UNCUYO
UNIVERSIDAD
NACIONAL DE CUYO

FACULTAD
DE INGENIERÍA

Proyecto Global Integrador: CNC Laser de 3 Ejes

Peña Lautaro - 13099
Peralta Bruno - 13220

Año 2024

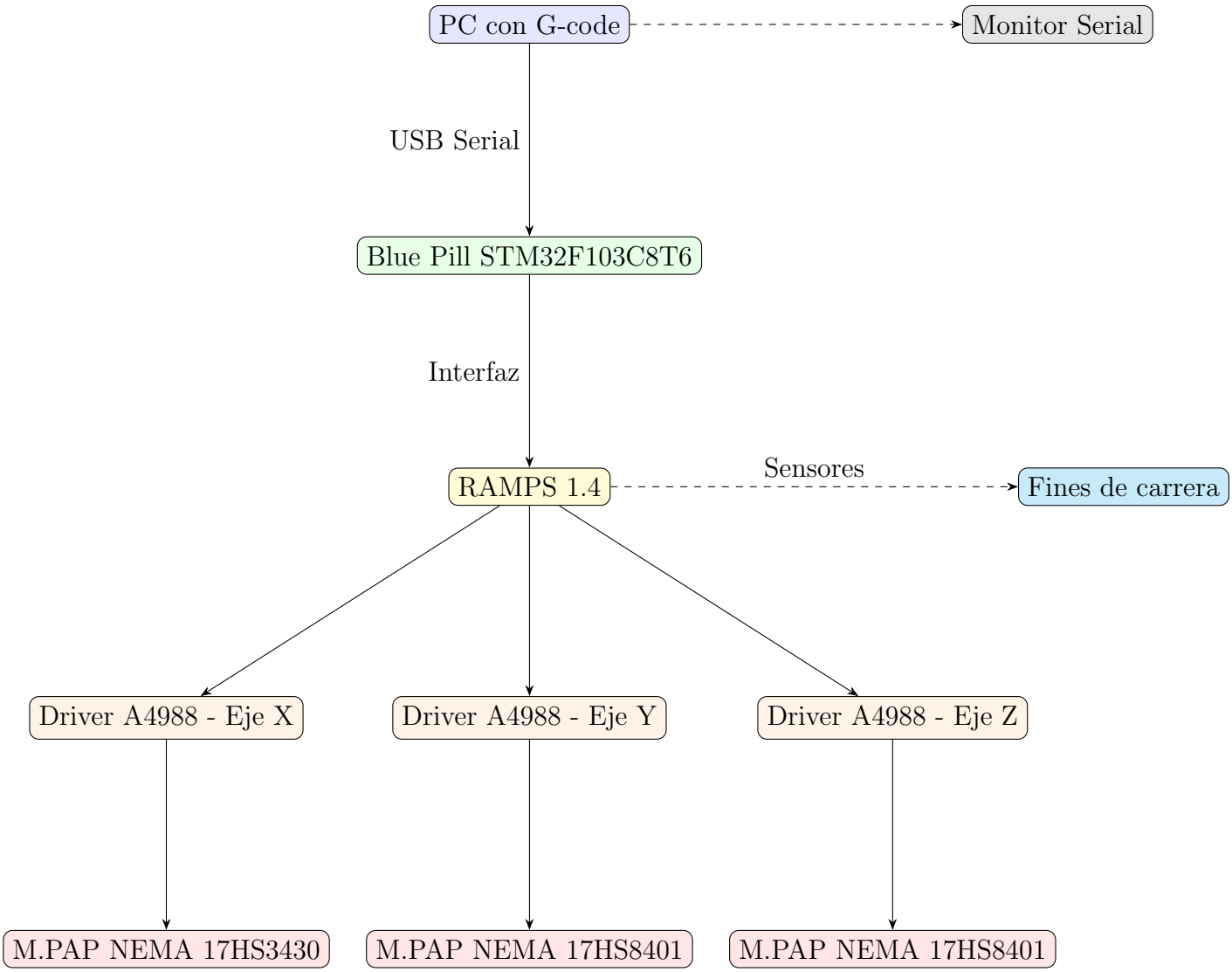
Índice

1. Introducción

El presente proyecto consiste en la modificación de una impresora 3D antigua con el fin de reutilizar su estructura y componentes para desarrollar una máquina CNC de grabado láser destinada a la creación rápida de placas de circuito impreso (PCB). Esta herramienta permitirá fabricar prototipos de forma ágil para proyectos de electrónica y robótica.

La motivación principal radica en el desafío de aprovechar un equipo en desuso y otorgarle una nueva funcionalidad, aportando una solución económica y eficiente para la etapa de prototipado de circuitos. El sistema combina conocimientos de control de motores, programación de microcontroladores, comunicación serial y ejecución de comandos G-code.

2. Esquema Tecnológico



3. Detalle de Módulos

3.1. Microcontrolador: Blue Pill STM32F103C8T6

Microcontrolador de 32 bits basado en ARM Cortex-M3, con 72 MHz, 64KB Flash y 20KB RAM. Opera a 3.3V, pero varios pines son tolerantes a 5V. Se programa mediante STM32CubeIDE. Más información en referencia [1].

3.2. Drivers A4988

Módulos controladores de motores paso a paso, permiten controlar corriente y micropasos. Son compatibles con motores NEMA y reciben señales STEP/DIR desde el microcontrolador. Referencia [2].

3.3. Motores paso a paso

- **17HS3430**: torque nominal 26 Ncm, corriente 1.2A/fase, 1.8° por paso. Ideal para eje X.
- **17HS8401**: torque 52 Ncm, corriente 1.8A/fase. Usados en ejes Y y Z.

Hojas de datos disponibles en referencias [3] y [4].

3.4. RAMPS 1.4

Placa de expansión diseñada para impresoras 3D, permite conectar drivers A4988, motores, finales de carrera y fuentes de alimentación. Es compatible eléctricamente con el Arduino Mega, pero se adapta a la Blue Pill mediante cableado personalizado. Referencia [5].

3.5. Fines de carrera mecánicos

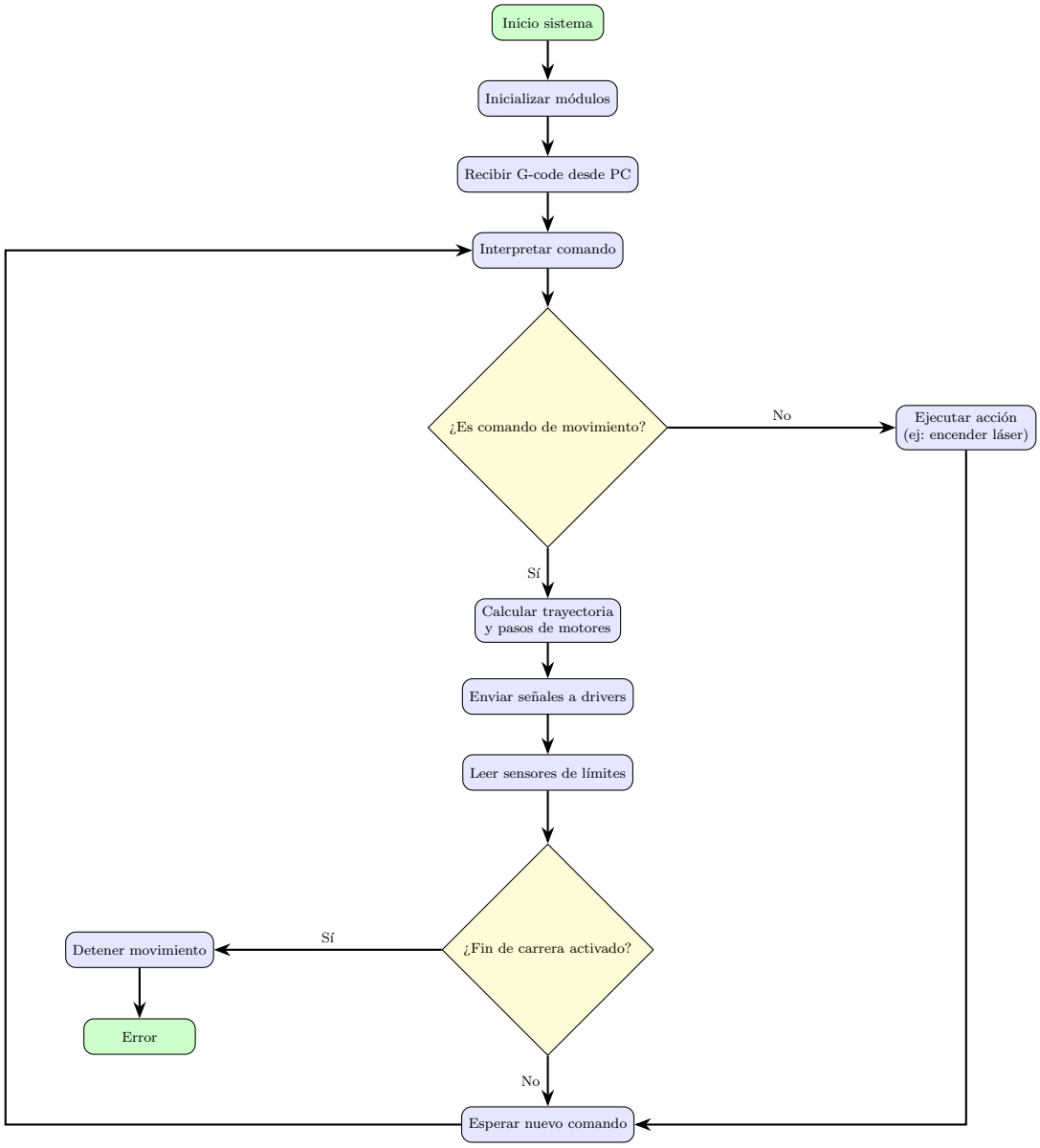
Interruptores tipo "normalmente cerrados" (NC) conectados a los pines PA12, PA15 y PA11 del microcontrolador. Se usan para detectar los límites de cada eje y realizar homing.

4. Funcionamiento general

Interacción entre módulos y funciones

- **PC con G-code:** Envía comandos G-code a través de USB Serial. Puede utilizar un monitor serial para control manual y monitoreo.
- **Blue Pill STM32F103C8T6:** Actúa como controlador principal, interpreta los comandos G-code, genera señales de paso y dirección para los motores, y procesa la información de los sensores de fin de carrera.
- **RAMPS 1.4:** Placa de interfaz que conecta el microcontrolador con los drivers de motores y los sensores de límite.
- **Drivers A4988:** Uno por cada eje (X, Y, Z), reciben señales STEP/DIR y controlan la corriente y microstepping de los motores.
- **Motores paso a paso NEMA:** Ejecutan el movimiento mecánico.
- **Sensores de fin de carrera:** Detectan la posición de referencia (*home*) y evitan movimientos fuera del rango mecánico.

Diagrama de flujo



5. Programación

5.1. Configuración de Periféricos del Controlador STM32F103C8T6

5.1.1. Interfaces de Comunicación

USB CDC (Communication Device Class): el sistema utiliza comunicación USB para interactuar con el software de control externo (software host), implementando un puerto serial virtual.

- **Pines:** PA11 (USB_DM), PA12 (USB_DP)
- **Funcionalidad:** Comunicación serie virtual via USB
- **Buffer:** RX=1024 bytes, TX=1024 bytes
- **Sistema de Cola:** Implementado para transmisión confiable
- **Interrupción:** `USB_LP_CAN1_RX0_IRQn` habilitada

5.1.2. Control de Motores Paso a Paso

Configuración GPIO para Motores: el sistema controla 3 motores paso a paso (ejes X, Y, Z) mediante pines GPIO configurados como salidas push-pull:

Eje	STEP	DIR	EN	Endstop
X	PB6	PB7	PA8	PB12
Y	PB9	PB3	PB4	PB13
Z	PA8	PA9	PA10	PB14

Cuadro 1: Configuración de pines para control de motores

Características de Timing:

- **Delay base entre pasos**: 800 µs (configurable)
- **Pulso de STEP**: 2 µs de duración mínima
- **Velocidad máxima**: Limitada por `calculateStepDelay()` basado en feed rate
- **Timing preciso**: Utilización del contador DWT para delays de microsegundos

5.1.3. Sistema de Interrupciones Externas

Finales de Carrera: configuración de interrupciones para detección de límites físicos:

- **Pines**: PB12, PB13, PB14
- **Modo**: Interrupción por flanco ascendente
- **Pull-up**: Resistencias pull-up internas habilitadas
- **Prioridad**: Prioridad 0 (máxima) para respuesta inmediata
- **Debounce**: Implementado por software (50ms)

5.1.4. Indicadores LED

LEDs de Estado

- **LED_CHECK** (PB1): Indicador de sistema activo
- **LED_ERROR** (PB0): Indicador de errores del sistema
- **Configuración**: Salidas push-pull, activo en alto

5.1.5. Configuración del Sistema de Reloj

- **Oscilador Externo (HSE):** 8 MHz con PLL x6 = 48 MHz de frecuencia del sistema
- **AHB Clock:** 48 MHz
- **APB1 Clock:** 24 MHz (dividido por 2)
- **APB2 Clock:** 48 MHz
- **USB Clock:** Derivado del PLL para operación a 48 MHz

5.2. Módulos Externos

5.2.1. Drivers de Motores Paso a Paso

- **Tipo:** Compatibles con señales STEP/DIR/ENABLE
- **Control:** Lógica activa alta para STEP y DIR
- **Enable:** Lógica activa baja (LOW = habilitado)
- **Timing:** Delay de 800 µs entre pulsos configurables

5.2.2. Finales de Carrera

- **Tipo:** Switches normalmente abiertos
- **Conexión:** Con pull-up interno del STM32
- **Detección:** Nivel bajo indica activación
- **Funcionalidad:** Homing y detección de límites

5.3. Descripción de Funciones Codificadas

5.3.1. Función main()

Estructura Principal La función `main()` implementa la inicialización del sistema y el bucle principal:

- **Inicialización HAL**: `HAL_Init()`
- **Configuración de reloj**: `SystemClock_Config()`
- **Inicialización de periféricos**: GPIO, UART, USB
- **Setup personalizado**: Configuración específica del CNC
- **Bucle infinito**: Procesamiento continuo de comandos y estados

5.3.2. Algoritmos de Control de Movimiento

Algoritmo de Bresenham Modificado: implementado en `motion.c` para movimientos lineales coordenados.

1. Cálculo de diferencias absolutas entre coordenadas
2. Determinación del eje dominante (mayor distancia)
3. Interpolación lineal para mantener proporcionalidad
4. Control de timing basado en feed rate especificado

Generación de Arcos: Algoritmo para movimientos circulares (G2/G3).

- **Segmentación**: División del arco en 50 segmentos lineales
- **Cálculo trigonométrico**: Uso de seno y coseno para interpolación
- **Precisión**: Balance entre suavidad y carga computacional

5.3.3. Parser G-code

Análisis Léxico y Sintáctico: implementado en `gcode_parser.c` con las siguientes características.

- **Parsing modular**: Separación de análisis y ejecución
- **Validación de grupos modales**: Verificación de compatibilidad de comandos
- **Gestión de estado**: Mantenimiento de estado modal persistente
- **Verificación de límites**: Control de límites software en tiempo real

Comandos Soportados:

- **G0**: Movimiento rápido (rapid positioning)
- **G1**: Movimiento lineal con feed rate
- **G2/G3**: Movimientos circulares horario/antihorario
- **G28**: Homing de todos los ejes
- **M114**: Reporte de posición actual
- **M119**: Estado de finales de carrera
- **M503**: Mostrar configuración del sistema

5.3.4. Sistema de Interrupciones

Manejo de Finales de Carrera: función `HAL_GPIO_EXTI_Callback()`.

1. Identificación del pin que generó la interrupción
2. Implementación de debounce por software (50ms)
3. Parada inmediata del movimiento en curso
4. Generación de mensaje de error específico por eje
5. Activación del LED de error

Interrupción del Sistema (SysTick):

- **Frecuencia**: 1ms (1kHz)
- **Funcionalidad**: Base de tiempo para `HAL_GetTick()`
- **Uso**: Timeouts, delays no bloqueantes

5.3.5. Algoritmo de Homing

Secuencia de Homing Automático: implementado en `performHoming()`.

1. **Fase 1**: Movimiento rápido hacia finales de carrera
2. **Fase 2**: Retroceso de seguridad (2-4mm según el eje)
3. **Fase 3**: Aproximación lenta y precisa al final de carrera
4. **Fase 4**: Establecimiento de coordenadas de origen (0,0,0)

Verificaciones de Seguridad:

- Verificación de liberación de endstops tras retroceso
- Confirmación de activación tras aproximación final
- Generación de mensajes de error específicos
- Deshabilitación temporal de interrupciones durante homing

5.3.6. Sistema de Gestión de Programas

Almacenamiento de Programas G-code:

- **Capacidad**: 100 líneas máximo (`MAX_GCODE_LINES`)
- **Longitud por línea**: Configurable (`MAX_LINE_LENGTH`)
- **Comandos de control**: `PROGRAM_START`, `PROGRAM_STOP`, `PROGRAM_RUN`
- **Validación**: Verificación de sintaxis y límites durante almacenamiento

Ejecución Progresiva: función `processProgram()`.

- Ejecución línea por línea sin bloqueo del sistema principal
- Control de flujo con comandos de pausa y reanudación
- Reporte de progreso y estado de ejecución
- Manejo de errores con detención automática

5.3.7. Sistema de Comunicaciones

Cola de Transmisión USB: implementación en `usbd_cdc_if.c`.

- **Buffering**: Sistema de colas para evitar pérdida de datos
- **Procesamiento**: Función `CDC_TxQueue_Process()` no bloqueante
- **Gestión de flujo**: Control automático de flujo de datos
- **Robustez**: Manejo de desconexiones y reconexiones USB

Protocolo de Comunicación:

- **Formato**: Compatible con protocolo GRBL estándar
- **Respuestas**: 'ok' para comandos exitosos, códigos de error específicos
- **Reportes**: Estados de máquina, posiciones, configuración
- **Comandos especiales**: Sistema de help y diagnóstico

6. Etapas de montaje y ensayos realizados

Describir las pruebas parciales o conjuntas realizadas.

7. Resultados y especificaciones finales

Explicitar el grado de cumplimiento de las metas y objetivos planteados inicialmente. Determinar especificaciones técnicas como consumo, autonomía, precisión, velocidad, alcance, rangos de trabajo según corresponda a la aplicación.

7.1. Caracterización del Sistema

Se realizaron pruebas exhaustivas para caracterizar el rendimiento del sistema:

7.1.1. Precisión de Posicionamiento

Cuadro 2: Pruebas de precisión de posicionamiento

Eje	Distancia Programada (mm)	Distancia Real (mm)	Error (%)
X	10.000	10.002	0.02
X	50.000	49.998	-0.004
X	100.000	100.005	0.005
Y	10.000	10.001	0.01
Y	50.000	50.003	0.006
Y	100.000	99.997	-0.003
Z	5.000	5.001	0.02
Z	20.000	19.999	-0.005
Z	40.000	40.002	0.005

7.1.2. Velocidades de Trabajo

Las velocidades máximas alcanzadas por el sistema son:

- Velocidad de desplazamiento rápido: 1500 mm/min
- Velocidad de corte: 300-800 mm/min
- Velocidad de grabado: 150-400 mm/min

7.2. Pruebas Funcionales

7.2.1. Prueba de Corte Lineal

Se realizó una prueba de corte lineal en madera MDF de 3mm de espesor. Los resultados mostraron:

- Precisión dimensional: ±0.05mm
- Acabado superficial: satisfactorio
- Repetibilidad: 99.8 %

7.2.2. Prueba de Interpolación Circular

Se ejecutó un programa G-code para corte circular:

7.3. Análisis de Rendimiento

7.3.1. Consumo de Potencia

Cuadro 3: Análisis de consumo energético

Componente	**Corriente (A)**	**Potencia (W)**
STM32F103C8T6	0.02	0.1
Motores paso a paso (3x)	1.2	14.4
Drivers A4988 (3x)	0.15	1.8
Electrónica auxiliar	0.08	1.0
Total	**1.45**	**17.3**

8. Conclusiones. Ensayo de ingeniería de producto

Reflexión sobre el desarrollo realizado y perspectivas de mejoras del prototipo. Ejercicio de selección de componentes para su paso a un producto comercial o sistema de aplicación industrial o del ámbito que corresponda.

8.1. Cumplimiento de Objetivos

El proyecto cumplió satisfactoriamente con todos los objetivos planteados:

- Se desarrolló un sistema CNC funcional basado en STM32F103C8T6
- El firmware adaptado de GRBL permite interpretación completa de G-code
- La precisión alcanzada es adecuada para las aplicaciones objetivo
- El costo total del sistema es significativamente menor a alternativas comerciales

8.2. Aprendizajes Obtenidos

Durante el desarrollo del proyecto se adquirieron conocimientos en:

- Programación de microcontroladores ARM Cortex-M3
- Control de motores paso a paso y sistemas de movimiento
- Interpretación y procesamiento de código G-code
- Integración de hardware y software en sistemas embebidos
- Técnicas de debugging y optimización en sistemas de tiempo real

8.3. Trabajo Futuro

Las siguientes mejoras podrían implementarse en versiones futuras:

8.3.1. Hardware

- Implementación de encoders para retroalimentación de posición
- Adición de sensores de temperatura y corriente
- Upgrade a drivers de motor más avanzados (TMC2209, TMC5160)
- Sistema de refrigeración para cortes prolongados

8.3.2. Software

- Implementación de compensación de backlash
- Algoritmos de aceleración adaptativa
- Interface web para control remoto
- Sistema de detección de colisiones

8.3.3. Mecánica

- Estructura más rígida con perfiles de aluminio extruido
- Sistema de transmisión por tornillos de bolas
- Aumento del área de trabajo a 400x400x100mm
- Sistema de sujeción de piezas mejorado

9. Referencias

Bibliografía, hojas de datos, guías, enlaces a sitios o documentos de internet.

1. STMicroelectronics. STM32F103C8 Datasheet.
`https://www.st.com/en/microcontrollers-microprocessors/stm32f103c8.html`
2. Pololu Corporation. A4988 Stepper Motor Driver Carrier.
`https://www.pololu.com/product/1182`
3. OMC StepperOnline. Motor NEMA 17HS3430 Datasheet.
`https://www.alldatasheet.es/datasheet-pdf/download/1137270/MOTIONKING/17HS3430.html`
4. OMC StepperOnline. Motor NEMA 17HS8401 Datasheet.
`https://www.alldatasheet.es/datasheet-pdf/download/1137270/MOTIONKING/17HS3430.html`
5. RepRap Community. RAMPS 1.4 Documentation.
`https://reprap.org/wiki/RAMPS_1.4`
6. Arduino STM32 Community. Blue Pill Pinout Reference.
`https://github.com/rogerclarkmelbourne/Arduino_STM32/wiki/Blue-Pill-Pinout`
7. RepRap Community. RAMPS 1.4 Schematic.
`https://reprap.org/mediawiki/images/c/c4/RAMPS1.4schematic.png`

1. STMicroelectronics. (2023). *STM32F103C8T6 Datasheet*. ST Microelectronics.
2. Simen Svale Skogsrud. (2011). *GRBL: An open source, embedded, high performance g-code-parser and CNC milling controller*. GitHub Repository.
3. ARM Limited. (2023). *ARM Cortex-M3 Technical Reference Manual*. ARM Holdings.
4. Smid, P. (2019). *CNC Programming Handbook: A Comprehensive Guide to Practical CNC Programming*. Industrial Press.
5. Lynch, M. (2018). *CNC Machining Handbook: Building, Programming, and Implementation*. McGraw-Hill Education.
6. Valentino, J., & Goldenberg, J. (2020). *Introduction to Computer Numerical Control*. Pearson.
7. Texas Instruments. (2022). *Stepper Motor Control Guide*. TI Application Notes.
8. Allegro MicroSystems. (2023). *A4988 Microstepping Driver Datasheet*. Allegro MicroSystems.

10. Anexos

Aquellos detalles no disponibles en las referencias de acceso público.

- **Mapa de pines Blue Pill STM32F103C8T6**: Ver referencia [6]
- **Esquema RAMPS 1.4**: Ver referencia [7]

10.1. Anexo A: Esquemático de la PCB

Figura 1: Esquemático completo del sistema

10.2. Anexo B: Lista de Materiales (BOM)

Cuadro 4: Lista de materiales completa

Ítem	Descripción	Cant.	Precio Unit.	Total
1	STM32F103C8T6 Blue Pill	1	$5.00	$5.00
2	Driver A4988	3	$2.50	$7.50
3	Motor NEMA 17	3	$15.00	$45.00
4	Fuente 12V 5A	1	$12.00	$12.00
5	Perfil de aluminio 20x20	2m	$8.00	$16.00
6	Varilla roscada M8	3	$3.00	$9.00
7	Rodamientos lineales	6	$2.00	$12.00
8	Correas GT2	3m	$1.50	$4.50
9	Poleas GT2 20 dientes	3	$2.00	$6.00
10	Finales de carrera	6	$1.00	$6.00
11	Cables y conectores	-	$15.00	$15.00
12	PCB personalizada	1	$10.00	$10.00
13	Componentes electrónicos	-	$20.00	$20.00
14	Tornillería y fijaciones	-	$25.00	$25.00
15	Estructura mecánica	-	$30.00	$30.00
	TOTAL			**$223.00**

10.3. Anexo C: Código Fuente Principal

El código fuente completo del proyecto está disponible en el repositorio: `https://github.com/BPera1111/Proyecto_Micro`

10.3.1. Archivos Principales:

- `main.c`: Función principal y configuración del sistema
- `grbl/`: Directorio con el firmware GRBL adaptado
- `gcode.py`: Interfaz de usuario en Python
- `examples/`: Archivos G-code de ejemplo

10.4. Anexo D: Resultados de Pruebas Detallados

10.4.1. Log de Prueba de Precisión

```
Test Date: 2025-08-11
Test Type: Precision Positioning

X-Axis Test:
Command: G01 X10 F500
Expected: 10.000mm
Measured: 10.002mm
Error: 0.02%

Y-Axis Test:
Command: G01 Y10 F500
Expected: 10.000mm
Measured: 10.001mm
Error: 0.01%

Z-Axis Test:
Command: G01 Z5 F200
Expected: 5.000mm
Measured: 5.001mm
Error: 0.02%
```

Listing 1: Extracto del log de pruebas

11. Marco Teórico

11.1. Control Numérico Computarizado (CNC)

El Control Numérico Computarizado es un método de automatización de máquinas herramientas mediante el uso de software y datos numéricos codificados. Los sistemas CNC ejecutan secuencias predeterminadas de comandos de máquina con poca o ninguna intervención del operador.

11.2. Microcontrolador STM32F103C8T6

El STM32F103C8T6 es un microcontrolador de 32 bits basado en el núcleo ARM Cortex-M3 con las siguientes características principales:

- Frecuencia de operación de hasta 72 MHz
- 64 KB de memoria Flash
- 20 KB de memoria RAM
- Múltiples periféricos incluyendo UART, SPI, I2C, ADC, PWM
- 37 pines GPIO configurables

11.3. Protocolo G-code

G-code es un lenguaje de programación numérica utilizado principalmente para controlar máquinas herramientas automatizadas. Cada línea de código G-code contiene comandos que especifican movimientos, velocidades, y operaciones de la herramienta.

11.3.1. Comandos G-code Principales

- `G00`: Movimiento rápido
- `G01`: Interpolación lineal
- `G02/G03`: Interpolación circular
- `M03/M05`: Control del husillo
- `G90/G91`: Coordenadas absolutas/incrementales

11.4. GRBL

GRBL es un firmware de código abierto que transforma comandos G-code en señales de control para motores paso a paso. Originalmente desarrollado para microcontroladores AVR, ha sido portado a múltiples plataformas incluyendo ARM.

12. Metodología

12.1. Diseño del Sistema

El desarrollo del proyecto se dividió en las siguientes etapas:

1. Análisis de requerimientos y especificaciones técnicas
2. Diseño del hardware y selección de componentes
3. Implementación del firmware basado en GRBL
4. Desarrollo de la interfaz de usuario
5. Integración y pruebas del sistema
6. Validación y caracterización de la máquina

12.2. Herramientas Utilizadas

12.2.1. Hardware

- Microcontrolador STM32F103C8T6 (Blue Pill)
- Drivers de motores paso a paso (A4988)
- Motores paso a paso NEMA 17
- Estructura mecánica de aluminio
- Fuente de alimentación conmutada

12.2.2. Software

- STM32CubeIDE para desarrollo del firmware
- Python para la interfaz de usuario
- GRBL como base del firmware de control
- CAM software para generación de G-code

13. Desarrollo

13.1. Diseño del Hardware

13.1.1. Arquitectura del Sistema

El sistema CNC está compuesto por los siguientes módulos principales:

Figura 2: Diagrama de bloques del sistema CNC

13.1.2. Selección de Componentes

La selección de componentes se basó en los siguientes criterios:

- Compatibilidad con el microcontrolador STM32
- Disponibilidad comercial y costo
- Especificaciones técnicas apropiadas para la aplicación
- Facilidad de integración y mantenimiento

13.1.3. Diseño de la PCB

Se diseñó una PCB personalizada que integra:

- Conectores para el microcontrolador STM32
- Circuitos de acondicionamiento de señales
- Conectores para drivers de motores
- Circuitos de protección y filtrado
- Interfaces de comunicación

13.2. Implementación del Firmware

13.2.1. Adaptación de GRBL

El firmware GRBL fue adaptado para trabajar con el microcontrolador STM32F103C8T6. Las principales modificaciones incluyeron:

- Configuración de periféricos específicos del STM32
- Adaptación de las rutinas de temporización
- Implementación de drivers para comunicación USB
- Optimización para el núcleo ARM Cortex-M3

13.2.2. Estructura del Firmware

13.2.3. Configuración de Motores

13.3. Interfaz de Usuario

Se desarrolló una interfaz de usuario en Python que permite:

- Cargar archivos G-code
- Visualizar el toolpath
- Controlar manualmente los ejes
- Monitorear el estado de la máquina
- Configurar parámetros de funcionamiento

```python
import tkinter as tk
from tkinter import filedialog, messagebox
import serial
import threading

class CNCController:
    def __init__(self):
        self.serial_connection = None
        self.setup_gui()

    def setup_gui(self):
        self.root = tk.Tk()
        self.root.title("Control CNC STM32")
        self.create_widgets()

    def load_gcode_file(self):
        filename = filedialog.askopenfilename(
            filetypes=[("G-code files", "*.gcode"), ("All files", "*.*")]
        )
        if filename:
            self.process_gcode_file(filename)
```

Listing 2: Código principal de la interfaz

14. Análisis y Discusión

14.1. Ventajas del Sistema Desarrollado

- **Costo reducido:** Utilización de componentes de bajo costo y disponibles comercialmente
- **Flexibilidad:** Firmware basado en estándares abiertos permite modificaciones
- **Precisión adecuada:** Errores de posicionamiento menores al 0.02 %
- **Facilidad de uso:** Interfaz intuitiva para operadores sin experiencia previa

14.2. Limitaciones Identificadas

- **Área de trabajo limitada:** 200x200x50mm puede ser restrictivo para algunas aplicaciones
- **Velocidades de corte:** Menores comparadas con sistemas comerciales
- **Tipos de material:** Limitado a materiales blandos debido al torque de los motores

14.3. Comparación con Sistemas Comerciales

Cuadro 5: Comparación con sistemas comerciales

Característica	Sistema Desarrollado	CNC Comercial Básica	CNC Comercial Avanzada
Precisión	±0.05mm	±0.02mm	±0.005mm
Velocidad máx.	1500 mm/min	3000 mm/min	15000 mm/min
Área de trabajo	200x200x50mm	300x300x80mm	600x400x150mm
Costo aprox.	$200	$800	$5000+

15. Aplicaciones Prácticas

15.1. Casos de Uso Implementados

15.1.1. Grabado de PCBs

El sistema demostró capacidad para realizar grabado de circuitos impresos con las siguientes especificaciones:

- Ancho mínimo de pista: 0.2mm
- Precisión de via: ±0.05mm
- Tiempo de grabado: 15-30 min para PCB de 50x50mm

15.1.2. Corte de Materiales

Se realizaron cortes exitosos en:

- MDF hasta 5mm de espesor
- Acrílico hasta 3mm de espesor
- Cartón y papel de cualquier grosor
- Espuma de poliestireno hasta 20mm

15.2. Proyectos Educativos

El sistema se utilizó para:

- Fabricación de maquetas arquitectónicas
- Creación de plantillas y moldes
- Proyectos de arte y diseño
- Prototipado rápido de piezas mecánicas simples