Bitácora de Trabajo:

Equipo Electrónica

Grupo 1.

- **Docentes**:
  - Osvaldo P. Ivani
  - Sebastián Amago Prato
  - Martín A. Ricciardelli
  - Juan Martín Hermida

- **Estudiantes**:
  - Dante G. Mele Ientile (Project Manager)
  - León A. Martin
  - Hernán A. Silva
  - Nicolás L. Fertonani
  - Laureano M. Rivera Pascua

- **Curso**: 4to Año Ciclo Superior - Electromecánica

- Pin Kill: 41 (pins_RAMPS.h)
- Void del pin kill en Marlin_main.cpp

```
900
901 void setup_killpin() {
902 #if HAS_KILL
903   SET_INPUT_PULLUP(KILL_PIN);
904 #endif
905 }
906
907 #if ENABLED(FILAMENT_RUNOUT_SENSOR)
908
909 void setup_filrunoutpin() {
910 #if ENABLED(ENDSTOPPULLUP_FIL_RUNOUT)
911   SET_INPUT_PULLUP(FIL_RUNOUT_PIN);
912 #else
913   SET_INPUT(FIL_RUNOUT_PIN);
914 #endif
915 }
916
917 #endif
918
919 void setup_powerhold() {
920 #if HAS_SUICIDE
```

```
#endif
918
919 void setup_powerhold() {
920 #if HAS_SUICIDE
921   OUT_WRITE(SUICIDE_PIN, HIGH);
922 #endif
923 #if HAS_POWER_SWITCH
924 #if ENABLED(PS_DEFAULT_OFF)
925   OUT_WRITE(PS_ON_PIN, PS_ON_ASLEEP);
926 #else
927   OUT_WRITE(PS_ON_PIN, PS_ON_AWAKE);
928 #endif
929 #endif
930 }
931
932 void suicide() {
933 #if HAS_SUICIDE
934   OUT_WRITE(SUICIDE_PIN, LOW);
935 #endif
936 }
937
```

DEFINICION DE MAIN

El elemento HTML <main> representa el contenido principal del <body> de un documento o aplicación

FUNCIONES INLINE

Se conocen como funciones inline a las funciones que, al compilar, no son llamadas en el código objeto, sino *insertadas* en la sección del código donde se las llame.

```
10806
10807 #if DISABLED(EMERGENCY_PARSER)
10808
10809         case 108: // M108: Cancel Waiting
10810           gcode_M108();
10811           break;
10812
10813         case 112: // M112: Emergency Stop
10814           gcode_M112();
10815           break;
10816
10817         case 410: // M410 quickstop - Abort all the planned moves.
10818           gcode_M410();
10819           break;
10820
```

IF EMERGENCY_PARSER ESTÁ DESHABILID0:

```
7304    M112: Emergency Stop
7305 */
7306 inline void gcode_M112() {
7307   kill(PSTR(MSG_KILLED));
7308 }
7309
7310
7311 /**
7312    M410: Quickstop - Abort all planned moves
7313
7314    This will stop the carriages mid-move, so most likely they
7315    will be out of sync with the stepper position after this.
7316 */
7317 inline void gcode_M410() {
7318   quickstop_stepper();
7319 }
7320
7321 #endif
```

- El motor está definido por default en el pin D9, y el ventilador en el D8

define MSG_ERR_KILLED "Printer halted. kill() called!"

is used only in this method

void kill(const char* lcd_msg) {

SERIAL_ERROR_START;

SERIAL_ERRORLNPGM(MSG_ERR_KILLED);

if ENABLED(ULTRA_LCD)

kill_screen(lcd_msg);

else

UNUSED(lcd_msg);

endif

delay(500); // Wait a short time

cli(); // Stop interrupts

thermalManager.disable_all_heaters();

disable_all_steppers();

if HAS_POWER_SWITCH

SET_INPUT(PS_ON_PIN);

endif

suicide();

while (1) {

if ENABLED(USE_WATCHDOG)

- watchdog_reset();

#endif

} // Wait for reset

}

Problem is that it can be called from many places:

1070: if (strcmp(command, "M112") == 0) kill(PSTR(MSG_KILLED));

/**

- Manage several activities:
  - Check for Filament Runout
  - Keep the command buffer full
  - Check for maximum inactive time between commands
  - Check for maximum inactive time between stepper commands
  - Check if pin CHDK needs to go LOW
  - Check for KILL button held down
  - Check for HOME button held down
  - Check if cooling fan needs to be switched on
  - Check if an idle but hot extruder needs filament extruded (EXTRUDER_RUNOUT_PREVENT)

    /

    void manage_inactivity(bool ignore_stepper_queue/=false*/) {

    .....

    9924: if (max_inactive_time && ELAPSED(ms, previous_cmd_ms + max_inactive_time)) kill(PSTR(MSG_KILLED));

which is probably turning of after idlle...

Then there are the temperature safes (can you recompile without the temperature safety to eliminate this code?

if DISABLED(BOGUS_TEMPERATURE_FAILSAFE_OVERRIDE)

- if (!killed) {
- Running = false;
- killed = true;
- kill(lcd_msg);
- }
- else

disable_all_heaters(); // paranoia

endif

if ENABLED(TEMP_SENSOR_1_AS_REDUNDANT)

if (e > HOTENDS)

else

if (e >= HOTENDS)

endif

- {
- SERIAL_ERROR_START;
- SERIAL_ERROR((int)e);
- SERIAL_ERRORLNPGM(MSG_INVALID_EXTRUDER_NUM);
- kill(PSTR(MSG_KILLED));
- return 0.0;

}

There is one when handling SD files (but you said it happens even without)

if (isFileOpen()) { //replacing current file by new file, or subfile call

if (push_current) {

if (file_subcall_ctr > SD_PROCEDURE_DEPTH - 1) {

SERIAL_ERROR_START;

SERIAL_ERRORPGM("trying to call sub-gcode files with too many levels. MAX level is:");

SERIAL_ERRORLN(SD_PROCEDURE_DEPTH);

kill(PSTR(MSG_KILLED));

return;

}

Then there is watch dog

// Watchdog timer interrupt, called if main program blocks >1sec and manual reset is enabled.

if ENABLED(WATCHDOG_RESET_MANUAL)

ISR(WDT_vect) {

SERIAL_ERROR_START;

SERIAL_ERRORLNPGM("Something is wrong, please turn off the printer.");

kill(PSTR("ERR:Please Reset")); //kill blocks //16 characters so it fits on a 16x2 display

while (1); //wait for user or serial reset

}

endif //WATCHDOG_RESET_MANUAL

It needs to be one of these, disable one by one each of them in the firmware with the Configuration.h to know which it is. It could even be even M112 kind, like emergency stop, it could be triggered by both min and max endstops triggered at the same time (that could happen if contacts got dirty, or the connector itself and is leaking current into wrong pin on the mcu).

If you have kill button then probably this is enabled as well

if HAS_KILL

- // Check if the kill button was pressed and wait just in case it was an accidental
- // key kill key press
- // ----------
- static int killCount = 0; // make the inactivity button a bit less responsive
- const int KILL_DELAY = 750;
- if (!READ(KILL_PIN))
- killCount++;
- else if (killCount > 0)
- killCount--;
-
- // Exceeded threshold and we can confirm that it was not accidental
- // KILL the machine
- // ----------

if (killCount >= KILL_DELAY) kill(PSTR(MSG_KILLED));

endif

So then any dirt or interference around that pin can trigger it by accident. Hope something will help you find out what the cause is.

- PASOS PARA MARLIN: https://marlinfw.org/docs/development/coding_standards.html

- ACLARACIÓN SOBRE LIBRERIAS/FUNCIONES, ETC. https://aprendiendoarduino.wordpress.com/2016/11/16/funciones-definidas-por-usuario-2/

- Kill display y como hacer o intentar hacer que cuando se apague quede en su posición: https://www.spainlabs.com/foros/tema-Cambiar-mensajes-de-error-Marlin

```
12705  #if HAS_KILL
12706
12707    // Check if the kill button was pressed and wait just in case it was an accidental
12708    // key kill key press
12709    // -------------------------------------------------------------------------------
12710    static int killCount = 0;   // make the inactivity button a bit less responsive
12711    const int KILL_DELAY = 750;
12712    if (!READ(KILL_PIN))
12713      killCount++;
12714    else if (killCount > 0)
12715      killCount--;
12716
12717    // Exceeded threshold and we can confirm that it was not accidental
12718    // KILL the machine
12719    // ------------------------------------------------------------------
12720    if (killCount >= KILL_DELAY) {
12721      SERIAL_ERROR_START();
12722      SERIAL_ERRORLNPGM(MSG_KILL_BUTTON);
12723      kill(PSTR(MSG_KILLED));
```

Lo de la imagen anterior está en Marlin_main

- Crear funciones básicas en Marlin y más especificaciones: https://hackaday.io/project/164156-lp3d-a-fully-lasercut-kit-3d-printer/log/166506-creating-custom-functions-in-marlin
-

\

- REED SWITCH NC Y NO

Control your Meccano models (or anything else) from your Windows PC!
Take a look at my new MECControl project at **meccontrol.com**

A reed switch consists of two or three springy metal reeds having plated, long-life contacts at the tips and encapsulated in a sealed glass tube. The two-reed type has normally open (NO) contacts which close when operated, and the three-reed type is a changeover, i.e. it has a pair of normally open (NO) and a pair of normally closed (NC) contacts. When the switch is operated, both these pairs change to the opposite state. Both types are shown in figure 1.

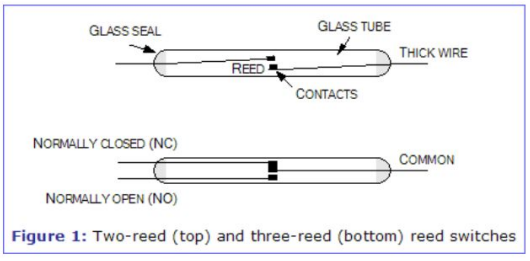


Figure 1: Two-reed (top) and three-reed (bottom) reed switches

- G-codes sender y más
  https://reprap.org/wiki/PCB_Milling#Software_suites
- Mapeo de Alturas con Marlin:
  https://martinvb.com/wp/using-marlins-auto-leveling-for-pcb-milling/
- Para mañana:
  https://reprap.org/forum/read.php?131,554184
  https://forum.v1engineering.com/t/pcb-milling-bed-levelling/8138/4

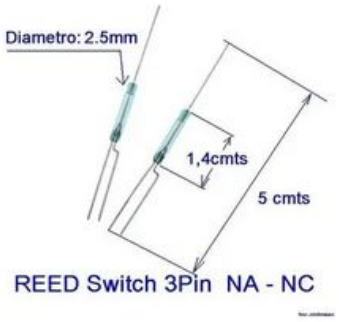
- Marlin para torpes - Linked
  https://es.slideshare.net/xprado/marlin-paratorpes102

- Guía completa: Configurar Marlin 2.0.x desde cero y no morir en el intento
  https://3dwork.io/configurar-marlin-2-0-x-desde-cero/

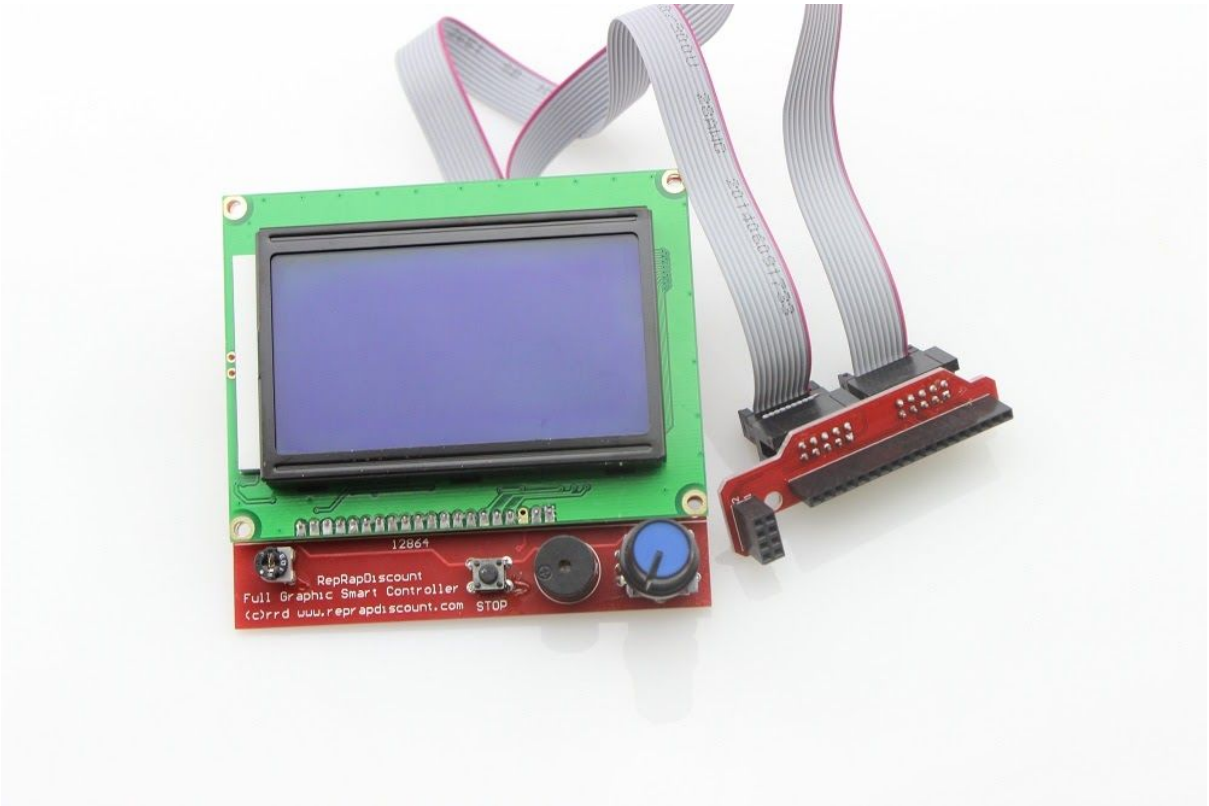
- Firmware definitivo (Marlin)
  https://www.zonamaker.com/impresion-3d/crea-impresora/14-crea-imp-firmaware-marlin

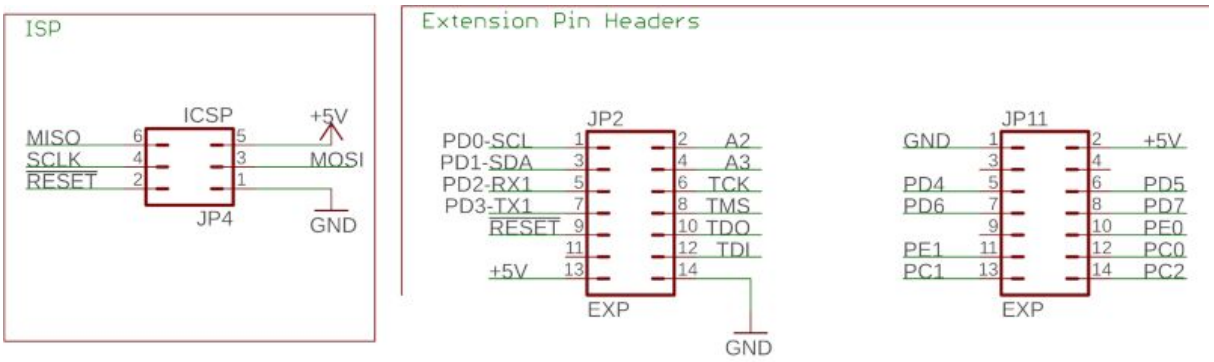
- Diagrama fuente switching
  https://www.diarioelectronicohoy.com/blog/imagenes/2010/11/TO-220AB.gif

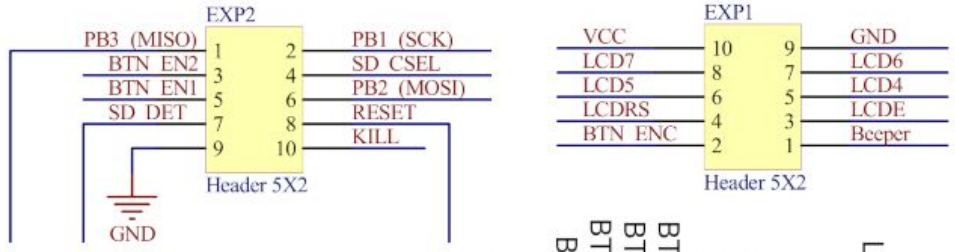
-

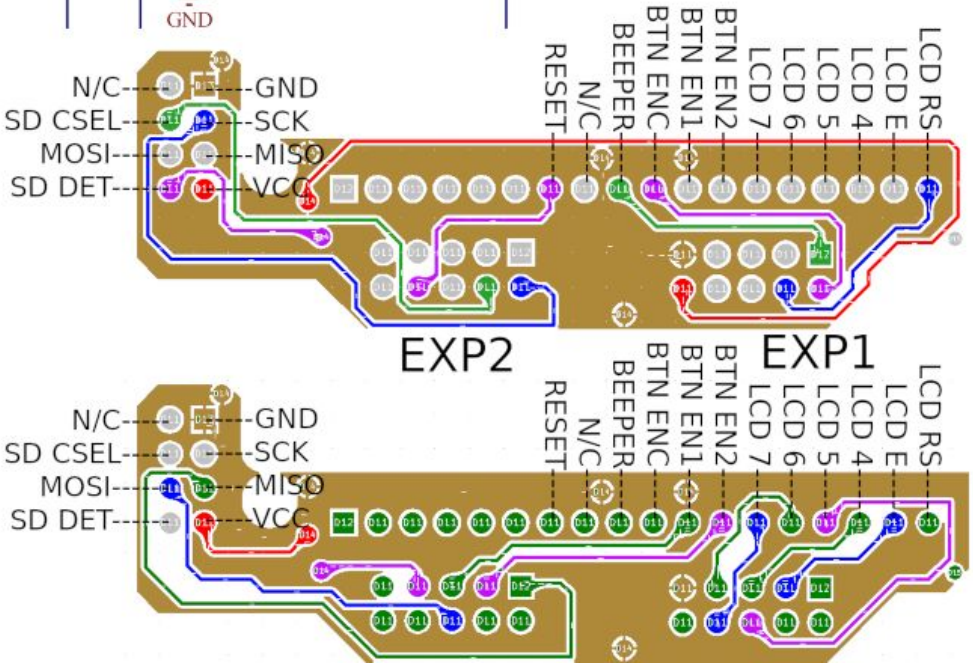

Diametro: 2.5mm
1,4cmts
5 cmts
REED Switch 3Pin NA - NC

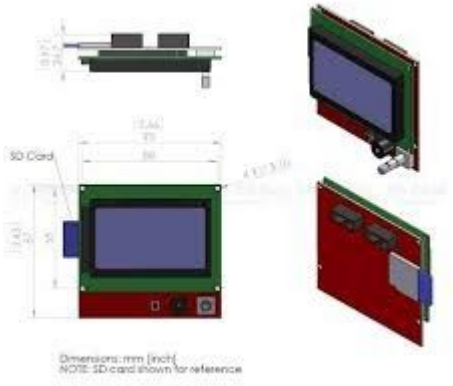

Diametro: 2.5mm
1,4cmts
5 cmts
REED Switch 3Pin NA - NC


12864
RepRapDiscount
Full Graphic Smart Controller
STOP


ISP
ICSP
MISO
SCLK
RESET
+5V
MOSI
GND
JP4
Extension Pin Headers
JP2
PD0-SCL
PD1-SDA
PD2-RX1
PD3-TX1
RESET
+5V
A2
A3
TCK
TMS
TDO
TDI
EXP
GND
JP11
GND
PD4
PD6
PE1
PC1
+5V
PD5
PD7
PE0
PC0
PC2
EXP


EXP2
PB3 (MISO)
BTN_EN2
BTN_EN1
SD_DET
PB1 (SCK)
SD_CSEL
PB2 (MOSI)
RESET
KILL
Header 5X2
GND
EXP1
VCC
LCD7
LCD5
LCDRS
BTN_ENC
GND
LCD6
LCD4
LCDE
Beeper
Header 5X2


N/C
SD CSEL
MOSI
SD DET
GND
SCK
MISO
VCC
RESET
N/C
BEEPER
BTN ENC
BTN EN1
BTN EN2
LCD 7
LCD 6
LCD 5
LCD 4
LCD E
LCD RS
EXP2
EXP1


SD Card

https://reprap.org/wiki/CNC_Gcode_controller

VIDEOS SOBRE EL CONTROLADOR CNC-GCODE:
https://www.youtube.com/watch?v=xxbBipRvp5I&t=1114s
https://www.youtube.com/watch?v=Y12oszEqwF8

Blogs sobre la app:
http://diwo.bq.com/cnc-gcode-controller-instalacion-y-uso/

Blog sobre GCODE:
https://gcodetutor.com/cnc-machine-training/cnc-g-codes.html
https://gcodetutor.com/articles.html
https://gcodetutor.com/cnc-machine-training/cnc-g-codes.html
https://gcodetutor.com/fanuc-training-course/cnc-mill-programming.html
https://gcodetutor.com/gcode-tutorial/cnc-m-codes.html
https://tormach.com/temporary-work-offsets-g92-g92-1-g92-2-and-g92-3

GCODE - MARLIN:
https://marlinfw.org/docs/gcode/G092.html
https://marlinfw.org/docs/gcode/G092.html

MAS DE LO MISMO
https://www.haascnc.com/service/codes-settings.type=gcode.machine=mill.value=G28.html

https://machmotion.com/cnc-info/g-code.html
https://books.google.com.ar/books?id=D4x8AwAAQBAJ&pg=PT278&lpg=PT278&dq=parameters+5161-5166&source=bl&ots=L_EJJrix3Z&sig=ACfU3U1WRtkiTr9Qp7rVWCB2XOVr2aQogg&hl=en&sa=X&ved=2ahUKEwjJqtXr7vzpAhWWHbkGHXYRCRkQ6AEwAHoECAkQAQ#v=onepage&q=home&f=false

HABER, COMO SE HACE LA PUESTA A CERO DE LA MAQUINA

- Primero, no te hagas el tonto y pone en la pestaña “Advanced Options” de la mejor app del mundo (CNC Gcode Controller) Homing para que la herramienta se vaya hasta los finales de carrera y se establezca de esta forma el cero máquina.

- Pasaste lo anterior?
- Si!!
- Muy bien!, pero te falta el 0 pieza nene.
- Uy, como hago eso?
- Primero no grites, lo que tenés que hacer es poner en “Delta” la distancia en donde queres que este tu cero pieza. Y una vez que estés en esa posición te vas a "Set Position" y pones X,Y y Z todo en 0
- Uhh, no se bien donde tiene que ir
- Tranquilo, tenemos todo preparado para tontos como vos, esto es APB. Anda a la pestaña “Simple Controls” y movete

hasta que masomenos a ojo veas que la herramienta está en el borde izquierdo de la placa pcb.
- OKOK
- Como okok, te acabo de salvar la vida.
- Bueeee tranquilo
- Ya ta?
- Si, ya fui a donde maso quiero que vaya
- Bien, hace lo de "Set Position", todo te tengo que decir viejo?
- Disculpá, ya ta
- Bueno listo, que mas queres, nos vemos en el próximo tutorial. Saludos!

El chino conecta el motor a D9

- Mos si es que el que viene se prende fuego https://reprapworld.com/products/electronics/components/mosfet_s/mosfet_n_ch_30v_150a_irlb8743pbf/

  Conector interlock C14 CHASSIS

http://www.sycelectronica.com.ar/articulo.php?codigo=TECLA-TR12-N

- Vídeo de explicación boost:
  https://www.youtube.com/watch?v=6WwCqxSaF_w
  https://www.youtube.com/watch?v=9tQR-i7b6WA
  https://www.youtube.com/watch?v=RRY0R2vqKIE
  https://www.youtube.com/watch?v=RRY0R2vqKIE

http://www.reprap.com/wiki/Heated_Bed_MOSFET_Power_Expansion_Module

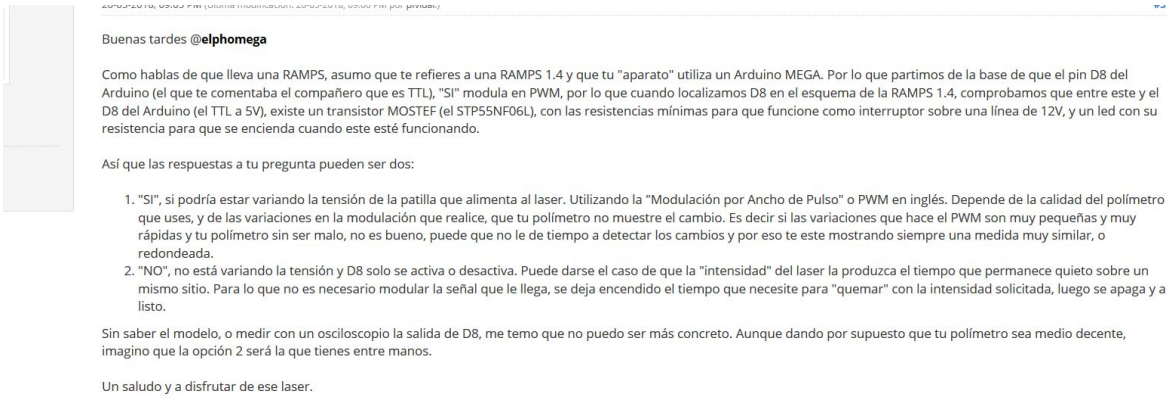
Tema MKS MOS - DISIPACIÓN DE CALOR, FUENTE DE 24 V - APROVECHAMIENTO MÁXIMO DE POTENCIA

https://github.com/Aus3D/RUMBA-Plus/issues/7 -> Hablan de cuanto se banca el MOS de Ramps

Video donde explica todo lo relacionado con la fuente de 24V y el MKS MOS
https://www.youtube.com/watch?v=6WwCqxSaF_w&list=PLBd0pmvp-_tDTLAkC3afL6U3rJCrxE6eh&index=2&t=1473s

TEMA MOTORES PASO A PASO Y SUS CABLES:

- Twist the wires (enrosca los cables del pap para disminuir la inducción
- 3 metros de cable no pasa na!
- Se pueden enrroscar por bobina en lugar de todos juntos. Así se hace en los servomotores industriales
- En cuanto a la vibración y al ruido de los motores, se colocan unos dampers



Buenas tardes @**elphomega**

Como hablas de que lleva una RAMPS, asumo que te refieres a una RAMPS 1.4 y que tu "aparato" utiliza un Arduino MEGA. Por lo que partimos de la base de que el pin D8 del Arduino (el que te comentaba el compañero que es TTL), "SI" modula en PWM, por lo que cuando localizamos D8 en el esquema de la RAMPS 1.4, comprobamos que entre este y el D8 del Arduino (el TTL a 5V), existe un transistor MOSTEF (el STP55NF06L), con las resistencias mínimas para que funcione como interruptor sobre una línea de 12V, y un led con su resistencia para que se encienda cuando este esté funcionando.

Así que las respuestas a tu pregunta pueden ser dos:

1. "SI", si podría estar variando la tensión de la patilla que alimenta al laser. Utilizando la "Modulación por Ancho de Pulso" o PWM en inglés. Depende de la calidad del polímetro que uses, y de las variaciones en la modulación que realice, que tu polímetro no muestre el cambio. Es decir si las variaciones que hace el PWM son muy pequeñas y muy rápidas y tu polímetro sin ser malo, no es bueno, puede que no le de tiempo a detectar los cambios y por eso te este mostrando siempre una medida muy similar, o redondeada.
2. "NO", no está variando la tensión y D8 solo se activa o desactiva. Puede darse el caso de que la "intensidad" del laser la produzca el tiempo que permanece quieto sobre un mismo sitio. Para lo que no es necesario modular la señal que le llega, se deja encendido el tiempo que necesite para "quemar" con la intensidad solicitada, luego se apaga y a listo.

Sin saber el modelo, o medir con un osciloscopio la salida de D8, me temo que no puedo ser más concreto. Aunque dando por supuesto que tu polímetro sea medio decente, imagino que la opción 2 será la que tienes entre manos.

Un saludo y a disfrutar de ese laser.

- Conexión de cable apantallado: Como conectarlo, porqué, etc. https://www.cemdal.com/2020/01/23/las-conexiones-a-masa-de-un-cable-apantallado-en-uno-o-dos-lados/

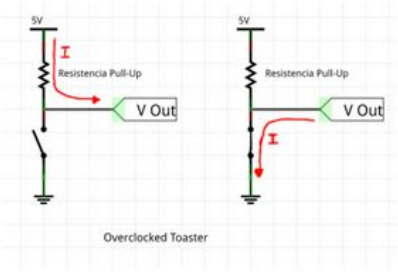
- OJO QUE PARA EL FINAL DE CARRERA HAY QUE COLOCAR LAS RESISTENCIAS PULL DOWN Y PULL UP EN EL MARLIN

- Makerbots tiene planitos de las cosas

- El final de carrera lo usamos como viene pero de forma empírica le metemos un pequeño filtro y un pull down o pull up si es necesario.

2.5 EndStop PullUp.

Las resistencias PullUp se emplean en electrónica digital para confirmar el estado de una entrada, mas información en http://ovtoaster.com/resistencias-pulldown-y-pullup/.



Arduino (el micro Atmel) tiene interiormente resistencias pullup en sus entradas digitales, y estas resistencias podemos activarlas o no. Desde este bloque le diremos al micro controlador si tiene que activarlas en todas las entradas correspondientes a los EndStop (finales de carrera) o solo en algunas. Solo es necesario activar las resistencias PullUp en caso de que tu EndStop sea un simple micro mecánico (imagen de abajo a la izquierda), si usas alguno de los que hay en el mercado que están montados sobre una PCB con algún componente SMD, (imagen de abajo a la derecha) no necesitas activarlas.

SI PULLUP

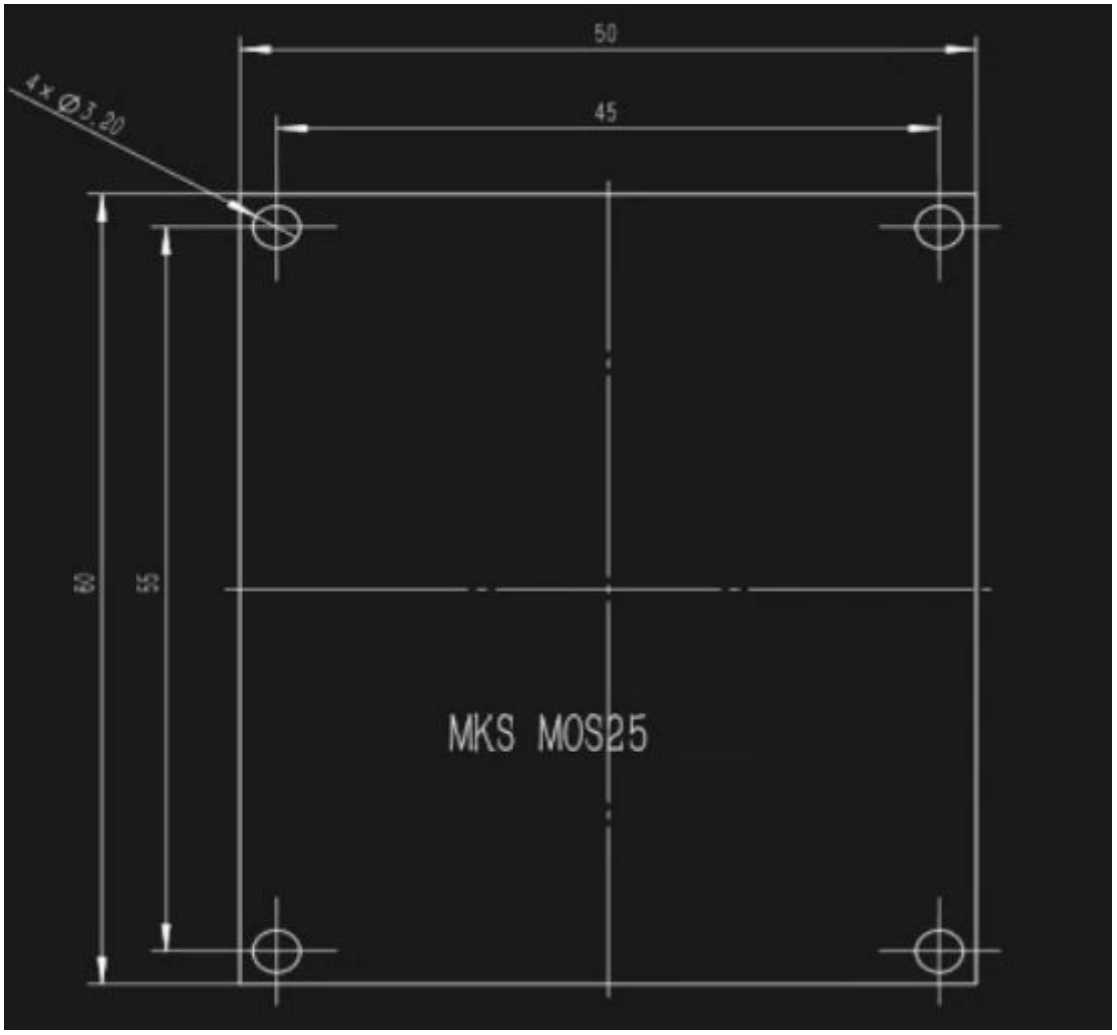
NO PULLUP

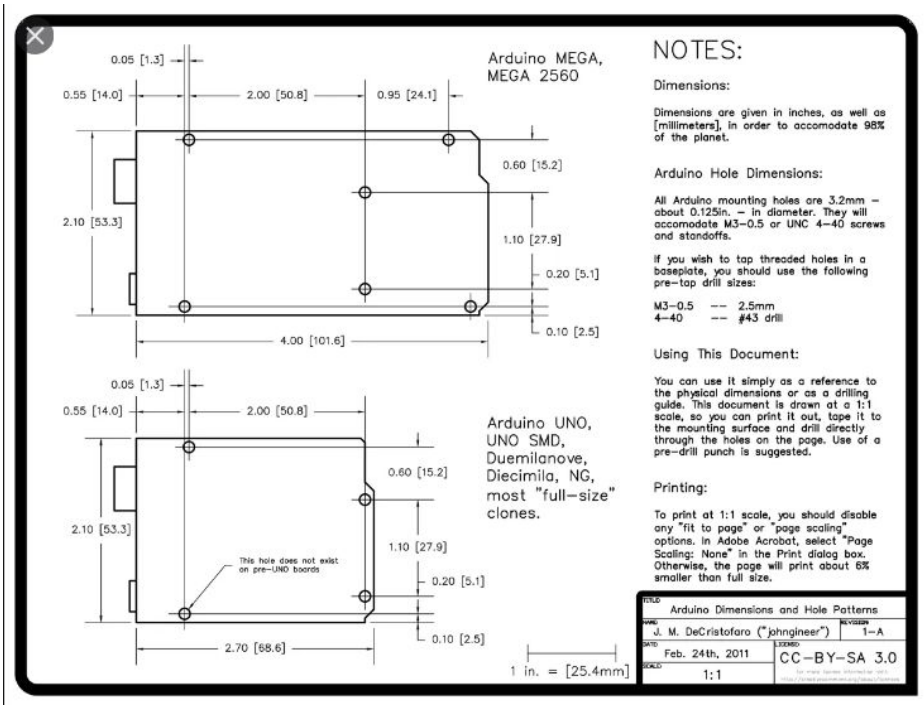
-

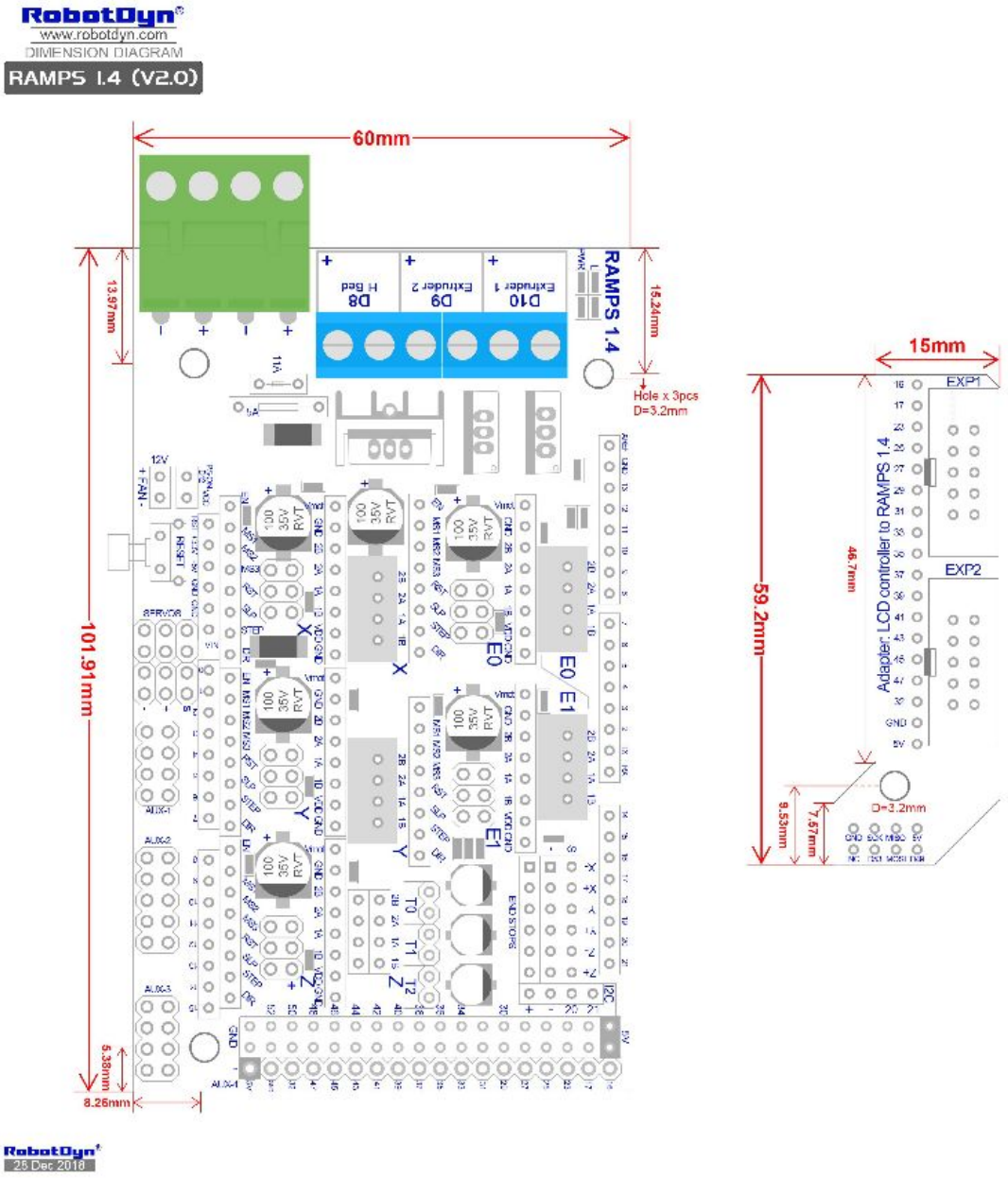
No hace falta usar dumpers o smoothers ya que en la práctica no hacen diferencia, usar mayor cantidad de micropasos.

Etapa 4

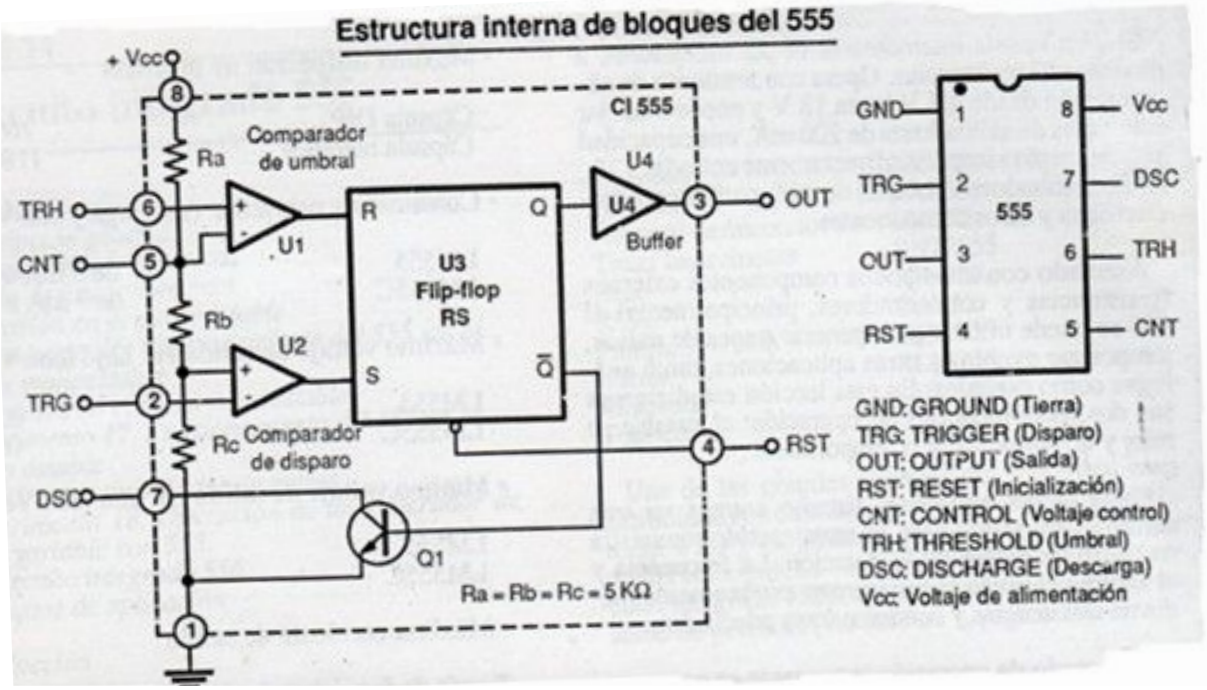
https://protosupplies.com/product/usb-type-b-female-to-2-54mm-header-breakout/


50
45
4 x Ø 3.20
60
55
MKS MOS25

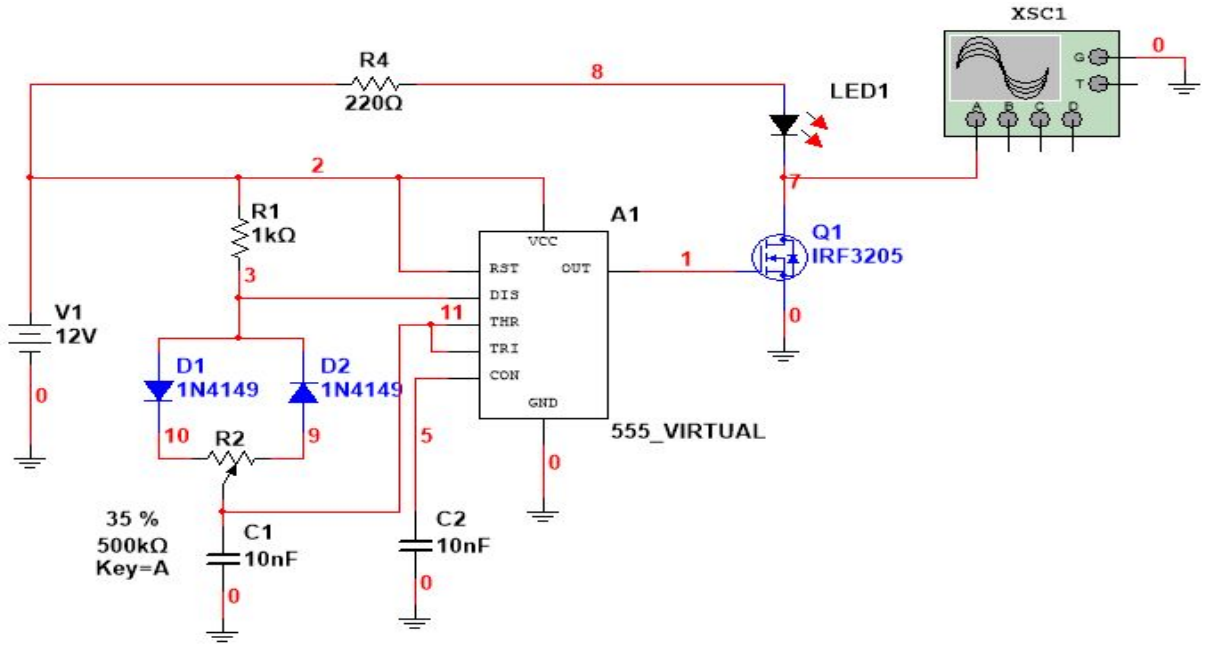

Arduino MEGA, MEGA 2560
0.05 [1.3]
0.55 [14.0]
2.00 [50.8]
0.95 [24.1]
0.60 [15.2]
2.10 [53.3]
1.10 [27.9]
0.20 [5.1]
0.10 [2.5]
4.00 [101.6]
Arduino UNO, UNO SMD, Duemilanove, Diecimila, NG, most "full-size" clones.
This hole does not exist on pre-UNO boards
2.70 [68.6]
1 in. = [25.4mm]
NOTES:
Dimensions:
Dimensions are given in inches, as well as [millimeters], in order to accomodate 98% of the planet.
Arduino Hole Dimensions:
All Arduino mounting holes are 3.2mm - about 0.125in. - in diameter. They will accomodate M3-0.5 or UNC 4-40 screws and standoffs.
If you wish to tap threaded holes in a baseplate, you should use the following pre-tap drill sizes:
M3-0.5 -- 2.5mm
4-40 -- #43 drill
Using This Document:
You can use it simply as a reference to the physical dimensions or as a drilling guide. This document is drawn at a 1:1 scale, so you can print it out, tape it to the mounting surface and drill directly through the holes on the page. Use of a pre-drill punch is suggested.
Printing:
To print at 1:1 scale, you should disable any "fit to page" or "page scaling" options. In Adobe Acrobat, select "Page Scaling: None" in the Print dialog box. Otherwise, the page will print about 6% smaller than full size.
TITLE: Arduino Dimensions and Hole Patterns
NAME: J. M. DeCristofaro ("johngineer")
REVISION: 1-A
DATED: Feb. 24th, 2011
LICENSED: CC-BY-SA 3.0
SCALE: 1:1



Circuito de Dimmer



Circuito Interno de LM555



Funcionamiento:

Por un lado, el circuito Integrado 555 (timer o compraador, ni idea) comprará (gracias a sus comparadores, obvio) la tensión de entrada de a tercios. Es decir, evaluará si en el pin Thr existe más ⅔ de la tensión de entrada, es decir, más de 8V. Y en el pin Tri evaluará si existe menos de ⅓, es decir, menos de 4V.

En un primer instante, el capacitor estará descargado y la corriente de carga circulará por un circuito RC compuesto por la resistencia de 1k, el valor resistivo del potenciómetro, el diodo D1 y, por su puesto, el Capacitor C1. Evidentemente, como el capacitor se encuentra

descargado, en el comparador U2 la tensión será mayor en el pin + que en él - y, por lo tanto, habrá un 1 a la salida del 555

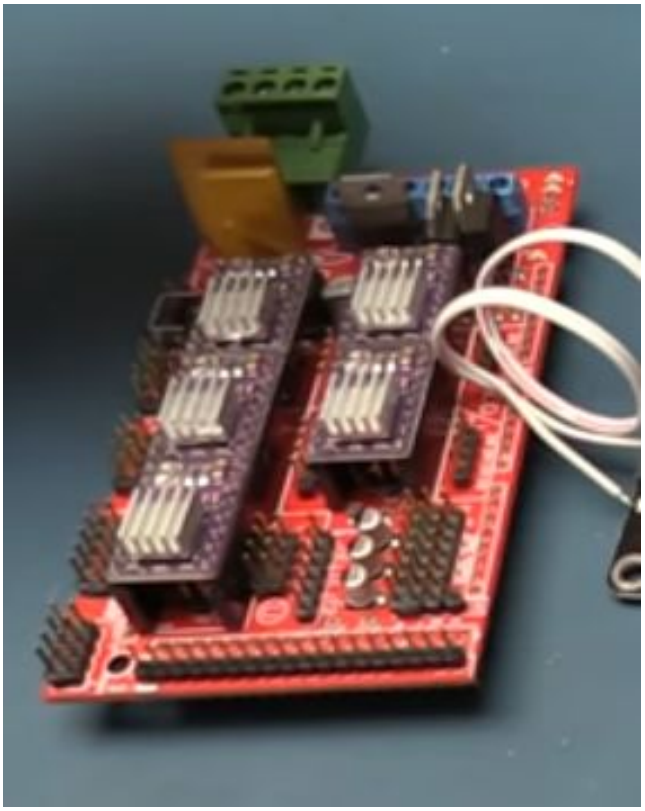
A medida que el capacitor se carga, la tensión de pin + del comparador U1 aumentará y será mayor a la tensión en el pin - del mismo. Esto generará que a la salida Q exista un 0 y, como consecuencia, la salida Q negada tenga un 1, lo que exitará la base del transistor de descarga interno del 555

EVIDENTEMENTE, LA BASE DEL TRANSISTOR DE DESCARGA DEBE TENER UNA RESISTENCIA LIMITADORA PQ SINO SE ROMPE TODO.

Entonces, dependiendo del valor resistivo que posea el potenciómetro, la constante de tiempo del capacitor tendrá un valor distinto, es decir, al variar el potenciómetro se varía el tiempo de carga y de descarga del capacitor. Como cada curva está asociada a un nivel alto y a un nivel bajo, dependiendo de la posición del potenciómetro modulamos el ancho de pulso.

El potenciómetro R2 nos sirve para poder ajustar el tiempo de duración del pulso, o, el ancho del pulso, debido a esto se obtiene una señal PWM (*Modulación por ancho de pulso*). El mosfet IRFZ24N funciona como una “llave On/Off” adecuada para manejar los niveles altos de corriente requeridos por la tira led.

Colocación de los disipadores de los driverso



BOOST STEP UP:
Todo micro:
https://www.todomicro.com.ar/electronica-tecnica/767-regulador-de-voltaje-step-up-hasta-83v-15a-1200w.html
Otra página pro:
https://www.microjpm.com/products/ad49812/

⇒

Designación [editar]

La designación estándar para una rosca UTS es un número indicando el diámetro nominal (mayor) de la rosca, seguido por el paso medido en roscas por pulgada. Para diámetros más pequeños que ¼ de pulgada, el diámetro se indica mediante un número entero definido en el estándar, para todos los otros diámetros, se da el número en pulgadas.

Este par de números está opcionalmente seguido de las letras UNC, UNF o UNEF o si la combinación del diámetro de paso es desde la serie "gruesa", "fina" o "extra fina" y podría también ser seguida por un grado de precisión.

Ejemplo: #6-32 UNC 2B (diámetro máximo: 0.1380 pulgadas, paso: 32 tpi)

Tornillos de Rosca Unificado - UNC, UNF y UNEF							
Diámetro Máximo (pulgada \ mm)		Densidad de Rosca (d, roscas por pulgadas) y pasos de rosca (p)					
		Gruesa (UNC)		Fina (UNF)		Extra fina (UNEF)	
		d (TPI)	p (pulgada \ mm)	d (TPI)	p (pulgada \ mm)	d (TPI)	p (pulgada \ mm)
#0	0.0600 \ 1.5240	None		80	0.012500 \ 0.3175	None	
#1	0.0730 \ 1.8542	64	0.015625 \ 0.3969	72	0.013888 \ 0.3528	None	
#2	0.0860 \ 2.1844	56	0.017857 \ 0.4536	64	0.015625 \ 0.3969	None	
#3	0.0990 \ 2.5146	48	0.020833 \ 0.5292	56	0.017857 \ 0.4536	None	
#4	0.1120 \ 2.8448	40	0.025000 \ 0.6350	48	0.020833 \ 0.5292	None	
#5	0.1250 \ 3.1750	40	0.025000 \ 0.6350	44	0.022727 \ 0.5773	None	
#6	0.1380 \ 3.5052	32	0.031250 \ 0.7938	40	0.025000 \ 0.6350	None	
#8	0.1640 \ 4.1656	32	0.031250 \ 0.7938	36	0.027778 \ 0.7056	None	
#10	0.1900 \ 4.8260	24	0.041667 \ 1.0583	32	0.031250 \ 0.7938	None	
#12	0.2160 \ 5.4864	24	0.041667 \ 1.0583	28	0.035714 \ 0.9071	32	0.031250 \ 0.7938
¼	0.2500 \ 6.3500	20	0.050000 \ 1.2700	28	0.035714 \ 0.9071	32	0.031250 \ 0.7938
5⁄16	0.3125 \ 7.9375	18	0.055556 \ 1.4111	24	0.041667 \ 1.0583	32	0.031250 \ 0.7938
⅜	0.3750 \ 9.5250	16	0.062500 \ 1.5875	24	0.041667 \ 1.0583	32	0.031250 \ 0.7938
7⁄16	0.4375 \ 11.1125	14	0.071428 \ 1.8143	20	0.050000 \ 1.2700	28	0.035714 \ 0.9071
½	0.5000 \ 12.7000	13	0.076923 \ 1.9538	20	0.050000 \ 1.2700	28	0.035714 \ 0.9071
9⁄16	0.5625 \ 14.2875	12	0.083333 \ 2.1167	18	0.055556 \ 1.4111	24	0.041667 \ 1.0583
⅝	0.6250 \ 15.8750	11	0.090909 \ 2.3091	18	0.055556 \ 1.4111	24	0.041667 \ 1.0583
¾	0.7500 \ 19.0500	10	0.100000 \ 2.5400	16	0.062500 \ 1.5875	20	0.050000 \ 1.2700
⅞	0.8750 \ 22.2250	9	0.111111 \ 2.8222	14	0.071428 \ 1.8143	20	0.050000 \ 1.2700
1	1.0000 \ 25.4000	8	0.125000 \ 3.1750	12	0.083333 \ 2.1167	20	0.050000 \ 1.2700

MARLIN PARA TORPES

1. Instrucciones:
   #define: Es una de las instrucciones principales. Le dice a marlin aquellos valores y variables que se encuentran definidos. Se pone generalmente la variable y luego el valor.
   #undef: #define pero negado, es decir, quita una variable previamente definida. Difiere con uncomment (//) en que esta instrucción sirve para decirle a marlin que esa variable ya no existe.
   #ifdef y ***#endif***: Es una combinación #define y #undef. El microprocesador solo lee las líneas que se encuentra dentro de este #ifdef. Tiene principio y fin. Solo se utiliza si la variable utilizada dentro de esta condición estuvo previamente definida con #define.

#ifndef y ***#endif***: A diferencia del anterior la condición se ejecuta si la variable dentro de ella **no** se encontraba definida previamente.

2. Velocidad del puerto serie:
   La velocidad que viene por defecto en Marlin es 250.000 baudios (comunicación más rápida y sin errores), sin embargo, algunas placas no lo soportan. En este caso lo ponemos en 115.200.
   #define BAUDRATE 250000

3. Cosas claves para nuestra aplicación. Todos lo siguiente se encuentra en la carpeta **Configuration.h**
   a) Tipo de placa controladora, en nuestro caso RAMPS 1.4.
   b) Debemos poner los datos tanto de autor (nombre de la empresa) como de fecha de compilación, de esta manera podremos saber información de la máquina y si la misma se encuentra actualizada con la última versión de compilación que hayamos creado y cargado. Esto lo haremos mediante las líneas:
   #define STRING_VERSION_CONFIG_H __DATE__ " " __TIME__ // build date and time. (una vez subido el Marlin al Arduino, se actualiza la fecha y hora a ese momento)
   #define STRING_CONFIG_H_AUTHOR "(none, default config)" // Who made the changes.
   En la última línea es necesario cambiar lo que se encuentra entre comillas (*None, default config*) con algún nombre o el nombre de la empresa.
   c) Debemos definir el puerto serie de nuestra computadora que se comunicará con Arduino, por default es el puerto 0. Este valor lo debemos cambiar segun el puerto serie COM de la computadora al cual conectaremos el Arduino o la propia máquina.
   #define SERIAL_PORT 0
   d) Definir si utilizamos las resistencias pullup para los finales de carrera que tienen algunas placas controladoras como arduino. Lo recomendable es activarlas en caso de que tengamos como final de carrera un simple microswitch sin filtro ni nada y desactivarlas cuando tengamos un final de carrera con filtros y demás (los que vienen ya montados en un PCB). La línea de código es:
   #define ENDSTOPPULLUPS // Comment this out (using/ at the start of the line) to disable the EndStop pullup resistors.
   En caso de que tengamos que activar solo algunas Pullups y no todas, comentamos la línea previamente citada y descomentamos dentro de la siguiente condición negada las que queramos.

```
#ifndef ENDSTOPPULLUPS
// fine EndStop settings: Individual pullups. will be ignored if ENDSTOPPULLUPS is defined
// #define ENDSTOPPULLUP_XMAX
// #define ENDSTOPPULLUP_YMAX
```

```
//      #define     ENDSTOPPULLUP_ZMAX
//      #define     ENDSTOPPULLUP_XMIN
//      #define     ENDSTOPPULLUP_YMIN
//      #define     ENDSTOPPULLUP_ZMIN
#endif
```

e) Chequear si los finales de carrera envían un 1 lógico al ser pulsados, caso contrario, se deberá invertir la lógica para lograr esto. Para poder invertir la los estados normales en Marlin tendremos que poner en "*false*" o "*true*"las siguientes líneas.

```
const bool X_MIN_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop.
const bool Y_MIN_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop.
const bool Z_MIN_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop.
const bool X_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop.
const bool Y_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop.
const bool Z_MAX_ENDSTOP_INVERTING = true; // set to true to invert the logic of the EndStop
```

f) Chequear que el movimiento de los ejes tenga dirección correcta. En caso de que la dirección de movimiento de alguno de los ejes sea incorrecta tendremos que invertir el estado de las siguientes líneas.

```
#define INVERT_X_DIR true // for Mendel set to false, for Orca set to true
#define INVERT_Y_DIR false // for Mendel set to true, for Orca set to false
#define INVERT_Z_DIR true // for Mendel set to false, for Orca set to true
```

Ejemplo: El eje X al hacer homing se mueve en dirección contraria y en Marlin estoy viendo que su estado es "*True*". Seteo este estado a "*False*" y listo.

e) Fijar la velocidad de desplazamiento al hacer homing con la línea:

```
#define HOMING_FEEDRATE {50*60, 50*60, 4*60, 0} // set the homing speeds (mm/min)
```

El orden de los valores son #define HOMING_FEEDRATE (X, Y, Z, Extrusor) y las unidades están en mm/s, pero marlin multiplica este valor por 60 para pasarlo a mm/min. Es decir, supongamos que queremos una velocidad de 3000 mm/min (una locura) ponemos 50*60. El autor recomienda velocidades de 30*60 en los ejes X y Y y 2*60 en el eje Z.

g) En el caso de ser necesario, se puede fijar una velocidad máxima la cual no podrá ser excedida incluso desde el control de la máquina. La línea correspondiente a esta función es:
#define DEFAULT_MAX_FEEDRATE {500, 500, 5, 25}
Las unidades se encuentran mm/s y corresponden al eje X, Y y Z respectivamente.
h) Por último, debemos fijarnos las aceleraciones máximas de los ejes por defecto cargadas en Marlin. Para evitar vibraciones y oscilaciones debemos tener las aceleraciones de los motores paso a paso de los ejes con bajas aceleraciones. Para modificar el valor de la aceleración de los ejes se modifican los valores de la siguiente variable definida:
#define DEFAULT_MAX_ACCELERATION {9000,9000,100,10000}
#define DEFAULT_ACCELERATION 3000
La primera línea tiene que ver con las aceleraciones máximas posibles que pueden alcanzar los ejes X, Y y Z respectivamente en mm/s^2 sin importar el valor que se incluya en el G-Code. Mientras que la segunda tiene que ver con la aceleración por defecto que tienen los motores paso a paso en caso de que no se especifique dentro del G-Code, también en mm/s^2.

4. **LCD (Full Graphic Smart Controller)**
   a) Lo primero que debemos establecer es el lenguaje con el cual se mostrarán los mensajes en la pantalla, para esto usamos la siguiente línea de código que se encuentra en el archivo **lenguaje.h**:
   //#define LANGUAGE_INCLUDE GENERATE_LANGUAGE_INCLUDE(en)
   Por defecto se encuentra en inglés (en), lo cambiaremos a español (es).

   b) Como nuestra pantalla tiene la posibilidad de insertar una tarjeta SD para ejecutar nuestro G-Code, deberemos activar la SD desde el archivo **Configuration.h** de Marlin.
   //#define SDSUPPORT // Enable SD Card Support in Hardware Console

   Si tenemos problemas con la tarjeta a la hora de iniciar nuestro código tenemos como posible solución poner el lector de tarjeta SD en modo lento, para esto descomentamos o definimos la línea:
   //#define SDSLOW // Use slower SD transfer mode (not normally needed -- uncomment if you're getting volume init error).
   A su vez, Marlin tiene la opción de poder chequear errores de envío y recepción de datos con la tarjeta SD constantemente mediante “CRC check”. Por defecto se encuentra desactivado, pero es recomendable activarlo. La línea a comentar es:

//#define SD_CHECK_AND_RETRY // Use CRC checks and retries on the SD communication.

c) Debido a que nuestra pantalla controladora posee un encoder rotativo (La perilla para seleccionar entre menús, ajuste de parámetro, etc.) debemos configurar la cantidad de pasos que debemos rotar para poder navegar entre los menús con las siguientes dos variables:
//#define ENCODER_PULSES_PER_STEP 1 // Increase if you have a high resolution encoder
//#define ENCODER_STEPS_PER_MENU_ITEM 5 // Set according to ENCODER_PULSES_PER_STEP or your liking

d) Lo siguiente que debemos hacer es descomentar la variable definida que corresponde a nuestra pantalla, esta línea a descomentar es:
//#define REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER
Cabe aclarar que para utilizar este panel tendremos que descargar e incluir la librería **U8glib**, la misma se puede descargar de **GitHub.**

e) Por último, desde Marlin podremos ajustar la frecuencia y la duración del Buzzer que posee la pantalla como feedback al presionar el encoder, esto lo haremos modificando el valor de las variables definidas de la siguiente línea:
//#define LCD_FEEDBACK_FREQUENCY_HZ 1000 // this is the tone frequency the buzzer plays when on UI feedback. ie Screen Click.
//#define LCD_FEEDBACK_FREQUENCY_DURATION_MS 100 // the duration the buzzer plays the UI feedback sound. ie Screen Click.

5. Tenemos la posibilidad de variar la frecuencia de las salidas D8, D9 y D10. Marlin tiene cargado por defecto frecuencias PWM bajas para que no hayan interferencias con el hardware, sin embargo, podremos cambiar el valor de dicha frecuencia modificando el siguiente valor:
#define SOFT_PWM_SCALE 0
Al reemplazar el valor “0” con un “1”, la frecuencia PWM se duplicará. Hay que tener en cuenta que al momento de aumentar este valor, la resolución de la frecuencia (que por defecto es 128) disminuye. Esta función nos servirá para el ajuste del ***Circuito de Seguridad*** y la misma se encuentra en la carpeta de Marlin **Configuration.h**

6. Podemos hacer un homing ultra rápido. Esto quiere decir que en vez de hacer homing en serie de los ejes X y Y (primero uno y después el otro), los ejes se desplazarán diagonalmente de manera tal de hacer homing de ambos ejes al mismo tiempo. Esto lo logramos descomentando la línea:
//#define QUICK_HOME //if this is defined, if both x and y are to be homed, a diagonal move will be performed initially.
Esta función se encuentra en la carpeta **Configuration_adv.h**

Comandos G

G0 y G1: Marlin no distingue entre uno y otro, para especificar la velocidad de avance del movimiento se utiliza la letra F. Se utilizan para dar la instrucción a los ejes de moverse una longitud específica. Ejemplo:
G1 X12 F2000
Con esto estamos diciendo que el eje X se mueve de manera lineal 12 milímetros a una velocidad de 2000 mm/min. Generalmente se utiliza G0 para diferenciar una operación de mecanizado con una operación en el aire.

G2 y G3: Realiza movimiento de arco, siendo G2 para arco en dirección horaria y G3 en dirección antihoraria. Se especifica el punto final en el eje X, el punto final en el eje Y, con la I se especifica centro de la circunferencia, en el eje X y con la J lo mismo que con la I pero en el eje Y. Sin embargo, este comando no es utilizado para la fabricación de PCB's. Ejemplo:
G2 X90.6 Y13.8 I5 J10
Con esto estamos diciendo que haremos un arco con dirección horaria, con punto final en X60.6 y en Y13.8, también le estamos diciendo que el centro de este arco se encuentra a 5 mm de la distancia actual en el eje X y a 10 mm de la distancia actual en el eje Y.

G4: Le da la instrucción a la máquina de hacer un retardo por un determinado tiempo, sin embargo no funciona como delay ya que no se para completamente, sino que sigue ejecutando otras instrucciones al mismo tiempo. Se puede expresar tanto en segundos como en milisegundos. Ejemplo:
G4 S1 (le dice a la máquina que espere 1 **segundo**)
G4 P1000 (le dice a la máquina que espere 1000 **milisegundos**, lo que es equivalente a 1 segundo)

G21: Con este comando le decimos a Marlin que utilizaremos el sistema de unidades métrico, es decir, que los valores que utilizamos se encuentran todos en milímetros. Ejemplo:
G21
G1 X12 F200

G20: Tiene la función de decirle a Marlin que los valores que se encuentran en el código se encuentran determinados con el sistema de unidades anglosajón, es decir, los valores se encuentran en pulgadas. Ejemplo:
G20
G1 X12 F200

G28: Este comando sirve para hacer homing. Se puede especificar en qué eje hacer homing o no se especifica nada y se interpreta como un homing general para todos los ejes. Ejemplo:
G28 (no especifico nada, entonces todos los ejes hacen homing)
G28 X Y (especifico que el homing solo se lleve a cabo en los ejes X e Y)

G90: Este comando indica a Marlin que todos los valores que se toman de los ejes son absolutos, es decir, que están todos referidos al punto de origen X0 Y0 Z0. Para el enrutamiento de las

PCB's generalmente se utiliza este método. Ejemplo:
G90
G1 X12 F1200

G91: Con esto le decimos a Marlin que todos los valores de los ejes son incrementales, es decir se mueven dicho valor tomando como referencia la posición actual y no la de origen como lo hacíamos en el G90. Ejemplo:
G91
G1 X12 F1200

G92: Es tal vez el comando más importante de todos para nuestra aplicación, con el le estaremos cambiando las coordenadas reales en las cuales se encuentra la herramienta. En caso de no especificar la nueva coordenada, Marlin interpretará que esa posición de la herramienta es el nuevo origen de la máquina. Ejemplo:
G92 X12 Y30 Z2 (especificamos coordenadas, por lo tanto la posición actual de la herramienta será X12 Y30 Z2).
G92 (no especificamos ninguna coordenada, es decir, ahora la posición actual de la herramienta es el origen).

Comandos M

M3: Le dice a Marlin que prenda el motor
M5: Le dice a Marlin que apague el motor
M114: Muestra la posición actual de la herramienta. Ejemplo:
G0 X25 Y14
M114

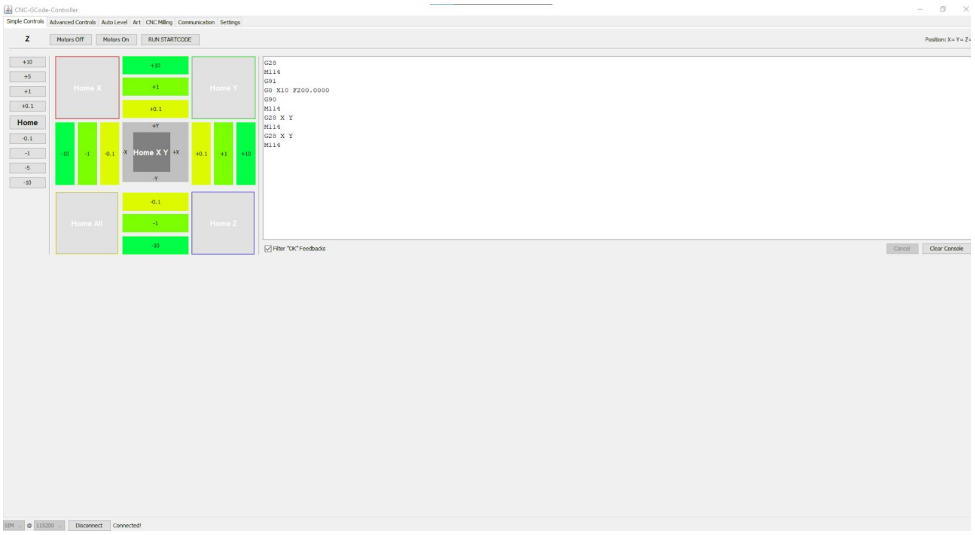
CNC GCode Controller

A continuación se llevará a cabo una serie de instrucciones de como utilizar el software "CNC GCode Controller".

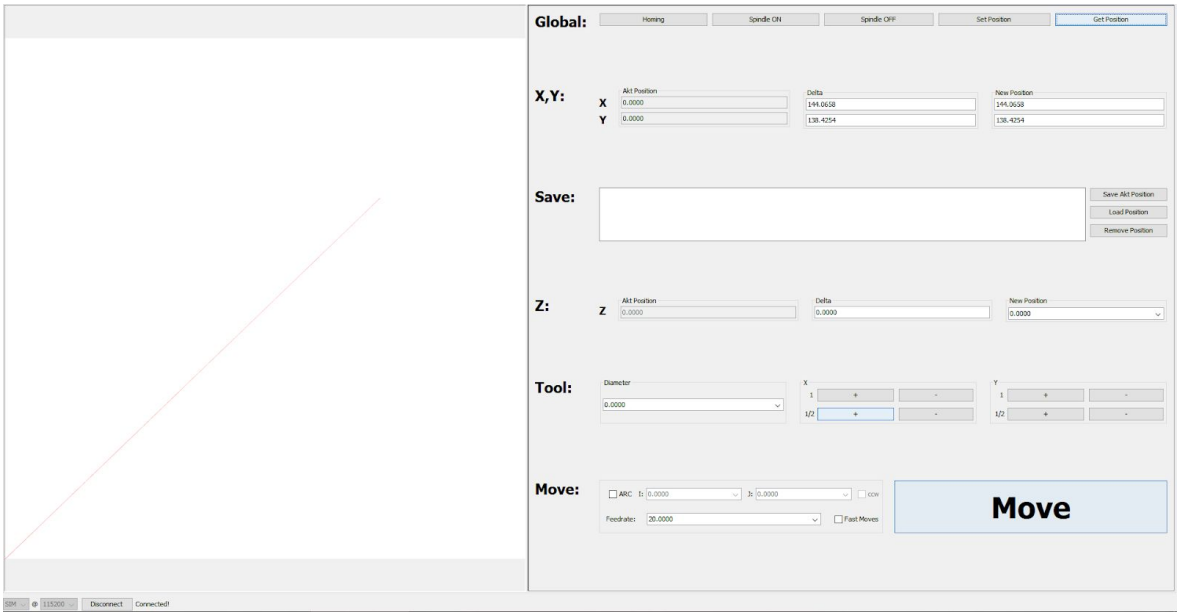
Esta aplicación nos servirá como medio para poder comunicarnos con nuestro Router CNC. Básicamente, con el software estaremos enviando comandos y coordenadas a la máquina para que esta lleve a cabo el enrutamiento, el agujereado y el corte de nuestra PCB.

Su interfaz se divide en siete pestañas, de las cuales seis serán de suma importancia. Las pestañas son:

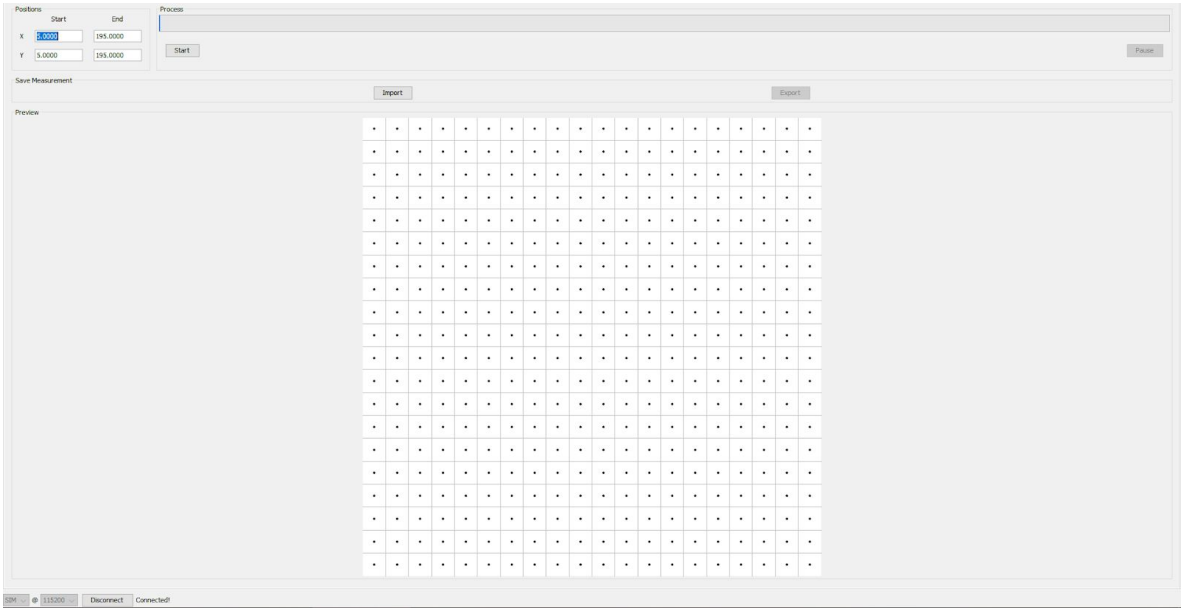
"Simple Controls": Como el nombre lo indica en inglés, sirve para realizar movimientos básicos de la máquina, como mover la herramienta en los tres ejes, realizar el homing (tanto individual para cada eje como general para los tres), prender y apagar el husillo e iniciar nuestro G-Code inicial.



“Advanced Controls”: En esta pestaña se encuentran los controles avanzados de la máquina, como por ejemplo el avance de la herramienta a una posición específica, prendido y apagado de husillo, la posición actual de la herramienta, hacer homing general, realizar movimientos complejos de arcos y guardar posiciones previas de la herramienta para poder eventualmente volver a cargarlas. Además de este amplio abanico de controles manuales, también tenemos un visualizador del recorrido de la herramienta en la parte izquierda.

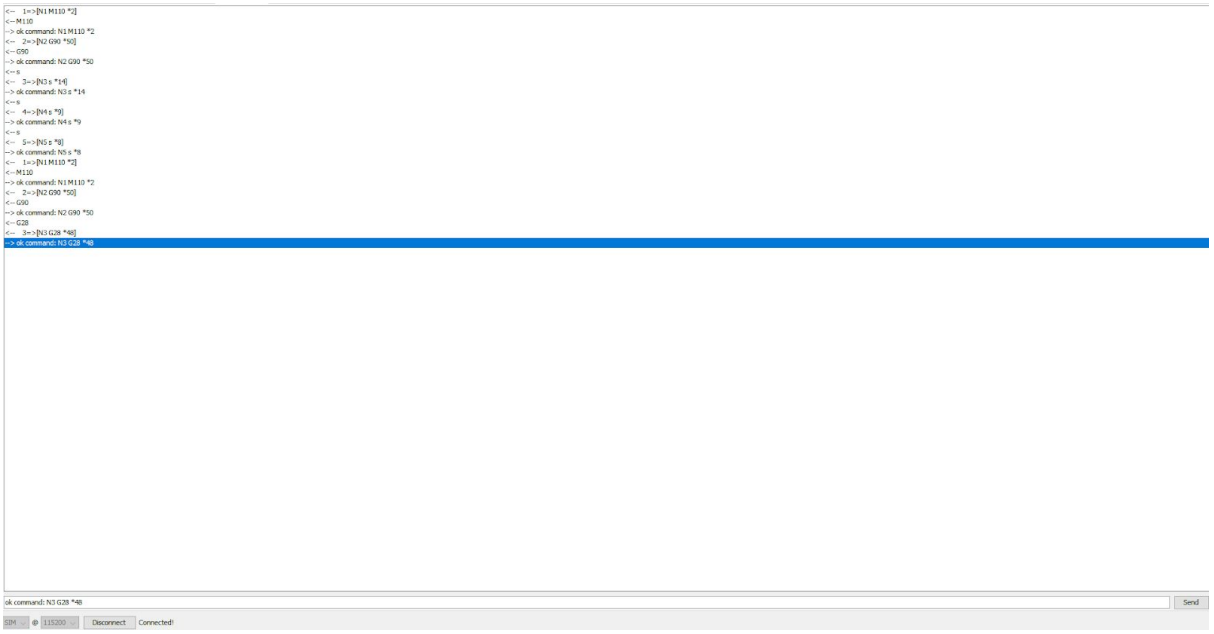


“Auto Level”: Con esta pestaña controlaremos y configuraremos todo lo referido con la autonivelación. Más adelante haremos hincapié en cómo realizar este procedimiento.

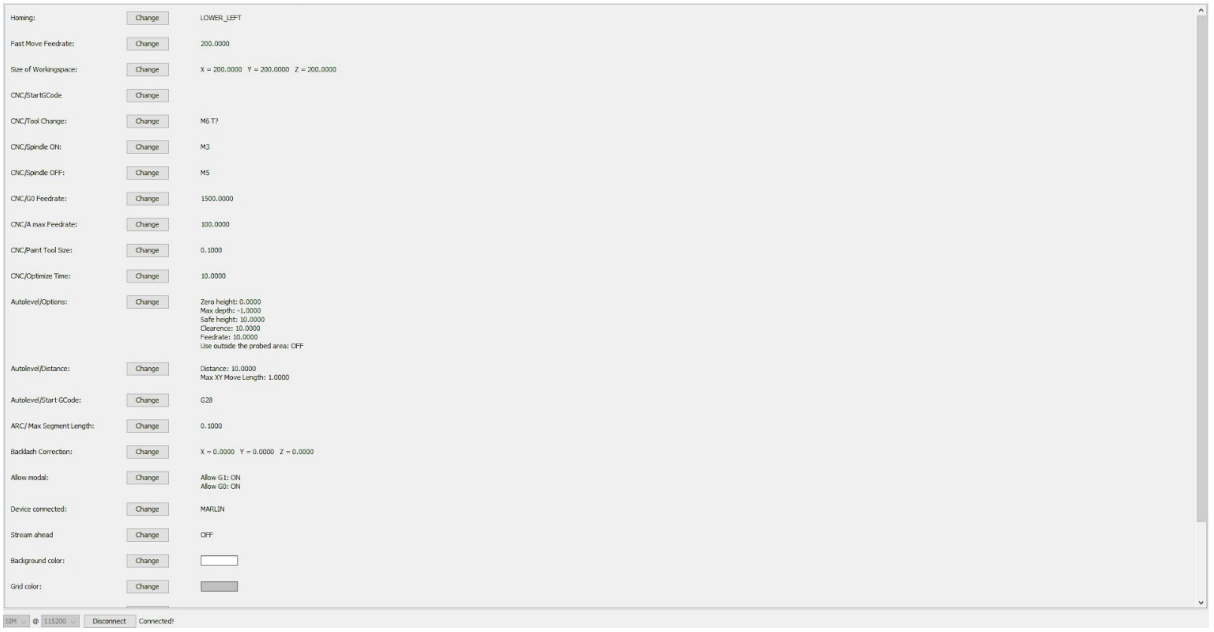


“CNC Milling”: A través de esta pestaña cargaremos nuestro G-Code, el cual puede ser optimizado. A su vez en la parte izquierda se nos mostrará una cuadrícula con un trazado de nuestro G-Code, podremos cambiar entre diferentes capas que corresponden a cada nivel del eje Z para poder ver el recorrido que hará la herramienta o cualquier otra información importante que se encuentre en niveles de altura diferentes. Este trazado cambiará de color a rojo cuando la herramienta ya lo haya terminado en la vida real. Otro dato importante es que podremos visualizar nuestro G-Code en la parte de derecha inferior, podremos editar línea por línea de nuestro código y además nos avisará si hay algún tipo de error en ellas poniéndose en color rojo o alguna advertencia en color amarillo.

“Communication”: Aquí veremos todos los comandos que son enviados a la máquina. También nos permitirá enviar comandos manualmente.



“Settings”: En este apartado configuraremos todo el software basándonos en las capacidades de nuestra máquina.



Pasos a Realizar para Utilizar el Software

1) Ir a la pestaña **“Settings”** y cambiar en base a nuestras necesidades la esquina los apartados:
   Homing: Cambiaremos donde se ubicará el homing total. Este parámetro será seleccionado en base al punto de origen que tomamos al momento de realizar la PCB que se requiera mecanizar
   Size of Workingspace: Cambiaremos el valor de esta variable en base al recorrido efectivo total que puede realizar nuestra herramienta en cada eje. Esta opción sirve como seguridad e indica al software el recorrido máximo capaz de realizar nuestra herramienta para que la misma no se choque eventualmente debido a que introducimos una coordenada que sobrepasa

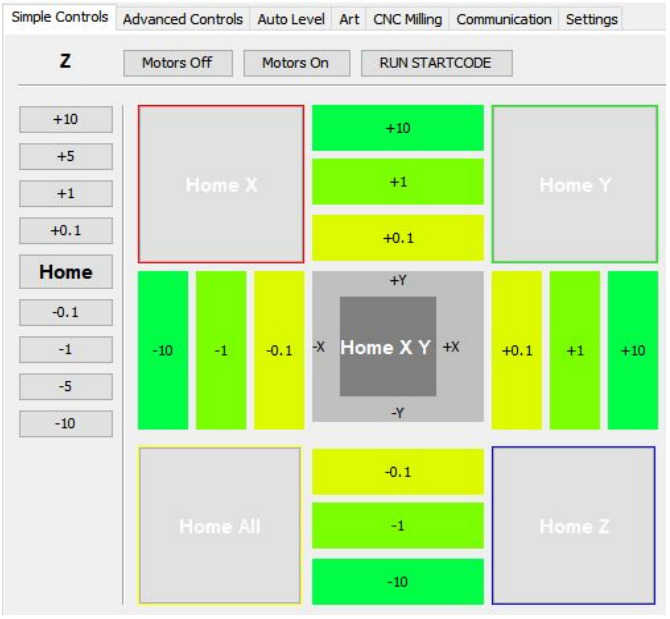
este límite. En otras palabras, es un final de carreras virtual.
CNC/StartGCode: Acá introduciremos el código que se ejecutará antes de comenzar el propio código de nuestra PCB. Generalmente no es necesario, pero existe la posibilidad.
CNC/Spindle ON: Establecemos con que comando prendemos el motor de husillo, el correcto como se mencionó anteriormente es M3.
CNC/Spindle OFF: Establecemos con que comando apagaremos el motor de husillo, el correcto como se mencionó anteriormente es M5.
Autolevel/Options: Con esta opción cambiaremos la altura cero, el máximo recorrido en el eje Z, la altura de retirada, la altura de finalizado y la velocidad de la herramienta durante la ejecución de la autonivelación.
Autolevel/Distance: Definiremos la distancia entre pruebas para la autonivelación.
Autolevel/StartGcode: Configuraremos el código inicial antes de comenzar la autonivelación al momento de iniciar este proceso. Por defecto está puesto el comando G28, es decir, el homing general.
Backlash Correction: Esta opción se completa de forma empírica, puesto que es necesario probar el movimiento de los ejes de la máquina. Nos permite corregir el backlash que se genera en la rosca del tornillo roscado.
Device Connected: Con esto elegiremos el firmware que estamos utilizando, en nuestro caso como ya sabemos utilizamos Marlin, por lo tanto, elegiremos esta opción.
Fast Move Feedrate: Configuraremos la velocidad de avance de los movimientos rápidos cargados manualmente mediante la pestaña **"Advanced Controls"**.

Configuraciones no utilizadas:
CNC/Tool Change: Podemos seleccionar el comando con el cual efectuaremos el cambio de herramienta. En nuestro caso no es necesario ya que utilizaremos una misma herramienta para todas las operaciones de mecanizado.
CNC/G0 Feedrate: Configuración de la velocidad de avance con el comando G0. No utilizaremos esta función ya que Marlin no hace distinción entre G0 y G1, sino que la velocidad de avance para cada movimiento se especificará en la línea del movimiento mediante la letra F.
CNC/Paint Tool Size: Con esta opción definiremos el diametro de nuestra herramienta. No es necesario especificarla ya que es un dato útil solo para el estado de simulación que posee el software.

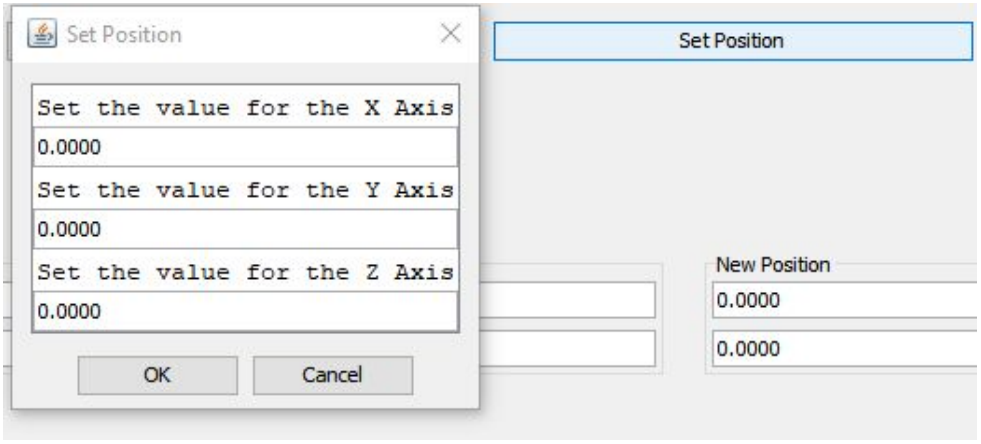
2) Luego de configurar la aplicación y una vez conectada la máquina con nuestra computadora, estableceremos en la aplicación el puerto de la compu en el cual se encuentra conectada la máquina, establecemos los baudios (ya especificados anteriormente) y presionaremos el botón **"connect"**. En la siguiente imagen no es posible conectar a ningún puerto ya que no se encuentra conectada la máquina.

3) Ahora que la comunicación de nosotros con la máquina es real, haremos el homing de todos los ejes y moveremos la herramienta manualmente mediante la pestaña de **"Simple Controls"**

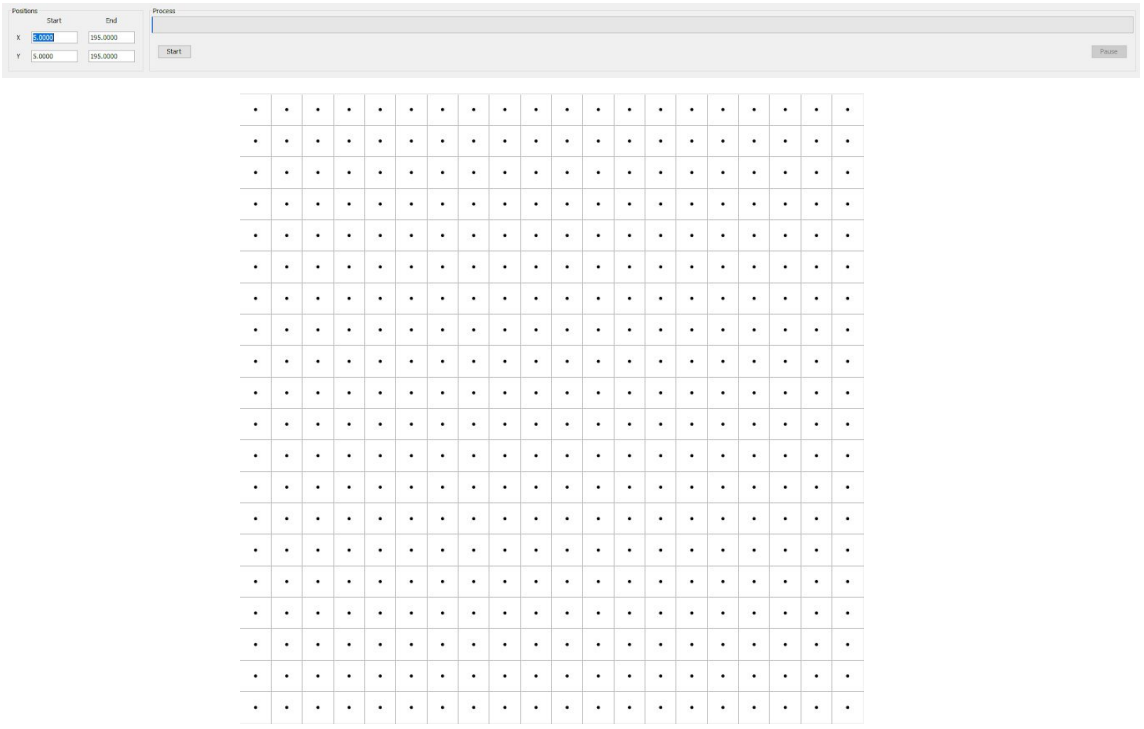
hasta alguna esquina de nuestra placa de cobre virgen previamente instalada en la mesa de la máquina.



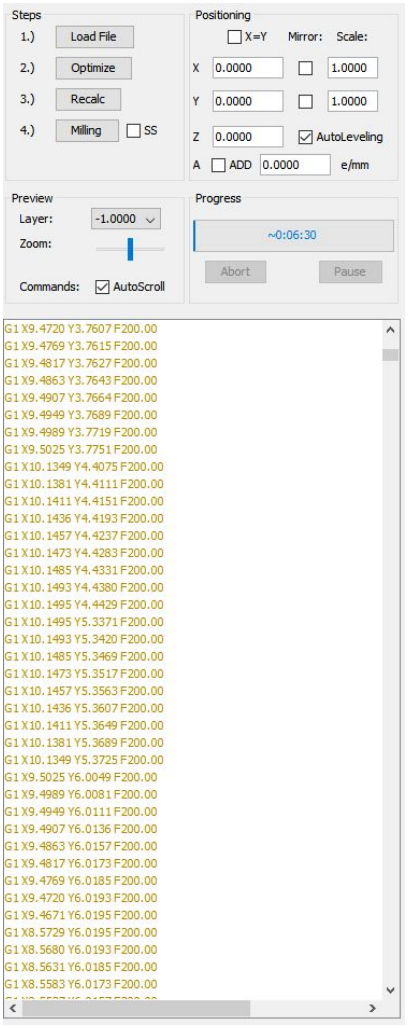
4) Una vez que la herramienta se encuentra aproximadamente sobre la esquina correspondiente de la placa, iremos a la pestaña **"Advanced Controls"** de la aplicación y haremos click sobre el botón **"Set Position"**, una vez hecho se desplegará una ventana que nos solicita ingresar el valor de esta nueva posición, el objetivo de hacer esto es establecer la esquina de la placa de cobre como el nuevo origen de la máquina, por lo tanto, pondremos estos valores en 0 y continuaremos presionando "Ok".



5) Llega el momento de realizar la autonivelación, para esto seleccionaremos la pestaña de **"Auto Level"**, en el apartado de "Position" seleccionaremos las medidas de nuestra placa de cobre o de la parte de la misma utilizada para el mecanizado. Se mostrará una grilla con los puntos en donde la herramienta bajará y tocará la placa. Una vez ingresado estos valores haremos click en "Start" y veremos que la herramienta empezará a moverse haciendo contacto múltiples veces con la placa.



6) Con el proceso de Auto nivelación ya realizado, procederemos a el proceso de mecanizado de nuestra placa virgen. Para esto iremos a la pestaña **"CNC Milling"** y cargaremos nuestro archivo contenedor del G-Code haciendo click en el botón "Load File" , veremos que se muestra el trazado de nuestra PCB. Luego haremos click en "Optimize", una vez optimizado se mostrará un cartel que nos muestra el tiempo ahorrado, este tiempo suele ser de unos segundos para aplicaciones simples, oprimimos "Recalc" para actualizar el G-Code y tildamos la casilla de **"Autoleveling"**. Finalmente clickeamos el botón "Milling" para comenzar el mecanizado.


Steps
1.) Load File
2.) Optimize
3.) Recalc
4.) Milling SS
Positioning
X=Y Mirror: Scale:
X 0.0000 1.0000
Y 0.0000 1.0000
Z 0.0000 AutoLeveling
A ADD 0.0000 e/mm
Preview
Layer: -1.0000
Zoom:
Commands: AutoScroll
Progress
~0:06:30
Abort Pause
G1 X9.4720 Y3.7607 F200.00
G1 X9.4769 Y3.7615 F200.00
G1 X9.4817 Y3.7627 F200.00
G1 X9.4863 Y3.7643 F200.00
G1 X9.4907 Y3.7664 F200.00
G1 X9.4949 Y3.7689 F200.00
G1 X9.4989 Y3.7719 F200.00
G1 X9.5025 Y3.7751 F200.00
G1 X10.1349 Y4.4075 F200.00
G1 X10.1381 Y4.4111 F200.00
G1 X10.1411 Y4.4151 F200.00
G1 X10.1436 Y4.4193 F200.00
G1 X10.1457 Y4.4237 F200.00
G1 X10.1473 Y4.4283 F200.00
G1 X10.1485 Y4.4331 F200.00
G1 X10.1493 Y4.4380 F200.00
G1 X10.1495 Y4.4429 F200.00
G1 X10.1495 Y5.3371 F200.00
G1 X10.1493 Y5.3420 F200.00
G1 X10.1485 Y5.3469 F200.00
G1 X10.1473 Y5.3517 F200.00
G1 X10.1457 Y5.3563 F200.00
G1 X10.1436 Y5.3607 F200.00
G1 X10.1411 Y5.3649 F200.00
G1 X10.1381 Y5.3689 F200.00
G1 X10.1349 Y5.3725 F200.00
G1 X9.5025 Y6.0049 F200.00
G1 X9.4989 Y6.0081 F200.00
G1 X9.4949 Y6.0111 F200.00
G1 X9.4907 Y6.0136 F200.00
G1 X9.4863 Y6.0157 F200.00
G1 X9.4817 Y6.0173 F200.00
G1 X9.4769 Y6.0185 F200.00
G1 X9.4720 Y6.0193 F200.00
G1 X9.4671 Y6.0195 F200.00
G1 X8.5729 Y6.0195 F200.00
G1 X8.5680 Y6.0193 F200.00
G1 X8.5631 Y6.0185 F200.00
G1 X8.5583 Y6.0173 F200.00