. This spindle is responsible for milling contour and drilling operation and the fact that it is based from solenoid, the operation will differ from normal CNC machine.

For standard CNC machine, the Z axis (responsible for holding spindle motor) was guided by stepper motor with linear bearing or rods that enables the travel of the spindle to plunge down to a work area.

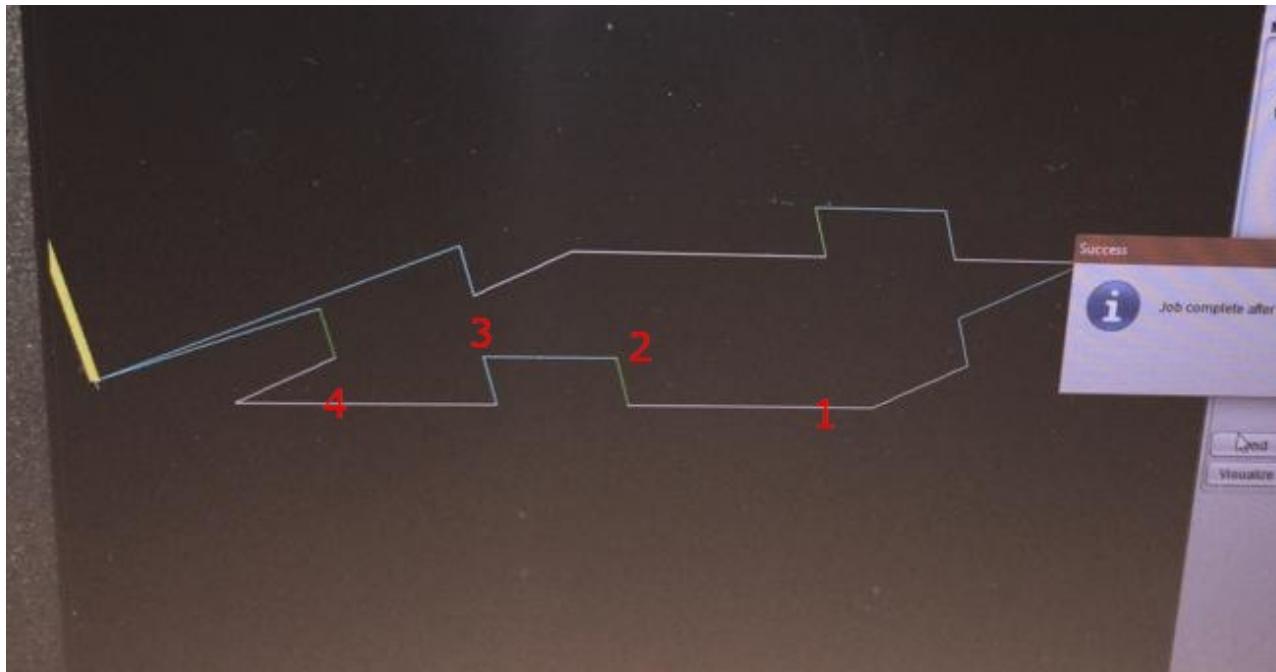


For this machine, originally a solenoid was used instead of stepper motor. In normal condition, the spindle will stay upwards due to the fact that it was held by spring. Once the solenoid is activated, the spindle will plunge down and when deactivated, the spindle will bounce back to the original position due to the spring action.



The firmware that was used (explained further in the next section) was based on standard milling operation where in order to drill / mill, the spindle will travel downwards and stays there. This caused problem with this machine where the solenoid does not latch when drilling / milling and it jitters.

A close inspection was made in the operation of the firmware by tapping signal at the pinout of the RAMPS 1.4 board (explained further in the next section) with the help of oscilloscope.



Above shows a mill operation. The operation is as follows:

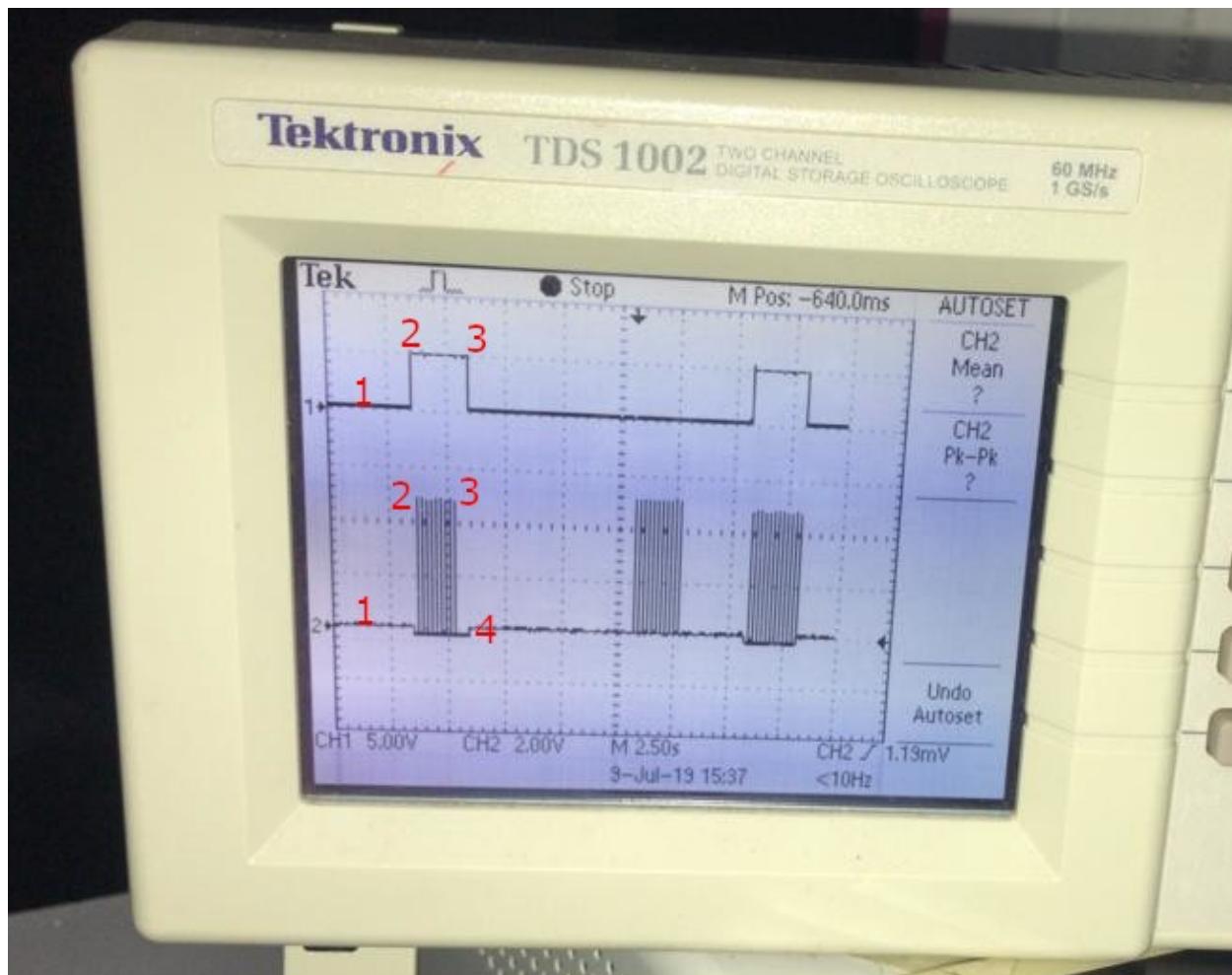
1 – Spindle is being plunged down and continues so

2 – Spindle moves up

(2-3) Spindle stays at top position

3 – Spindle plunges down to continue milling

4 – Spindle is being plunged down and continues so

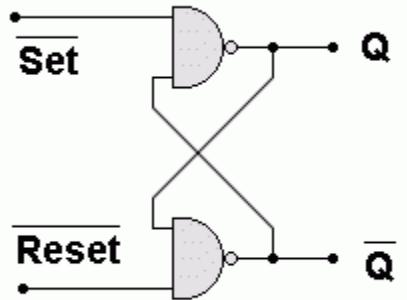


Above shows the signals from RAMPS board. Channel 1 indicates signal for direction and Channel 2 indicates signal for motor stepping.

The jittering of the solenoid was because Channel 2 sends out multiple signals that corresponds with the step angle of the stepper motor that control the travel distance. This is due to the fact that the firmware as made assuming the Z axis was using stepper motor with linear rails/ bearing to drive it.

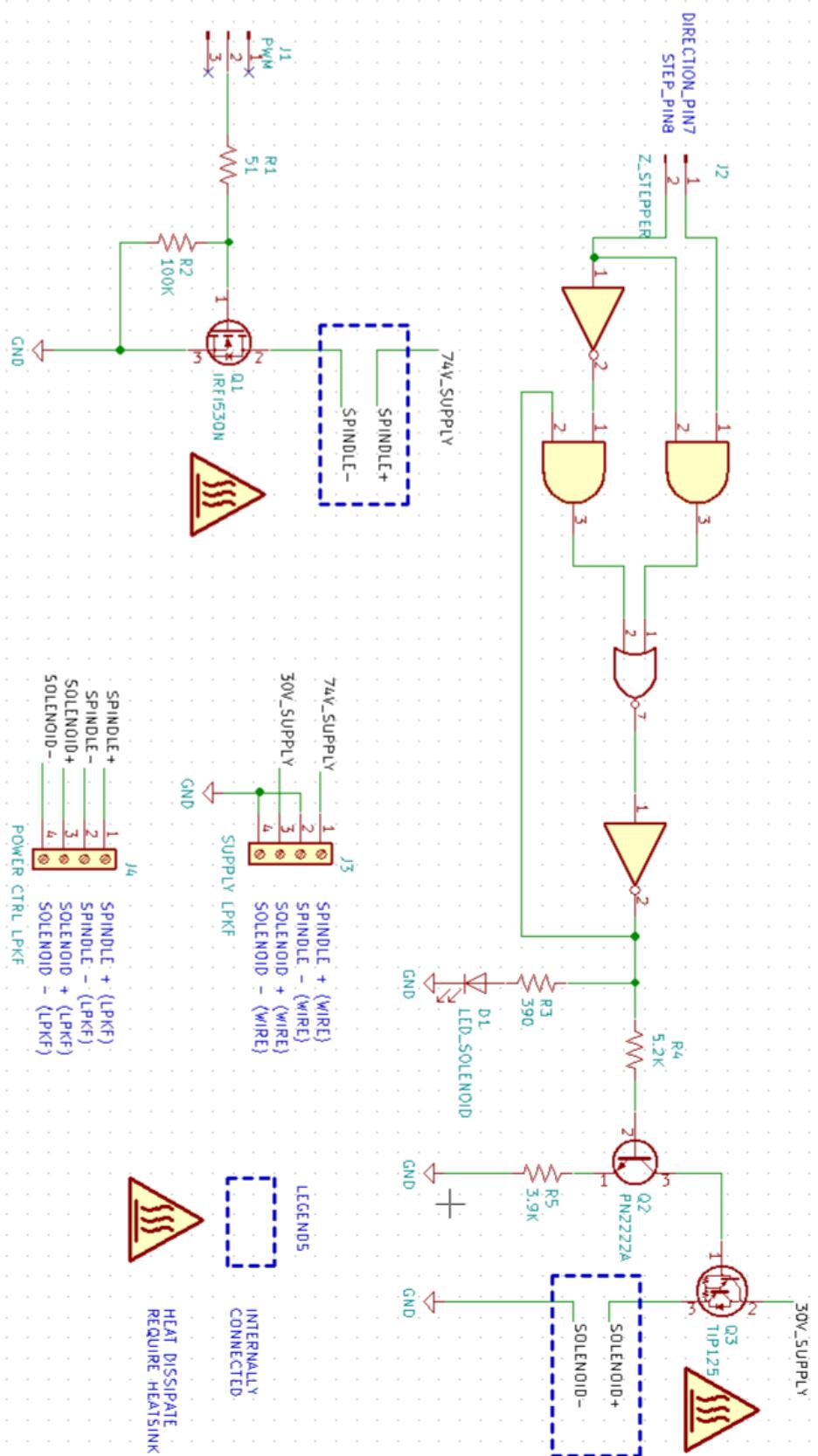
A latching mechanism had designed to overcome this problem.

A logic gate was designed based from bi stable latches operation (SN74LS75 IC) to latch the solenoid until there is a change in the direction signal.

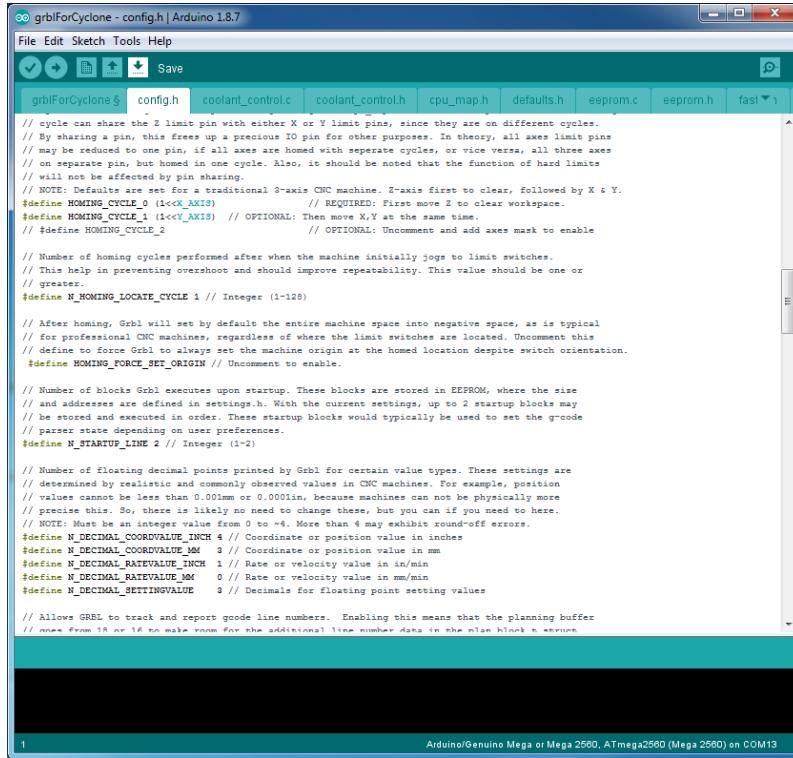


The PCB schematic is as following. The PN2222 Transistor and TIP125 Darlington Transistor was configured in pair in order to amplify the latching signal from the logic gate current to the solenoid.

This eliminated the jitters of the solenoid during the milling process.



GRBL Firmware



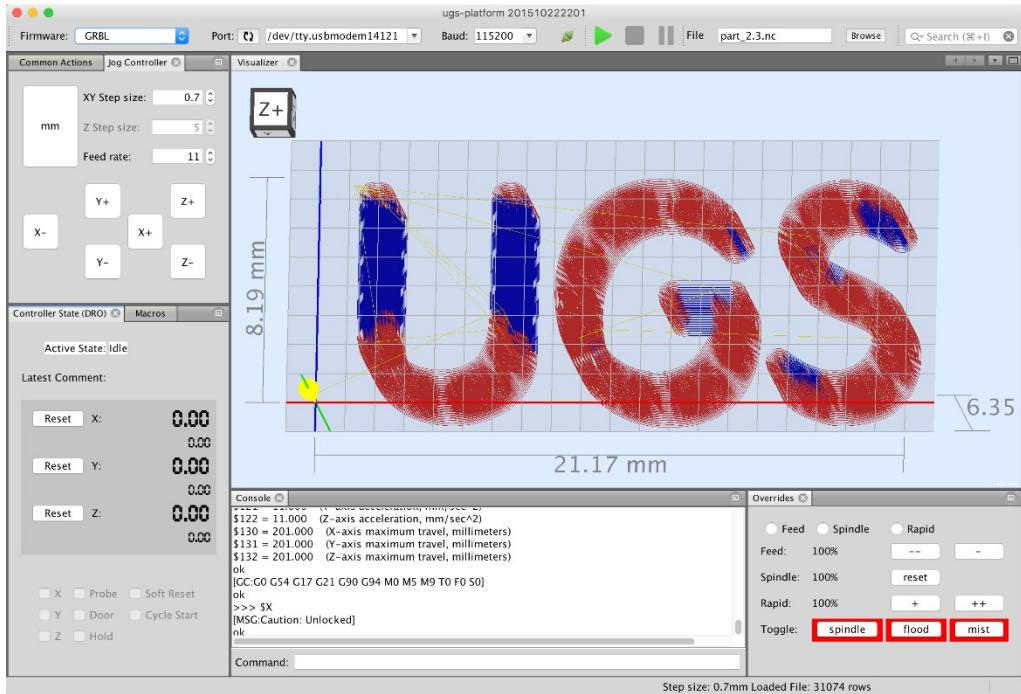
The screenshot shows the Arduino IDE interface with the file "grblForCyclone - config.h" open. The code in the editor is the GRBL configuration file, which includes various defines and comments related to machine setup and performance. The status bar at the bottom indicates the board is an Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on COM13.

```
// cycle can share the Z limit pin with either X or Y limit pins, since they are on different cycles.  
// By sharing a pin, this frees up a precious IO pin for other purposes. In theory, all axes limit pins  
// may be reduced to one pin, if all axes are homed with separate cycles, or vice versa, all three axes  
// on separate pin, but homed in one cycle. Also, it should be noted that the function of hard limits  
// will not be affected by pin sharing.  
// NOTE: Defaults are set for a traditional 3-axis CNC machine. Z-axis first to clear, followed by X & Y.  
#define HOMING_CYCLE_0 (1<<_X_AXIS) // REQUIRED: First move Z to clear workspace.  
#define HOMING_CYCLE_1 (1<<_Y_AXIS) // OPTIONAL: Then move X,Y at the same time.  
#define HOMING_CYCLE_2 // OPTIONAL: Uncomment and add axes mask to enable  
  
// Number of homing cycles performed after when the machine initially jogs to limit switches.  
// This help in preventing overshoot and should improve repeatability. This value should be one or  
// greater than zero.  
#define N_HOMING_LOCATE_CYCLE 1 // Integer (1-128)  
  
// After homing, Grbl will set by default the entire machine space into negative space, as is typical  
// for professional CNC machines, regardless of where the limit switches are located. Uncomment this  
// define to force Grbl to always set the machine origin at the homed location despite switch orientation.  
#define HOMING_FORCE_SET_ORIGIN // Uncomment to enable.  
  
// Number of blocks Grbl executes upon startup. These blocks are stored in EEPROM, where the size  
// and addresses are defined in settings.h. With the current settings, up to 2 startup blocks may  
// be stored and executed in order. These startup blocks would typically be used to set the g-code  
// parser state depending on user preferences.  
#define N_STARTUP_LINE 2 // Integer (1-2)  
  
// Number of floating decimal points printed by Grbl for certain value types. These settings are  
// determined by realistic and commonly observed values in CNC machines. For example, position  
// values are often less than 0.01mm or 0.0001in, because machines can not be physically more  
// precise than that. So, there is little benefit in changing them, unless you can if you need to here.  
// NOTE: Must be an integer value from 0 to <4, since <4 may exhibit round-off errors.  
#define N_DECIMAL_COORDVALUE_INCH 4 // Coordinate or position value in inches  
#define N_DECIMAL_COORDVALUE_MM 3 // Coordinate or position value in mm  
#define N_DECIMAL_RATEVALUE_INCH 1 // Rate or velocity value in in/min  
#define N_DECIMAL_RATEVALUE_MM 0 // Rate or velocity value in mm/min  
#define N_DECIMAL_SETTINGVALUE 3 // Decimals for floating point setting values  
  
// Allows GREL to track and report gcode line numbers. Enabling this means that the planning buffer  
// grows from 16 to 16 to make room for the additional line number data in the plan block n struct
```

A firmware was needed in order to control the movement for stepper motor and spindle based from design. An Open Sourced GRBL motion control firmware was flashed onto the Arduino Mega microcontroller. The GRBL accepts G-Code commands and was mostly used for Open Sourced CNC based project such as CNC Mill Machine and CNC Laser Cutting Machine. The fact that the codes was written in C with Arduino library, configuring the library to suit this project was a breeze.

(Github library - <https://github.com/grbl/grbl>)

G Code Sender Application



With the use of GRBL firmware, a G-Code commands can be sent through serial interfaces (e.g move X axis for 4cm or start spindle at 4000 RPM). But with milling operations, it usually consist of thousands of line of commands. Here, the Universal G-Code sender application helps in managing the commands. This applications also provides a visualizer for the commands sent.

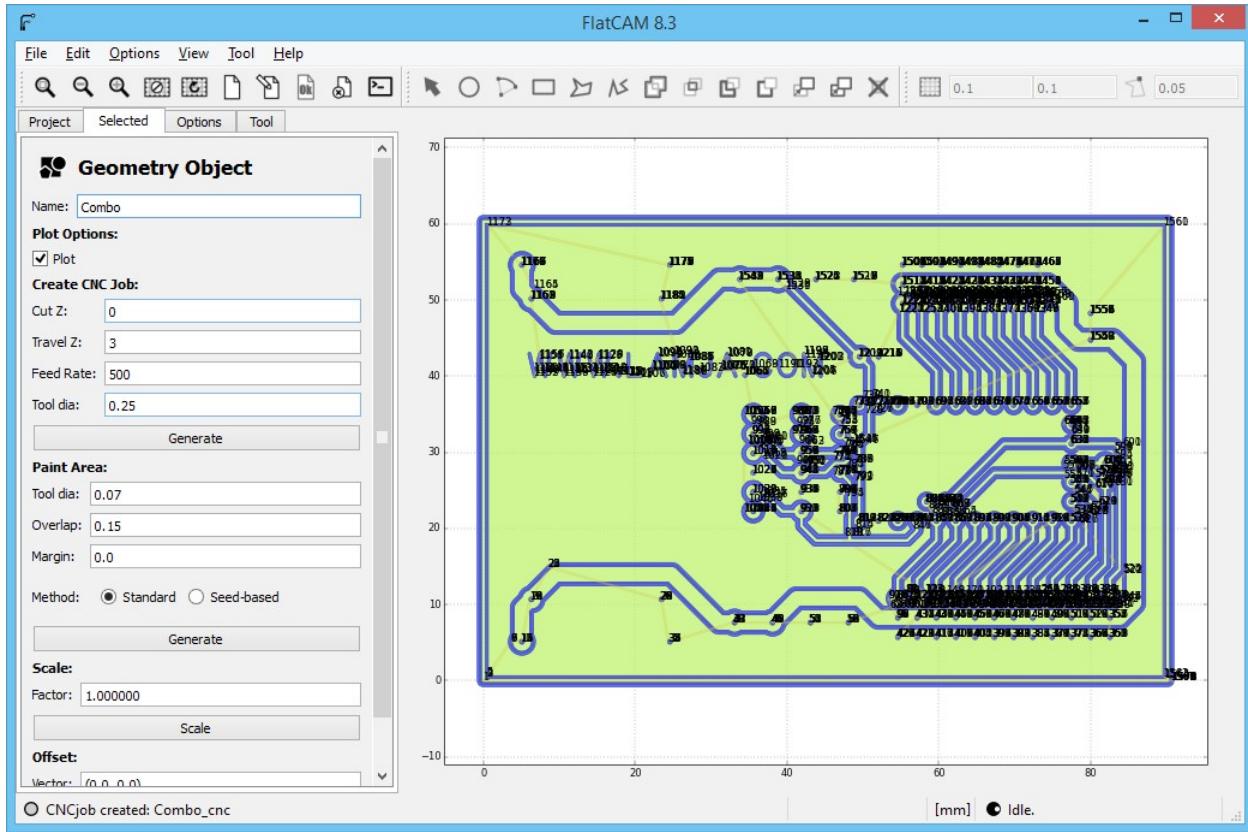
(https://winder.github.io/ugs_website/#universal-gcode-sender)

Example for G-Code Command

for PCB Milling

```
LPKF Pitch DIA 0.2 PASS 3 0.5.txt - ...
File Edit Format View Help
Z1
G21
G90
G94
F600.00
G00 Z1.0000
M03 S1000
G4 P2
G00 X4.6104Y5.2250
G01 Z-1.0000
G01 X4.5965Y5.2510
G01 X4.5843Y5.2778
G01 X4.5740Y5.3054
G01 X4.5654Y5.3336
G01 X4.5586Y5.3622
G01 X4.5540Y5.3913
G01 X4.5511Y5.4206
G01 X4.5500Y5.4500
G01 X4.5511Y5.4794
G01 X4.5540Y5.5087
G01 X4.5586Y5.5378
G01 X4.5654Y5.5664
G01 X4.5740Y5.5946
G01 X4.5843Y5.6222
G01 X4.6031Y5.6621
G01 X4.6180Y5.6876
G01 X4.6343Y5.7120
G01 X4.6521Y5.7355
G01 X4.6716Y5.7575
G01 X4.6925Y5.7784
G01 X4.7145Y5.7979
G01 X4.7380Y5.8157
G01 X4.7624Y5.8320
G01 X4.7879Y5.8469
G01 X4.8143Y5.8598
G01 X4.8415Y5.8710
G01 X4.8694Y5.8806
G01 X4.8978Y5.8881
G01 X4.9267Y5.8939
G01 X4.9559Y5.8978
G01 X4.9926Y5.8999
G01 X12.8122Y5.9000
G01 X12.8416Y5.8989
G01 X12.8563Y5.8978
G01 X12.8855Y5.8939
G01 X12.9000Y5.8914
G01 X12.9266Y5.8846
G01 X12.9568Y5.8760
```

PCB Milling CAM Application



In order to mill a PCB, a CAM software is needed to detect the contour of the design and set the proper parameters in order to achieve the proper clearance. Flatcam reads Gerber file (PCB layout design file) and Excellon (PCB drill design file) and output a G-Code commands for mill operations.

Final Result

The machine are now able to mill PCB with precision. Below shows a successful 100 pin TQFP footprint with a pitch of 0.3mm milled with the machine and compared side by side with PCB made by etching process.

