Laboratorio 1:

Adriel Castillo, Gabriel González
Carrera: Ingeniería en Mecatrónica, Universidad Tecnológica (UTEC)
Fray Bentos, Uruguay
adriel.castillo@estudiantes.utec.edu.uy, gabriel.gonzalez@estudiantes.utec.edu.uy
25 de septiembre de 2025

Resumen— Este informe detalla el desarrollo de un laboratorio centrado en la programación de bajo nivel del microcontrolador ATmega328P en lenguaje ensamblador. Se abordaron cuatro problemas principales: el control de una punzonadora mediante una máquina de estados finitos (FSM), la visualización de mensajes en una matriz de LEDs con control por UART, la generación de señales analógicas mediante un DAC R-2R y una LUT, y el control de un plotter monocromático. Si bien la implementación física de la punzonadora y el DAC no fue completada, se lograron avances significativos en el diseño lógico y se implementaron con éxito los sistemas de control para la matriz de LEDs y el plotter, validando la capacidad de manejar periféricos, actuadores y comunicación serial a través de código optimizado

Palabras Clave: ATmega328P, Lenguaje Ensamblador (ASM), Sistemas Embebidos, Máquina de Estados Finitos (FSM), GPIO, Matriz de LEDs, Multiplexación, Plotter, Comunicación Serial, UART.

I.Introducción

El control de sistemas electromecánicos a bajo nivel es un pilar fundamental en la ingeniería mecatrónica. La capacidad de interactuar directamente con el hardware, gestionando cada ciclo de reloj y cada bit de los registros, permite crear soluciones altamente eficientes y deterministas, cruciales en aplicaciones de tiempo real. El microcontrolador ATmega328P, basado en la arquitectura AVR RISC, es una plataforma ideal para este aprendizaje debido a su simplicidad y al acceso directo que ofrece a sus periféricos.

Este laboratorio se diseñó para aplicar los conceptos de programación en lenguaje ensamblador (ASM) en la resolución de problemas prácticos de automatización y control. El trabajo abarcó desde el diseño de lógica secuencial mediante máquinas de estados finitos (FSM) para gobernar un proceso industrial (punzonadora), hasta el manejo de interfaces de salida como matrices de LEDs y la generación de señales analógicas, culminando en el control de un plotter para la ejecución de trayectorias precisas. A través de estos desafíos, se buscó consolidar la comprensión de la interacción entre software y hardware, el uso de protocolos de comunicación como USART y la importancia de las buenas prácticas de desarrollo, como el control de versiones con GitHub.

II.Objetivos

Objetivo General:

Analizar, diseñar e implementar diferentes sistemas digitales basados en el microcontrolador ATmega328P programado con assembler, integrando conceptos de GPIO, comunicación serial, LUT con DAC R-2R y control de actuadores, con el fin de comprender la interacción entre hardware, software y protocolos de comunicación en aplicaciones de automatización

Objetivos específicos:

1. Ensamblar una punzonadora con cinta e implementar el control de esta mediante máquinas de estados y comunicación USART.
2. Programar la visualización de mensajes desplazables en una matriz de LEDs, controlados a través de UART
3. Generar señales analógicas específicas utilizando una LUT y un convertidor digital-analógico R-2R.
4. Desarrollar un sistema de control de un plotter monocromático empleando ensamblador en el ATmega328P y comunicación serial.

III.Materiales utilizados

Materiales electrónicos:

- Microcontrolador ATmega328P
- Kit de Fischertechnik
- Protoboard y cables de conexión
- Resistencias (para red R-2R de 8 bits)
- Matriz de LEDs
- Osciloscopio digital
- Fuente de alimentación
- Multímetro digital

Software:

- Microchip Studio
- GitHub
- Software de procesamiento de texto (Word, LaTeX, etc.) para la elaboración del informe.
- Herramientas para capturar evidencias visuales (cámara o teléfono móvil).

IV.Marco teórico

Este laboratorio integra varios conceptos fundamentales de la electrónica digital y de microprocesadores.

A. Lenguaje ensamblador en microcontroladores

El lenguaje ensamblador (Assembly o ASM) es un lenguaje de bajo nivel que establece una relación casi directa entre las instrucciones escritas por el programador y las operaciones que ejecuta la Unidad Central de Procesamiento (CPU) del microcontrolador. Cada instrucción en ensamblador corresponde a una instrucción en código máquina representada en binario, lo que permite un control muy preciso del hardware.

En el caso del ATmega328P, ensamblador forma parte de la arquitectura AVR RISC de 8 bits, diseñada por Atmel

(hoy Microchip). Algunas de sus características principales son:

- **Conjunto reducido de instrucciones (RISC)**: alrededor de 130 instrucciones, la mayoría se ejecuta en un solo ciclo de reloj.
- **Registros de propósito general (R0–R31)**: permiten operaciones aritméticas y lógicas sin necesidad de acceder continuamente a la memoria, aumentando la eficiencia.
- **Memoria separada (Harvard Architecture)**: programa e instrucciones se almacenan en memorias diferentes (Flash y SRAM).
- **Instrucciones orientadas al control de hardware**: manipulación directa de bits en registros de E/S, temporizadores, interrupciones y comunicación serial.

Ventajas del uso de ensamblador

- **Control absoluto del hardware**: se pueden manipular registros y periféricos a nivel de bit.
- **Código altamente optimizado**: permite reducir el tamaño del programa y mejorar la velocidad de ejecución.
- **Determinismo**: cada instrucción tiene un tiempo de ejecución conocido, lo que es crítico en aplicaciones de tiempo real como generación de señales (DAC R-2R) o control de actuadores (punzonadora y plotter).

Desventajas frente a lenguajes de alto nivel

- **Mayor complejidad de programación**: escribir en ensamblador requiere conocer la arquitectura interna del microcontrolador.
- **Baja portabilidad**: el código ensamblador está diseñado para una arquitectura específica (en este caso AVR), y no puede trasladarse fácilmente a otros microcontroladores.
- **Mantenimiento difícil**: los programas extensos en ASM pueden volverse complicados de leer y depurar.

B. GPIO en microcontoladores

Los puertos de entrada/salida de propósito general (GPIO) permiten al microcontrolador interactuar con el entorno físico, ya sea recibiendo señales de sensores o controlando actuadores. EN el caso del ATmega328P, los puertos están organizados en registros (PORTx, DDRx, PINx), donde:

- DDRx define si el pin es entrada o salida
- PORTx controla el nivel lógico de salida o activa resistencias de pull-up en entradas
- PINx permite leer el estado de los pines

El control de una punzonadora con cinta transportadora se basa en esta característica, donde los GPIO se utilizan para accionar motores, leer sensores de posición y manejar LEDs indicadores.

C. Comunicación serial USART

La comunicación USART (Universal Synchronous/Asynchronous Receiver-Transmitter) es un protocolo estándar en los microcontroladores para enviar y recibir datos seriales. En el ATmega328P se puede configurar en modo asincrónico, utilizando registros como UBRR(baud rate), UDR(registro de datos) y UCSRA/B/C (control y estado).

En este laboratorio, USART cumple funciones clave:
Permitir el monitoreo de los estados de la punzonadora en tiempo real
Recibir comandos externos para iniciar el proceso o seleccionar el tipo de carga
Interactuar con la matriz de LEDs, habilitando al usuario a elegir entre un mensaje desplazable o figuras predefinidas.

D. Máquinas de estados finitos (FSM)

Una máquina de estados finitos se encarga de modelar el comportamiento secuencial de un steam digital, permitiendo organizar procesos en etapas. En la punzonadora con cinta transportadora se definen los siguientes estados:
En espera: El sistema queda inactivo hasta recibir un inicio.
Alimentación:La cinta posiciona la pieza bajo el punzón.
Posicionado: Detección de la pieza lista.
Punzonado: Accionamiento del punzón.
Descarga: Retiro de la pieza procesada.
Fin de ciclo: Vuelve a esperar el comando.

El uso de FSM asegura claridad en el control, facilita la depuración y permite escalar el sistema.

E. LUT y conversor DAC R-2R

Un DAC R-2R es un convertidor de digital a analógico implementado únicamente con resistencias. Consiste en una red en escalera donde cada bit controla un interruptor que conecta a referencia de voltaje o tierra. La salida es una tensión proporcional al valor binario de entrada.

$$Vout = Vref \cdot \frac{D}{2^N}$$

Donde:

- Vref es la tensión de referencia
- D es el valor digital aplicado
- N es el número de bits

El uso de una LUT (Look-up Table) permite precalcular los valores digitales que representan una onda (senoidal, triangular, cuadrada, etc.) y reproducir secuencialmente enviando esos datos al DAC. De esta forma se obtiene una señal periódica observable en el osciloscopio.

F. Matriz de LEDs y control UART

Una matriz de LEDs es un conjunto de diodos dispuestos en filas y columnas, lo que permite controlar múltiples píxeles con menos pines mediante multiplexación. En este laboratorio se utiliza una matriz DOLANG o una matriz común, capaz de mostrar mensajes o imágenes.

El control se implementa en ensamblador utilizando **UART**, de manera que:

- Al recibir datos, el sistema selecciona el modo de visualización.
- Se puede optar por mensaje desplazable o imágenes predefinidas.
- El desplazamiento se logra actualizando columnas de forma secuencial con retardos programados.

G. Control de plotter mediante microcontrolador

El plotter monocromático utilizado se controla a través de un PLC, pero el microcontrolador ATmega328P envía las señales mediante transistores MOSFET o relés. Los movimientos posibles son: arriba, abajo, izquierda, derecha, además de un solenoide para trazar.

El control digital requiere sincronización en los tiempos de conmutación para garantizar figuras precisas (círculo, triángulo, cruz). Además, se establece una comunicación

serial con el microcontrolador para que el usuario seleccione qué figura dibujar.

H. Importancia del GitHub y la trazabilidad del código

El uso de control de versiones como GitHub garantiza que las implementaciones en ensamblador queden registradas, con trazabilidad de cambios y posibilidad de colaboración. Esta práctica es fundamental en ingeniería moderna, donde la reproducibilidad y documentación del software son tan importantes como el diseño del hardware.

V.Metodología

El laboratorio fue desarrollado en varias etapas, cada una correspondiendo a los problemas planteados en la consigna. Todas las implementaciones se realizaron utilizando el microcontrolador ATmega328P, programado en lenguaje ensamblador y verificando los resultados físicamente.

El desarrollo del laboratorio se inició con la revisión de la documentación del kit Fischertechnik, a fin de comprender la estructura y las recomendaciones de montaje del modelo de cinta transportadora con punzonadora. Posteriormente, se procedió al armado del sistema físico, sustituyendo el controlador original del kit por el microcontrolador ATmega328P. Si bien se avanzó en la comprensión del montaje y en la definición de la lógica de control, el prototipo no logró ponerse en funcionamiento durante la práctica, por lo que la implementación se limitó al diseño de la máquina de estados finitos (FSM) correspondiente. Dicho modelo incluyó los estados de espera, alimentación, punzonado, descarga y fin de ciclo, junto con las transiciones definidas por sensores y retardos temporizados. Asimismo, se planificó la integración de la comunicación vía USART para el envío de estados y la recepción de comandos externos, aunque esta parte quedó en fase conceptual.

En el problema B, se abordó la implementación de una matriz de LEDs destinada a mostrar mensajes desplazables. Para ello, se construyó una tabla de búsqueda (LUT) que contenía los patrones binarios de cada carácter a visualizar. El desplazamiento se logró mediante la multiplexación secuencial de columnas y la incorporación de retardos calibrados, con el objetivo de asegurar un desplazamiento suave y legible. A través de la comunicación serial, el sistema ofreció un menú de inicio, desde el cual el usuario podía seleccionar entre el modo de mensaje desplazable o la presentación de figuras predefinidas desarrolladas en la evaluación continua.

El problema C, orientado a la generación de una señal analógica con un DAC R-2R de 8 bits, no se desarrolló en la práctica. La participación del equipo se limitó únicamente a la lectura y análisis de la consigna, sin avanzar en el diseño de la LUT, la implementación del código en ensamblador ni la verificación de la señal en osciloscopio.

Por último, los problemas D y E se centraron en el control de un plotter monocromático. La comunicación entre el microcontrolador y el PLC del equipo se realizó mediante transistores MOSFET, que permitieron accionar los movimientos verticales y horizontales, así como el solenoide encargado del trazado. Se desarrollaron rutinas en ensamblador que permitieron dibujar distintas figuras (triángulo, círculo y cruz), además de una secuencia que ejecutaba todas las figuras de manera automática. El sistema fue programado para recibir órdenes a través de UART, de modo que el usuario pudiera seleccionar la figura a trazar. Adicionalmente, se verificó la correcta interacción con el pulsador de RESET, que devuelve el puntero a la posición inicial, y con el pulsador de emergencia, que detiene toda la operación inmediatamente.

En síntesis, la metodología aplicada permitió desarrollar de forma parcial los objetivos del laboratorio: se implementó efectivamente el control de la matriz de LEDs, se diseñaron las máquinas de estados de la punzonadora y las rutinas de dibujo para el plotter, mientras que la punzonadora física no pudo ponerse en marcha y el DAC R-2R quedó sin avances prácticos más allá de la lectura inicial de la consigna.

VI.Resultados

Los resultados obtenidos durante el desarrollo del laboratorio fueron parciales y diferenciados según cada problema planteado en la consigna.

Problema A:

En el caso de la punzonadora con cinta transportadora, no se logró la implementación física del sistema debido a limitaciones técnicas en el montaje y puesta en marcha del modelo Fischertechnik. No obstante, se alcanzó el diseño completo de la máquina de estados finitos (FSM) que describe el comportamiento esperado del sistema (fig.1). Esta representación permitió establecer de manera clara las transiciones entre estados de espera, alimentación, posicionado, punzonado, descarga y fin de ciclo. Si bien no fue posible validar experimentalmente el prototipo, el desarrollo teórico del modelo constituye un insumo fundamental para futuras implementaciones.

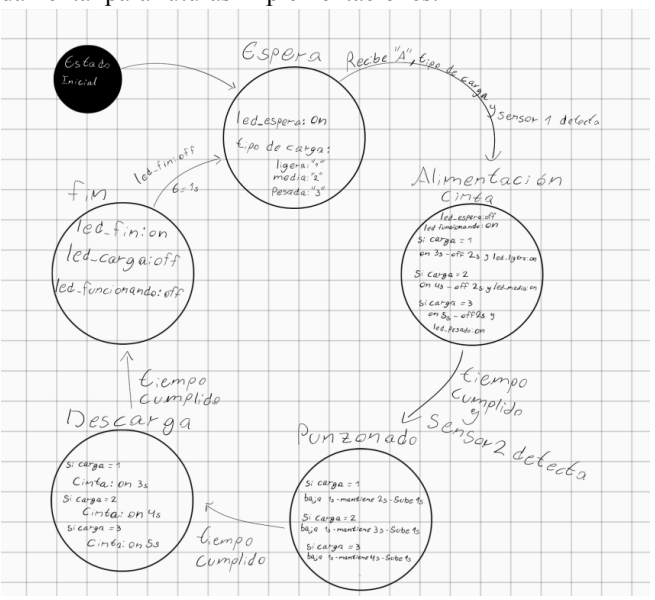

Fig 1. Máquina de estados de punzonadora con cinta transportadora

Como se observa en la figura 1, el ciclo comienza con una transición inmediata al estado Espera al energizar el sistema. La activación del proceso requiere la concurrencia de tres eventos de entrada: el comando de inicio (Recibe "A"), la clasificación del Tipo de Carga (1, 2 o 3), y la confirmación de la presencia del material por el Sensor 1.

La característica distintiva del FSM reside en su capacidad para modular los tiempos de operación en función

de la carga identificada. Esta modulación se aplica en las tres etapas de acción del proceso:

Alimentación: La duración del movimiento de la cinta es directamente proporcional al índice de carga. Se observó que las Cargas 1, 2 y 3 requerían 3s, 4s y 5s de movimiento, respectivamente. Esta variación es fundamental para posicionar correctamente el material. La transición a la siguiente etapa solo se autoriza luego de que las piezas queden detenidas durante 2 segundos y el objeto ha sido confirmado en la posición de trabajo por el Sensor 2.

Punzonado: Se confirmó que todas las cargas atraviesan el estado de Punzonado, pero el tiempo de tratamiento varía. El tiempo que el punzón se mantiene presionado es ajustado: 2s para la Carga 1 (Ligera), 3s para la Carga 2 (Media), y 4s para la Carga 3 (Pesada). Este resultado valida que el FSM adapta la intensidad del procesamiento requerida por cada material.

Descarga: Para la expulsión del material, el movimiento de la cinta mantiene la lógica de modulación de tiempos (3s, 4s, 5s).

Finalmente, al completarse la tarea de descarga, el sistema transiciona a fin y retorna al estado espera tras un retardo fijo de 1s, completando el ciclo de control y quedando listo para una nueva operación.

```
;Estado de inicio = espera
ldi r16, espera
sts estado, r16

main:
    rcall datos ;recibe comandos A,1,2,3      You, 2 weeks ago • Update codigo
    rcall botones ;revisa el boton de inicio

;Lee el estado actual

    lds r16, estado
    cpi r16, espera
    breq esperando
    cpi r16, energizar
    breq alimentacion
    cpi r16, posicionar
    breq posicionado
    cpi r16, punzonar
    breq punzonada
    cpi r16, descarga
    breq descargado
    cpi r16, fin
    breq finalizado

    rjmp main
```

Fig 2. Máquina de estados definida en assembler (código incompleto)

El núcleo del sistema es una máquina de estados que cicla continuamente entre la recepción de comandos y la ejecución del estado actual. Cada estado implementa la lógica específica de esa fase del proceso.

Problema B:

```
MainLoop:
    RCALL Caraf
    RJMP MainLoop

Caraf:
    LDI R16, 0X00
    LDI R16, 0b00000100
    OUT PORTD, R16
    LDI R16, 0X00
    LDI R16, 0b11001100
    OUT PORTB, R16
    LDI R16, 0X00
    LDI R16, 0b00111100
    OUT PORTC, R16
    //--------
    RCALL Mic
    LDI R16, 0X00      adrielcastillo, 46 minutes ago • codigo carita feliz
    LDI R16, 0b00001000
    OUT PORTD, R16
    LDI R16, 0X00
    LDI R16, 0b00110100
    OUT PORTB, R16
    LDI R16, 0X00
    LDI R16, 0b00111011
    OUT PORTC, R16
    RCALL Mic
    //--------
    LDI R16, 0X00
    LDI R16, 0b00010000
    OUT PORTD, R16
    LDI R16, 0X00
    LDI R16, 0b00101000
    OUT PORTB, R16
    LDI R16, 0X00
    LDI R16, 0b0010110
    OUT PORTC, R16
    RCALL Mic
```

Fig 3. Fragmento del código de la cara feliz

El código utilizado para el manejo de la matriz de leds convierte al microcontrolador en un proyector de imágenes para la matriz.. Su truco es que no enciende todos los LEDs a la vez, sino que lo hace tan rápido que nuestro cerebro cree que sí.

El programa enciende una sola columna de la matriz, dibuja una línea vertical de píxeles en ella por una milésima de segundo, y luego la apaga. Inmediatamente, salta a la siguiente columna y repite el proceso con un nuevo patrón. Este ciclo de "encender y saltar" se repite una y otra vez a través de todas las columnas a una velocidad altísima.

Debido al efecto llamado persistencia de la visión, nuestro ojo no es capaz de ver el parpadeo ni las columnas individuales. En su lugar, une todas las líneas en una sola imagen completa y estática.

Mediante la técnica de multiplexación por barrido de columna, se programaron rutinas en ensamblador que permiten mostrar patrones fijos, como los emojis solicitados en la guía. Las pruebas físicas confirmaron que las imágenes se visualizan de forma estable y clara.

Sin embargo, no se completó la implementación de la funcionalidad de mensaje desplazable. Aunque se logró el control fundamental de la matriz, el desarrollo del algoritmo para el desplazamiento dinámico del texto quedó inconcluso por limitaciones de tiempo en el laboratorio.

Problema C:

El problema C, vinculado a la generación de señales analógicas mediante un DAC R-2R de 8 bits, no se desarrolló en la práctica. La participación del grupo se limitó únicamente a la lectura y comprensión de la consigna, sin avances en la construcción de la red resistiva, el diseño de la LUT ni la verificación de la señal en el osciloscopio.

Por último, los problemas D y E, asociados al control del plotter monocromático, se completaron exitosamente. Se implementó la comunicación entre el microcontrolador y el PLC a través de transistores MOSFET, lo que permite accionar los movimientos en los ejes y el solenoide de trazo. Se programaron en ensamblador las secuencias necesarias

para dibujar figuras básicas —triángulo, círculo y cruz—, además de una rutina que ejecutaba todas las figuras en secuencia. La selección de figuras se realizó mediante comandos enviados por UART, y se validó la operación del pulsador de RESET y del botón de emergencia, asegurando la seguridad del sistema. Las pruebas físicas confirmaron la correcta ejecución de las trayectorias dentro del área definida del plotter, cumpliendo satisfactoriamente con los requisitos establecidos en la consigna.

```
39  MAIN_LOOP:
40      RCALL USART_RX
41
42      CPI R17, '1'
43      BREQ Trian
44      CPI R17, '2'
45      BREQ Cruz
46      CPI R17, '3'
47      BREQ Circulo
48      CPI R17, 'T'
49      BREQ Todos
50
51      RCALL USART_TX
52      RJMP MAIN_LOOP
53  Trian:
54      RCALL pos
55      RCALL Tri
56      RJMP MAIN_LOOP
57
58  Cruz:
59      RCALL Pos
60      RCALL Equis
61      RJMP MAIN_LOOP
62  Circulo:
63      RCALL Pos
64      RCALL Cir
65      RJMP MAIN_LOOP
66  Todos:
67      RCALL Tod
```

Fig 4. Fragmento de código de la ploteadora

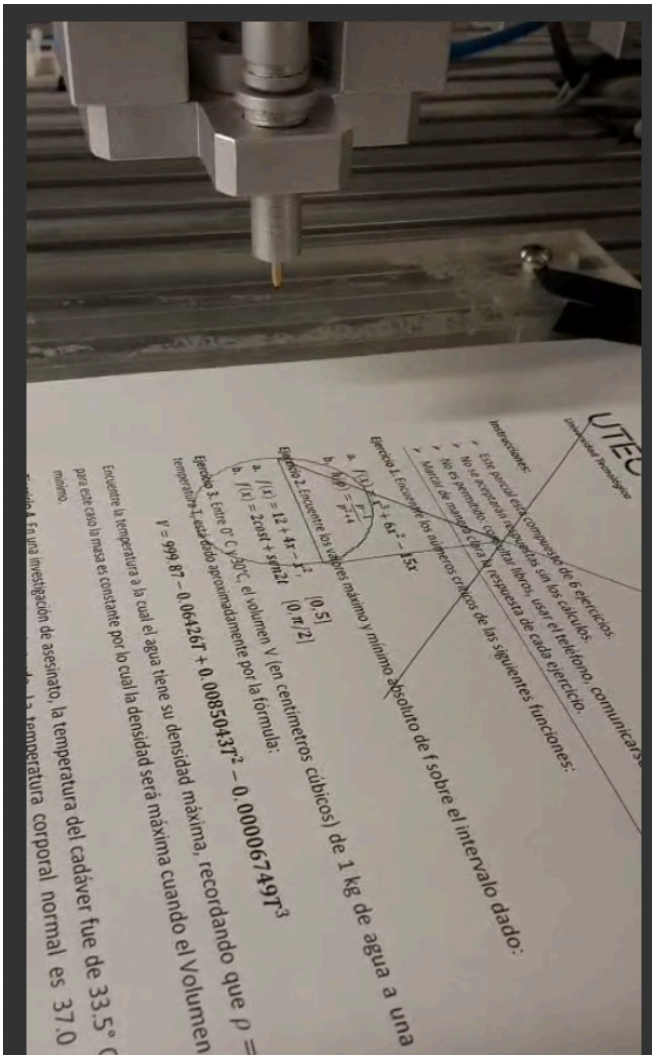

Fig 5. Dibujos con la plotter

VII.Conclusiones

La forma en que se desarrolló el laboratorio permitió integrar conceptos fundamentales de microprocesadores, comunicación serial y control digital mediante programación en ensamblador. Si bien no todos los problemas pudieron resolverse de forma práctica, los resultados obtenidos constituyen un avance importante en la consolidación de competencias técnicas y en la comprensión de la relación entre hardware y software en sistemas embebidos.

En primer lugar, la experiencia con la punzonadora evidenció la utilidad de las máquinas de estados como herramientas de modelado, aun cuando el programa no pudo ser realizado, por lo cual no se logró hacer la punzonadora funcionar. El diseño teórico permite comprender la secuencia de operación del sistema y sentó las bases para futuras implementaciones, resaltando el valor de separar el diseño lógico de la validación práctica.

En segundo lugar, la implementación de la matriz de LEDs demostró la efectividad de las rutinas en ensamblador para gestionar procesos de multiplexación y sincronización temporal. Este resultado confirmó la importancia de planificar estructuras de datos como las LUT para optimizar el control de dispositivos visuales.

De la misma manera, el problema del DAC R-2R no pudo desarrollarse, lo cual refleja la necesidad de una mejor gestión del tiempo y de los recursos. No obstante, la lectura de la consigna permitió reconocer la importancia del manejo de señales analógicas en aplicaciones embebidas, lo que será relevante para trabajos futuros.

Por otro lado, el plotter constituyó el logro más significativo del laboratorio, ya que se implementó y verificó físicamente el control de movimientos, la selección de figuras y las rutinas de seguridad. Este resultado evidenció la capacidad del grupo para integrar programación en ensamblador, control de hardware externo mediante transistores MOSFET y comunicación serial, alcanzando una solución funcional alineada con los objetivos propuestos.

En conjunto, el laboratorio permitió reforzar la importancia del uso de ensamblador como herramienta de control de bajo nivel, la planificación mediante máquinas de estados, y la comunicación serial como medio de interacción entre sistemas. Aunque los resultados fueron parciales en algunos aspectos, la experiencia contribuyó al desarrollo de una visión más integral de los sistemas digitales y de los desafíos inherentes a su implementación práctica.

VIII.Referencias

[1] Martínez, J., y Díaz, M. (2025). *Guía del 1er Laboratorio - GPIO*. Unidad Curricular: Tecnologías de Microprocesamiento. Universidad Tecnológica (UTEC), Fray Bentos, Uruguay.

[2] Microchip Technology Inc. (2018). *ATmega48A/PA/88A/PA/168A/PA/328/P Datasheet*. DS40002061B.

[3] Fischertechnik GmbH. (2018). *ROBOTICS BT Smart Beginner Set - Assembly instruction*. 163330.

[4] Fischertechnik GmbH. *ROBOTICS BT Smart Beginner Set - Cuaderno adjunto*.

IX.Apéndices

Apéndice A: Recursos Digitales

1. Repositorio de Código en GitHub

Enlace:https://github.com/GabiiST16/Tec.Micro/tree/e3c52027cd0be7d873d61adab154d20f5d3e97c3/Laboratorios/Lab1

Este repositorio contiene todo el código fuente en lenguaje ensamblador (.asm) desarrollado para el control del plotter y la matriz de LEDs.

2. Evidencia en Video (Google Drive)

La siguiente carpeta de Google Drive contiene los videos que demuestran el funcionamiento de los sistemas implementados, incluyendo la matriz de LEDs mostrando el mensaje y las figuras, y el plotter dibujando las formas geométricas.

Enlace:https://drive.google.com/drive/folders/1Wx-3IJdTCwUL3bhP4eVwVk7p2S7ap9FT?usp=sharing

Apéndice B: Ploteadora

```
1
2   .include "m328pdef.inc"
3   .CSEG
4   .ORG 0x0000
5   RJMP Seteo
6   Seteo:
7       ; Inicializar Stack Pointer al final de la SRAM
8       LDI R16, LOW(RAMEND)
9       OUT SPL, R16
10      LDI R16, HIGH(RAMEND)
11      OUT SPH, R16
12      LDI R17, 0xFC
13      OUT DDRD, R17
14
15  MainLoop:
16      RCALL Pos
17  Main2:
18      RJMP Main2
19  Mseg:
20      LDI R21, 40
21      LDI R22, 100
22      LDI R23, 189
23  L1:
24      DEC R23
25      BRNE L1
26      DEC R22
27      BRNE L1
28      DEC R21
29      BRNE L1
30      NOP
31      RET
32
33  Pos:
34      LDI R17, 0X00       adrielcastillo, last week • Update codigo …
35      LDI R17, 0X80
36      OUT PORTD, R17
37      RCALL Mseg
38      RCALL Mseg
39      RCALL Mseg
40      RCALL Mseg
41      RCALL Mseg
42      LDI R17, 0X00
43      OUT PORTD, R17
```

Fig 6. Código para probar el funcionamiento de la plotter

```
115 tri:
116     LDI R17, 0X00
117     LDI R17, 0X04
118     OUT PORTD, R17
119     RCALL Mseg
120     LDI R17, 0X00
121     LDI R17, 0X14
122     OUT PORTD, R17
123     RCALL Mseg
124     LDI R17, 0X00
125     OUT PORTD, R17
126     LDI R17, 0X44
127     OUT PORTD, R17
128     RCALL Mseg
129     LDI R17, 0X00
130     OUT PORTD, R17
131     LDI R17, 0XA4
132     OUT PORTD, R17
133     RCALL Mseg
134     LDI R17, 0X08
135     OUT PORTD, R17
136     RCALL Mseg
137     LDI R17, 0X00
138     OUT PORTD, R17
139     RET
```

Fig 7. Fragmento de código para hacer el triángulo

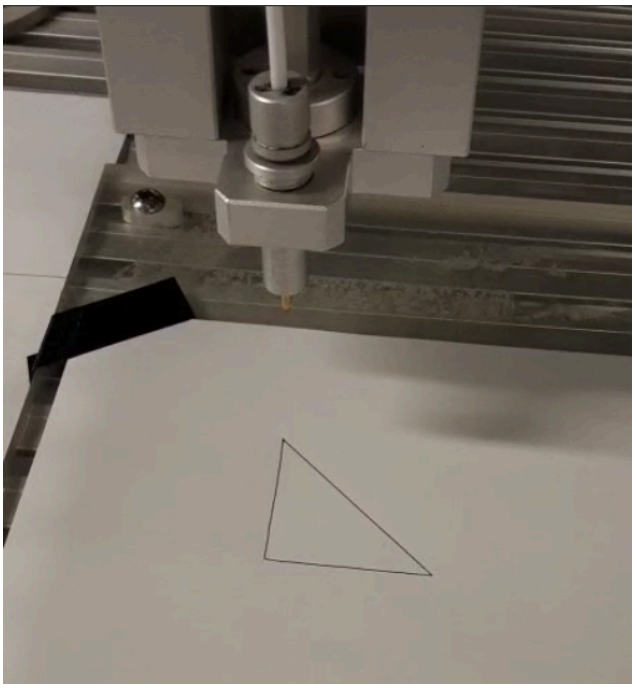

Fig 8. Triángulo dibujado por la plotter

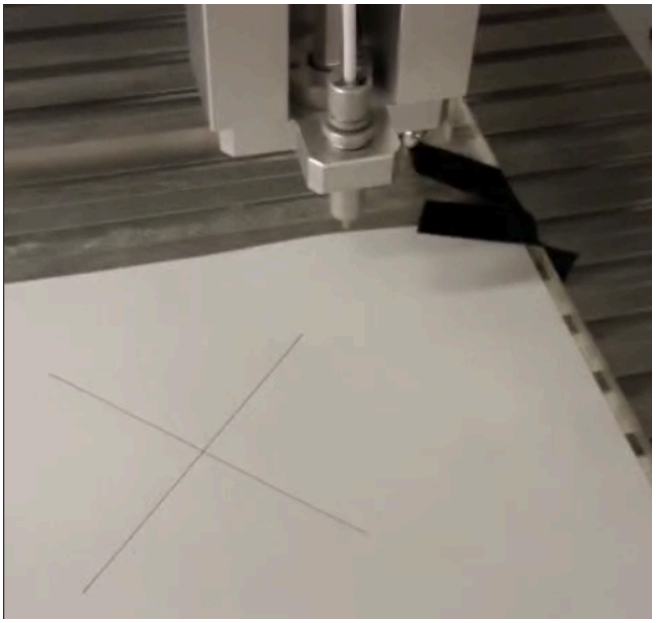

Fig 9. Cruz dibujada por la plotter

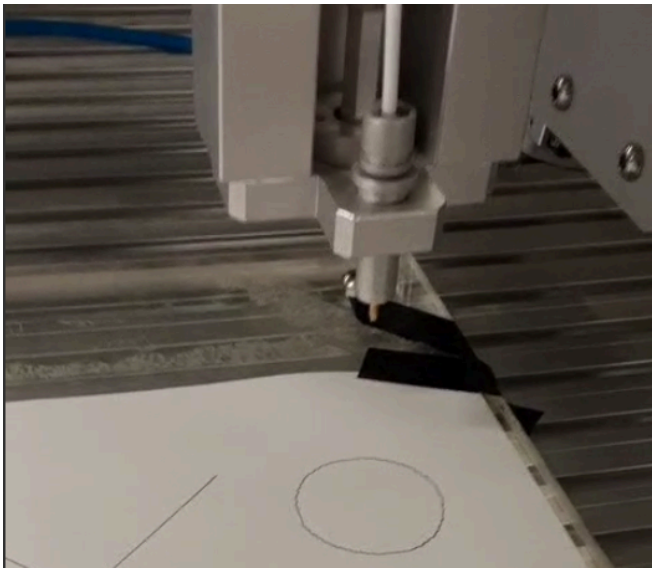

Fig 10. Circulo dibujado por la plotter

Apéndice C: Matriz de LEDs

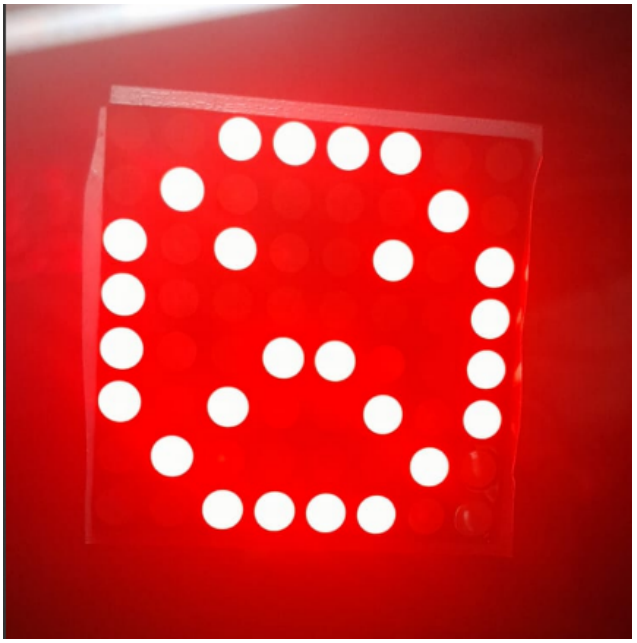

Fig 11. Carita triste

Fig 12. Carita feliz

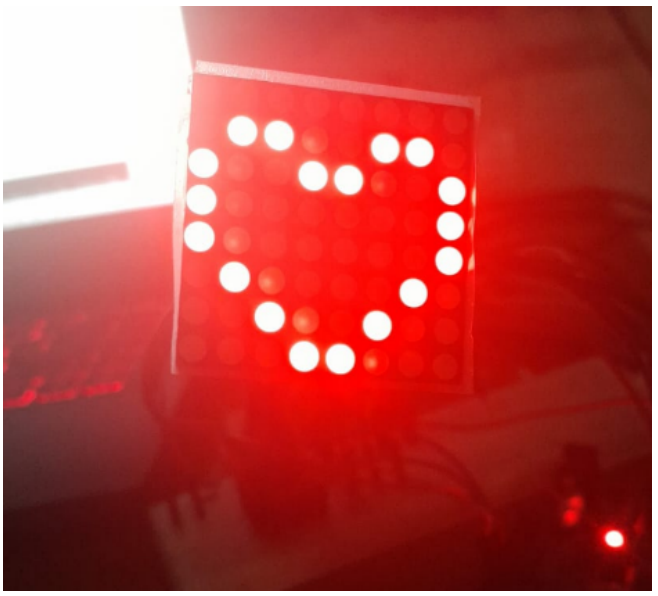

Fig 13. Corazón

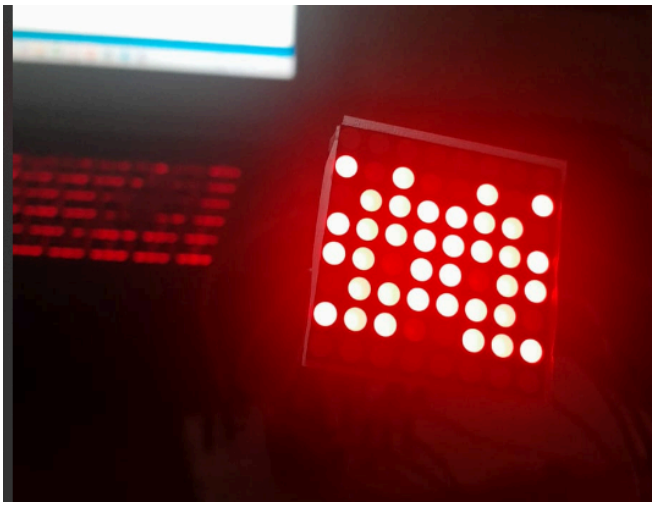

Fig 14. Alien

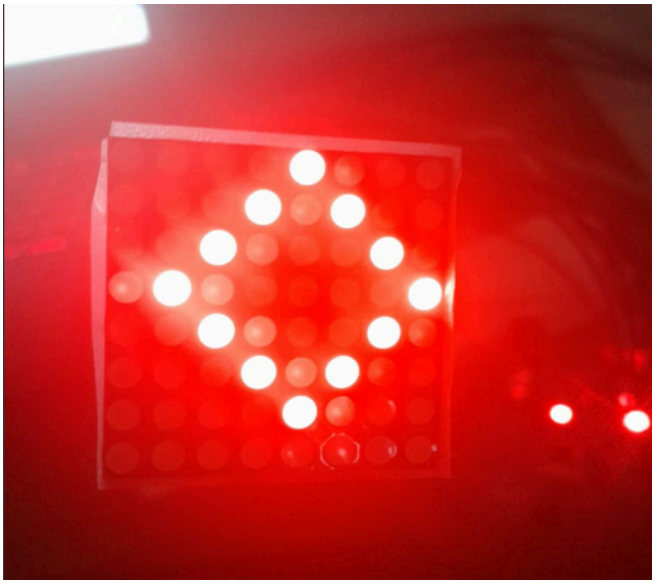

Fig 15. Rombo