



## **Mini CNC potter usando microcontrolador**

**Curso:** Introducción a sistemas embebidos

**Docente:** Francisco Camacho

### **Integrantes:**

Cesar Bryan Ruiz Villavicencio  
Geraldine Mabel Victorio Huacho  
Carlos Alberto Leon Goicochea

- **Introducción:**

**Problemática:**

La problemática identificada se centra en la necesidad de contar con un sistema compacto, de bajo costo y fácil de implementar que permita realizar trazados precisos, automatizados y repetibles. En muchos entornos educativos y de prototipado rápido no se dispone de herramientas CNC accesibles que faciliten procesos como dibujo técnico, marcado de piezas, o experimentación con control numérico.

Adicionalmente, los métodos manuales suelen presentar errores humanos, baja precisión y poca repetibilidad, lo que limita la calidad del trabajo y dificulta el aprendizaje de conceptos de automatización.

Por ello, se plantea el desarrollo de una mini CNC plotter basada en un microcontrolador, capaz de ejecutar movimientos controlados mediante instrucciones digitales, aumentando la precisión, reduciendo el esfuerzo del usuario y ofreciendo una plataforma económica para el aprendizaje de control de motores, electrónica y programación.

**Investigaciones anteriores:**

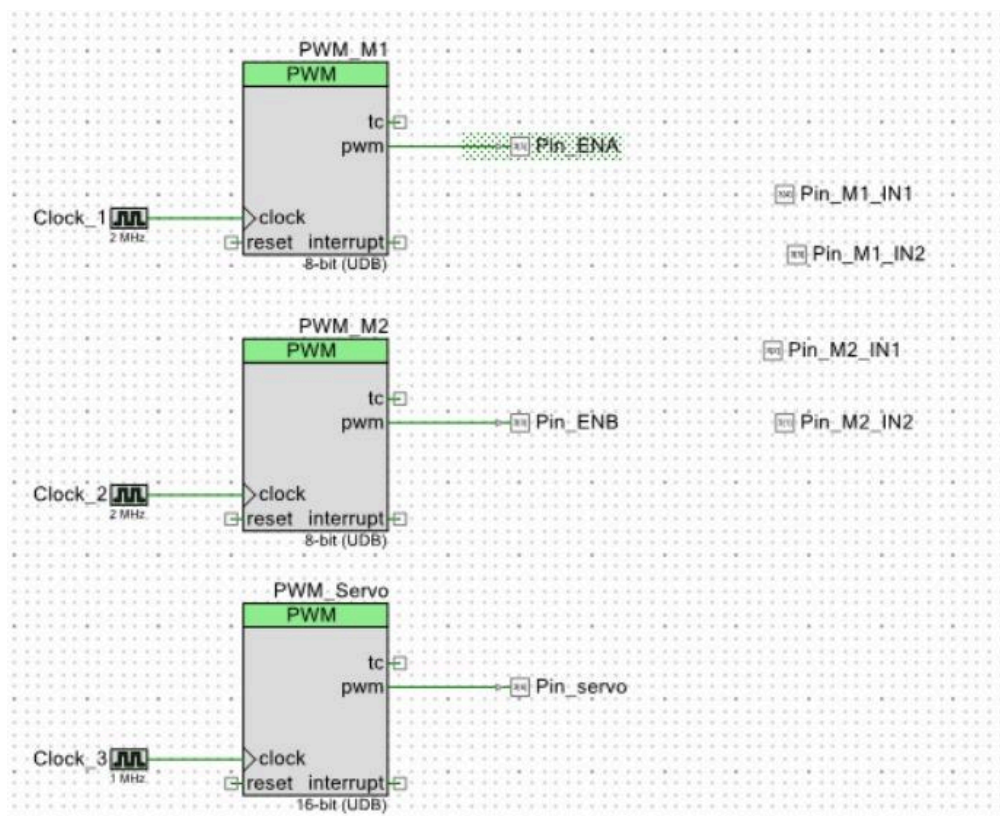
Para sustentar técnicamente el desarrollo de la mini CNC plotter controlada mediante un microcontrolador, se realizó una investigación exhaustiva de diversas fuentes académicas, técnicas y de la comunidad maker. En primer lugar, se analizó el artículo “2D CNC Plotter Integrated with Arduino”, publicado en el *Journal of Microprocessor and Microcontroller Research*, donde se explica la construcción de una CNC de bajo costo basada en motores paso a paso y la utilización de un Arduino para gestionar el control de posición, velocidad y secuencias de movimiento. Esta fuente permitió comprender la lógica fundamental del control numérico y los requisitos mínimos de hardware para implementar un sistema de trazado automatizado. Asimismo, el trabajo “Design and Implementation of 2D CNC Plotter Using Arduino”, publicado en el *International Journal of Research and Scientific Innovation (IJRSI)*, ofreció un enfoque más detallado sobre el diseño mecánico, el uso de controladores de motor como el A4988 y los procesos de interpretación de instrucciones G-code, lo cual resultó clave para establecer los principios de funcionamiento de la máquina y la interacción entre software y hardware. Para complementar la teoría con referencias prácticas, se revisó el repositorio *CNC-pen-plotter* disponible en GitHub, que proporciona código fuente, esquemas eléctricos y detalles de construcción de un plotter similar, permitiendo comparar el sistema de control utilizado en este proyecto con implementaciones reales ya probadas. Además, se tomaron como referencia proyectos ampliamente documentados de plataformas maker como Hackster.io y Hackaday.io, entre ellos “Arduino Based Mini CNC 2D Plotter” y “Mini Arduino CNC”, que presentan guías paso a paso para construir

máquinas CNC compactas a partir de componentes reciclados o económicos, destacando soluciones prácticas para la estructura mecánica, calibración de ejes y control de precisión. En conjunto, todas estas fuentes aportaron un panorama amplio sobre el estado actual de las CNC de pequeño formato, permitiendo identificar buenas prácticas, ventajas, limitaciones y tendencias de diseño que fueron esenciales para desarrollar un prototipo funcional, económico y adecuado a los objetivos del proyecto.

- **Desarrollo:**

1. Diagrama de bloques:

Psoc:



En el diseño del PSOC, cada módulo PWM está conectado de manera estratégica para controlar los actuadores de la CNC plotter. El bloque PWM\_M1 se enlaza al pin Pin\_ENA, encargado de habilitar y regular la velocidad del motor del eje X, mientras que las señales Pin\_M1\_IN1 y Pin\_M1\_IN2 controla el sentido de giro mediante un puente H externo. De forma similar, el bloque PWM\_M2 se conecta al pin Pin\_ENB, controlando la velocidad del motor del eje Y, y sus señales Pin\_M2\_IN1 y Pin\_M2\_IN2 determinan la dirección de movimiento. Para el control del eje Z, el módulo PWM\_Servo está conectado directamente a Pin\_servo, desde donde se generan pulsos de 1 a 2

ms necesarios para posicionar el SG90. Cada PWM utiliza un clock independiente configurado a 2 MHz o 3 MHz según el nivel de resolución requerido, garantizando precisión tanto en los motores lineales como en el servo. Estas conexiones permiten que el PSoC controle de forma coordinada los tres ejes de la CNC, asegurando movimientos estables, precisos y totalmente sincronizados con la lógica del programa.

## 2. Componentes:

1. **Lectoras de DVD:** Los mecanismos de las lectoras de DVD son fundamentales para la mini CNC porque contienen motores lineales precisos y guías que permiten movimientos controlados en los ejes X y Y. Estos sistemas ya están diseñados para desplazarse suavemente y con alta resolución, por lo que reutilizarlos reduce costos y facilita la construcción de una estructura mecánica compacta y ligera. Además, su diseño permite obtener movimientos repetitivos, ideales para trabajos de trazado y dibujo donde se requiere exactitud en la posición del lapicero.



2. **Servo sg90:** El servo SG90 es importante en la mini CNC porque se utiliza para mover el eje Z, permitiendo subir y bajar el lapicero con precisión. Gracias a su control mediante señales PWM, este servo puede posicionarse de forma rápida y exacta, lo que asegura que el lápiz ejerza la presión adecuada sobre la superficie o se levante totalmente cuando sea necesario. Su tamaño reducido, bajo consumo y facilidad de control lo convierten en la opción ideal para mecanismos de elevación en CNC pequeñas.



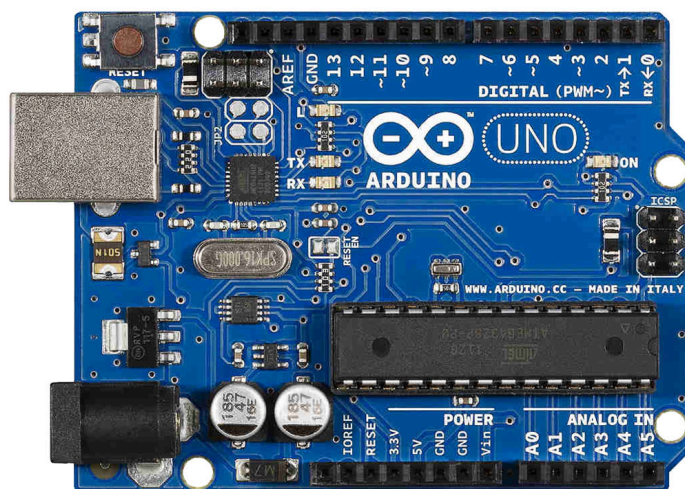
- 3. Lapicero:** El lapicero actúa como la herramienta principal de dibujo dentro de la mini CNC, ya que es el encargado de realizar los trazos sobre la superficie de trabajo. Es un elemento ligero, fácil de fijar y reemplazar, y permite obtener líneas definidas sin requerir un sistema de corte más complejo. Su simplicidad lo hace adecuado para pruebas de precisión, diseños gráficos y demostraciones de funcionamiento del sistema CNC, manteniendo el proyecto accesible y seguro.



- 4. Triplay:** El triplay funciona como la base estructural de la mini CNC, proporcionando un soporte rígido y estable para todos los componentes mecánicos y electrónicos. Su resistencia a vibraciones y su facilidad para ser perforado o modificado permiten montar los motores, guías y tarjetas de control de manera segura. Al brindar una plataforma firme, el triplay mejora la precisión del movimiento y evita desalineamientos durante la operación del trazado.



5. **Arduino uno:** El Arduino Uno es esencial porque actúa como el controlador principal de la mini CNC, procesando los comandos enviados desde el software y generando las señales necesarias para accionar los motores y el servo. Su arquitectura simple, amplia documentación y facilidad de programación lo convierten en una plataforma ideal para proyectos educativos y prototipos. Gracias al Arduino, es posible coordinar los movimientos en los tres ejes y ejecutar trayectorias de dibujo de manera confiable.



6. **Lm9023:** El LM9023 es importante porque funciona como el driver que controla los motores de las lectoras de DVD, permitiendo regular su velocidad y sentido de giro. Este componente actúa como un puente H eficiente que protege tanto al Arduino como a los motores, asegurando una operación estable. Gracias a su capacidad para manejar corrientes apropiadas para motores pequeños, el LM9023 permite un control preciso de los desplazamientos en los ejes X y Y sin sobrecargar la electrónica principal.



- 7. Portabaterías:** El portabaterías es necesario para alojar y organizar las baterías que alimentan la mini CNC, manteniendo una conexión segura y estable. Su función es asegurar que las baterías se mantengan firmes, evitando falsos contactos o desconexiones mientras la máquina está en funcionamiento. Además, permite reemplazar o recargar las baterías fácilmente, garantizando la autonomía del sistema cuando no se utiliza alimentación por USB.



- 8. Interruptor :**El interruptor cumple un papel fundamental como elemento de seguridad, ya que permite encender y apagar todo el sistema de manera rápida y controlada. Gracias a él, es posible cortar la energía en caso de fallos, movimientos inesperados o durante la manipulación de los componentes. Esto evita daños en la electrónica y protege al usuario, siendo un elemento simple pero indispensable en cualquier máquina CNC.



9. **Baterías de litio:** Las baterías de litio proporcionan la energía necesaria para que la mini CNC funcione de manera autónoma y estable. Su alta densidad energética, bajo peso y capacidad de entregar corriente suficiente para motores y electrónica las hacen ideales para un proyecto portátil. Además, permiten que el sistema opere sin depender de fuentes externas, ofreciendo mayor flexibilidad durante las pruebas y demostraciones del funcionamiento de la CNC.



10. **Teclado matricial:** El teclado matricial es un componente esencial dentro de la mini CNC, ya que proporciona una interfaz física intuitiva para que el usuario pueda seleccionar las figuras a dibujar, como las opciones asociadas a las teclas “1” y “2”. Su diseño basado en filas y columnas permite detectar múltiples teclas mediante un número reducido de pines, lo que lo hace ideal para proyectos compactos como este. Además, su integración con el PSoC permite que cada pulsación sea identificada de manera precisa y rápida, asegurando que la CNC ejecute la figura correspondiente sin necesidad de una computadora externa. Gracias a su simplicidad, bajo costo y eficacia, el teclado matricial mejora la interacción con la máquina y hace el sistema más autónomo y práctico.



11. **Psoc:** El PSoC cumple un papel fundamental en la mini CNC plotter porque actúa como el módulo encargado de procesar las órdenes provenientes del usuario y gestionar el sistema de entrada. Gracias a su arquitectura programable y a sus bloques digitales configurables, el PSoC permite implementar de manera eficiente la lectura del teclado matricial mediante el escaneo de filas y columnas, además de manejar interrupciones que aseguran una respuesta rápida y estable. Su flexibilidad facilita la integración con el resto del sistema, permitiendo enviar las señales correctas para seleccionar y ejecutar las figuras que la CNC debe dibujar. Esto convierte al PSoC en un componente clave que aporta versatilidad, velocidad de procesamiento y una interfaz de control más inteligente.



### 3. Tabla de pines:

Pines para el Psoc:

<input type="checkbox"/>	Pin_ENA	P3[5]	▼	34	▼	✓
<input type="checkbox"/>	Pin_ENB	P3[0]	▼	29	▼	✓
<input type="checkbox"/>	Pin_M1_IN1	P3[4]	▼	33	▼	✓
<input type="checkbox"/>	Pin_M1_IN2	P3[3]	▼	32	▼	✓
<input type="checkbox"/>	Pin_M2_IN1	P3[2]	▼	31	▼	✓
<input type="checkbox"/>	Pin_M2_IN2	P3[1]	▼	30	▼	✓
<input type="checkbox"/>	Pin_servo	P3[6]	▼	36	▼	✓

La asignación de pines del PSoC se organizó para mantener separados y ordenados los controles de cada actuador de la CNC. El motor del eje X utiliza P3[5] como pin ENA para la señal PWM y P3[4] y P3[3] para IN1 e IN2, que definen su dirección. El motor del eje Y se conecta de forma similar: P3[0] como ENB y los pines P3[2] y P3[1] para sus entradas de dirección. Finalmente, el pin P3[6] se asignó al servo, ya que requiere un PWM dedicado para controlar su posición. Esta distribución agrupada facilita el cableado y asegura un control estable de los motores y el servo dentro del sistema.

#### 4. Diagrama de flujo:

Diagrama de flujo del psoc:

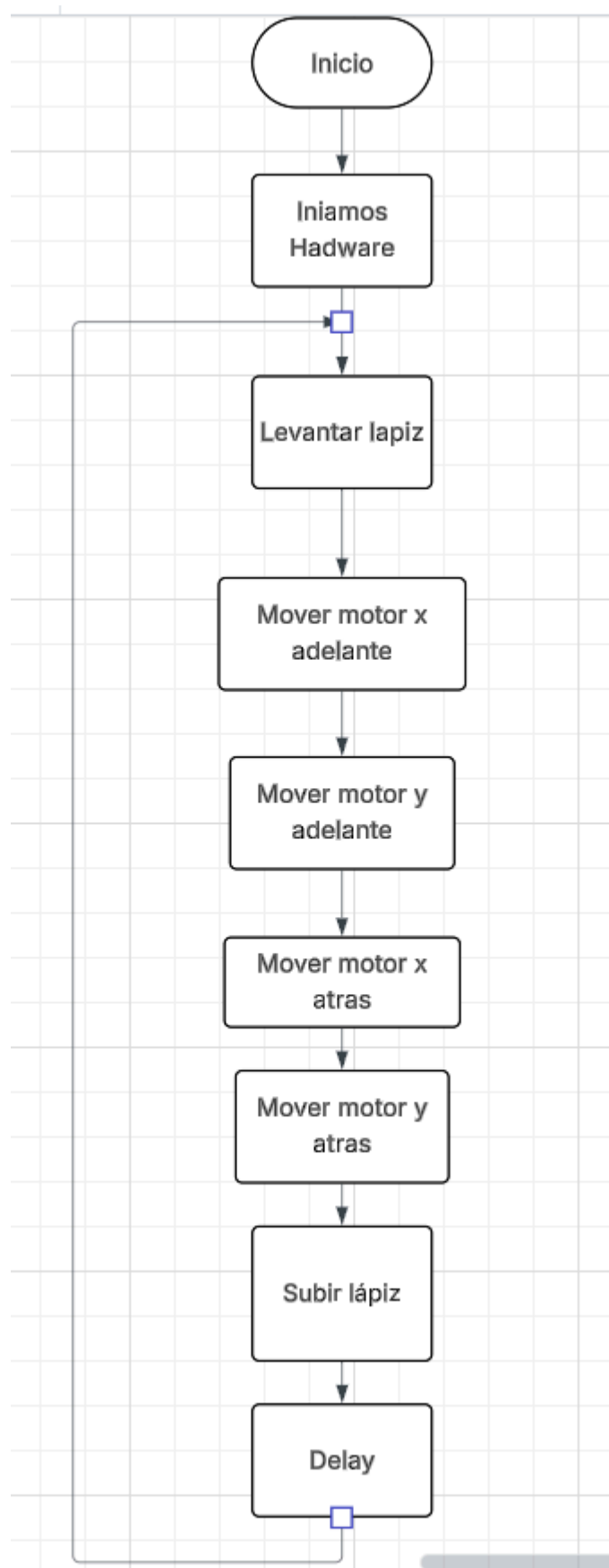
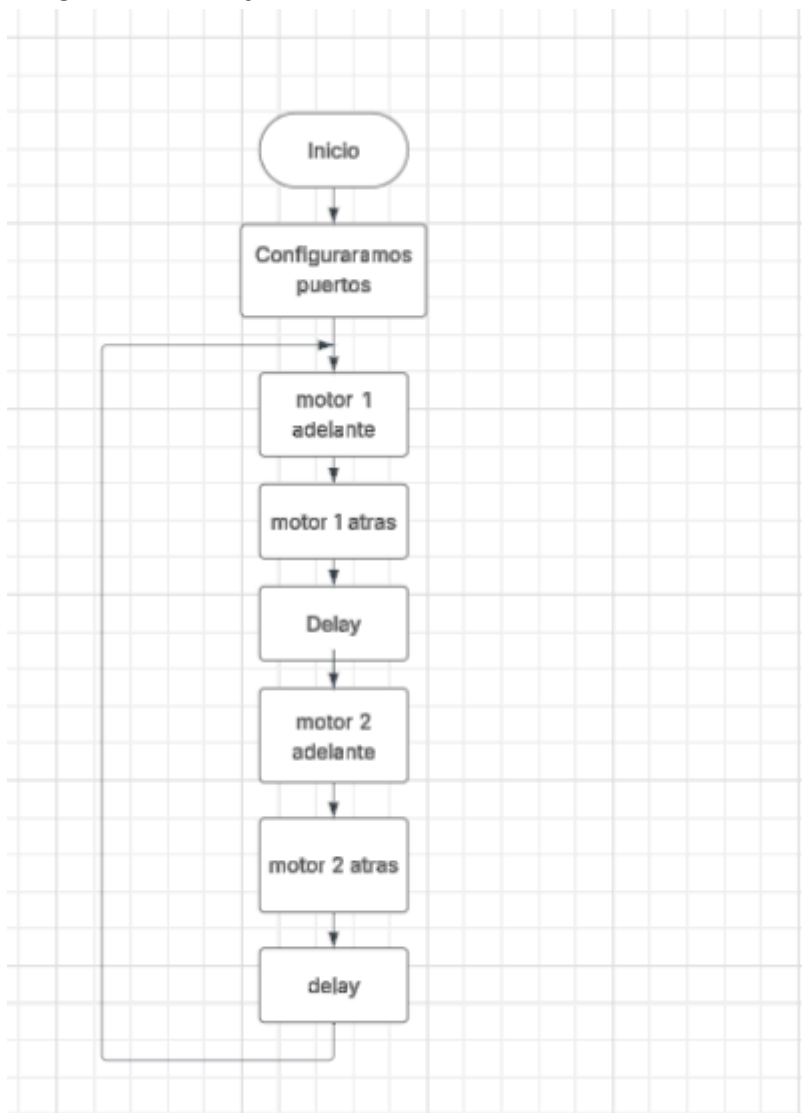


Diagrama de flujo assembler:



- **Resultados:**

1. Validación de etapas:

Código de validación para el teclado matricial y comunicación UART:

```
.include "m328Pdef.inc"
```

```
; Configuración de velocidad (9600 baudios a 16MHz)
```

```
.equ F_CPU = 16000000
```

```

.equ BAUD = 9600
.equ UBRR_VAL = (F_CPU/(16*BAUD))-1

.cseg
.org 0x00
    rjmp RESET

RESET:
    ; 1. Inicializar Stack Pointer
    ldi r16, low(RAMEND)
    out SPL, r16
    ldi r16, high(RAMEND)
    out SPH, r16

    ; 2. Configurar UART (Transmisor)
    ldi r16, high(UBRR_VAL)
    sts UBRR0H, r16
    ldi r16, low(UBRR_VAL)
    sts UBRR0L, r16

    ldi r16, (1<<TXEN0) ; Habilitar Transmisor
    sts UCSR0B, r16

    ldi r16, (1<<UCSZ01)|(1<<UCSZ00) ; 8 bits de datos, 1 stop bit
    sts UCSR0C, r16

    ; 3. Configurar Puertos para Teclado
    ; Configurar PB0-PB3 como SALIDAS (Filas)
    ldi r16, 0b00001111
    out DDRB, r16 ; DDR en 1 = Salida

    ; Configurar PD4-PD7 como ENTRADAS con PULL-UP (Columnas)
    cbi DDRD, 4 ; Aseguramos que sea entrada (0)
    cbi DDRD, 5
    cbi DDRD, 6
    cbi DDRD, 7

    ldi r16, 0b11110000 ; Escribir 1 en el Puerto de entrada...
    out PORTD, r16 ; ...activa las resistencias Pull-Up internas

MAIN_LOOP:
    rcall TECLADO_SCAN ; Rutina que devuelve tecla en R24
    tst r24 ; ¿Se presionó algo? (0 = nada)
    breq MAIN_LOOP ; Si es 0, repetir

    rcall UART_SEND ; Enviar el valor de R24
    rcall DELAY_ANTIREBOTE
    rjmp MAIN_LOOP

```

UART\_SEND:

```
; Esperar a que el buffer esté vacío
lds r17, UCSR0A
sbrs r17, UDRE0
rjmp UART_SEND
; Enviar dato
sts UDR0, r24
ret
```

;-----

; Subrutina: TECLADO\_SCAN

;-----

TECLADO\_SCAN:

```
ldi r24, 0 ; Limpiamos R24 (asumimos que no hay tecla)
```

; --- ESCANEEO FILA 1 (PB0 = 0) ---

```
ldi r16, 0b11111110 ; Poner PB0 en LOW, resto HIGH
```

```
out PORTB, r16
```

```
nop ; Pequeña pausa para estabilidad
```

```
nop
```

; revisar Columnas (PD4-PD7)

```
sbis PIND, 4 ; ¿PD4 es LOW? (Tecla '1')
```

```
ldi r24, '1'
```

```
sbis PIND, 5 ; ¿PD5 es LOW? (Tecla '2')
```

```
ldi r24, '2'
```

```
sbis PIND, 6 ; ¿PD6 es LOW? (Tecla '3')
```

```
ldi r24, '3'
```

```
sbis PIND, 7 ; ¿PD7 es LOW? (Tecla 'A')
```

```
ldi r24, 'A'
```

```
tst r24 ; ¿Encontramos algo?
```

```
brne FIN_SCAN ; Si R24 != 0, terminamos y retornamos
```

; --- ESCANEEO FILA 2 (PB1 = 0) ---

```
ldi r16, 0b11111101 ; Poner PB1 en LOW
```

```
out PORTB, r16
```

```
nop
```

```
nop
```

```
sbis PIND, 4 ; (Tecla '4')
```

```
ldi r24, '4'
```

```
sbis PIND, 5 ; (Tecla '5')
```

```
ldi r24, '5'
```

```
sbis PIND, 6 ; (Tecla '6')
```

```
ldi r24, '6'
```

```
sbis PIND, 7 ; (Tecla 'B')
```

```
ldi r24, 'B'
```

```
tst r24
```

```
brne FIN_SCAN
```

```

; --- ESCANEO FILA 3 (PB2 = 0) ---
ldi r16, 0b11111011 ; Poner PB2 en LOW
out PORTB, r16
nop
nop
sbis PIND, 4      ; (Tecla '7')
ldi r24, '7'
sbis PIND, 5      ; (Tecla '8')
ldi r24, '8'
sbis PIND, 6      ; (Tecla '9')
ldi r24, '9'
sbis PIND, 7      ; (Tecla 'C')
ldi r24, 'C'
tst r24
brne FIN_SCAN

```

```

; --- ESCANEO FILA 4 (PB3 = 0) ---
ldi r16, 0b11110111 ; Poner PB3 en LOW
out PORTB, r16
nop
nop
sbis PIND, 4      ; (Tecla '*')
ldi r24, '*'
sbis PIND, 5      ; (Tecla '0')
ldi r24, '0'
sbis PIND, 6      ; (Tecla '#')
ldi r24, '#'
sbis PIND, 7      ; (Tecla 'D')
ldi r24, 'D'

```

FIN\_SCAN:

```

; Restaurar filas a HIGH (reposo)
ldi r16, 0b00001111
out PORTB, r16
ret

```

```

;-----
; Subrutina: DELAY_ANTIREBOTE
; Propósito: Generar una pausa de aprox 20ms a 16MHz
;-----

```

DELAY\_ANTIREBOTE:

```

ldi r20, 64 ; Contador externo (Ajusta este valor para cambiar la duración)

```

Lazo1:

```

ldi r21, 255 ; Contador medio

```

Lazo2:

```

ldi r22, 255 ; Contador interno

```

Lazo3:

```

dec r22 ; Restar 1
brne Lazo3 ; Si no es 0, repetir

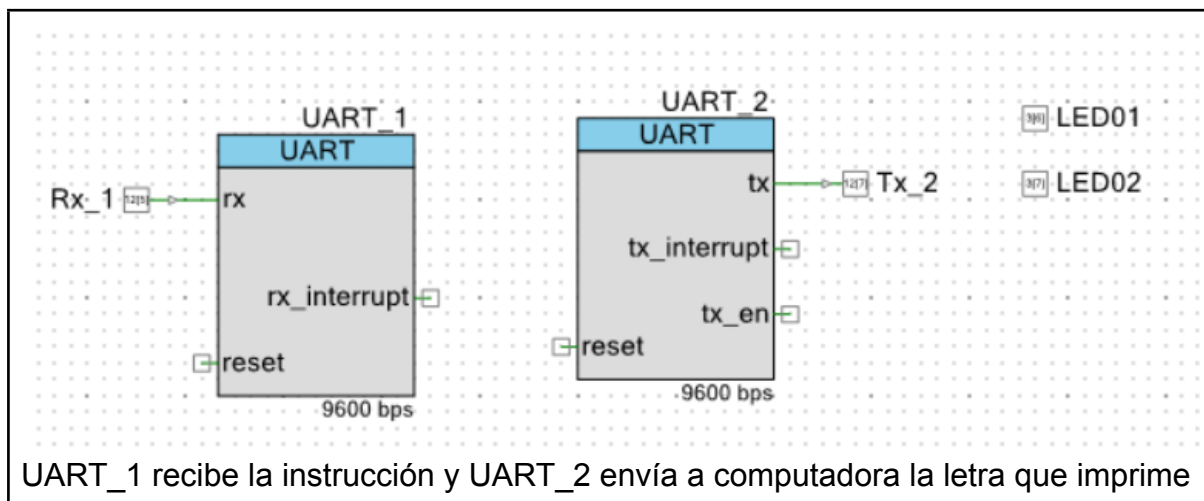
```

```

dec r21
brne Lazo2      ; Si no es 0, repetir lazo medio
dec r20
brne Lazo1      ; Si no es 0, repetir lazo externo
ret

```

Recibe instrucciones en el Psoc:



UART\_1 recibe la instrucción y UART\_2 envía a computadora la letra que imprime

Código

```

#include "project.h"

int main(void)
{
    CyGlobalIntEnable; /* Enable global interrupts. */
    UART_1_Start();
    UART_2_Start();

    for(;;)
    {
        uint8 dato;

        // 1. Revisar si UART_1 (ATmega) tiene algo
        if (UART_1_GetRxBufferSize() > 0)
        {
            // 2. Leer el dato del ATmega
            dato = UART_1_GetByte();

            // 3. Escribir el dato hacia la PC (UART_2)
            UART_2_PutChar(dato);
        }
    }
}

```

```
switch(dato)
{
    case '1':
        LED01_Write(1);
        break;

    case '2':
        LED02_Write(1);
        break;

    case '3':
        LED01_Write(0);
        break;

    case '4':
        LED02_Write(0);
        break;
}
}
}
}

/* [] END OF FILE */
```

2. Validación del sistema integrado.
3. Análisis de los recursos usados: memoria y potencia.

Para el Psoc

Resource Type	: Used	: Free	: Max	: % Used
Digital Clocks	3	5	8	37.50 %
Analog Clocks	0	4	4	0.00 %
CapSense Buffers	0	2	2	0.00 %
Digital Filter Block	0	1	1	0.00 %
Interrupts	0	32	32	0.00 %
IO	10	38	48	20.83 %
Segment LCD	0	1	1	0.00 %
CAN 2.0b	0	1	1	0.00 %
I2C	0	1	1	0.00 %
USB	0	1	1	0.00 %
DMA Channels	0	24	24	0.00 %
Timer	0	4	4	0.00 %
UDB				
Macrocells	15	177	192	7.81 %
Unique P-terms	15	369	384	3.91 %
Total P-terms	15			
Datapath Cells	4	20	24	16.67 %
Status Cells	3	21	24	12.50 %
StatusI Registers	3			
Control Cells	3	21	24	12.50 %
Control Registers	3			
Opamp	0	4	4	0.00 %
Comparator	0	4	4	0.00 %
Delta-Sigma ADC	0	1	1	0.00 %
LPF	0	2	2	0.00 %
SAR ADC	0	2	2	0.00 %
Analog (SC/CT) Blocks	0	4	4	0.00 %
DAC				
VIDAC	0	4	4	0.00 %

## Para assembler

Segment	Begin	End	Code	Data	Used	Size	Use%
[.cseg]	0x000000	0x0000b6	182	0	182	32768	0.6%
[.dseg]	0x000100	0x000100	0	0	0	2048	0.0%
[.eseg]	0x000000	0x000000	0	0	0	1024	0.0%

Assembly complete, 0 errors. 0 warnings

- **Conclusiones:**

El desarrollo de la mini CNC plotter utilizando Arduino programado en lenguaje ensamblador permitió demostrar que es posible implementar un sistema de dibujo automático de bajo costo y buena precisión aprovechando componentes reciclados y módulos electrónicos básicos. La máquina logró ejecutar correctamente figuras geométricas simples, como el cuadrado y el triángulo, evidenciando que la comunicación entre los motores, el servo y el controlador se realiza de manera adecuada. Además, la integración del teclado matricial permitió agregar una interfaz interactiva, donde las teclas “1” y “2” funcionan como comandos directos para seleccionar la figura a dibujar, aportando mayor practicidad y facilidad de uso. En conjunto, la CNC exhibió un funcionamiento estable y coherente con los objetivos planteados, validando la viabilidad del diseño tanto a nivel mecánico como electrónico, y demostrando que el uso de ensamblador permite un control más preciso y eficiente del hardware involucrado.

- **Anexos:**

**Código assembler:**

```
; --- Configuración inicial ---  
.include "m328Pdef.inc"  
.DEF temp = r16  
.DEF contador_pasos = r17  
.DEF retardo1 = r18  
.DEF retardo2 = r19  
.DEF retardo3 = r20  
  
.ORG 0x0000  
    rjmp RESET  
  
RESET:
```

```

; Configurar Puertos (Salidas para motores)
ldi temp, 0xFF
out DDRB, temp ; Puerto B (Motor 1)
out DDRD, temp ; Puerto D (Motor 2)

; =====
; TRADUCCIÓN DEL VOID LOOP()
; =====
LOOP_PRINCIPAL:

; --- 1. motor1.step(256, FORWARD) ---
ldi contador_pasos, 0 ; 0 en un registro de 8 bits = 256 iteraciones al
decrementar
LOOP_M1_FWD:
call PASO_MOTOR1_ADELANTE
call DELAY_VELOCIDAD ; Controla las RPM
dec contador_pasos
brne LOOP_M1_FWD

; --- 2. motor2.step(256, FORWARD) ---
ldi contador_pasos, 0
LOOP_M2_FWD:
call PASO_MOTOR2_ADELANTE
call DELAY_VELOCIDAD
dec contador_pasos
brne LOOP_M2_FWD

; --- 3. delay(500) ---
call DELAY_MEDIO_SEGUNDO

; --- 4. motor1.step(256, BACKWARD) ---
ldi contador_pasos, 0
LOOP_M1_BCK:
call PASO_MOTOR1_ATRAS
call DELAY_VELOCIDAD
dec contador_pasos
brne LOOP_M1_BCK

; --- 5. motor2.step(256, BACKWARD) ---
ldi contador_pasos, 0
LOOP_M2_BCK:
call PASO_MOTOR2_ATRAS
call DELAY_VELOCIDAD
dec contador_pasos

```

```

brne LOOP_M2_BCK

; --- 6. delay(500) ---
call DELAY_MEDIO_SEGUNDO

rjmp LOOP_PRINCIPAL

```

```

; =====
; SUBROUTINAS (Lo que la librería AFMotor hace por ti)
; =====

```

#### PASO\_MOTOR1\_ADELANTE:

```

; Secuencia simple de 4 pasos (Bipolar) en Puerto B
; Esto se puede mejorar con tablas, pero así es visual:
ldi temp, 0b00001001 ; Paso 1
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00001100 ; Paso 2
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00000110 ; Paso 3
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00000011 ; Paso 4
out PORTB, temp
call DELAY_CORTO
ret

```

#### PASO\_MOTOR1\_ATRAS:

```

; Secuencia inversa
ldi temp, 0b00000011 ; Paso 4
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00000110 ; Paso 3
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00001100 ; Paso 2
out PORTB, temp
call DELAY_CORTO
ldi temp, 0b00001001 ; Paso 1
out PORTB, temp
call DELAY_CORTO
ret

```

; (Repetir lógica similar para MOTOR2 en PORTD...)

PASO\_MOTOR2\_ADELANTE:

ret ; (Abreviaremos aquí por espacio)

PASO\_MOTOR2\_ATRAS:

ret

; --- Retardos ---

DELAY\_VELOCIDAD:

; Retardo pequeño para definir la velocidad de giro

ldi retardo1, 50

d\_v: dec retardo1

brne d\_v

ret

DELAY\_CORTO:

; Pequeña pausa entre bobinas

ldi retardo1, 200

d\_c: dec retardo1

brne d\_c

ret

DELAY\_MEDIO\_SEGUNDO:

; Retardo largo (aprox 500ms)

ldi retardo3, 20

loop\_ext:

ldi retardo2, 255

loop\_mid:

ldi retardo1, 255

loop\_int:

dec retardo1

brne loop\_int

dec retardo2

brne loop\_mid

dec retardo3

brne loop\_ext

ret

### **Código psoc:**

```
#include "project.h"
```

```
// =====
```

```

//  Funciones Motores
// =====

// Motor 1 (eje X)
void Motor1_Forward()
{
    Pin_M1_IN1_Write(1);
    Pin_M1_IN2_Write(0);
}

void Motor1_Backward()
{
    Pin_M1_IN1_Write(0);
    Pin_M1_IN2_Write(1);
}

void Motor1_Stop()
{
    Pin_M1_IN1_Write(0);
    Pin_M1_IN2_Write(0);
}

// Motor 2 (eje Y)
void Motor2_Forward()
{
    Pin_M2_IN1_Write(1);
    Pin_M2_IN2_Write(0);
}

void Motor2_Backward()
{
    Pin_M2_IN1_Write(0);
    Pin_M2_IN2_Write(1);
}

void Motor2_Stop()
{
    Pin_M2_IN1_Write(0);
    Pin_M2_IN2_Write(0);
}

// =====
//  Servo SG90
// =====

```

```

// 0° → 1ms (duty 5%)
// 90° → 1.5ms (duty 7.5%)
// 180° → 2ms (duty 10%)

void Servo_WriteAngle(float angle)
{
    // duty = 5% + (angle/180)*5%
    float duty = 5 + (angle / 180.0) * 5.0;
    uint16 compare = (uint16)((duty / 100.0) * 20000); // periodo 20 ms

    PWM_Servo_WriteCompare(compare);
}

void Lapis_Arriba()
{
    Servo_WriteAngle(0);
}

void Lapis_Abajo()
{
    Servo_WriteAngle(40); // Ajusta según tu mecanismo
}

// =====
// Programa principal
// =====

int main(void)
{
    CyGlobalIntEnable;

    PWM_M1_Start();
    PWM_M2_Start();
    PWM_Servo_Start();

    // Velocidad de motores
    PWM_M1_WriteCompare(180); // Ajusta según fuerza
    PWM_M2_WriteCompare(180);

    // Preparación
    Lapis_Arriba();
    CyDelay(1000);

    for(;;)

```

```

{
  // =====
  //  DIBUJAR CUADRADO
  // =====

  Lapisz_Abajo();
  CyDelay(500);

  // LADO 1 (derecha)
  Motor1_Forward();
  CyDelay(800);
  Motor1_Stop();

  // LADO 2 (arriba)
  Motor2_Forward();
  CyDelay(800);
  Motor2_Stop();

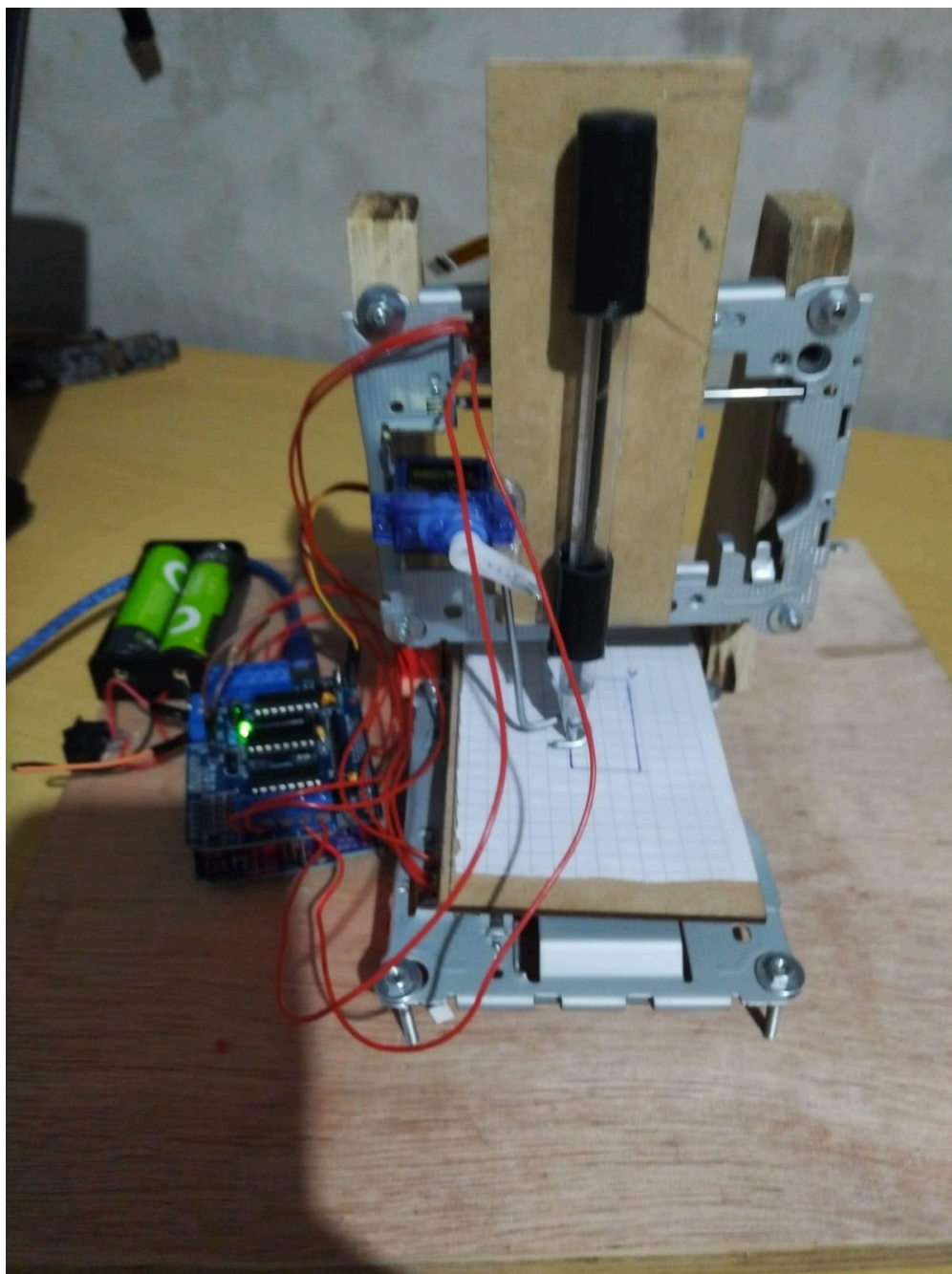
  // LADO 3 (izquierda)
  Motor1_Backward();
  CyDelay(800);
  Motor1_Stop();

  // LADO 4 (abajo)
  Motor2_Backward();
  CyDelay(800);
  Motor2_Stop();

  Lapisz_Arriba();
  CyDelay(2000);
}
}

```

Fotos:



Link del video:

<https://drive.google.com/drive/folders/1tVvLnElmjGjJHoVGhbqjBNzeUjR38Lz>

- **Referencias:**

Wankhede, V. W., Ambilduke, A., Pandharam, D., Khubalkar, I., Pote, K., & Shambharkar, S. (s. f.). *2D CNC Plotter Integrated with Arduino*. Journal of Microprocessor and Microcontroller Research. Recuperado de <https://matjournals.net/engineering/index.php/JoMMR/article/view/1494>

Suresh, R., Yussuf, A., Shijin, A. J., Venkatesh, V., & Ahmed, I. A. (2025). *Design and Implementation of 2D CNC Plotter Using Arduino*. International Journal of Research and Scientific Innovation. Recuperado de <https://rsisinternational.org/journals/ijrsi/articles/design-and-implementation-of-2d-cnc-plotter-using-arduino/>

Maker 101. (2024). *Build a Simple 3D Printed CNC Plotter Machine*. Hackster.io. Recuperado de <https://www.hackster.io/mertarduino/build-a-simple-3d-printed-cnc-plotter-machine-1e9811>