

UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA
CARRERA DE ESPECIALIZACIÓN EN SISTEMAS
EMBEBIDOS



MEMORIA DEL TRABAJO FINAL

Controlador de maquina CNC de 3 ejes

Autor:
Pablo Slavkin

Director:
Ing. Juan Manuel Cruz

Jurados:
Esp. Ing. Eric Pernia
Ing. Danilo Zechin
Dr. Ing. Pablo Gómez

Este trabajo fue realizado en las Ciudad Autónoma de Buenos Aires, entre enero de 2018 y diciembre de 2018 .

Resumen

Esta memoria describe el desarrollo e implementación de un controlador para una máquina CNC de 3 ejes. Este trabajo fue requerido por Wolfcut S.A. que le servirá de base en la implementación de lectura de fiduciales, cambio automático de herramientas y otros desafíos técnicos.

Se utilizó el sistema operativo FreeRTOS, un cortex M4 como controlador, el stack lwIP para la comunicación Ethernet, un driver de motores controlado por SPI, una UART como interfaz de debug, control de versiones con git, pruebas unitarias para componentes críticos, técnicas multitareas en el software de control y Kicad en el diseño de PCB de un driver de motores.

Agradecimientos

En primer lugar deseo expresar mi agradecimiento al director de este trabajo, el experimentado Ing. Juan Manuel Cruz por haber dedicado parte de su valioso tiempo y estar disponible para atender mis inquietudes.

Por último, deseo agradecer al Dr. Ing. Pablo Gómez, al Ing. Danilo Zechin y al Esp. Ing. Eric Pernia por aceptar ser jurados y conceder parte de su tiempo para analizarlo y corregirlo.

Índice general

Resumen	III
1. Introducción General	1
1.1. Introducción	1
1.2. Requerimientos	3
1.3. Diagrama de actividades (AON)	3
1.4. Objetivos y alcances	3
2. Introducción Específica	5
2.1. Diagrama en bloques de la máquina CNC	5
2.2. Generación de archivos GCode	6
3. Diseño e Implementación	9
3.1. Plataforma embebida	9
3.2. Driver de motores	10
3.2.1. Sincronización de relojes	10
3.3. Desarrollo de hardware	11
3.4. Arquitectura del firmware	12
3.4.1. Protocolo SPI	15
3.5. Software de control	16
4. Ensayos y Resultados	19
4.1. Pruebas funcionales del hardware	19
4.1.1. Pruebas de recorrido	19
4.1.2. Pruebas de velocidad	19
4.1.3. Pruebas de aceleración	21
Aceleración insuficiente	22
Aceleración suficiente	23
4.2. Pruebas en prototipo funcional	23
5. Conclusiones	27
5.1. Conclusiones generales	27
5.2. Próximos pasos	27
A. GCode utilizados en los ensayos	29
B. Esquemáticos	31
B.1. Esquemático de PCB de driver de motores	31
Bibliografía	35

Índice de figuras

1.1. Máquina CNC fabricada por Wolfcut.	1
1.2. Diagrama en bloques de una máquina CNC controlada por PC. Se destaca como el bloque nivelación de mesa requiere al agregado de otros bloques intermedios para integrarse a la arquitectura básica de la máquina.	2
1.3. Diagrama en bloques de una máquina CNC controlada por un sistema embebido	2
1.4. Diagrama de actividades AON elaborado previamente al inicio del desarrollo del trabajo. Se comprobó su utilidad al completar los objetivos a tiempo.	4
2.1. Diagrama de bloques de la máquina CNC. En recuadro amarillo los bloques implementados en este trabajo.	6
2.2. Captura de pantalla de RhinoCam. Se muestra como el resultado de las operaciones de maquinado dan por resultado el icono de Openhard con una media esfera en su centro.	7
3.1. Plataforma de desarrollo TIVA C TM4C1294 (imagen tomada de Texas Instruments, http://www.ti.com/tool/sw-ek-tm4c1294xl)	9
3.2. Chip de control de motor (izq.) y placa de desarrollo utilizada (der.) (imágenes tomada de [1])	10
3.3. Conexión SPI encadenada. La salida de unos de los drivers es la entrada del siguiente, y el último devuelve su salida como entrada del primero cerrando el anillo	11
3.4. Conexión de CLK encadenada. El primer driver genera el reloj de referencia, el resto toma la señal y la regenera para el próximo	11
3.5. Vista 3D de la placa de control de motores desarrollada en Kicad	12
3.6. Placa de control de motores doble faz fabricada como prototipo	12
3.7. Flujo de datos relacionados al procesamiento de los comandos GCodes. Las sentencias son inicialmente almacenadas en una cola que luego son preprocesadas para adelantar el cálculo matemático. El resultado es enviado por SPI, cuando el comando previo ha concluido.	13
3.8. Ejemplo de envío de un comando a 3 dispositivos encadenados (imagen tomada de [11])	15
3.9. Pantalla principal CuiPap. Trazado de eje X-Y en tiempo real del recorrido (recuadro XY). Posición del eje Z (recuadro Z). Líneas de sentencia GCode por enviar (recuadro Sender). Información de debug (recuadro Serial).	17
3.10. Pantalla derivada de CuiPap. Trazado de los ejes X, Y y Z en tiempo real a mitad de un trabajo de maquinado	17

4.1. Comparación entre el recorrido teórico del GCode (rojo con marcas cuadradas) y el real (azul con marcas 'x'). Aunque parecen coincidir divergen en ciertas zonas mínimamente debido al error en las precisiones de las velocidades.	20
4.2. Error amplificado entre el recorrido real (marcas cuadradas) y el teórico (marcas 'x'). El error se debe a la precisión de las velocidades relativas de cada eje, pero convergen perfectamente en los puntos definidos por el camino del GCode.	20
4.3. Prueba de velocidad en 3 ejes. La zona plana de la velocidad demuestra que la distancia a recorrer permitió completar la pendiente de aceleración y alcanzar la velocidad objetivo	21
4.4. Ensayo de aceleración insuficiente. La forma triangular de las curvas de velocidad indicadas con 'x' revela que no se logra la velocidad constante por falta de distancia, en cambio cambia abruptamente la pendiente de aceleración por una desaceleración. El recorrido (línea llena) refleja la forma cuadrática típica de una curva de movimiento con aceleración constante.	22
4.5. Ensayo de aceleración suficiente. Las curvas de velocidad marcadas con 'x' alcanzan rápidamente la forma plana que representa la velocidad objetivo constante, mientras que las curvas de posición, indicadas con líneas llenas, describen una recta cuando la velocidad es constante.	23
4.6. Prototipo controlando un pantógrafo CNC. Sobre la mesa de madera se colocó una hoja cuadriculada en la que se trazan los ensayos. El prototipo realizado al lado de la fuente de alimentación reemplaza el controlador original ubicado en el gabinete metálico en la parte inferior de la máquina.	24
4.7. Prototipo de hardware. Los 3 drivers de motores apilados conectado a los motores de la máquina e interconectados con el controlador a través de SPI.	25
4.8. Máquina trazando el logo Openhard sobre una hoja cuadriculada. Se adaptó un bolígrafo al cabezal de la máquina que actúa de trazador y permite realizar las validaciones.	25
4.9. Comparación de trazado de logo con aceleración adecuada (derecha) e inadecuada (izquierda).	26

...a mi tía y maestra Tita Perla Slavkin.

Capítulo 1

Introducción General

1.1. Introducción

Durante muchos años la empresa española Wolfcut [4] ha liderado el segmento de máquinas de control numérico (CNC), como muestra la imagen 1.1, diseñadas para el mercado de aberturas, cartelería comercial, fábrica de cajas, estudios de diseño, fab labs, entre otros.



FIGURA 1.1: Máquina CNC fabricada por Wolfcut.

Hasta el presente la empresa llevó adelante su negocio con un sistema de control basado en una notebook, un sistema operativo de uso general, un software de control y comunicación por USB, Ethernet o puerto paralelo. En esta configuración el software realiza las tareas de tiempo real que manejan los motores y opera la máquina, figura 1.2.

La dificultad para modificar el hardware, el software de control, el aumento en la complejidad de sus máquinas y la demanda por nuevas aplicaciones, impiden a la empresa satisfacer a sus clientes.

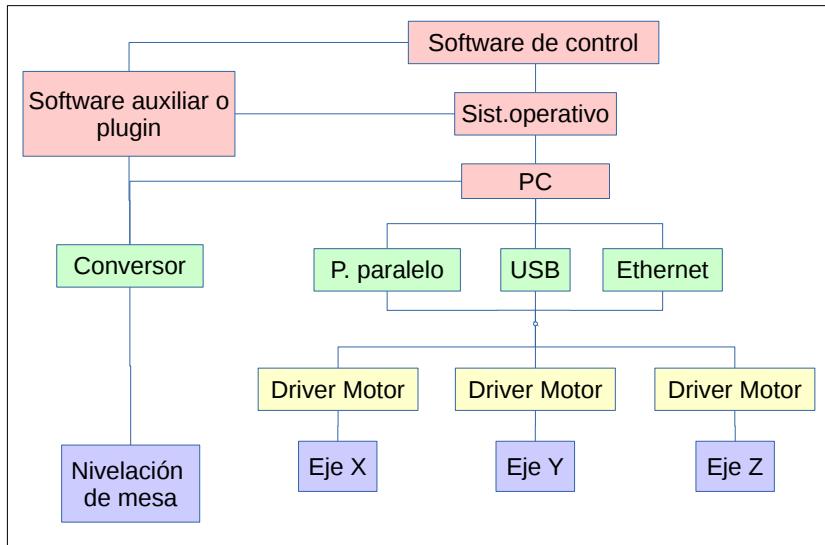


FIGURA 1.2: Diagrama en bloques de una máquina CNC controlada por PC. Se destaca como el bloque nivelación de mesa requiere al agregado de otros bloques intermedios para integrarse a la arquitectura básica de la máquina.

Por ejemplo, si la máquina requiere lectura de marcas, el fabricante se ve obligado a usar otro producto de software y hardware, lo que dificulta su uso.

El mercado de controladores genéricos para máquinas CNC ha demostrado que las nuevas versiones no siempre son compatibles con las anteriores. Esto implica a la empresa, reemplazar la electrónica en su conjunto, ante la falta o avería de una sola pieza.

Actualmente, la empresa mantiene una asociación entre máquinas y piezas utilizadas, a fin de facilitar el soporte y servicio técnico. Si bien esta logística ha funcionado hasta ahora, no asegura ser exitosa a largo plazo. Con el fin de progresar, ha encargado el desarrollo de un nuevo controlador embebido, con una topología como indica la figura 1.3 y lograr así la tecnología y documentación sobre las cuales proyectar su negocio.

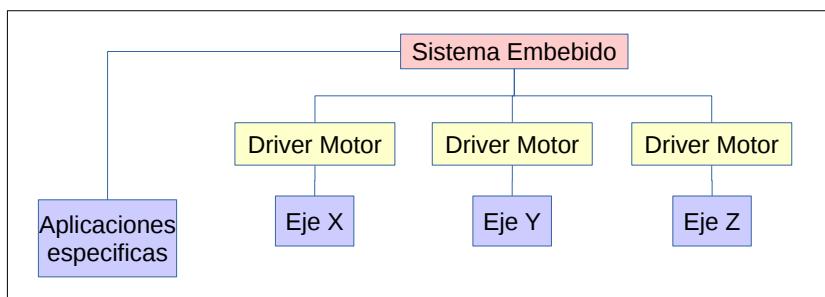


FIGURA 1.3: Diagrama en bloques de una máquina CNC controlada por un sistema embebido

Este enfoque reemplaza la PC, el sistema operativo, drivers y software de control, por un único sistema embebido, con capacidades como:

- Comandar los motores en tiempo real.

- Posibilidad de desarrollo de la interfaz de usuario para diferentes sistemas operativos.
- Comunicación a través de Ethernet, USB y/o RS232.
- Almacenamiento de los archivos GCode en el propio embebido.

Se observa que el software de control no requiere características de tiempo real, es decir, manejo de motores, finales de carrera, curvas de aceleración y solo interactúa con el usuario para seleccionar el trabajo a realizar, configurar la máquina y visualizar el avance.

Con los conceptos aprendidos en ingeniería de software se confeccionó la lista de requerimientos que se detalla a continuación.

1.2. Requerimientos

- 1 Grupo de requerimientos asociados con el parser
 - 1.1 Deberá recibir las sentencias Gcode en modo ASCII
 - 1.2 Contará con una validación de cada sentencia.
 - 1.3 Solo un subset de sentencias GCode serán válidas.
- 2 Grupo de requerimientos asociados al movimiento
 - 2.1 Podrá mover hasta 3 motores en ambas direcciones.
 - 2.2 Podrá ejecutar 10k micropulsos por segundo o más.
 - 2.3 Permitirá parametrizar la rampa de aceleración y desaceleración.
 - 2.4 Permitirá detener el movimiento en cualquier instancia de la ejecución.
- 3 Grupo de requerimientos asociados al hardware
 - 3.1 Se utilizará una UART TTL o equivalente para recibir las sentencias Gcode.

1.3. Diagrama de actividades (AON)

Con los conocimientos adquiridos en la materia gestión de proyectos, se utilizó el diagrama de actividades que se muestra en la figura 1.4. Este diagrama junto con el diagrama de Gannt planificado sirvieron de guía para concluir el trabajo a tiempo.

1.4. Objetivos y alcances

Los objetivos y alcances del proyecto, entre otros, son:

- Poner en marcha un prototipo de máquina CNC.
- Mostrar al cliente el concepto de control embebido.

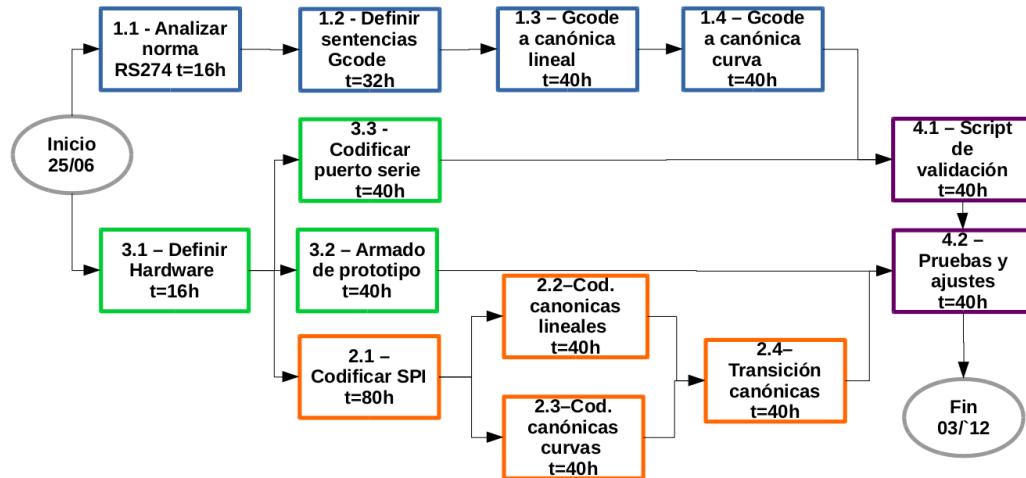


FIGURA 1.4: Diagrama de actividades AON elaborado previamente al inicio del desarrollo del trabajo. Se comprobó su utilidad al completar los objetivos a tiempo.

- Destacar las ventajas.
- Delimitar el alcance de sus posibilidades.
- El prototipo se limitará a realizar movimientos suficientes para los ensayo.
- El software de control estará acotado a la ejecución de ensayos y validaciones.

Capítulo 2

Introducción Específica

Este capítulo describe una introducción detallada del trabajo realizado. Inicialmente, se muestra un diagrama en bloques del controlador y posteriormente algunas de las técnicas aplicadas.

2.1. Diagrama en bloques de la máquina CNC

El desarrollo se dividió en varios bloques como lo muestra la figura 2.1 y se describen brevemente:

- Software de monitoreo y gestión
Es el encargado de enviar los archivos GCode a la máquina, mover libremente los tres ejes en tiempo real y visualizar la posición del cabezal de corte.
- Bloques html/Win/Mac/Android/IOs, PC/Mobil
El software, puede correr en cualquier sistema operativo y hardware de uso general, debido a que no requiere características de tiempo real.
- USB/Ethernet
Son las interfaces físicas habilitadas en la plataforma de desarrollo para la comunicación con el software de gestión.
- Controlador embebido
Es el encargado de realizar las tareas de tiempo real y actuar de intermedio entre el software y el hardware. Cuenta con capacidades para manejar la máquina de manera autónoma al utilizar archivos GCode almacenados internamente.
- Fuente de energía
Provee la energía suficiente para el movimiento de los motores y la electrónica de control.
- Driver de motores
Recibe comandos a través de la interfaz SPI y genera la secuencia de pulsos necesarios para mover los motores paso a paso.
- Mesa
Representa la mesa donde se realizan los trabajos de maquinado.
- Motor Eje X/Y/Z
Representan los motores de cada eje de la máquina.

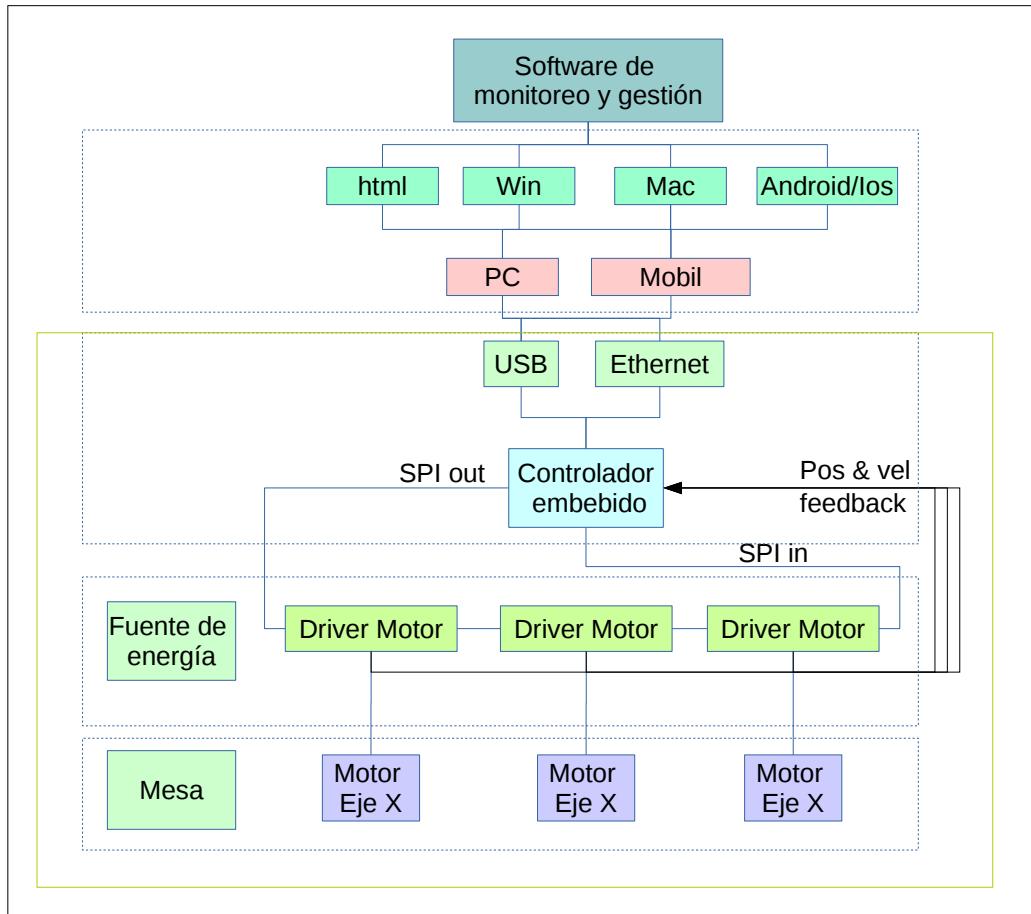


FIGURA 2.1: Diagrama de bloques de la máquina CNC. En recuadro amarillo los bloques implementados en este trabajo.

2.2. Generación de archivos GCode

Durante las pruebas, se utilizaron archivos GCode [9] que se obtuvieron usando el plugin RhinoCam¹ sobre el CAD Rhinoceros². En la figura 2.2 se muestra una captura de pantalla con la figura y los procesos de maquinado para obtener la pieza. Se puede observar el ícono Openhard con el agregado de una media esfera en su centro para poner a prueba movimientos combinados en los tres ejes durante el recorrido de su superficie.

¹RhinoCam, plugin de Rhino para generación de trabajos de maquinado por CNC. URL: <https://mecsoft.com/rhinocam-software/>.

²Rhinoceros, CAD para desarrollo de figuras 3D en general. URL: <https://www.rhino3d.com/>.

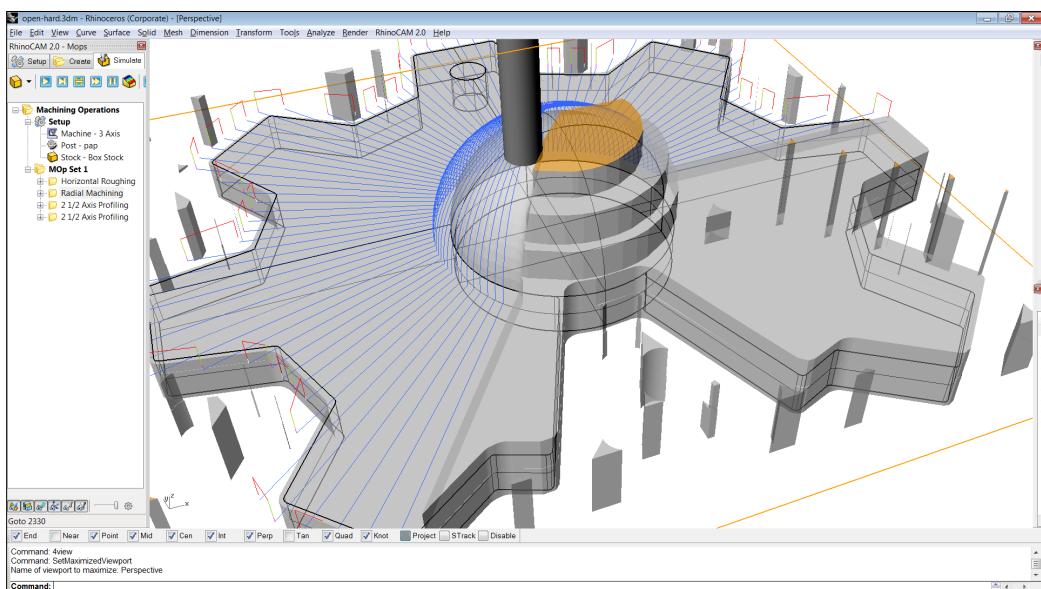


FIGURA 2.2: Captura de pantalla de RhinoCam. Se muestra como el resultado de las operaciones de maquinado dan por resultado el icono de Openhard con una media esfera en su centro.

Capítulo 3

Diseño e Implementación

En las siguientes secciones se exponen los criterios de diseño empleados y la descripción técnica detallada de los principales aspectos del producto.

3.1. Plataforma embebida

Para la selección de la plataforma se tomaron en cuenta los requisitos de conectividad (UART y opcionalmente Ethernet), capacidad de cómputo, costos y disponibilidad. Se eligió una plataforma de desarrollo de la firma Texas Instruments¹ modelo TIVA C TM4C1294 (figura 3.1) con las siguientes características:

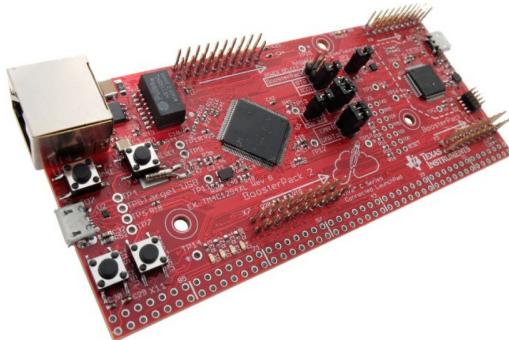


FIGURA 3.1: Plataforma de desarrollo TIVA C TM4C1294 (imagen tomada de Texas Instruments, <http://www.ti.com/tool/sw-ek-tm4c1294xl>)

- Microcontrolador ARM cortex M4 @ 120 Mhz - suficiente para el procesamiento necesario.
- 512k de flash y 256k de ram - permite ejecutar freeRTOS con lwIP que son los componentes de firmware de mayor tamaño de código.
- Unidad de punto flotante - esencial para la conversión de escala de coordenadas GCode y cálculo trigonométrico.
- Conectividad Ethernet con soporte para lwIP - permite tener el producto accesible por Ethernet con poco esfuerzo y gran versatilidad.
- Interfaz de programación ICDI - permite la depuración por USB y acceso a una UART útil para comandar la máquina .

¹Texas Instruments. URL: <https://www.ti.com>.

- Interfaz SPI - indispensable para la comunicación con los drivers de motores.

3.2. Driver de motores

Para el driver de los motores se optó por un circuito integrado powerstep01 [12] de la firma STMicroelectronics² en una plataforma de evaluación X-NUCLEO-IHM03A1 como se observa en la imagen 3.2 y cuenta con las siguientes características:

- Interfaz de control SPI - permite comunicar varias plataformas de desarrollo encadenadas, figura 3.3.
- Generación de hasta 128 micropasos - ideal para un movimiento suave y preciso del motor.
- Generación de rampas de aceleración - simplifica los cálculos requeridos y garantiza uniformidad en los movimientos.
- Seteo de velocidad con precisión - necesaria para un movimiento coordinado de más de un motor.
- Corriente de motor configurable - permite ajustar el diseño a los parámetros de los motores elegidos.
- Etapa de potencia integrada - simplifica el diseño del hardware y garantiza un acople óptimo entre la lógica y el manejo de los motores.

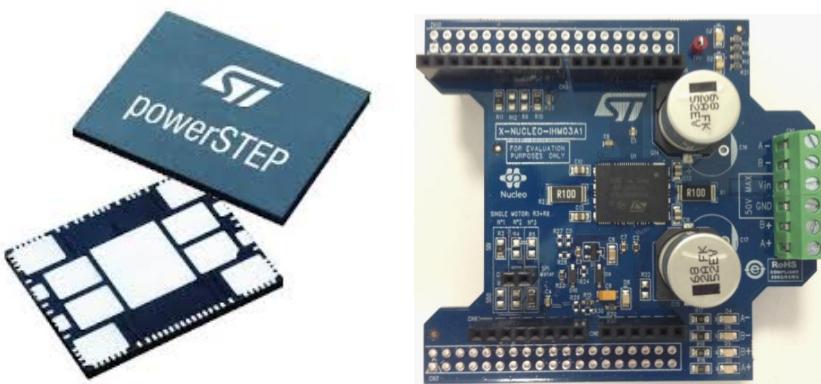


FIGURA 3.2: Chip de control de motor (izq.) y placa de desarrollo utilizada (der.) (imágenes tomada de [1])

3.2.1. Sincronización de relojes

Al poner en marcha dos motores se observó falta de sincronismo en sus movimientos. Este inconveniente se debió a que cada chip utiliza por defecto su propio reloj de referencia.

El sincronismo es una propiedad fundamental para realizar movimientos en mas de un eje sin desviarse de las trayectorias. Para sincronizar los controladores se

²ST Microelectronic. URL: <https://www.st.com>.

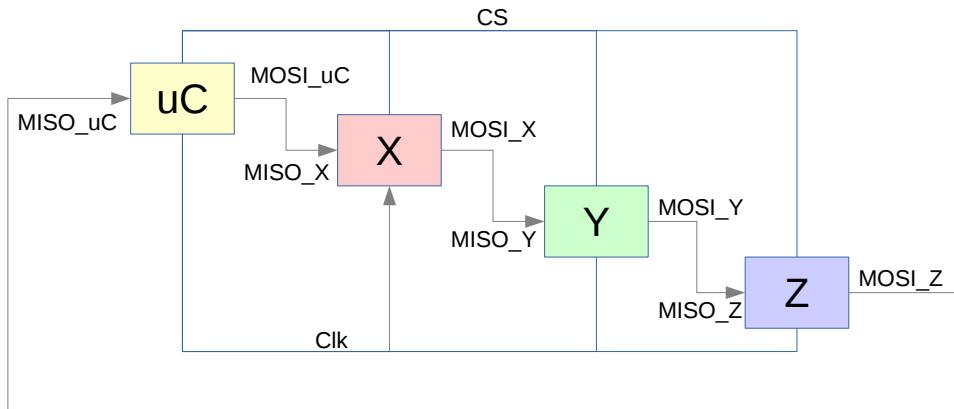


FIGURA 3.3: Conexión SPI encadenada. La salida de unos de los drivers es la entrada del siguiente, y el último devuelve su salida como entrada del primero cerrando el anillo

ideó una conexión donde uno de los chips utiliza su propio reloj de referencia y genera una salida sincrónica utilizada por el siguiente. Éste a su vez regenera la señal que servirá de entrada para el próximo lo que permite establecer una cadena de relojes como se aprecia en la figura 3.4.

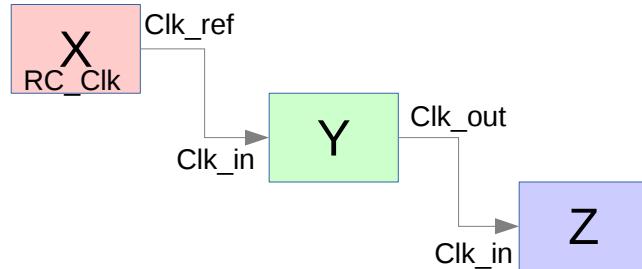


FIGURA 3.4: Conexión de CLK encadenada. El primer driver genera el reloj de referencia, el resto toma la señal y la regenera para el próximo

3.3. Desarrollo de hardware

En base a la plataforma de desarrollo utilizada en el prototipo se diseñó el PCB mostrado en la figura 3.5, donde se agregaron características surgidas durante el trabajo que no estaban implementadas en la herramienta original, como:

- Leds de debug y alimentación.
- Agujeros de montaje.
- PCB de 2 capas, en comparación con las 4 capas de la herramienta de ST.
- Conectores de potencia enchufables para facilitar la instalación y recambio.
- Conectores RJ45 para señales.
- Sincronización de relojes.

- Encadenamiento de interfaz SPI.

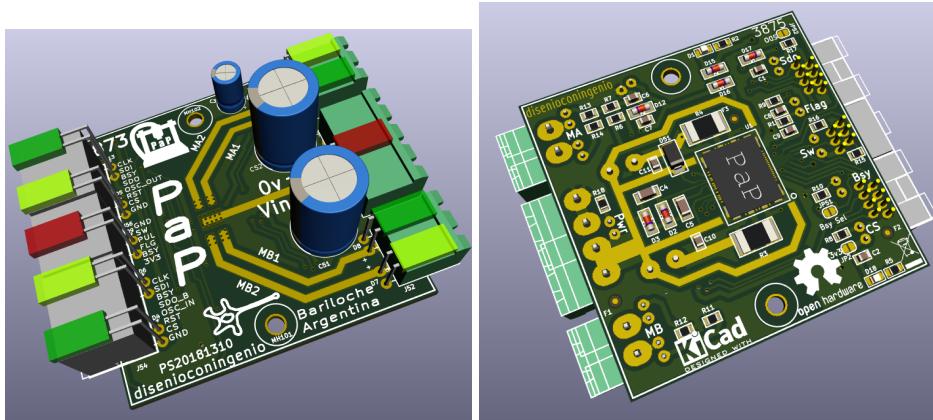


FIGURA 3.5: Vista 3D de la placa de control de motores desarrollada en Kicad

Se utilizó el software Kicad aprendido durante la carrera y se fabricaron PCB's prototipos que se aprecian en la figura 3.6 los cuales están en instancia de montaje. En el apéndice B se destacan parte de los esquemáticos utilizados donde se puede observar en detalle las configuraciones antes mencionadas.



FIGURA 3.6: Placa de control de motores doble faz fabricada como prototipo

3.4. Arquitectura del firmware

Sobre la base del sistema operativo de tiempo real freeRTOS [6] se desarrolló el firmware del controlador.

Entre otros, se aprovecharon los siguientes servicios: colas, semáforos, mutex, drivers, y el port de lwIP [3] ya disponibles para esta plataforma.

En la figura 3.7 se muestra de manera simplificada el flujo del sistema de recepción, procesamiento y ejecución de las sentencias GCode.

Las sentencias GCode son recibidas desde la UART o Ethernet y son almacenadas en una cola de preprocesamiento. Una tarea continuamente lee dicha cola y realiza trabajos preliminares sobre cada linea, incluidos los siguientes cálculos

- Distancia desde la coordenada anterior a la solicitada.
- Dirección del movimiento para cada eje.
- Velocidad objetivo para cada eje.
- Aceleración y desaceleración para cada eje.

Estas operaciones involucran el cálculo de una raíz cuadrada y operaciones intensivas en punto flotante. Al realizarlos por adelantado, como se destaca en el algoritmo 3.1, la siguiente tarea, resumida en el algoritmo 3.2, se limitará al envío de los comandos y lectura de la posición en tiempo real. De esta manera se logra minimizar el tiempo de espera entre comandos sucesivos y en consecuencia mayor fluidez de movimiento en la máquina.

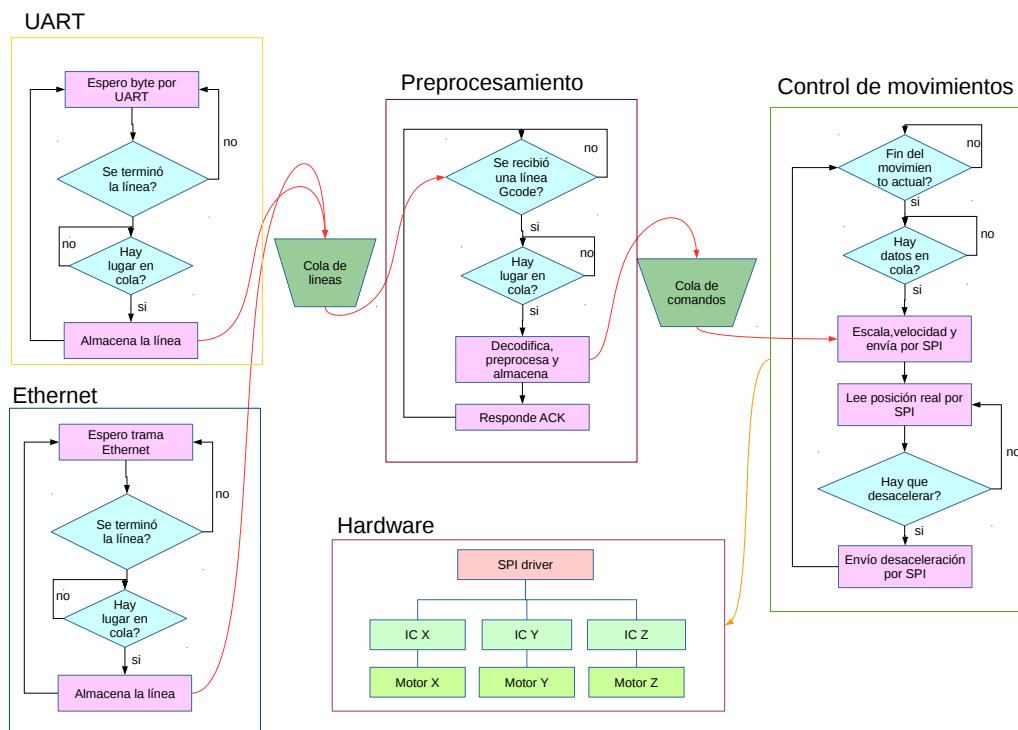


FIGURA 3.7: Flujo de datos relacionados al procesamiento de los comandos GCodes. Las sentencias son inicialmente almacenadas en una cola que luego son preprocesadas para adelantar el cálculo matemático. El resultado es enviado por SPI, cuando el comando previo ha concluido.

```

1 int Cmd_Gcode_GL( struct tcp_pcb* tpcb , int argc , char *argv [] )
2 static Motor_t Motor;
3 uint8_t i ;
4 for (i=0;i<NUM_AXES; i++) {
5     Motor.Pos [ i ] = Motor.Target [ i ];
6     Motor.Actual_Pos[ i ] = Motor.Actual_Target[ i ];
7 }
8
9 for (i=1;i<argc ; i++) {

```

```

10    switch(argv[i][0]) {
11        case 'X':
12            Motor.Actual_Target[0] = usrtotf(argv[i]+1,NULL);
13            Motor.Target[0]       = Motor.Actual_Target[0]*GMotor.Scale
14            [0];
15            break;
16        case 'Y':
17            Motor.Actual_Target[1] = usrtotf(argv[i]+1,NULL);
18            Motor.Target[1]       = Motor.Actual_Target[1]*GMotor.Scale
19            [1];
20            break;
21        case 'Z':
22            Motor.Actual_Target[2] = usrtotf(argv[i]+1,NULL);
23            Motor.Target[2]       = Motor.Actual_Target[2]*GMotor.Scale
24            [2];
25            break;
26        case 'F':
27            Set_Max_Vel(&Motor, usrtotf(argv[i]+1,NULL));
28            break;
29        default:
30            break;
31    }
32    \\calculos que implican operaciones de punto flotante
33    Dir      ( &Motor );
34    Delta    ( &Motor );
35    Distance ( &Motor ); \\raiz cuadrada
36    Vel      ( &Motor );
37    Accel    ( &Motor );
38    xQueueSend(Moves_Queue,&Motor,portMAX_DELAY);
39 }

```

ALGORITMO 3.1: Función de preprocesamiento de la línea GCode
y almacenamiento en la cola de comandos

```

1 void Moves_Parser(void* nil)
2 {
3     Motor_t Motor;
4     while(1) {
5         while(xQueueReceive(Moves_Queue,&Motor,portMAX_DELAY)!=pdTRUE)
6             ;
7         Acc_Dec_Steps ( &Motor );
8         Set_Max_Speed ( Aux_Vel );
9         Set_Acc      ( Motor.Acc );
10        Set_Dec      ( Motor.Dec );
11
12        if(Run_Or_Goto(&Motor)==false) {
13            Run      ( Motor.Dir, Motor.Vel );
14            Clear_Goto ( &Motor );
15            while(Busy_Read()==0)
16                vTaskDelay ( pdMS_TO_TICKS(20 ) );
17            while(Time2Goto(&Motor)==false) {
18                Abs_Pos   ( Motor.Pos );
19                Delta    ( &Motor );
20                vTaskDelay ( pdMS_TO_TICKS(20 ) );
21            }
22        }
23        Goto ( Motor.Target );
24        while(Busy_Read()==0)
25            vTaskDelay ( pdMS_TO_TICKS(20 ) );
26    }

```

27 }

ALGORITMO 3.2: Función de control de movimiento. Decide el comando a enviar y espera el momento para la desaceleración mientras lee la posición actual

3.4.1. Protocolo SPI

Para el correcto funcionamiento de la máquina es indispensable que los movimientos de los motores estén perfectamente sincronizados. Estos movimientos son indicados usando la interfaz SPI en la cual cada dispositivo se comporta como un esclavo y recibe las instrucciones del controlador que actúa como maestro.

La comunicación encadenada de dispositivos permite que los comandos enviados sean ejecutados en el mismo instante, dado que son interpretados en el flanco ascendente del pin CS. En la figura 3.8 se detalla el envío de un comando a 3 dispositivos encadenados. Mientras la señal CS esta en cero lógico, los datos pasan de un chip al siguiente y recién son interpretados al cambiar el estado de CS.

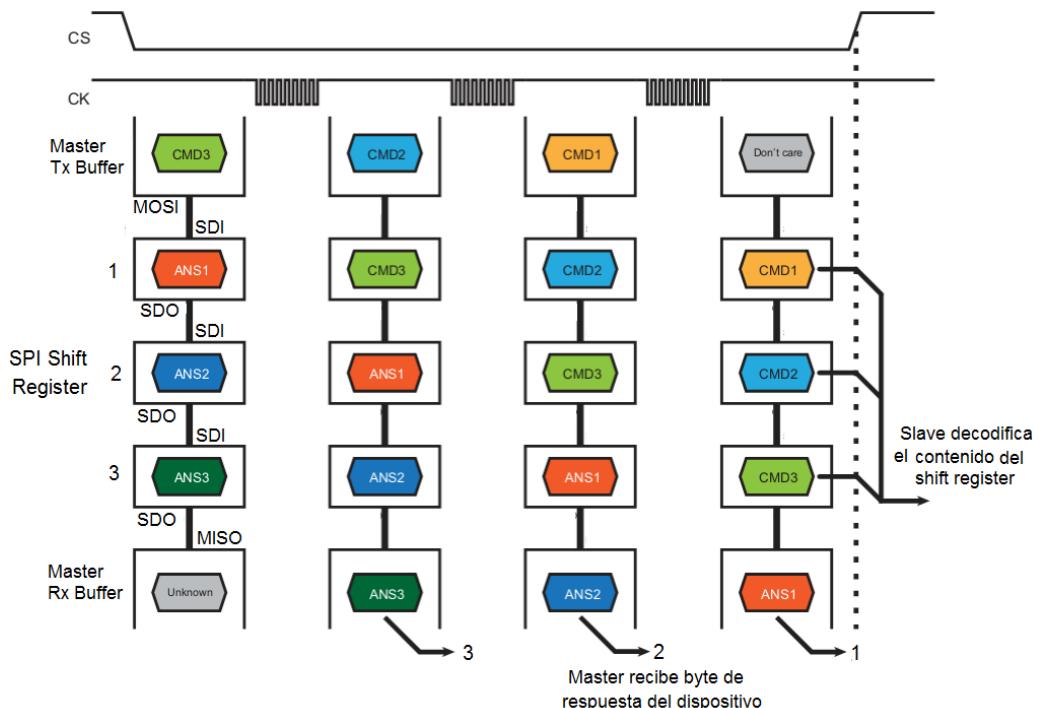


FIGURA 3.8: Ejemplo de envío de un comando a 3 dispositivos encadenados (imagen tomada de [11])

En la función listada en 3.3 se muestra el procedimiento empleado para el envío de los comandos por SPI. En el mismo se distingue el aprovechamiento de los servicios de freeRTOS y los drivers de SPI de Texas para la plataforma de desarrollo usada. Notar como el primer byte enviado corresponde al último dispositivo de la cadena.

```

1 void Send_Cmd2Spi( struct  tcp_pcb*  tpcb , Spi_Params*  Params)
2 {
3     uint32_t  Ans;
4     uint8_t  i ;

```

```

5   int8_t n;
6   //protege de acceso multiple al spi desde otra tarea
7   xSemaphoreTake(Busy_Sem, portMAX_DELAY);
8   //los comandos tienen longitud variable, se envian todos los
9   //indicados
10  for(i=0;i<=Params->Len;i++) {
11      //se baja el pin CS y comienza la transmision de datos
12      Cs_Lo();
13      //para todos los dispositivos, indexando desde el ultimo hasta el
14      //primero, debido a que el primer comando sera recibido por el ultimo
15      //dispositivo de la cadena.
16      for(n=NUM_AXES-1;n>=0;n--) {
17          //envio de un byte
18          MAP_SSIDataPut(SSI2_BASE, Params->Data[n][i]);
19          //lectura de la respuesta del ultimo dispositivo de la cadena
20          MAP_SSIDataGet(SSI2_BASE,&Ans);
21          //almaceno la respuesta
22          Params->Data[n][i]=(uint8_t)Ans;
23      }
24      //si hay una operacion en curso, justo antes de enviar el ultimo
25      //parametro se espera a que culmine, de otro modo seria invalida la
26      //operacion.
27      if(Wait_Busy==true && i==(Params->Len)) {
28          //espera en modo bloqueante ya que esto se ejecuta en el
29          //contexto de una tarea.
30          while(Busy_Read()==0)
31          ;
32          //desmarca la solicitud de espera para los proximos comandos
33          Wait_Busy=false;
34      }
35      //sube el CS para que el chip procese el comando recibido
36      Cs_Hi();
37  }
38  //libera el semaforo
39  xSemaphoreGive(Busy_Sem);
40 }
```

ALGORITMO 3.3: Función de envío y recepción de comandos por SPI. El primer byte enviado es leído por el ultimo esclavo de la cadena

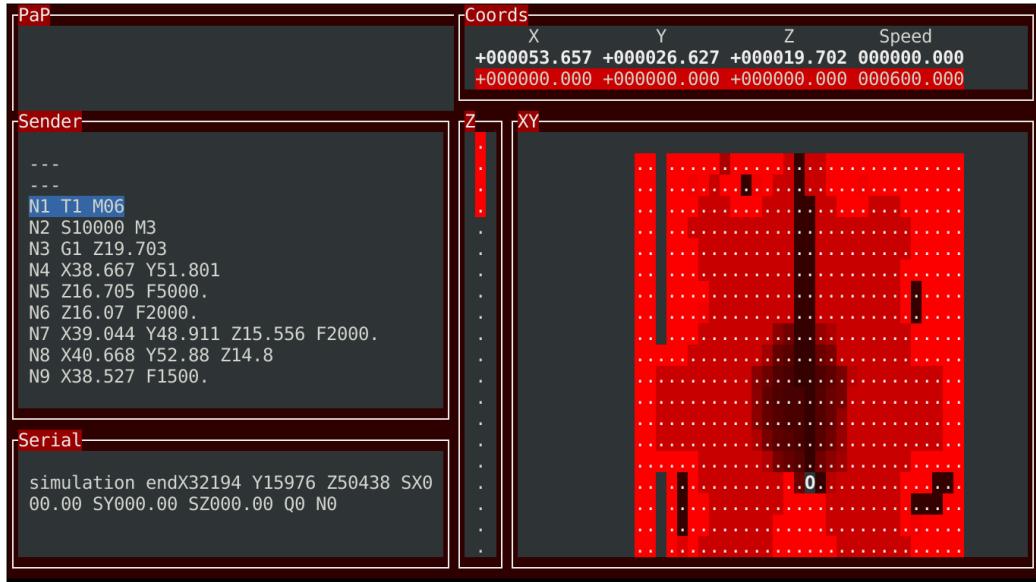
3.5. Software de control

El software de control que se muestra en la figura 3.9 se desarrolló con el fin de realizar los ensayos y no para el operario de la máquina, sin embargo su código en C++ permitirá reutilizar el código de las clases en un futuro desarrollo del software de usuario.

Se priorizó la posibilidad de acceso remoto, la portabilidad y la eficiencia, por lo que se decidió desarrollarlo para linea de comandos sobre Ncurses³. De esta manera se lo puede utilizar accediendo remotamente por telnet o ssh desde cualquier sistema operativo con requisitos mínimos de transferencia de datos y del sistema en general.

Esta herramienta archiva todas las transacciones entre la PC y el controlador para un análisis posterior. También posibilita visualizar el recorrido de la máquina en

³Thomas Dickey. 2018. URL: <https://www.gnu.org/software/ncurses/>.



un gráfico en tres dimensiones usando la herramienta Gnuplot⁴. En la figura 3.10 se muestra dicho gráfico a mitad del trabajo de maquinado.

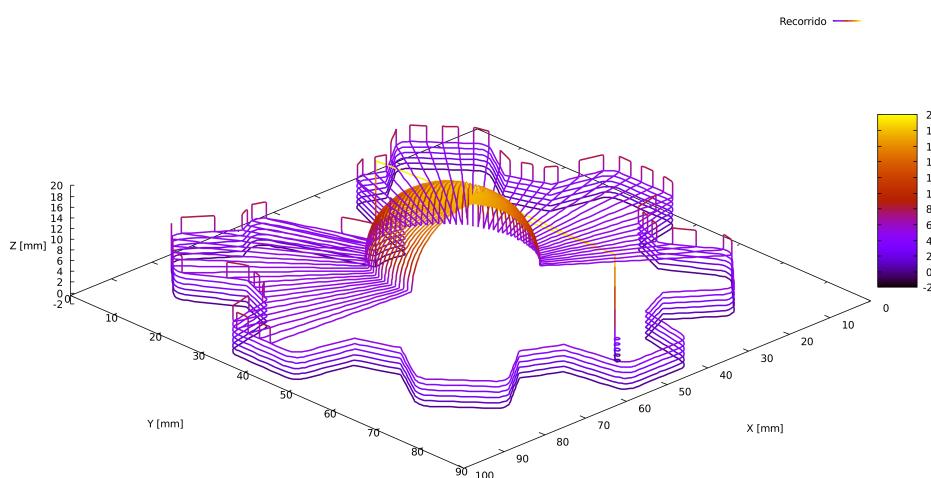


FIGURA 3.10: Pantalla derivada de Cuipap. Trazado de los ejes X, Y y Z en tiempo real a mitad de un trabajo de maquinado

Entre las funciones del software se destacan:

- Seleccionar y enviar un archivo GCode al controlador.
- Mover la máquina en todas las direcciones mediante el uso de las teclas.
- Escalar la velocidad de movimiento sobre la solicitada en el GCode.

⁴Herramienta portable para la generación de gráficos. URL: <http://www.gnuplot.info/>.

- Visualizar en tiempo real el movimiento de los 3 ejes en consola y en un gráfico de tres dimensiones.
- Visualizar información de debug
- Archivar la información del recorrido real para un posterior análisis.

Capítulo 4

Ensayos y Resultados

En este capítulo se describen algunos de los ensayos utilizados para validar el desarrollo de firmware, hardware y software

4.1. Pruebas funcionales del hardware

Con la ayuda del software de control desarrollado, Cuiap, destacado en la figura 3.9, se realizaron adquisiciones en tiempo real de la posición y la velocidad de cada eje cada 50 mseg. Para cada una de las pruebas se escribieron secuencias GCode que resaltan las características de interés en el análisis del recorrido, la velocidad y la aceleración.

4.1.1. Pruebas de recorrido

Para probar el recorrido de un determinado camino se utilizó la figura geométrica del logo *Openhard* en 3 dimensiones como se muestra en la figura 4.1. El código se puede encontrar en el apéndice A. En la figura 4.1 se pueden apreciar los caminos teóricos del GCode superpuestos con el recorrido real adquirido.

En el error amplificado en la figura 4.2 se revelan diferencias en ciertos tramos del recorrido. Sin embargo cuando el movimiento llega a un punto definido por una sentencia GCode, la medición muestra que las curvas convergen.

Además se puede ver como la pendiente de movimiento cambia para alcanzar el objetivo en el momento que comienza la desaceleración. Las diferencias entre el camino teórico y el real se deben al error en la precisión de las velocidades de cada eje.

4.1.2. Pruebas de velocidad

Se puede observar en la figura 4.3 que se optó por un recorrido con velocidades crecientes en cada eje. Las aceleraciones elegidas fueron suficientes para que se alcancen las velocidades objetivo. Se pudo constatar que las velocidades solicitadas correspondieron con las velocidades reales obtenidas durante la ejecución del recorrido.

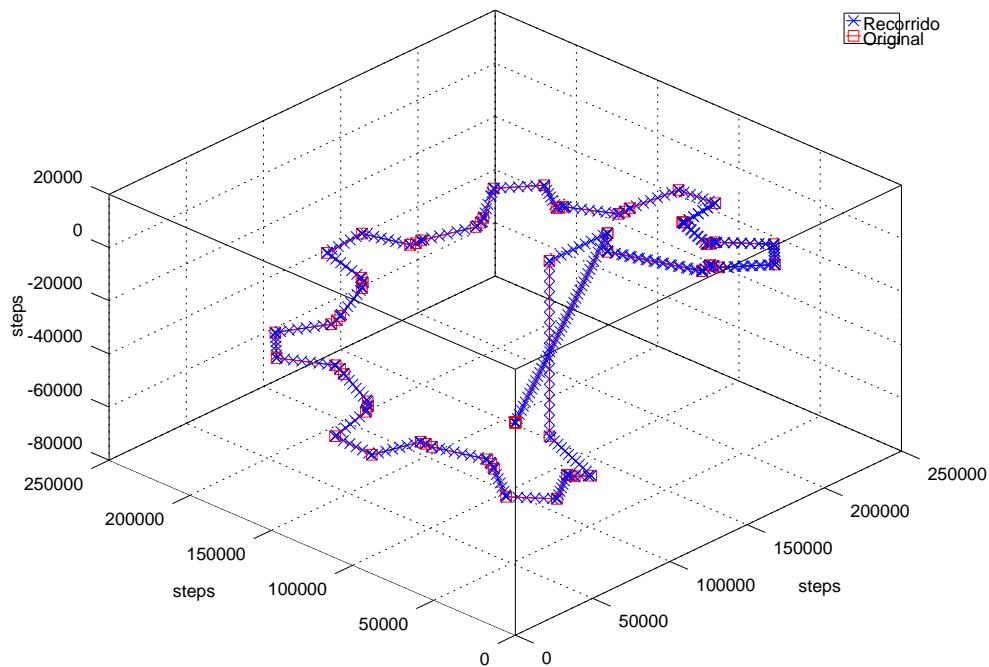


FIGURA 4.1: Comparación entre el recorrido teórico del GCode (rojo con marcas cuadradas) y el real (azul con marcas 'x'). Aunque parecen coincidir divergen en ciertas zonas mínimamente debido al error en las precisiones de las velocidades.

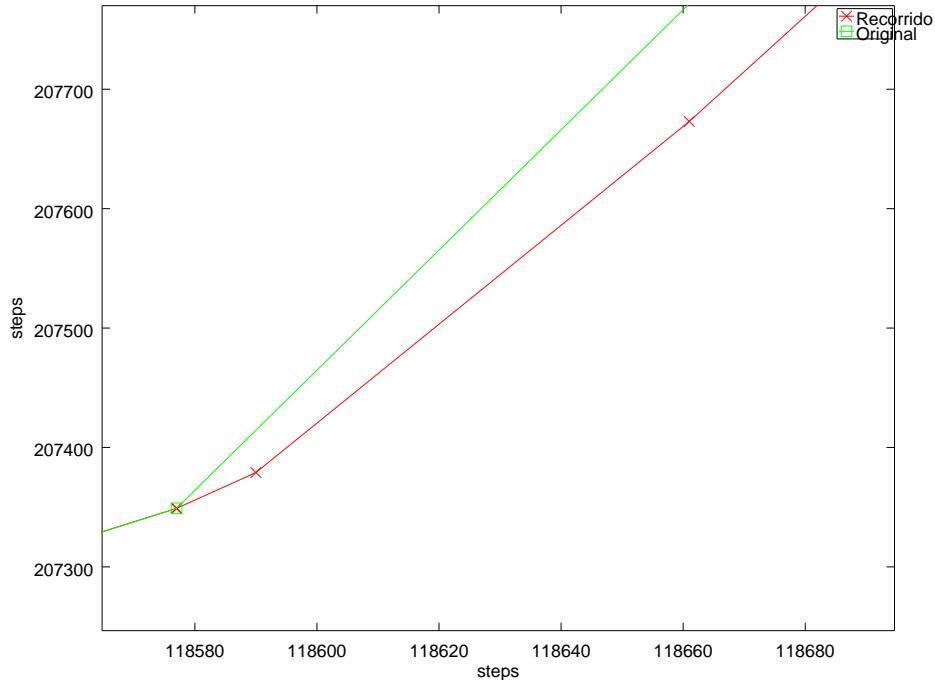


FIGURA 4.2: Error amplificado entre el recorrido real (marcas cuadradas) y el teórico (marcas 'x'). El error se debe a la precisión de las velocidades relativas de cada eje, pero convergen perfectamente en los puntos definidos por el camino del GCode.

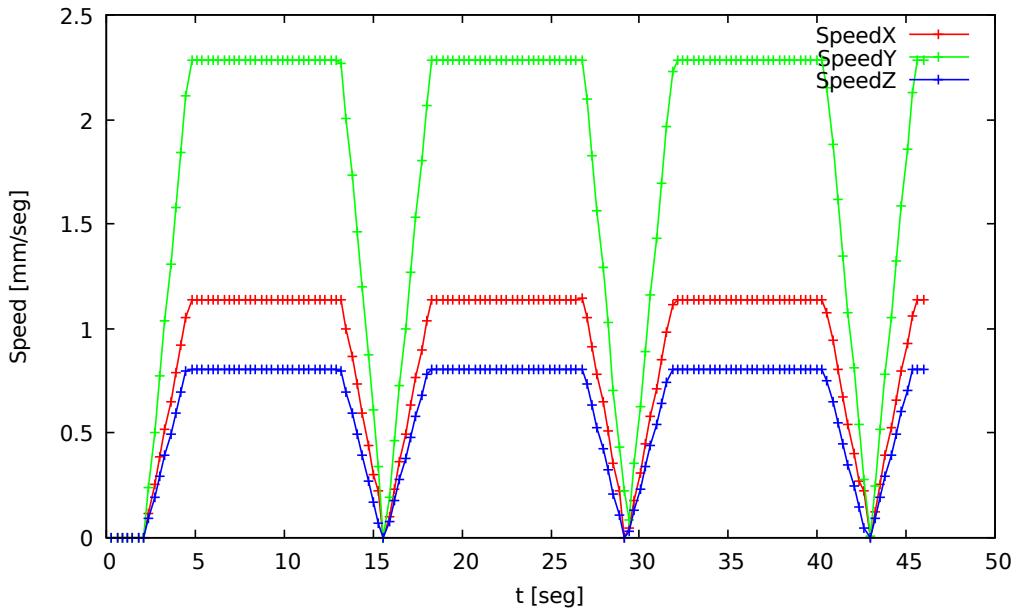


FIGURA 4.3: Prueba de velocidad en 3 ejes. La zona plana de la velocidad demuestra que la distancia a recorrer permitió completar la pendiente de aceleración y alcanzar la velocidad objetivo

4.1.3. Pruebas de aceleración

En los ensayos de aceleración se destacan dos situaciones que se desprenden de las ecuaciones 4.2 y 4.1 de movimiento clásico en una dimensión:

$$X = X_0 + V_0 t + \frac{1}{2} a t^2 \quad (4.1)$$

$$V_f = V_0 + a t \quad (4.2)$$

Como en este trabajo solo se implementó la técnica conocida como *exact stop* (*parrada exacta*), cada movimiento se inicia con $V_0 = 0$.

Si t_f es el tiempo requerido para alcanzar la velocidad V_f , se obtiene la ecuación 4.3:

$$\frac{V_f}{a} = t_f \quad (4.3)$$

Si X_f es la distancia en la cual la velocidad alcanza V_f y para simplificar se considera que $X_0 = 0$, al reemplazar 4.3 en 4.1 se obtiene la ecuación 4.4:

$$X_f = \frac{1}{2} a \left(\frac{V_f}{a} \right)^2 \quad (4.4)$$

$$X_f = \frac{V_f^2}{2a} \quad (4.5)$$

Esto determina la distancia que deberá recorrer cada eje para alcanzar la velocidad objetivo con la aceleración a .

Si se aplica el mismo cálculo para la desaceleración y se nombra X_g a la distancia recorrida partiendo con velocidad V_f hasta velocidad 0, y D a la distancia entre dos puntos consecutivos se obtiene la desigualdad 4.6:

$$X_f + X_g > D \quad (4.6)$$

Esto significa que la aceleración y desaceleración son suficientes para que el motor alcance la velocidad objetivo.

Y para el caso contrario se tiene la desigualdad 4.7:

$$X_f + X_g \leq D \quad (4.7)$$

La cual implica que el motor nunca alcanzará la velocidad objetivo debido a una aceleración insuficiente.

Para estos dos casos se realizaron ensayos que se detallan en los siguientes párrafos.

Aceleración insuficiente

En este ensayo, cuya representación se puede observar en la figura 4.4, la aceleración y desaceleración no son suficientes lograr la velocidad objetivo en el espacio disponible cumpliendo con la desigualdad 4.7.

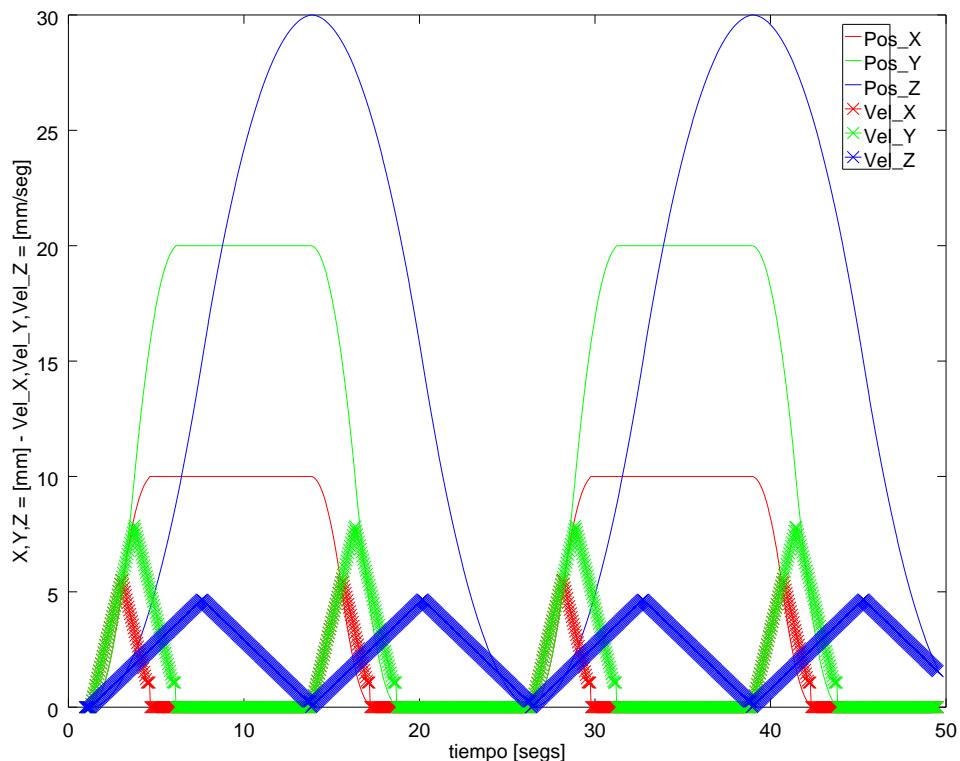


FIGURA 4.4: Ensayo de aceleración insuficiente. La forma triangular de las curvas de velocidad indicadas con 'x' revela que no se logra la velocidad constante por falta de distancia, en cambio cambia abruptamente la pendiente de aceleración por una desaceleración. El recorrido (línea llena) refleja la forma cuadrática típica de una curva de movimiento con aceleración constante.

Aceleración suficiente

En este ensayo representado en la figura 4.5, se muestra la situación donde la aceleración y desaceleración permiten alcanzar la velocidad objetivo, desigualdad 4.6

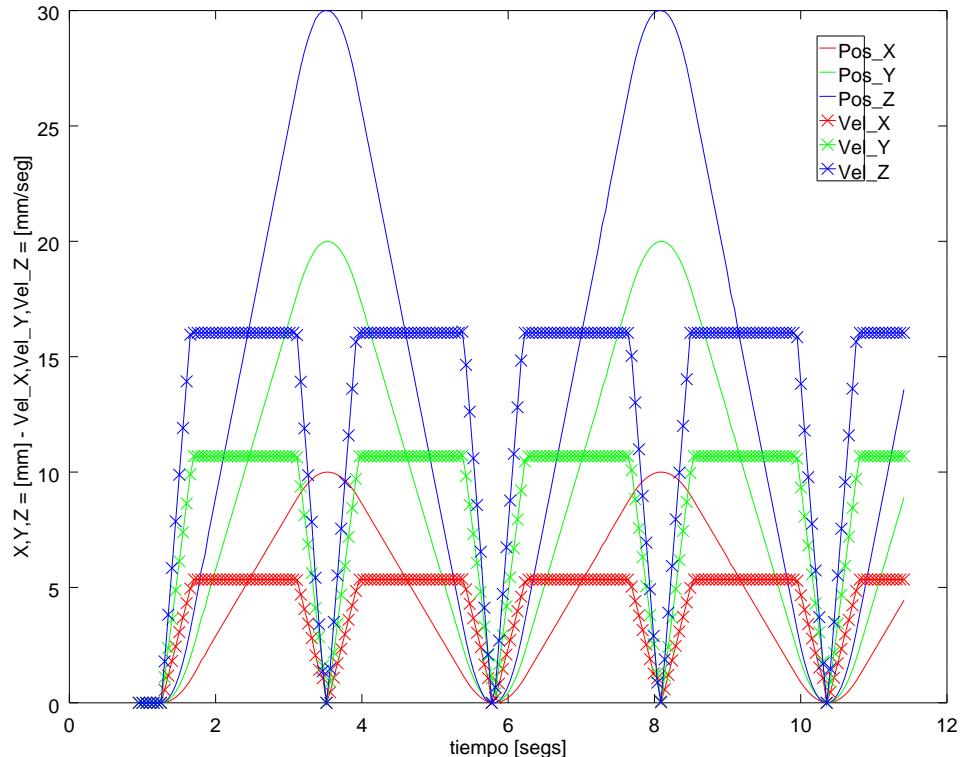


FIGURA 4.5: Ensayo de aceleración suficiente. Las curvas de velocidad marcadas con 'x' alcanzan rápidamente la forma plana que representa la velocidad objetivo constante, mientras que las curvas de posición, indicadas con líneas llenas, describen una recta cuando la velocidad es constante.

4.2. Pruebas en prototipo funcional

En las figuras 4.6 y 4.7 se puede apreciar la implementación del prototipo controlando un pantógrafo. Se adaptó un bolígrafo como trazador para realizar pruebas de validación, como se puede ver en la figura 4.8.

Utilizando el logo de Openhard en 2 dimensiones se probaron situaciones de aceleración adecuada e inadecuada para la máquina en ensayo.

En la figura 4.9 se puede ver que, cuando la aceleración no es la correcta, la máquina genera oscilaciones en su trayectoria. Esto se debe a un efecto típico de los motores paso a paso y la falta de amortiguación mecánica de esta máquina en particular. En cambio cuando el recorrido se realiza con una aceleración adecuada, el recorrido se ve uniforme y sin oscilaciones notables.

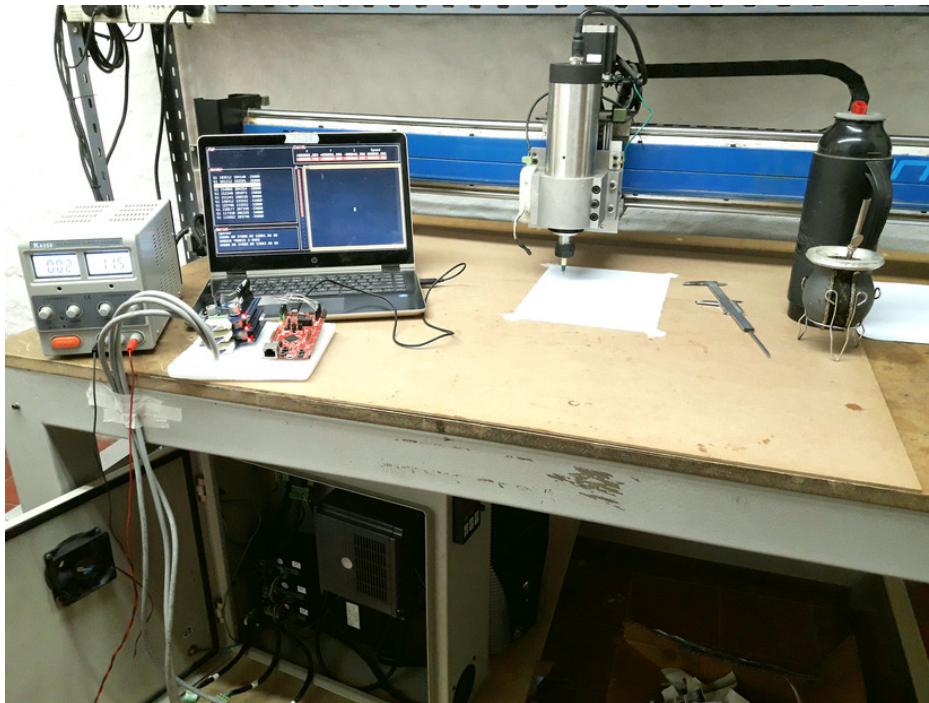


FIGURA 4.6: Prototipo controlando un pantógrafo CNC. Sobre la mesa de madera se colocó una hoja cuadriculada en la que se trazan los ensayos. El prototipo realizado al lado de la fuente de alimentación reemplaza el controlador original ubicado en el gabinete metálico en la parte inferior de la máquina.

La selección de las rampas de aceleración y desaceleración son fundamentales para el correcto funcionamiento y dependen de muchos factores entre los cuales se destacan:

- Masa de los ejes.
- Corriente de excitación de los motores.
- Amortiguamiento en el acople mecánico.
- Calidad de los motores, masa del eje, calidad del núcleo, calidad mecánica, resistencia e inductancia del bobinado, etc..

Dado que el sistema de control permite seleccionar por software la aceleración independiente de cada eje y las corrientes de excitación de los motores, se puede calibrar el funcionamiento de la máquina una vez construida dentro de cierto rango.

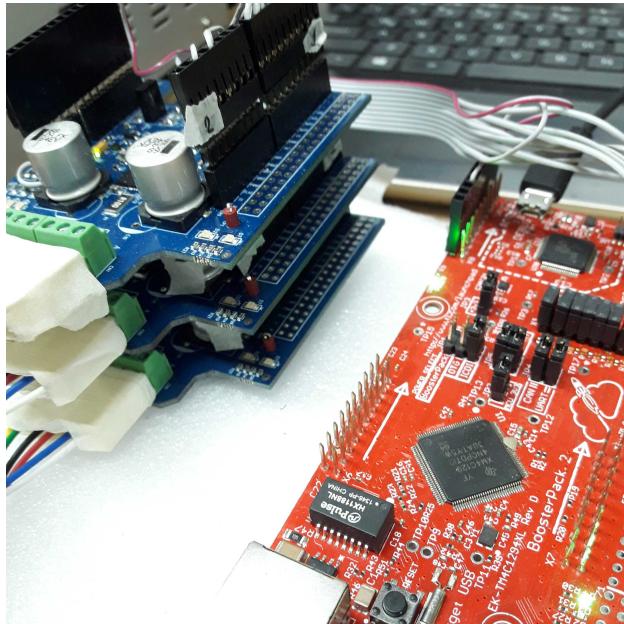


FIGURA 4.7: Prototipo de hardware. Los 3 drivers de motores apilados conectado a los motores de la máquina e interconectados con el controlador a través de SPI.

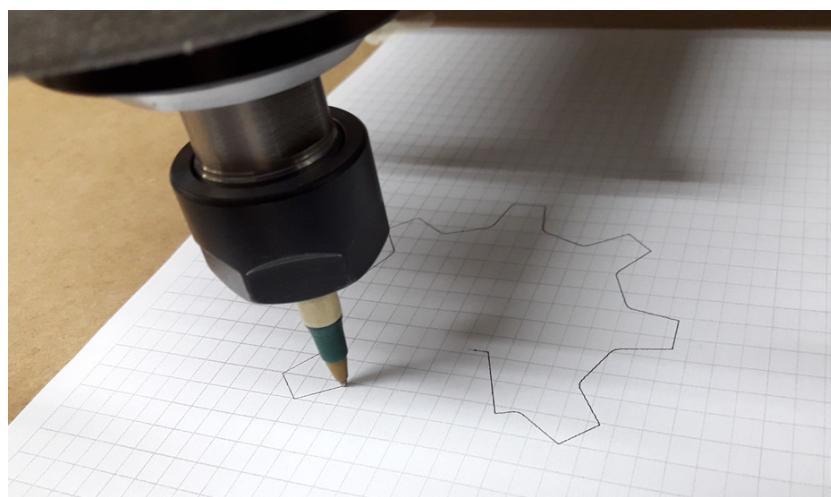


FIGURA 4.8: Máquina trazando el logo Openhard sobre una hoja cuadriculada. Se adaptó un bolígrafo al cabezal de la máquina que actúa de trazador y permite realizar las validaciones.

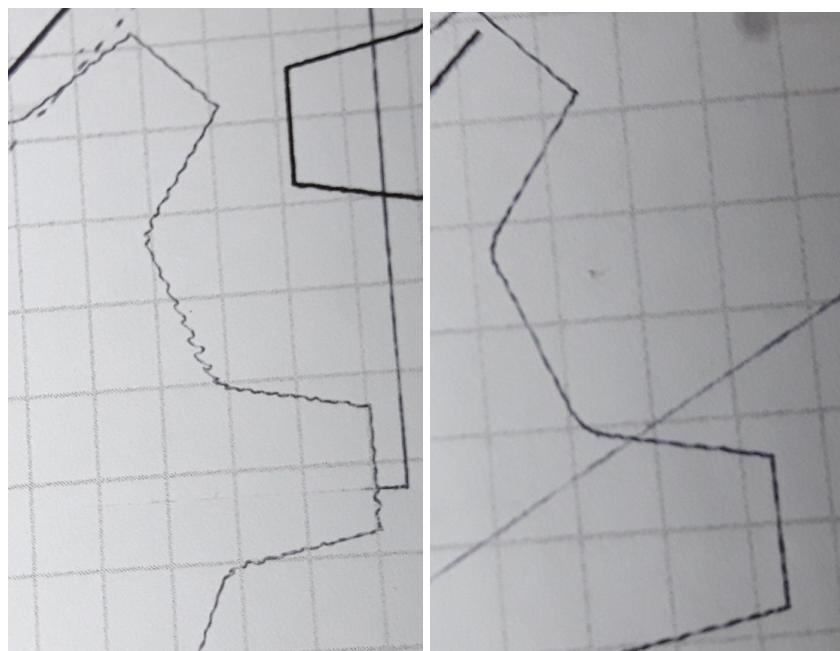


FIGURA 4.9: Comparación de trazado de logo con aceleración adecuada (derecha) e inadecuada (izquierda).

Capítulo 5

Conclusiones

5.1. Conclusiones generales

Se logró poner en marcha una máquina CNC con el uso del software, firmware y hardware desarrollados. De esta manera se reemplazó el controlador basado en una PC y drivers del tipo pulso/dirección por un controlador embebido. Se alcanzaron prestaciones equivalentes en un rango acotado de comandos y para cierto tipo de trabajos. Durante la implementación del driver de motor por SPI se pusieron en evidencia las siguientes ventajas:

- Reducir los requisitos temporales del procesador comparados con el método de pulso/dirección
- Alcanzar aceleración, desaceleración y velocidad constante sin fluctuaciones.
- Contar con micropasos seleccionables por software sin degradar el rendimiento.
- Encadenar drivers permitiendo comandar varios motores con la misma interfaz.
- Adaptar el driver a los requisitos del motor y máquina por software.

También se encontraron desventajas como:

- Baja resolución de máxima y mínima velocidad, aceleración y desaceleración: parámetros que degradan la performance en trayectorias de múltiples ejes.
- No contar con cola de comandos para ejecutar en secuencia y sin demoras.
- Tasa máxima de SPI insuficiente para más de 4 motores.

5.2. Próximos pasos

Si bien se lograron los objetivos planeados aún restan implementar características deseables y, basado en la experiencia obtenida, hay ciertos aspectos que justifican un nuevo análisis. Algunos de estos tópicos son:

- Reducir el tiempo entre sucesivos comandos para evitar detenciones.

- Realizar un cálculo adelantado de los comandos a ser enviados para sucesivos movimientos.
- Implementar los esquemas de *camino exacto* y *modo continuo* que permitirán extender el tipo de máquinas y trabajos posibles de realizar.
- Desarrollar el driver de sensores, fines de carrera y manejo del motor de corte.
- Implementar el almacenamiento interno de archivos GCode para la ejecución autónoma del maquinado.
- Documentar la API completa para permitir el desarrollo del software de control multiplataforma.

Apéndice A

GCode utilizados en los ensayos

El siguiente es el GCode que describe el recorrido de la figura geométrica del logo openhard, pero con una componente en el eje Z.

```

1 G1 153901 89540 5000
2 G1 153901 89540 -2000
3 G1 171040 47563 -3000
4 G1 179666 50972 -4000
5 G1 182029 51048 -5000
6 G1 184609 50023 -6000
7 G1 205723 35738 -7000
8 G1 221722 51737 -8000
9 G1 206878 73948 -9000
10 G1 206329 76248 -10000
11 G1 206815 78981 -11000
12 G1 217092 103395 -12000
13 G1 218710 105120 -13000
14 G1 221258 106219 -14000
15 G1 246290 111048 -15000
16 G1 246290 133674 -16000
17 G1 220087 138883 -17000
18 G1 218073 140122 -18000
19 G1 216484 142398 -19000
20 G1 206488 166928 -20000
21 G1 206412 169291 -21000
22 G1 207437 171871 -22000
23 G1 221722 192985 -23000
24 G1 205723 208984 -24000
25 G1 183512 194140 -25000
26 G1 181212 193591 -26000
27 G1 178479 194077 -27000
28 G1 154065 204354 -28000
29 G1 152340 205972 -29000
30 G1 151241 208520 -30000
31 G1 146412 233552 -31000
32 G1 123786 233552 -32000
33 G1 118577 207349 -33000
34 G1 117338 205335 -34000
35 G1 115062 203746 -35000
36 G1 90532 193750 -36000
37 G1 88169 193674 -37000
38 G1 85589 194699 -38000
39 G1 64475 208984 -39000
40 G1 48476 192985 -40000
41 G1 63320 170774 -41000
42 G1 63869 168473 -42000
43 G1 63383 165741 -43000
44 G1 53106 141327 -44000
45 G1 51488 139602 -45000

```

```

46 G1 48940 138503 -46000
47 G1 23909 133674 -47000
48 G1 23909 111048 -48000
49 G1 50111 105839 -49000
50 G1 52125 104600 -50000
51 G1 53714 102324 -51000
52 G1 63710 77794 -52000
53 G1 63786 75431 -53000
54 G1 62761 72851 -54000
55 G1 48476 51737 -55000
56 G1 64475 35738 -56000
57 G1 86686 50582 -57000
58 G1 88987 51131 -58000
59 G1 91719 50645 -59000
60 G1 99158 47563 -60000
61 G1 116297 89540 -61000
62 G1 116297 89540 5000

```

Se lista a continuación el GCode utilizado en los ensayos de velocidad:

```

1 F 10
2 G1 1000 0 0
3 F 20
4 G1 1000 1000 0
5 F 30
6 G1 1000 1000 0
7 F 40
8 G1 1000 1000 1000
9 F 50
10 G1 0 0 0

```

Se lista a continuación el GCode utilizado en los ensayos de aceleración:

```

1 G1 20000 0 0
2 G1 0 0 0
3 G1 20000 0 0
4 G1 0 0 0
5 G1 0 20000 0
6 G1 0 0 0
7 G1 0 20000 0
8 G1 0 0 0
9 G1 0 0 20000
10 G1 0 0 0
11 G1 0 0 20000
12 G1 0 0 0

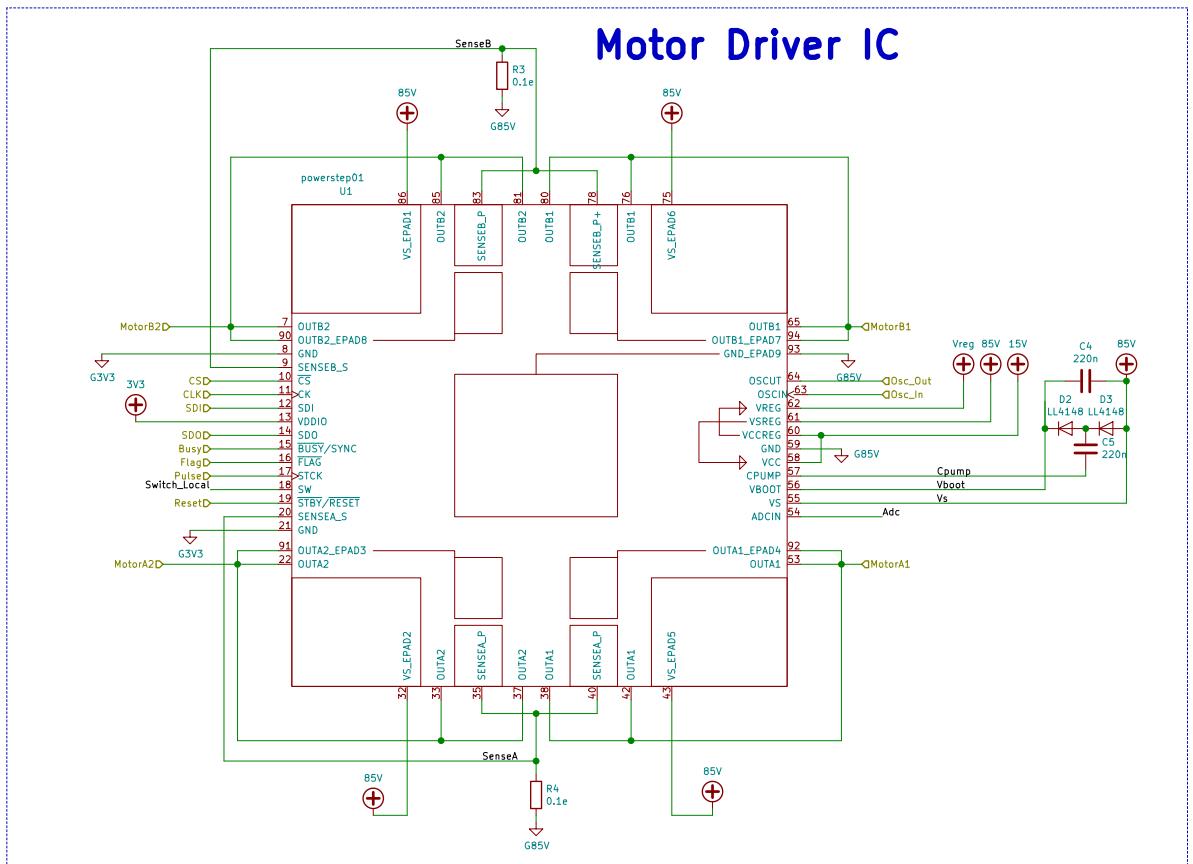
```

Apéndice B

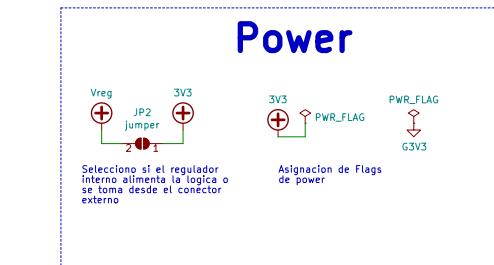
Esquemáticos

B.1. Esquemático de PCB de driver de motores

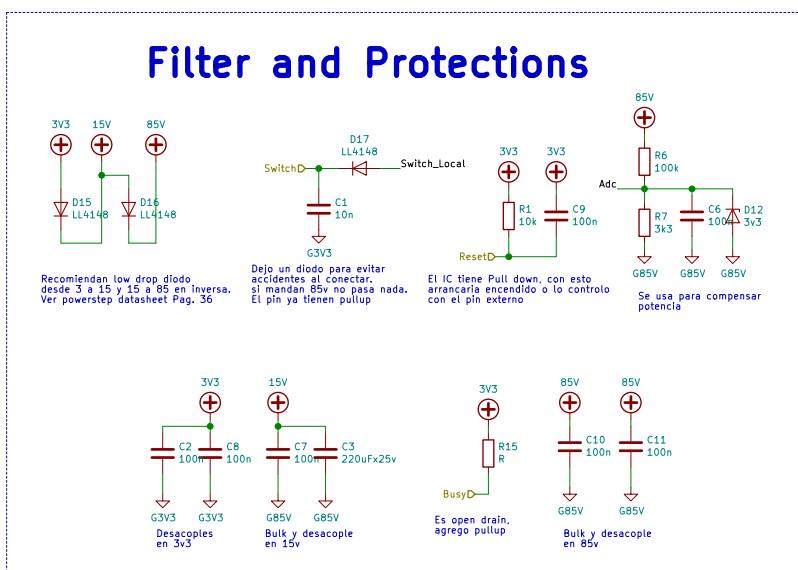
Se detalla a continuación lo mas relevante del esquemático del driver de motor desarrollado como material de consulta para alguno de los análisis realizados en esta memoria. El proyecto completo puede consultarse en https://github.com/pslavkin/cese_kicad.git.



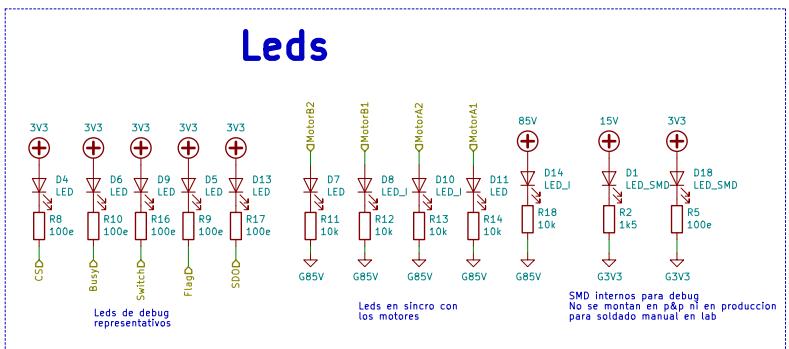
Power



Filter and Protections



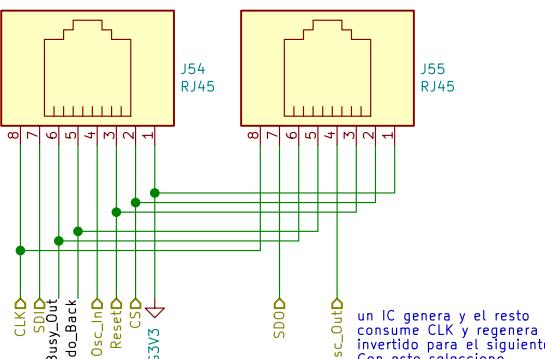
Leds



Pablo Slavkin
disenioconingenio

Sheet: /motor/
File: motor.sch
Title: PaP
Size: A3 | Date: 2018-09-21
KiCad E.D.A. kicad 5.0.0-fee4fd166ubuntu16.04.1 | Rev: 1.0
Id: 2/4

Spi Daisy Chain



Decido hacer el cruce de SPI en las fichas y no en el cable, de modo que los cables son siempre rectos

Pins Select

JP51
jumper
Busy_D₂ D₁ Busy_Out

si busy va de step, no va en cadena SPI, sino en conector individual

JP52
jumper
Sdo_Back D₂ D₁ SDO

con este puente se evita el uso de un stub externo

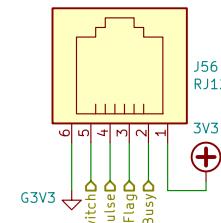
Motor

J52 Conn_01x02 2 MotorB1
1 MotorB2

J53 Conn_01x02 2 MotorA2
1 MotorA1

en dos conectores para facilitar el ruteo y que no se pueda conectar la alimentación por error

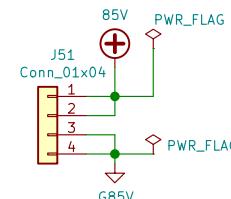
Parallel Conn



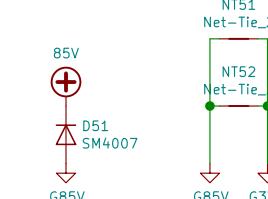
El fin de carrera y el step es individual por modulo asi que va por fuera de la cadena SPI

El busy si se usa como salida para manejar pulso a otro driver es individual

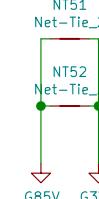
Main Power



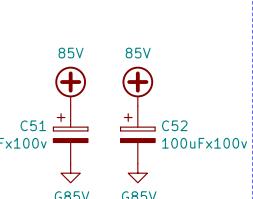
Contemplo doble pin del conector para positivo y doble para negativo para permitir encadenar alimentacion y/o suplir mas corriente



protección inversion de polaridad



Se conectan desde 2 pines de g3v3 a 1 de g85v



según pruebas 200uF es suficiente Uso 2 de 100 para que no quede tan alto

Pablo Slavkin
[diseñoconingenio](#)

Sheet: /connector/
File: connector.sch

Title: PaP

Size: A4 Date: 2018-09-21
KiCad E.D.A. kicad 5.0.0-fee4fd166ubuntu16.04.1

Rev: 1.0
Id: 3/4

Bibliografía

- [1] URL: STMicroelectronics https://www.st.com/content/st_com/en/products/motor-drivers/stepper-motor-drivers/powerstep01.html.
- [2] Thomas Dickey. 2018. URL: <https://www.gnu.org/software/ncurses/>.
- [3] Adam Dunkels. *lwIP*. 2018. URL: http://www.nongnu.org/lwip/2_1_x/index.html.
- [4] Fabricante de maquinas CNC. URL: <https://wolfcut.es>.
- [5] Herramienta portable para la generación de gráficos. URL: <http://www.gnuplot.info/>.
- [6] IEEE. Amazon FreeRTOS. Amazon, 2018. URL: <https://aws.amazon.com/freertos/>.
- [7] *RhinoCam, plugin de Rhino para generacion de trabajos de maquinado por CNC*. URL: <https://mecsoft.com/rhinocam-software/>.
- [8] *Rhinoceros, CAD para desarrollo de figuras 3D en general*. URL: <https://www.rhino3d.com/>.
- [9] RS274 interpreter. URL: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=823374.
- [10] ST Microelectronic. URL: <https://www.st.com>.
- [11] STMicroelectronics. *AN4290 Application note*. URL: https://www.st.com/content/ccc/resource/technical/document/application_note/d7/dd/a8/f3/db/3e/49/6f/DM00082126.pdf/files/DM00082126.pdf/jcr:content/translations/en.DM00082126.pdf.
- [12] STMicroelectronics. *System-in-package integrating microstepping controller and 10 A power MOSFETs*. Nov. de 2017. URL: <https://www.st.com/resource/en/datasheet/powerstep01.pdf>.
- [13] Texas Instruments. URL: <https://www.ti.com>.