

# Laboratorio 1: GPIO en ASM

**Sofía Modernell, Guadalupe Sosa, Victoria Etcheverry**

Carrera: Ingeniería en Mecatrónica, Universidad Tecnológica (UTEC)

Fray Bentos, Uruguay

victoria.etcheverry@estudiantes.utec.edu.uy, sofia.modernell@estudiantes.utec.edu.uy, guadalupe.sosa@estudiantes.utec.edu.uy

26 de septiembre de 2025

**Resumen**—En el presente laboratorio se abordó el diseño, programación y control de sistemas mecatrónicos utilizando el microcontrolador ATmega328P, integrando hardware y software en dispositivos prácticos. Se implementaron cuatro bloques principales: cinta transportadora con punzonadora, matriz de LEDs 8x8, conversor DAC R-2R con tabla de búsqueda y plotter. Cada proyecto permitió reforzar competencias en control de sistemas, programación en lenguaje ensamblador y verificación experimental. Las prácticas incluyeron la gestión de modos de operación, control de actuadores, comunicación UART y generación de señales analógicas a partir de datos digitales. Se documentaron los resultados y se analizaron las limitaciones técnicas, destacando la importancia de la planificación y verificación rigurosa en sistemas mecatrónicos.

**Keywords:** ATmega328P, Mechatronic systems, Assembly language programming, Conveyor belt, LED matrix, R-2R DAC, Plotter, UART communication, Hardware-software integration

**Abstract**—This laboratory focused on the design, programming, and control of mechatronic systems using the ATmega328P microcontroller, integrating hardware and software in practical devices. Four main projects were carried out: a conveyor belt with puncher, an 8x8 LED matrix, an R-2R DAC with a lookup table, and a plotter. Each project reinforced skills in system control, assembly language programming, and experimental validation. Activities included operation mode management, actuator control, UART communication, and analog signal generation from digital data. Results were documented and technical limitations were analyzed, highlighting the importance of planning, verification, and rigorous implementation in mechatronic systems.

## I. INTRODUCCIÓN

**E**n el presente laboratorio, realizamos actividades de programación en lenguaje ensamblador aplicadas al microcontrolador ATmega328P, explorando la interacción entre software y hardware a través de sistemas electrónicos y mecánicos. Durante la práctica, desarrollamos distintos proyectos que nos permitieron experimentar directamente cómo la programación en bajo nivel controla periféricos, gestiona señales digitales y temporizaciones, y coordina dispositivos mecánicos en sistemas embebidos.

Las actividades incluyeron la implementación de una cinta transportadora con punzonadora, la programación de una matriz de LEDs de 8x8, la construcción de un conversor digital-analógico (DAC) R-2R y el control de un plotter capaz de dibujar figuras predeterminadas. Cada sistema presentó desafíos particulares en términos de programación, sincronización de acciones y control de entradas y salidas digitales, lo que nos permitió aplicar conceptos de electrónica, comunicación serial y control de movimiento de manera práctica.

Durante el desarrollo, planificamos y diseñamos diagramas de máquina de estados para organizar las secuencias de el

sistemas que lo ameritaba, configuramos los puertos y registros del microcontrolador, definimos variables y constantes, y desarrollamos rutinas que ejecutan acciones temporizadas y responden a señales externas. También utilizamos USART y UART para establecer comunicación serial, permitiendo seleccionar opciones y monitorear el funcionamiento de los sistemas en tiempo real.

Además, realizamos simulaciones previas de los códigos para comprobar su funcionamiento antes de la implementación física, y posteriormente conectamos los componentes siguiendo las configuraciones definidas en el código.

## II. OBJETIVOS

### *Objetivo general*

Fortalecer las habilidades en el diseño y control de sistemas mediante la implementación de soluciones basadas en el microcontrolador ATmega328P, desarrollando la capacidad de integrar hardware y software para resolver problemas prácticos. Asimismo, fomentar la comprensión del trabajo con periféricos, la comunicación serial y la generación de señales, promoviendo un enfoque integral que combine teoría, práctica y documentación teórica.

### *Objetivos específicos:*

- Construir y programar el modelo de cinta transportadora con punzonadora, gestionando modos de operación, selección de carga y comunicación serial via USART.
- Implementar una matriz de LEDs controlada por el microcontrolador que despliegue mensajes desplazables y permita interacción con el usuario mediante UART.
- Desarrollar una Look Up Table (LUT) de una señal brindada e implementarla en un conversor DAC R-2R de 8 bits y verificar su comportamiento mediante osciloscopio.
- Programar el funcionamiento de un plotter mediante el microcontrolador, permitiendo la selección y trazado de figuras geométricas predefinidas a través de comunicación serial.
- Gestionar los códigos en lenguaje ensamblador utilizando GitHub, con control de versiones y documentación de las modificaciones realizadas.

### III. MATERIALES UTILIZADOS

TABLE I  
MATERIALES UTILIZADOS

MATERIALES	CANTIDAD
Arduino Uno Rev3	1
Protoboard	1
Kit de Fischertechnik	1
Fuente de DC	1
Pulsadores	4
LEDs roja	2
LEDs verde	1
LEDs amarilla	3
Driver L298	1
Resistencia 2k	9
Resistencia 1k	7
Osciloscopio	1
Matriz LEDs 8x8	1
Cables	-
Pen Plotter	1
Microchip Studio	-

### IV. MARCO TEÓRICO

#### Arduino UNO Rev3

El Arduino Uno Rev3 es una placa de desarrollo ampliamente utilizada en el ámbito educativo y en el prototipado de sistemas electrónicos, principalmente por su facilidad de uso, bajo costo y la gran comunidad que respalda su ecosistema. Esta placa se basa en el microcontrolador ATmega328P.

El dispositivo funciona con una tensión nominal de 5 V regulados y puede ser alimentado tanto a través del puerto USB como mediante una fuente externa, recomendándose un rango de voltaje entre 7 V y 12 V para asegurar un rendimiento estable. Esta flexibilidad en la alimentación facilita su integración en diferentes entornos y aplicaciones.

En cuanto a su arquitectura de pines, el Arduino Uno dispone de 14 pines digitales de entrada/salida, de los cuales 6 pueden ser utilizados como salidas PWM, además de 6 entradas analógicas identificadas como A0 a A5. La placa incluye también pines de referencia y alimentación, como VIN, AREF, conexiones a tierra, así como salidas de 3.3 V y 5 V para alimentar periféricos externos. Asimismo, incorpora interfaces de comunicación que permiten su interacción con otros dispositivos electrónicos: UART (pines 0 y 1), SPI (pines 10 a 13) e I<sup>2</sup>C (pines A4 y A5). Esta diversidad de opciones convierte al Arduino Uno en una plataforma versátil para la interconexión de sensores, actuadores y módulos de expansión.

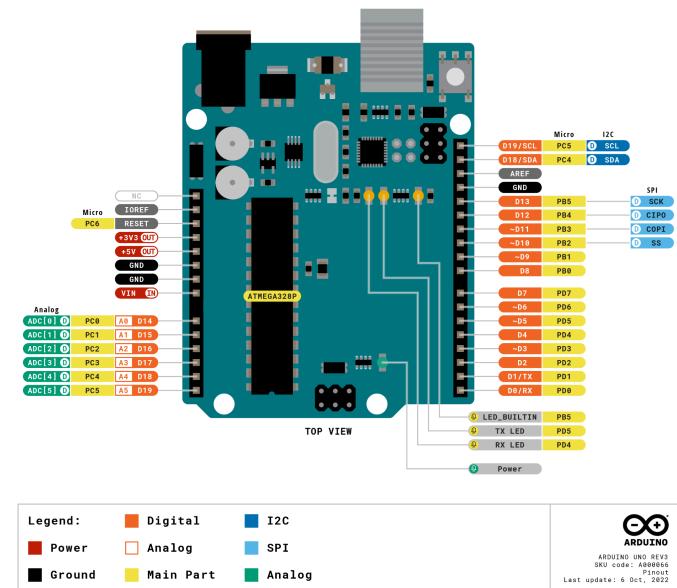


Fig.1 Arduino Pinout

#### ATmega328p

El ATmega328P es un microcontrolador de 8 bits de la serie megaAVR desarrollado por Atmel, ampliamente utilizado en sistemas embebidos y proyectos de electrónica de bajo consumo y bajo costo. Su arquitectura se basa en un microcontrolador RISC, lo que le permite ejecutar instrucciones en un solo ciclo de reloj, alcanzando hasta 1 MIPS, optimizando la velocidad de procesamiento y el consumo energético.

Entre sus principales características se destacan:

- Memoria: 32 KB de memoria flash (ISP) con capacidad de lectura mientras se escribe, 1 KB de memoria EEPROM y 2 KB de SRAM.
- Entradas/Salidas: Hasta 23 líneas de E/S de propósito general, 32 registros de propósito general y 6 canales PWM.
- Convertidor A/D: 6 canales de 10 bits, que permiten la adquisición de señales analógicas de los sensores.
- Temporizadores: 3 temporizadores flexibles con modo de comparación y un temporizador “watchdog” programable.
- Interfaz de comunicación: Soporta USART, SPI y comunicación serie de 2 hilos.
- Modos de ahorro de energía: Cinco modos configurables por software, que optimizan la autonomía del dispositivo en aplicaciones portátiles.
- Rango de operación: Funciona entre 1,8 y 5,5 V, lo que lo hace compatible con distintos sistemas de alimentación.

El ATmega328P se utiliza en proyectos que requieren control preciso de periféricos, adquisición de datos y transmisión de información, siendo la base de plataformas como Arduino Uno y Nano.



Fig.2 Microcontrolador ATmega328P.

#### *Ensamblador*

El lenguaje ensamblador es un lenguaje de programación de bajo nivel que proporciona una representación simbólica de las instrucciones de máquina específicas de una arquitectura de procesador. A diferencia de los lenguajes de alto nivel, el lenguaje ensamblador está estrechamente vinculado al hardware, lo que permite un control preciso sobre los recursos del sistema. Cada arquitectura de procesador tiene su propio conjunto de instrucciones, y el lenguaje ensamblador utiliza mnemónicos para representar estas instrucciones, facilitando su comprensión y escritura por parte del programador.

El programa ensamblador traduce el código fuente escrito en lenguaje ensamblador a código máquina ejecutable por el microprocesador. Este proceso de traducción es generalmente directo, con una correspondencia uno a uno entre las instrucciones mnemotécnicas y las instrucciones de máquina. Sin embargo, algunos ensambladores modernos ofrecen características adicionales, como el soporte para macros, que permiten la reutilización de fragmentos de código y facilitan la programación.

El uso del lenguaje ensamblador es particularmente relevante en situaciones donde se requiere un alto rendimiento, un control detallado del hardware o la optimización de recursos limitados, como en sistemas embebidos o en el desarrollo de controladores de dispositivos. Aunque los lenguajes de alto nivel han ganado popularidad debido a su facilidad de uso y portabilidad, el lenguaje ensamblador sigue siendo una herramienta valiosa en áreas donde el control preciso y la eficiencia son esenciales.

#### *UART y USART*

Los protocolos USART (Universal Synchronous Asynchronous Receiver Transmitter) y UART (Universal Asynchronous Receiver Transmitter) son fundamentales en sistemas embebidos para la transmisión de datos entre dispositivos. Ambos permiten la comunicación serial, pero presentan diferencias clave en su funcionamiento y aplicaciones.

El UART es un protocolo de comunicación serial asíncrona que utiliza dos líneas: una para la transmisión (TX) y otra para la recepción (RX). Su principal característica es que no requiere una señal de reloj externa, ya que la sincronización entre el transmisor y el receptor se establece mediante la configuración previa de parámetros como la velocidad de transmisión (baud rate), bits de datos, paridad y bits de parada. Esta simplicidad lo hace ideal para comunicaciones de corto alcance y baja velocidad, como la conexión entre microcontroladores y módulos periféricos.

Por otro lado, el USART extiende las capacidades del UART al permitir tanto comunicación asíncrona como síncrona.

Además de las líneas TX y RX, incorpora dos señales adicionales: XCK (Clock) y XDIR (Direction). La señal XCK proporciona un reloj para la sincronización de datos en modo síncrono, mientras que XDIR indica la dirección de la transmisión en comunicaciones bidireccionales. Estas características permiten al USART operar a mayores velocidades y con mayor precisión en entornos donde la sincronización es crítica, como en comunicaciones industriales o entre dispositivos que requieren alta tasa de transferencia de datos.

En términos de compatibilidad, es común que los microcontroladores implementen interfaces USART que puedan configurarse para operar en modo UART, adaptándose así a diferentes necesidades de comunicación sin requerir hardware adicional. Esta flexibilidad facilita el diseño de sistemas que pueden cambiar entre modos síncronos y asíncronos según los requisitos específicos de la aplicación.

#### *Diagrama de máquina de estados*

El diagrama de máquina de estados en UML es un diagrama de comportamiento que describe cómo un objeto cambia entre distintos estados a lo largo de su ciclo de vida, en función de eventos que ocurren. Un estado representa una situación en la que el objeto cumple ciertas condiciones o ejecuta actividades específicas, mientras que una transición muestra el paso de un estado a otro, provocado por un evento y opcionalmente condicionado por una guarda o acompañado de una acción.

Dentro del diagrama se incluyen símbolos estandarizados como el estado inicial (círculo sólido) y el estado final (círculos concéntricos). También es posible modelar estados compuestos, que contienen subestados, así como regiones ortogonales, que permiten representar comportamientos concurrentes. En algunos casos se emplean estados de historia, que retoman el subestado previamente activo en lugar de reiniciar.

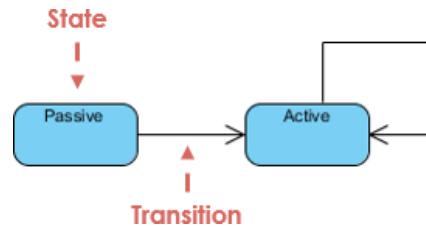


Fig.3 Estado y transición en diagramas de máquinas de estado Punzonadora con cinta transportadora

En el kit Smart Beginner se incluye un sistema que simula una cinta transportadora con punzonadora, destinado a la enseñanza de automatización y control. Este sistema permite experimentar con el transporte de objetos a lo largo de una banda motorizada, mientras se ejecuta un mecanismo de perforación de manera sincronizada. La banda transportadora está diseñada para desplazarse de forma continua mediante un pequeño motor eléctrico, permitiendo que los objetos se muevan a lo largo de un recorrido predefinido dentro del área del kit.

El mecanismo de punzonadora del kit se acciona en coordinación con el movimiento de la cinta, realizando perforaciones en los objetos de manera controlada. Este proceso permite comprender cómo se integran movimiento y acción sobre los materiales en sistemas automatizados, sin requerir equipos industriales de gran escala. La combinación de la banda y la punzonadora en el kit permite a los estudiantes

observar y controlar aspectos como la temporización, la sincronización y la precisión de las acciones mecánicas.

El uso de este sistema en el kit educativo facilita la comprensión de conceptos de automatización, como la secuencia de operaciones, la coordinación de componentes mecánicos y la programación de movimientos. A través de la manipulación de la cinta y la punzonadora, los estudiantes pueden experimentar de manera práctica con principios de ingeniería, control y procesos industriales, en un entorno seguro y de escala reducida.

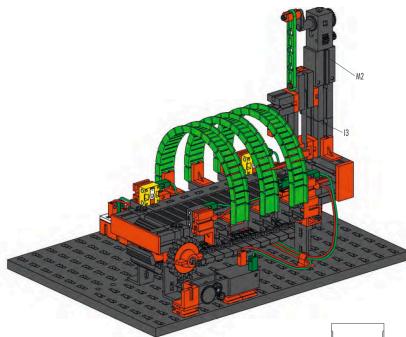


Fig.4 Cinta transportadora con punzonadora  
Puente H- Driver L298

Un puente H es una configuración circuital con diodos que permite controlar el giro de un motor, funcionando como controlador de la polaridad aplicada al motor. Existen drivers que ya integran esta electrónica, facilitando el control de motores sin necesidad de armar el circuito completo. Un ejemplo clásico es el L293; en esta sección se analiza el driver L298.

El driver L298 permite controlar dos motores de corriente continua o motores paso a paso con una corriente máxima de 2 amperios. Actualmente existen módulos comerciales que incluyen el driver soldado y conectores para las entradas y salidas, facilitando su uso. En el módulo típico del driver L298 se encuentran los siguientes componentes:

Regulador de voltaje LM7805, que permite alimentar la lógica del módulo.

Conectores de salida Output A y Output B para conectar los motores.

Terminales de control (control inputs), que incluyen pines de habilitación para cada motor.

Un jumper que permite decidir si se usa o no el regulador de voltaje LM7805.

Es importante considerar el voltaje con el que se alimentará el módulo. Aunque el driver soporta tensiones de hasta 36 V, si la alimentación es similar a la del microcontrolador (entre 6 y 12 V), no es necesario activar el regulador, ya que el L298 tolera esos niveles. En cambio, si se alimenta con tensiones superiores a 12 V, es obligatorio activar el jumper para usar el regulador; de lo contrario, el módulo puede dañarse rápidamente.

Para controlar un motor de corriente continua con el driver L298, normalmente se utilizan dos pines de control que determinan el sentido de giro del motor, enviando señales digitales de uno o cero según corresponda.

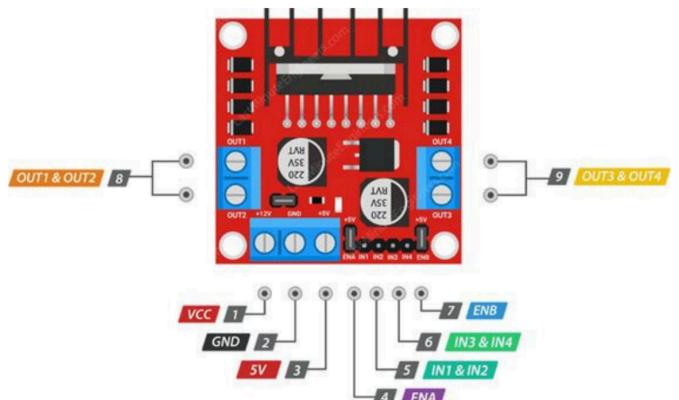


Fig.5 Driver L298 Pinout

#### DAC R-2R

Un DAC R-2R es un convertidor digital-a-analógico que usa únicamente dos valores de resistor (R y 2R) configurados en una red tipo escalera (“ladder”) para convertir un número binario en una señal analógica proporcional.

En su operación, cada bit digital (en lógica alta o baja) se aplica en distintos puntos de la escalera de resistores, generando divisiones de voltaje que luego se combinan y son convertidas a salida analógica mediante un amplificador operacional.

Se puede derivar una expresión general para el voltaje de salida. Si hay n bits, el valor de la salida es la suma ponderada de los bits en posiciones distintas, y el “paso” mínimo (LSB) es  $V_{LSB} = V_{IN}/2^n$ .

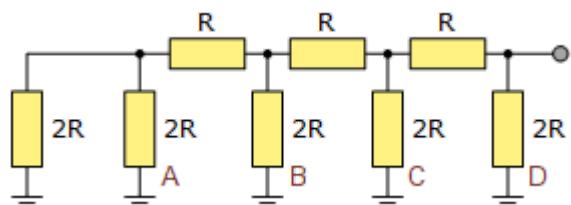


Fig.6 R-2R DAC

#### Matriz de LEDs 8x8

La matriz LED 8×8 es un dispositivo compuesto por 64 diodos emisores de luz dispuestos en ocho filas y ocho columnas, que permiten representar símbolos, caracteres o animaciones simples mediante el encendido controlado de cada punto luminoso. El principio de funcionamiento se basa en la multiplexación, que consiste en activar de forma secuencial cada fila o columna a gran velocidad, aprovechando la persistencia visual del ojo humano para percibir una imagen continua.

El control directo de la matriz requiere hasta 16 pines de salida, lo que resulta poco eficiente al trabajar con microcontroladores de bajo número de pines como el Arduino Uno

En el ámbito de la programación, el control de la matriz puede realizarse de forma directa o mediante librerías específicas para Arduino, que simplifican la escritura de caracteres, el encendido de puntos individuales y la generación de animaciones. Gracias a estas herramientas, es posible implementar efectos de desplazamiento, rotación o transiciones

fluidas, lo que convierte a la matriz LED  $8 \times 8$  en una opción versátil y didáctica para proyectos de visualización electrónica.

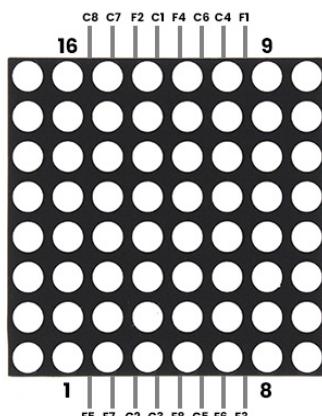


Fig.7 Matriz de LEDs  $8 \times 8$  Pinout

#### Pen Plotter

Un pen plotter es un dispositivo automatizado diseñado para dibujar imágenes o diagramas sobre una superficie plana utilizando un bolígrafo, marcador u otro instrumento de escritura. A diferencia de las impresoras convencionales, que aplican tinta mediante procesos de impresión, los pen plotters mueven el instrumento de manera controlada sobre los ejes X e Y, permitiendo reproducir con precisión figuras vectoriales.

En su funcionamiento típico, la pen plotter utiliza motores paso a paso para controlar el movimiento horizontal y vertical del bolígrafo, mientras que un mecanismo adicional permite levantar y bajar el instrumento de escritura según sea necesario. La máquina puede ser controlada mediante comandos de control de movimiento, como los que se encuentran en el G-code, lo que permite programar trayectorias complejas y repetir dibujos de manera consistente.

Este tipo de dispositivos se emplea tanto en proyectos educativos como en aplicaciones de prototipado rápido y automatización, ofreciendo una plataforma práctica para comprender conceptos de control de movimiento, sincronización de ejes y coordinación mecánica, sin necesidad de equipos industriales complejos.



Fig.8 Pen Plotter

TABLE II  
CONEXIONES PEN PLOTTER

Pin Digital	Conexión
D2	Bajar solenoide
D3	Subir solenoide
D4	Movimiento hacia abajo
D5	Movimiento hacia arriba
D6	Movimiento hacia la derecha
D7	Movimiento hacia la izquierda

#### V. METODOLOGÍA

En el presente laboratorio se llevaron a cabo actividades basadas en la programación en lenguaje ensamblador aplicadas al microcontrolador ATmega328P. La práctica estuvo compuesta por cuatro actividades principales: la ejecución de una cinta transportadora con punzonadora, la implementación de una matriz LED, el desarrollo de un conversor DAC y la utilización de un plotter. A continuación, se detalla el procedimiento correspondiente a cada una de estas etapas.

##### Parte A:

En esta actividad se planteó la implementación de un código para una cinta transportadora con punzonadora. El sistema debía realizar el siguiente procedimiento: elección de carga, avance de la cinta para posicionar la carga debajo de la punzonadora, detención de la cinta para posteriormente punzar la carga y, finalmente, retorno de la cinta y la carga a su posición inicial. La elección de carga debía contemplar tres opciones: carga ligera, carga media y carga pesada. Cada una de estas presentaba un tiempo especificado para realizar las funciones: avance 3 segundos y pausa de 2 segundos (carga ligera), avance 4 segundos y pausa 2 segundos (carga media), avance 5 segundos y pausa 3 segundos (carga pesada). Este proceso debía funcionar mediante dos tipos de comunicación: USART y lógica cableada.

Iniciando con la resolución de esta actividad, se desarrolló un diagrama de máquina de estados, el cual permitió una clara visualización del problema para su posterior solución. Dicho diagrama cuenta con seis estados: en espera, elección de carga, posicionado, punzonado, descarga y fin del ciclo. Este sistema se vinculó con entradas físicas (pulsadores), tiempos de espera y comunicación serial para controlar las salidas.

Respecto al código, se inició con la configuración de los puertos del microcontrolador, preparando los registros que se utilizaron como variables, habilitando temporizadores e inicializando el USART a 9600 baudios. Posteriormente, se programó el envío de un mensaje de inicio para confirmar que el sistema está activo, quedando en espera en el bucle principal. Dentro de este bucle, se leen las entradas del sistema (pulsadores y sensores) constantemente, comparando cada lectura con la anterior para detectar cambios de estados en el sistema. Si el estado cambia, el programa transmite por USART un mensaje indicando la nueva condición, sincronizando la lógica con una salida informativa.

La lógica principal se basa en el recorrido de los estados mencionados: en el primer estado, el sistema permanece inactivo hasta que se reciba un comando (mediante USART) o una señal física desde un pulsador, si esto sucede, se cambia al siguiente estado, seleccionando una de las cargas disponibles,

encendiendo leds que indican la elección. Este estado, se vincula con el siguiente, activando la salida con los intervalos de tiempo correspondientes, esperando una señal externa para continuar (sensores de posición). Más adelante, se ejecuta una secuencia de pulsos que activa la punzonadora, seguida de un avance mecánico hasta llegar al estado final, en el cual se enciende un LED de cierre segundos antes de reiniciar el ciclo completo.

Los intervalos de tiempo se implementaron mediante un temporizador que genera retardos de un segundo, multiplicados por un contador para alargar las interrupciones según sea necesario. A su vez, la comunicación USART permite controlar y monitorear el flujo: al recibir caracteres específicos, se avanza en la máquina de estados, se selecciona la carga, o incluso se reinicia el sistema en cualquier momento.

Previo a la implementación física, se simuló el funcionamiento del código en PICsimLab considerando todas las entradas del sistema como pulsadores y todas las salidas como LEDs.

Posteriormente, se ensambló la cinta transportadora con punzonadora utilizando los kits Fischertechnik, siguiendo paso a paso el manual de instrucciones. Una vez finalizado el armado, se procedió a conectar la cinta con el microcontrolador y la fuente de alimentación. Para estas conexiones, se consideraron las declaraciones del código desarrollado previamente.

Con respecto a la conexión entre la cinta y el microcontrolador, se conectó el sensor de inicio a PC2 y el sensor de finalización a PD6. Por otra parte, los motores (cinta y punzonadora) se conectaron a un puente H y posteriormente al arduino: la cinta a PB5 y PC3 y la punzonadora a PC0. Esta conexión extra al puente H fue necesaria debido a que el arduino trabaja con 5 V y la cinta con aproximadamente 9 V. Además, se requirió utilizar todos los pines del arduino, lo que impidió utilizar la adaptación del kit para su propia batería, ya que esto limitaba los pines de conexión.

Posteriormente, se estableció el vínculo entre la lógica cableada y el arduino, para esto se realizaron las siguientes asignaciones: botón de inicio a PD2, botón de carga ligera a PD3, botón de carga mediana a PD4, botón de carga pesada a PD5, led de espera a PB0, led de funcionamiento a PB1, led de carga ligera a PB2, led de carga mediana a PB3, led de carga pesada a PB4 y led de finalización a PC1.

Finalmente, se corroboró el correcto funcionamiento de todos los casos posibles. La cinta transportadora con punzonadora respondió de manera adecuada tanto a los comandos enviados por USART como a la lógica cableada, cumpliendo con las secuencias de avance, posicionamiento, punzonado y descarga definidas en la máquina de estados.

#### *Parte B.*

En esta actividad, se planteó el desarrollo de un código capaz de comunicar al usuario, mediante USART, un conjunto de opciones predefinidas para posteriormente representarlas en una matriz de LEDs de 8x8. Las opciones implementadas fueron: un mensaje desplazante y dos figuras (una cara feliz y un corazón). Los requisitos funcionales establecidos incluyeron: una clara visualización del mensaje mediante un desplazamiento continuo y suave, una velocidad de scroll ajustable, comunicación exclusivamente mediante USART y una clara documentación del código para facilitar su reutilización.

El programa está organizado en tres bloques principales que interactúan para controlar una matriz de LEDs y la comunicación USART. Primero, se inicializa la USART para habilitar la transmisión y recepción de datos seriales, permitiendo mostrar mensajes en la terminal y recibir la selección del usuario. Al iniciar, el programa muestra un mensaje de bienvenida y un menú con las opciones disponibles: “1: Mensaje desplazante”, “2: Carita Feliz” y “3: Corazón”. Una vez seleccionada una opción, el programa ejecuta la rutina correspondiente.

En el caso de la opción 1, se activa el mensaje desplazante (scroll), el cual se implementa almacenando cada columna de los caracteres del mensaje en un buffer de memoria. El desplazamiento se logra mediante un índice llamado offset que recorre las columnas del buffer, mostrando progresivamente ocho columnas a la vez en la matriz de LEDs. Cada columna se envía a los pines correspondientes de las filas y columnas del display mediante multiplexado: se activa una fila a la vez, cargando los bits de la columna actual en los puertos de salida y aplicando un retardo muy breve antes de pasar a la siguiente fila. La velocidad del desplazamiento se controla mediante un contador de frames, al llegar al final del mensaje, el scroll se reinicia de manera continua, creando un efecto de movimiento fluido.

Por otra parte, al seleccionar las opciones 2 o 3, el programa permite mostrar figuras fijas: una carita feliz o un corazón, activando secuencialmente las columnas de la matriz para cada fila según la forma de la figura.

Además, mientras se ejecuta cualquier opción, el programa revisa constantemente si hubo un cambio de selección mediante la USART, permitiendo que se pueda cambiar dinámicamente entre scroll y las figuras sin necesidad de reiniciar el sistema.

Posteriormente a la realización del código, se procedió a realizar la conexión física de la matriz LEDs y el microcontrolador. Esta etapa consistió principalmente en asociar cada pin de la matriz con el correspondiente pin del Arduino, asegurando que esta correspondencia siguiera lo definido en el código. Para proteger los LEDs y limitar la corriente se colocaron resistencias en los pines correspondientes a las columnas de la matriz. La conexión se realizó siguiendo un esquema claro y ordenado, garantizando que cada fila y cada columna pudiera ser controlada de manera independiente mediante las rutinas de multiplexado implementadas en el código. La configuración exacta de los pines utilizados y su correspondencia con la matriz se detalla en el apartado de resultados como “TABLE V”.

Finalmente, se verificó el correcto funcionamiento del sistema y se realizó la documentación requerida.

#### *Parte C:*

En este apartado se requirió la realización de un código capaz de implementar una LUT que muestre mediante un osciloscopio una señal analógica, implementando además un conversor DAC R-2R de 8 bits.

En primer lugar, se inicializó el sistema configurando la pila y estableciendo el puerto D como salida digital, de modo que cada valor escrito en este puerto fuera convertido en un voltaje proporcional por el DAC. Posteriormente, se implementó una LUT en memoria de programa que contenía valores correspondientes a la forma de onda a generar, los cuales fueron recorridos secuencialmente mediante el registro puntero

Z. Para garantizar la temporización adecuada, se configuró el Timer0 en modo CTC utilizando el registro OCR0A como comparador, lo que permitió establecer una frecuencia de muestreo estable y ajustar la frecuencia de la señal. En cada coincidencia del temporizador, se cargó el siguiente valor de la LUT y se envió al puerto de salida, reiniciando el puntero al llegar al final de la tabla para lograr una señal periódica y continua.

Posteriormente, se ensambló la parte física, implementando red resistiva R-2R utilizando resistencias de 2k ohm y 1k ohm, donde cada bit del puerto de salida del microcontrolador se conectó a una rama de la red de resistencias. En esta disposición, cada resistencia de 1k ohm se conecta al pin correspondiente del microcontrolador y a una resistencia de 2k ohm que va a tierra, mientras que otra resistencia de 2k ohm continúa hacia el siguiente escalón. Repitiendo esto para los ocho bits, se obtuvo en el nodo final la salida analógica proporcional al valor digital aplicado en el puerto, asegurando que el bit más significativo aportará la mayor contribución en voltaje y el menos significativo la menor. Finalmente, el último nodo se conectó a la entrada del osciloscopio, permitiendo visualizar y analizar la señal generada por el sistema.

#### Parte D y E:

En esta última actividad se desarrolló un sistema de control para un plotter con solenoide, utilizando lenguaje ensamblador y el microcontrolador ATmega328P. El objetivo fue implementar un programa capaz de recibir comandos por UART y dibujar figuras predeterminadas (un triángulo, una cruz o ambas de manera secuencial) sobre la superficie de trabajo.

El código comienza con la inicialización de la pila y la configuración de los periféricos necesarios. Se habilita la comunicación UART a 9600 baudios para enviar mensajes informativos y recibir la opción seleccionada por el usuario. Además, se configuran como salidas los pines digitales destinados a controlar tanto los motores del sistema de movimiento como el solenoide encargado de subir y bajar el marcador. Se utilizó además un LED de prueba en PB5 como indicador del estado de ejecución.

El sistema transmite al inicio un mensaje de bienvenida y un menú de selección con tres opciones: “1: Triángulo”, “2: Cruz” y “T: Ambas figuras”. El usuario selecciona la figura deseada ingresando el carácter correspondiente en la terminal, lo que actualiza la variable de control interna. Al recibir una opción válida, el sistema enciende el LED, envía un mensaje de confirmación por UART y llama a una función para posicionar el cabezal. Al finalizar, se asegura que el solenoide quede en la posición elevada y el sistema retorna al menú, listo para recibir nuevas instrucciones.

La lógica principal del trazado de las figuras se basa en una serie de subrutinas que controlan el solenoide para subir y bajar el marcador, junto con rutinas de movimiento que activan los motores del cabezal durante tiempos específicos. Para trazar el triángulo, el programa primero baja el marcador y luego ejecuta tres movimientos coordinados: hacia abajo, hacia la izquierda, y finalmente una diagonal hacia arriba y a la derecha. La rutina para la cruz baja el marcador y traza dos líneas perpendiculares, una vertical (moviéndose arriba y abajo) y otra horizontal (moviéndose izquierda y derecha). La rutina que dibuja ambas figuras simplemente ejecuta las dos anteriores en secuencia, asegurando que el cabezal se posiciona

en el centro entre cada una. Los retardos se implementaron mediante una subrutina de temporización en milisegundos, ajustando la duración de cada desplazamiento.

Las conexiones se encontraban previamente realizadas, solo fue necesario analizar las asociaciones de los pines para asegurar declaraciones correctas en el código y conectar el arduino a una terminal para enviar el código

## VI. RESULTADOS

En este apartado se presenta y analiza cada resultado o avance obtenido mediante la práctica, lo cuál incluye documentación visual de las actividades y fragmentos importantes de sus códigos.

### Parte A:

#### Máquina de Estados:

En el Apéndice I se presenta el diagrama de la máquina de estados, mientras que en el Apéndice II se muestra el código implementado para su realización. Dicho diagrama describe el funcionamiento secuencial del sistema, el cuál se explicará brevemente. El Estado 0 representa la condición inicial o de espera, donde todos los actuadores están apagados y únicamente el LED de espera se encuentra encendido, pasando al Estado 1 al recibir la orden de inicio desde un pulsador o por USART. En el Estado 1, se activa el LED de funcionamiento y, según la señal de entrada (sensor de inicio y pulsadores de carga), se selecciona uno de los tres subestados que definen el tipo de carga. Posteriormente, en el Estado 2, la cinta avanza con el LED correspondiente a la carga seleccionada encendido, manteniendo este estado durante un tiempo determinado antes de pasar a una etapa de espera y luego a la punzonadora. El Estado 3 ejecuta la acción de punzonado, manteniendo el LED de carga activo según el subestado en curso, y tras un tiempo específico retorna a la espera o avanza al siguiente estado. En el Estado 4, la cinta retrocede con el LED de carga correspondiente encendido, hasta completar el tiempo asignado para cada subestado. Finalmente, el Estado 5 indica la finalización del ciclo con el LED de fin encendido, regresando después de 5 segundos al estado inicial. Adicionalmente, en cualquier punto del proceso es posible forzar el retorno al Estado 0 mediante la recepción de la orden USART = 'R', asegurando así una condición de reinicio inmediato del sistema.

#### Documentación:

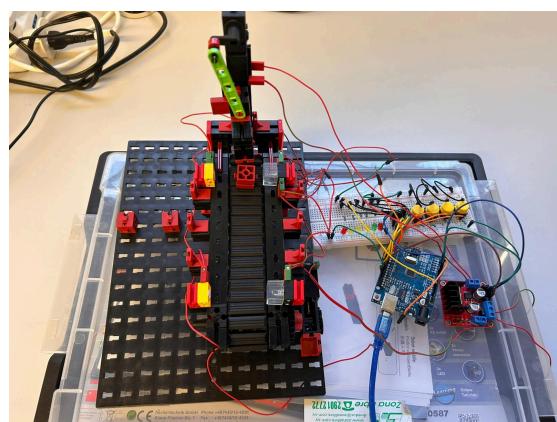


Fig. 9 Configuración apartado A - Cinta transportadora con punzonadora

En la presente imagen se observa la conexión realizada entre el microcontrolador y la cinta transportadora, incluyendo la interacción mediante pulsadores.

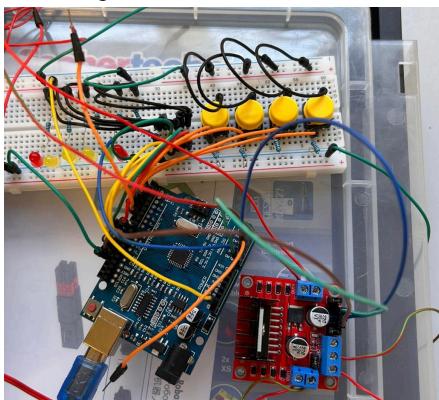


Fig. 10 Configuración apartado A - Lógica cableada

En este caso, se aprecia puntualmente la parte de la comunicación mediante lógica cableada. Los pulsadores funcionan como entradas del sistema, seleccionando el comienzo y la carga a transportar. Por otra parte, las LEDs funcionan como salidas, comunicando al usuario en qué estado se encuentra el sistema.

TABLE III  
MAPEO DE CONEXIONES PARTE A - ENTRADAS

ENTRADAS	PUERTO	PIN
Btn_inicio	D	PD2
Btn_ligera	D	PD3
Btn_media	D	PD4
Btn_pesada	D	PD5
Sensor_fin	D	PD6
Sensor_inicio	C	PC2

En esta tabla se especifica cómo se realizó la conexión del microcontrolador respecto a las entradas del sistema, puntualmente pulsadores y sensores.

TABLE IV  
MAPEO DE CONEXIONES PARTE A - SALIDAS

ENTRADAS	PUERTO	PIN
Led_espera	B	PB0
Led_funciona	B	PB1
led_ligera	B	PB2
Led_media	B	PB3
Led_pesada	B	PB4
Motor_Cinta_B	B	PB5
Motor_Cinta_S	C	PC3
Motor_Punzon	C	PC0
Led_fin	C	PC1

En esta segunda tabla se observa las conexiones del microcontrolador con las salidas del sistema (Motores y LEDs).

Parte B:

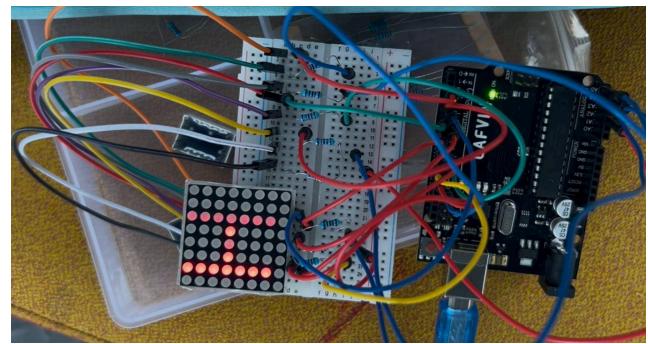


Fig. 11 Configuración apartado B - Matriz Led

En esta figura se observa la conexión entre la matriz led y el arduino, como se mencionó previamente, se conectan resistencias a las columnas para limitar el paso de corriente. Los pines se conectan al arduino como se especifica en la tabla a continuación.

TABLE V  
MAPEO DE CONEXIONES PARTE B

PINES MATRIZ	PINES ARDUINO
F1 - 9	PD2
F2 - 14	PD3
F3 - 8	PD4
F4 - 12	PD5
F5 - 1	PD6
F6 - 7	PD7
F7 - 2	PB0
F8 - 5	PB1
C1 - 13	PB2
C2 - 3	PB3
C3 - 4	PB4
C4 - 10	PB5
C5 - 6	PC5
C6 - 11	PC4
C7 - 15	PC3
C8 - 16	PC2

Tabla que refleja la interacción entre la matriz y el arduino.

TABLE VI  
CORAZÓN EN MATRIZ LED

	1	1			1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
	1	1	1	1	1	1
		1	1	1	1	
			1	1		

En esta tabla vemos una demostración de cómo se recolectaron los datos para pasarlo a la matriz led, este método consta de crear una tabla de 8x8 y colocar un “1” en donde se prenderán las LEDs para la figura que se pretende mostrar, posterior a esto, se escribe en binarios los datos de las filas, tal así como se muestra a continuación:

Fila 1 = 0b00000000

Fila 2 = 0b01100110  
 Fila 3 = 0b11111111  
 Fila 4 = 0b11111111  
 Fila 5 = 0b01111110  
 Fila 6 = 0b00111100  
 Fila 7 = 0b00011000  
 Fila 8 = 0b00000000

Este procedimiento se realizó para todas las letras del mensaje deslizante. Una vez hecho esto, se puede visualizar en código de la siguiente manera:

.db 0b00000000, 0b01100110, 0b11111111, 0b11111111,  
 0b01111110, 0b00111100 , 0b00011000, 0b00000000

*Parte C:*

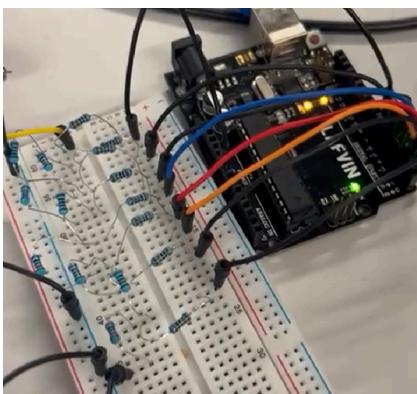


Fig. 12 Configuración - Conversor DAC

En esta figura se observa la red resistiva aplicada para el conversor, recordando que se realizó con resistencias de 2k ohm y 1k ohm.

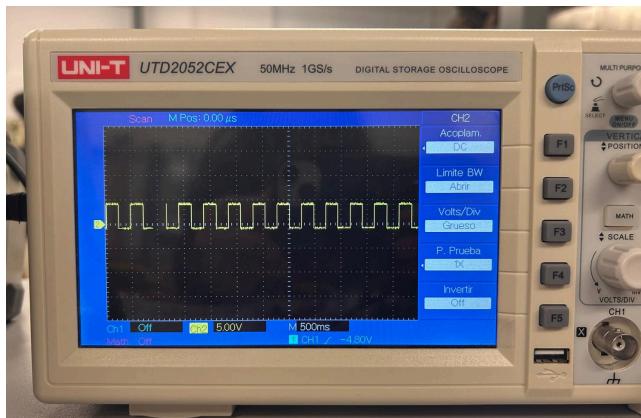


Fig. 13 Señal Analógica - Conversor DAC

En la presente figura se observa la señal analógica lograda gracias al conversor DAC.

*Parte E y D:*

```
Terminal 0
Bienvenido al sistema de dibujo
Seleccione una opcion:
1: Triangulo 2: Cruz T: Ambas figuras
Dibujando figura...
Ambas figuras seleccionadas
Bienvenido al sistema de dibujo
Seleccione una opcion:
1: Triangulo 2: Cruz T: Ambas figuras
Dibujando figura...
Triangulo seleccionado
Bienvenido al sistema de dibujo
Seleccione una opcion:
1: Triangulo 2: Cruz T: Ambas figuras
```

Fig. 14 Mensaje de bienvenida UART - Plotter

En esta figura se visualiza el mensaje de bienvenida mediante UART, como se muestra, el mensaje se ve desde la terminal virtual.

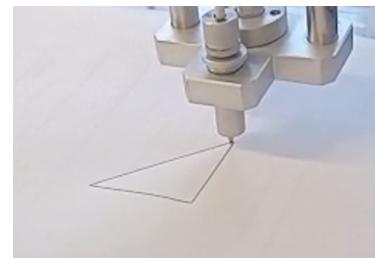


Fig. 15 Figura Triángulo - Plotter

En la presente imagen se observa una de las figuras dibujadas por el Plotter: un triángulo.

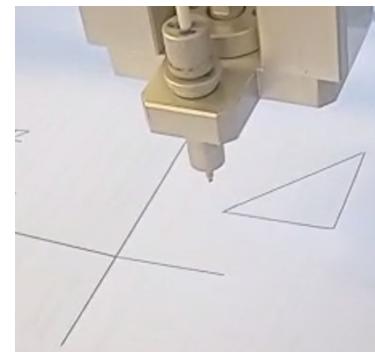


Fig. 16 Figura Cruz - Plotter

Mientras que, en esta imagen, se muestra la segunda figura (Cruz) junto a la anteriormente mencionada.

En este caso el plotter se encontraba en la opción 3, por esta razón, se aprecian ambas figuras.

*Generalidades:*

En todas las consignas se utilizaron conceptos básicos que son especialmente necesarios para el correcto funcionamiento del código. A continuación se explican algunos de estos conceptos y se adjuntan ejemplos de los mismos:

*Configuración de Puertos y registros:*

Los códigos implementados utilizan registros temporales, definidos con la directiva .def. Estos registros se emplean para almacenar datos intermedios, índices de bucles, máscaras o flags durante la ejecución, facilitando la manipulación de información de manera eficiente. Además, en esta sección, se asignan y configuran los pines de los microcontroladores para interactuar con los periféricos correspondientes

Se puede apreciar en este fragmento del código B - matriz led:

.def fila = r20

```
.def columnas = r21
.def indice = r22
.def flags = r23
.def deb_der = r24
.def deb_izq = r25
```

*Uso de constantes:*

Los códigos implementados utilizan constantes necesarias, definidas con .equ, que facilitan la configuración de parámetros como el ancho de mensajes, velocidades de scroll, baudios de UART o direcciones de memoria. Ejemplo extraído del código D - Plotter:

```
.equ BAJA_SOLENOIDE = 2
.equ SUBIR_SOLENOIDE = 3
.equ MOV_ABAJO = 4
.equ MOV_ARRIBA = 5
.equ MOV_DERECHA = 6
.equ MOV_IZQUIERDA = 7
```

*Uso de bucles y control de flujo:*

Se implementan bucles loop y controles condicionales (cpi, brne, breq) para iterar sobre datos o estados, permitiendo recorrer LUTs, barrido de leds o movimiento secuencial de motores.

*Interrupciones y temporizadores:*

Los cuatro programas pueden depender de la temporización para tareas periódicas. En el caso del DAC, se utiliza un timer para actualizar la salida de manera constante; en la matriz LED, la temporización permite el refresco del multiplexado; y en el plotter y la cinta, se controla la activación del solenoide y los movimientos de los motores. Aunque la implementación específica varía en cada caso, todos comparten la lógica de usar registros de temporización y flags para ejecutar acciones de manera precisa y ordenada. A continuación se muestra las interrupciones (por polling) y temporizadores aplicadas en el código A - cinta transportadora:

```
; Espera de 1 segundo usando Timer0
```

*espera\_1s:*

```
ldi r17,0
e1s:
    sbis TIFR0,TOV0
    rjmp e1s
    ldi r16,(1<<TOV0)
    out TIFR0,r16
    inc r17
    cpi r17,61
    brne e1s
    ret
```

*; Espera múltiple*

*espera\_multi:*

```
rcall espera_1s
dec CONTADOR
brne espera_multi
ret
```

*Manipulación de datos binarios:*

Todos los códigos realizan operaciones a nivel de bit para interactuar directamente con el hardware. Se emplean instrucciones como sbi, cbi, ldi, out, and y or para manejar pines individuales o grupos de bits según las necesidades de cada periférico. Esta manipulación binaria es esencial para

controlar salidas digitales, actualizar registros de la LUT, encender o apagar LEDs y mover motores de forma controlada. Un ejemplo aplicable en este caso, es el siguiente fragmento del código C - Conversor DAC:

```
main_loop:
    wait_flag:
        sbis TIFR0, OCF0A
        ; limpiar bandera
        ldi tmp, (1<<OCF0A)
        out TIFR0, tmp
        ; siguiente muestra desde la LUT
        lpm sample, Z+      ; leer byte desde memoria de programa
        out PORTD, sample   ; escribir byte completo en PORTD (R-2R)
```

En este fragmento se manipulan datos binarios a nivel de registro para controlar el hardware: se utiliza lpm sample, Z+ para leer un byte desde la LUT almacenada en la memoria de programa y luego out PORTD, sample para enviar ese valor al puerto D, de manera que cada bit del registro controla un pin individual del microcontrolador, permitiendo manejar un DAC R-2R.

*Estructura modular:*

Cada programa sigue una estructura modular clara que facilita la lectura y depuración. Esta estructura incluye la configuración inicial del programa (.cseg, .org), la inicialización de los periféricos, el bucle principal donde se ejecuta la lógica central y, en caso de ser necesario, subrutinas o rutinas de interrupción. Este enfoque organizado permite separar las distintas funciones y simplifica tanto la implementación como el mantenimiento del código.

*Stack pointer:*

Todos los programas hacen uso del Stack Pointer para gestionar el almacenamiento temporal de direcciones de retorno y datos durante la ejecución de subrutinas o interrupciones. Esto permite que el microcontrolador mantenga un control seguro sobre el flujo del programa, asegurando que las llamadas a subrutinas, los retornos y las interrupciones no corrompan los registros temporales ni los datos del sistema. Fragmento extraído del código B - Matriz led, a modo de ejemplo:

```
ldi r16, high(RAMEND)
out SPH, r16
ldi r16, low(RAMEND)
out SPL, r16
```

*UART:*

Los códigos que requieren comunicación serial utilizan la UART para transmitir y recibir información con otros dispositivos o con la computadora. Esto incluye la configuración de velocidad de transmisión (baud rate), activación de transmisor y receptor, y la lectura/escritura de datos mediante registros de la UART. La comunicación serial permite enviar comandos, datos de control o información de estado, y es común a los programas de DAC, matriz LED y plotter cuando se requiere monitoreo o control externo.

A continuación se muestra como se implementó en el código B - matriz led, teniendo en cuenta que se usó esta estructura para todos los demás ítems.

*; Inicialización UART*

```

ldi temp, high(UBRR_VALUE)
sts UBRR0H, temp
ldi temp, low(UBRR_VALUE)
sts UBRR0L, temp
ldi temp, (1<<RXEN0)|(1<<TXEN0)
sts UCSR0B, temp
ldi temp, (1<<UCSZ01)|(1<<UCSZ00)
sts UCSR0C, temp

```

Donde: UBRR0H / UBRR0L configura la velocidad de transmisión (baud rate), UCSR0B habilita el receptor (RXEN0) y el transmisor (TXEN0) y UCSR0C configura la comunicación a 8 bits (UCSZ01=1 y UCSZ00=1).

; Recepción y chequeo de información

UART\_CheckReceive:

```

lds temp,UCSR0A
sbrs temp,RXC0
ret
lds temp,UDR0
cpi temp,'1'
breq set1
cpi temp,'2'
breq set2
cpi temp,'3'
breq set3
ret
set1:
ldi opcion_actual,1
ret
set2:
ldi opcion_actual,2
ret
set3:
ldi opcion_actual,3
ret

```

En esta parte se verifica si hay un dato recibido (mediante RXC0): si no hay retorna inmediatamente, si hay un carácter recibido (“1”, “2” o “3”) actualiza la variable (opcion\_actual) que controla la matriz.

; Transmisión de un carácter

UART\_Transmit:

```

lds temp,UCSR0A
sbrs temp,UDRE0
rjmp UART_Transmit
sts UDR0,r18
ret

```

Donde se espera hasta que el buffer de transmisión esté vacío (UDRE0) y se envía el byte en r18 al registro UDR0 para transmitirlo por la UART.

; Envío de un string

UART\_SendString:

```

lpm r18,Z+
cpi r18,0
breq fin_str
rcall UART_Transmit
rjmp UART_SendString
fin_str:

```

ret

En este caso, se lee un string desde la memoria del programa (FLASH) usando el puntero Z y se transmite cada carácter hasta encontrar un 0 que indique el final del string.

; Transmisión de un carácter

MostrarMenu:

```

ldi ZH,high(WelcomeMsg<<1)
ldi ZL,low(WelcomeMsg<<1)
rcall UART_SendString
ldi ZH,high(MenuMsg<<1)
ldi ZL,low(MenuMsg<<1)
rcall UART_SendString
ldi ZH,high(MenuOpciones<<1)
ldi ZL,low(MenuOpciones<<1)
rcall UART_SendString
ret

```

En este fragmento, se cargan los punteros Z con los mensajes y se llama a UART\_sendString para enviar cada mensaje por la UART, mostrando el menú inicial al usuario.

## VII. CONCLUSIONES

Durante la realización de este laboratorio se abordó el diseño, programación y control de distintos sistemas mecatrónicos utilizando el microcontrolador ATmega328P, con el objetivo de integrar de manera efectiva el hardware y software en la operación de dispositivos prácticos. Se trabajó en la implementación de cuatro bloques principales: la cinta transportadora con punzonadora, la matriz de LEDs, el conversor DACR-2R con tabla de búsqueda y el plotter, cada uno de los cuales permitió reforzar competencias en control de sistemas, programación en lenguaje ensamblador y verificación experimental de resultados.

En el caso de la cinta transportadora con punzonadora se logró cumplir con los objetivos de gestión de modos de operación, detección de objetos mediante sensores y coordinación con la cinta transportadora. Para esta implementación, se decidió usar un puente H (Driver L298) en lugar de la F5 board prevista para el control de los motores, dado que esta última no ofrece los suficientes pines de conexión para manejar simultáneamente los LEDs, pulsadores y sensores de la cinta. Se identificó una limitación en el comportamiento final del mecanismo: al activarse el sensor, la punzonadora no mantiene la posición baja por el tiempo esperado, sino que realiza un par de ciclos de bajar y subir repetidamente. A pesar de esta desviación respecto al comportamiento esperado, el resto de las funciones operan correctamente, permitiendo validar la programación desarrollada y reforzando la comprensión de la integración entre hardware y software en un sistema mecatrónico.

La implementación de la matriz de LEDs permitió desplegar mensajes desplazables de manera efectiva, así como interactuar con el usuario a través del monitor serial. El sistema respondió correctamente a los comandos enviados, mostrando los patrones programados de forma clara y sin errores de visualización. Esta práctica facilitó la comprensión del manejo de periféricos de salida, la programación secuencial y la comunicación mediante UART. La operación estable de la matriz de LEDs evidencia que los objetivos planteados para este bloque se cumplieron satisfactoriamente, consolidando la

familiarización con este tipo de dispositivos y su control mediante microcontrolador.

Durante la implementación del conversor DAC R-2R con tabla de búsqueda, se logró generar la señal analógica a partir de datos digitales propuestos de manera efectiva, obteniendo una onda cuadrada con un duty cycle de 50%. La señal fue verificada mediante el osciloscopio y mostró un comportamiento consistente con lo esperado. Esta práctica permite comprender y aplicar los principios de conversión digital-analógica, así como el uso de tablas de búsqueda en lenguaje ensamblador para controlar con precisión la salida del DAC.

Aunque la resolución del DAC de 8 bits y el ruido presente en el sistema experimental impusieron ciertas limitaciones, la señal generada permitió validar la estrategia de generación de señales y reforzó el aprendizaje sobre la integración de componentes digitales y analógicos.

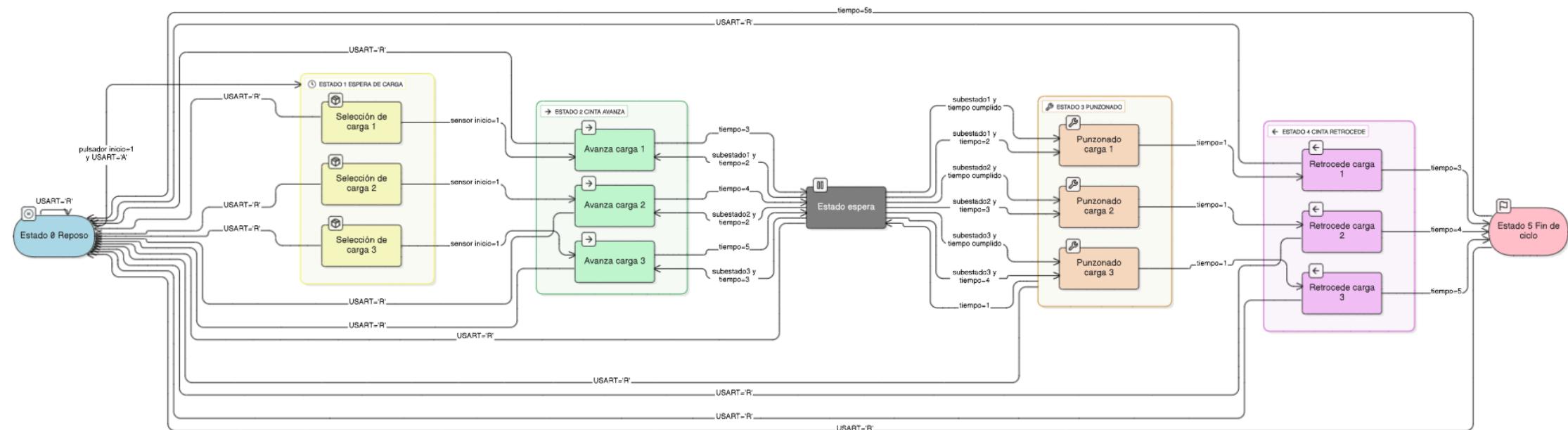
En cuanto a la implementación del plotter, se utilizó correctamente el monitor serial para la interacción y se comprendieron en detalle las formas de movimiento y la combinación de las mismas para generar figuras. La práctica permitió familiarizarse con el plotter como herramienta fundamental para el trazado automático, consolidando conocimientos sobre el control secuencial de actuadores y la coordinación de movimientos.

De las figuras propuestas, se lograron dibujar exitosamente dos de las tres planteadas: el triángulo y la cruz, sin presentar ningún inconveniente durante su ejecución. Sin embargo, la generación del círculo resultó demasiado compleja; a pesar de los varios intentos realizados, no se logró encontrar la lógica adecuada para su implementación, limitación atribuible parcialmente a nuestra experiencia como programadoras. No obstante, no afectó el aprendizaje adquirido. En general, los demás objetivos propuestos para el bloque del plotter se cumplieron satisfactoriamente, permitiendo validar los conceptos de control de movimientos.

En resumen, el laboratorio permitió consolidar el aprendizaje sobre diseño y control de sistemas mecatrónicos mediante microcontroladores, fortaleciendo la capacidad de integración de componentes físicos y su programación, generar señales analógicas apartir de datos digitales, controlar actuadores y periféricos, y validar experimentalmente los resultados obtenidos. A través de la experiencia adquirida, se comprendieron las limitaciones técnicas y las consideraciones prácticas que influyen en el desempeño de los sistemas, así como la importancia de la planificación, verificación y documentación rigurosa en la implementación de soluciones.

## VIII. APÉNDICE

#### Apéndice I. Diagrama de máquina de estados Parte A: Cinta transportadora con Punzonadora:



Apéndice II.Código para diagrama de máquina de estados  
Parte A: Cinta transportadora con Punzonadora:

```

title Diagrama de flujo: Máquina de estados
para control de cinta y punzonadora
direction right

Estado 0 Reposo [shape: oval, color:
lightblue, icon: pause-circle]
Estado 1 Espera de carga [color: yellow,
icon: clock] {
    Selección de carga 1 [icon: package,
color: yellow]
    Selección de carga 2 [icon: package,
color: yellow]
    Selección de carga 3 [icon: package,
color: yellow]
}
Estado 2 Cinta avanza [color: green, icon:
arrow-right] {
    Avanza carga 1 [icon: arrow-right, color:
green]
    Avanza carga 2 [icon: arrow-right, color:
green]
    Avanza carga 3 [icon: arrow-right, color:
green]
}
Estado espera [color: gray, icon: pause]
Estado 3 Punzonado [color: orange, icon:
tool] {
    Punzonado carga 1 [icon: tool, color:
orange]
    Punzonado carga 2 [icon: tool, color:
orange]
    Punzonado carga 3 [icon: tool, color:
orange]
}
Estado 4 Cinta retrocede [color: purple,
icon: arrow-left] {
    Retrocede carga 1 [icon: arrow-left,
color: purple]
    Retrocede carga 2 [icon: arrow-left,
color: purple]
    Retrocede carga 3 [icon: arrow-left,
color: purple]
}
Estado 5 Fin de ciclo [shape: oval, color:
pink, icon: flag]

// Relaciones y transiciones
// Inicio
Estado 0 Reposo > Estado 1 Espera de carga:

```

```

pulsador inicio=1 y USART='A'

// Selección de subestado en Estado 1
// Transición a Estado 2 según sensor inicio
Selección de carga 1 > Avanza carga 1:
sensor inicio=1
Selección de carga 2 > Avanza carga 2:
sensor inicio=1
Selección de carga 3 > Avanza carga 3:
sensor inicio=1

// Estado 2: Avance de cinta y transición a
espera o siguiente estado
Avanza carga 1 > Estado espera: tiempo=3
Avanza carga 2 > Estado espera: tiempo=4
Avanza carga 3 > Estado espera: tiempo=5

// Estado espera: regreso a Estado 2 o
avance a Estado 3 según temporización
// Estado espera > Estado 3 Punzonado (según
temporización)
Estado espera > Punzonado carga 1:
subestado1 y tiempo cumplido
Estado espera > Punzonado carga 2:
subestado2 y tiempo cumplido
Estado espera > Punzonado carga 3:
subestado3 y tiempo cumplido

// Estado 3: Punzonado y transición a espera
o siguiente estado
// Estado espera > Punzonado carga X (loop
si corresponde)
Estado espera > Punzonado carga 1:
subestado1 y tiempo=2
Estado espera > Punzonado carga 2:
subestado2 y tiempo=3
Estado espera > Punzonado carga 3:
subestado3 y tiempo=4

// Estado 3 > Estado 4 Cinta retrocede
Punzonado carga 1 > Retrocede carga 1:
tiempo=1
Punzonado carga 2 > Retrocede carga 2:
tiempo=1
Punzonado carga 3 > Retrocede carga 3:
tiempo=1

// Estado 4: Retroceso y transición a Estado
5
Retrocede carga 1 > Estado 5 Fin de ciclo:
tiempo=3

```

```
Retrocede carga 2 > Estado 5 Fin de ciclo:  
tiempo=4  
Retrocede carga 3 > Estado 5 Fin de ciclo:  
tiempo=5  
  
// Estado 5: Fin de ciclo y reinicio  
// Transición global de reset desde  
cualquier estado  
Estado 0 Reposo > Estado 0 Reposo: USART='R'  
Avanza carga 1 < Estado espera: subestado1 y  
tiempo=2  
Avanza carga 2 < Estado espera: subestado2 y  
tiempo=2  
Avanza carga 3 < Estado espera: subestado3 y  
tiempo=3  
Estado 0 Reposo < Estado 5 Fin de ciclo:  
tiempo=5s  
Estado 0 Reposo < Estado 1 Espera de carga:  
USART='R'  
Estado 0 Reposo < Estado 1 Espera de carga:  
USART='R'  
Estado 0 Reposo < Selección de carga 2:  
USART='R'  
Estado 0 Reposo < Selección de carga 3:  
USART='R'  
Estado 0 Reposo < Avanza carga 1: USART='R'  
Estado 0 Reposo < Avanza carga 2: USART='R'  
Estado 0 Reposo < Avanza carga 3: USART='R'  
Estado 0 Reposo < Estado espera: USART='R'  
Estado 0 Reposo < Retrocede carga 1:  
USART='R'  
Estado 0 Reposo < Retrocede carga 2:  
USART='R'  
Estado 0 Reposo < Retrocede carga 3:  
USART='R'  
Estado 0 Reposo < Estado 5 Fin de ciclo:  
USART='R'  
Estado espera < Estado 3 Punzonado: tiempo=1  
Estado 0 Reposo < Estado 3 Punzonado:  
USART='R'
```

## IX. EVIDENCIAS

S. Modernell, V. Etcheverry, y G. Sosa, [https://github.com/Victoriaetech/Microcontroladores/tree/main/Laboratorios/Laboratorio%201]. [Accedido: Septiembre, 2025]

## REFERENCIAS

[1] Arduino, “Arduino Uno Rev3 – Reliable ATmega328P Board with Digital & Analog I/O,” *Arduino Store*. [Online]. Available: <https://store-usa.arduino.cc/products/arduino-uno-rev3>. [Accessed: Sep. 26, 2025].

[2] Wikipedia, “Atmega328,” *Wikipedia*, 20-Mar-2024. [Online]. Available: <https://es.wikipedia.org/w/index.php?title=Atmega328&oldid=158934266>. [Accessed: Sep. 20, 2025].

[3] Wikipedia, “Lenguaje ensamblador,” *Wikipedia, la enciclopedia libre. Enlace*. [Accedido: 26-sep-2025].

[4] Cadence PCB Solutions, “USART vs. UART: What's the Difference?”, *Cadence PCB Design & Analysis*, 8 de febrero de 2023. [Enlace](#). [Accedido: 26-sep-2025].

[5] Visual Paradigm, “What is State Machine Diagram?”, *Visual Paradigm*. [Online]. Available: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/>. [Accessed: Sep. 26, 2025].

[6] Fischertechnik, “Smart Beginner Set,” Fischertechnik. Available: <https://www.fischertechnik.biz/fischertechnik-bt-smart-beginner-set>. [Accessed: 26-sep-2025].

[7] Fischertechnik, “163330\_bt\_smart beginner,” Fischertechnik. Available: [https://ev1.utec.edu.uy/moodle/pluginfile.php/1281615/mod\\_resource/content/0/163330\\_bt\\_smart\\_beginner.pdf](https://ev1.utec.edu.uy/moodle/pluginfile.php/1281615/mod_resource/content/0/163330_bt_smart_beginner.pdf). [Accessed: 26-sep-2025].

[8] Electronics Tutorials, “R-2R DAC (R-2R Digital-to-Analogue Converter),” *Electronics Tutorials*, [Online]. Available: <https://www.electronics-tutorials.ws/combination/r-2r-dac.html>. [Accessed: 26-Sep-2025].

[9] Talos Electronics. (s.f.). Puente H Doble L298N. Recuperado el 21 de junio de 2025, de <https://www.taloselectronics.com/blogs/tutoriales/puente-h-doble-l298n>

[10] Uelectronics, “Cómo conectar y programar la Matriz LED 8×8 1088AS,” *Uelectronics Blog*, 20-Mar-2024 (aproximado). [Online]. Available: <https://blog.uelectronics.com/tarjetas-desarrollo/arduino/como-programar-y-a-hacer-la-conexion-matriz-led8x8/>. [Accessed: Sep. 20, 2025].