Cuarta Práctica de Laboratorio

*Tecnologías de Microprocesamiento

1st Luis Bouvier
Universidad Tecnológica del Uruguay
Fray Bentos, Río Negro, Uruguay
luis.bouvier@estudiantes.utec.edu.uy

2nd Marcos Casanova
Universidad Tecnológica del Uruguay
Fray Bentos, Río Negro, Uruguay
marcos.casanova@estudiantes.utec.edu.uy

3rd Santiago Moizo
Universidad Tecnológica del Uruguay
Fray Bentos, Río Negro, Uruguay
santiago.moizo@estudiantes.utec.edu.uy

***Abstract*—**
***Index Terms*—**

I. INTRODUCCIÓN

El Laboratorio N.º 4 de la unidad curricular Tecnologías de Microprocesamiento tiene como finalidad profundizar en el uso de sistemas embebidos basados en el microcontrolador ATmega328P, aplicando protocolos de comunicación digital, sensores y módulos externos en distintos entornos prácticos. La actividad propone aplicar los protocolos SPI, I²C y UART, así como integrar sensores como el DHT11 y el MPU6050, pantallas LCD y matrices RGB WS2812, con el propósito de procesar información del entorno y generar respuestas visuales o de control en tiempo real.

Asimismo, el laboratorio busca desarrollar arquitecturas maestro–esclavo entre microcontroladores, generar animaciones programadas, y controlar una matriz LED mediante inclinación, permitiendo fortalecer la programación modular y el uso coordinado de periféricos simultáneos. Finalmente, se propone aplicar estos conocimientos en un entorno más complejo, integrando comunicación, control de actuadores y respuesta inmediata en la actividad orientada al mini–fútbol robótico.

Objetivo General:

Integrar y aplicar los periféricos avanzados del microcontrolador ATmega328P incluyendo UART, SPI, I²C, ADC y PWM para desarrollar sistemas embebidos capaces de sensar, procesar y comunicar información en tiempo real, implementando soluciones distribuidas y aplicaciones interactivas basadas en módulos externos y dispositivos digitales.

Objetivos Específicos

- Implementar la comunicación Maestro-Esclavo mediante los protocolos SPI e I²C, coordinando el envío y recepción de datos entre dos microcontroladores.
- Leer, procesar y transmitir información proveniente de sensores externos, como el DHT11 y el MPU6050, aplicando conversión ADC y comunicación digital según corresponda.
- Desarrollar animaciones y sistemas gráficos en matrices RGB WS2812, Controlando los tiempos de actualización y el flujo de datos sin bloquear la ejecución del programa.
- Controlar dispositivos en función del sensado, utilizando técnicas de procesamiento filtrado básico y mapeo de datos para generar respuestas visuales o de movimiento.
- Integrar múltiples periféricos dentro de un mismo proyecto embebido, aplicando programación modular y garantizando la correcta sincronización entre lectura, procesamiento y salida.
- Aplicar los conocimientos adquiridos para diseñar un sistema interactivo final, combinando comunicación, sensado y control en un entorno dinámico.

II. MARCO TEÓRICO

ATmega328P

El ATmega328P es un microcontrolador AVR de 8 bits ampliamente utilizado en sistemas embebidos debido a su arquitectura tipo Harvard, eficiencia energética y gran disponibilidad de periféricos internos. Opera típicamente a 16 MHz, dispone de 32 KB de Flash, 2 KB de SRAM y 1 KB de EEPROM, además de 23 pines GPIO, tres temporizadores, un conversor ADC de 10 bits, y módulos de comunicación UART, SPI e I²C.

Timers, ADC, UART, SPI, I2C:

Los temporizadores permiten medir intervalos de tiempo, generar interrupciones periódicas y producir señales PWM. El ATmega328P incluye 3 tipos de Timers:

- **Timer0: 8 bits, usado comúnmente para PWM rápido**
- **Timer1: 16 bits, usado idealmente para control preciso de servos y motores.**
- **Timer2: 8 bits, utilizado para modulación y tareas temporizadas adicionales**

Los modos más utilizados son CTC (Clear Timer on Compare match) y Fast PWM.

ADC (Analog Conversor Digital) El conversor Analógico-digital del ATmega328P (ADC) posee 10 bits de resolución, permitiendo mapear señales entre 0.5V a valores entre 0 y 1023. Se usa para leer sensores como potenciómetros, LDRs, joysticks, y sensores analógicos en general. El ADC admite varias referencias de voltaje y prescalers para ajustar el tiempo de conversión.

El ADC del ATmega328P dispone de 8 canales analógicos (ADC0–ADC7) multiplexados mediante el registro ADMUX, el cual define tanto el canal de entrada como la referencia de voltaje (AVcc, AREF o referencia interna). El proceso de

conversión se controla a través del registro ADCSRA, donde se activan el módulo ADC, el inicio de conversión (ADSC) y el prescaler, que determina la frecuencia de muestreo. El prescaler es fundamental para garantizar un tiempo de conversión adecuado, normalmente ajustado a una frecuencia interna entre 50 kHz y 200 kHz para obtener mediciones precisas.

La conversión se inicia al configurar el bit ADSC, y concluye cuando este vuelve a cero. El resultado queda disponible en los registros ADCL y ADCH, combinados para formar el valor digital de 10 bits. La ecuación general de conversión es:

Vin=ADC x Vref1023

Donde Vref es típicamente de 5 V cuando se utiliza AVcc como referencia.

El ADC es ampliamente utilizado en sistemas embebidos para adquirir señales continuas y transformarlas en información procesable. En este laboratorio su uso fue esencial en tareas como la lectura del LM35 en el control de temperatura, la detección de posición mediante potenciómetros en el control de motor, la medición de inclinación del joystick y otros sensores dependientes de tensión. Su integración con PWM y comunicación serial permite implementar sistemas de control en lazo cerrado que responden dinámicamente a cambios en el entorno físico.

UART (Universal Asynchronous Receiver-Transmitter)

La UART (Universal Asynchronous Receiver-Transmitter) es un periférico de comunicación serial asíncrona, utilizado comúnmente para el intercambio de datos punto a punto entre un microcontrolador y un dispositivo externo, como una computadora, un módulo Bluetooth o una terminal serial. Su funcionamiento no requiere un reloj compartido, ya que ambos dispositivos deben estar configurados a la misma velocidad de transmisión (baud rate).

La comunicación se basa en la transmisión secuencial de bits, estructurados en tramas que incluyen: un bit de inicio (start bit), 8 bits de datos, un bit opcional de paridad y uno o más bits de parada (stop bits). En el ATmega328P, la UART se configura mediante los registros UBRR0H/L (baud rate), UCSR0A/B/C (control y formato de trama). El modo más utilizado es 9600 bps, 8N1, que corresponde a 8 bits de datos, sin paridad y 1 bit de parada.

Las funciones UART permiten enviar caracteres individuales, cadenas completas y recibir comandos del usuario, siendo esenciales para depuración, monitoreo o control remoto de sistemas embebidos.

Serial Asíncrono y Baud Rate

La comunicación serial asíncrona es un método de transmisión de datos en el que los dispositivos intercambian información sin compartir un reloj común. En lugar de utilizar una señal de sincronización, cada trama enviada incluye un bit de inicio, una cantidad fija de bits de datos, un bit opcional de paridad y uno o más bits de parada, permitiendo que el receptor identifique el comienzo y fin de cada byte.

En el ATmega328P, la UART opera bajo este esquema, donde ambos dispositivos deben estar configurados a la misma velocidad de transmisión, conocida como baud rate. El baud rate expresa la cantidad de símbolos por segundo enviados a través de la línea serial; valores comunes son 9600, 19200 o 115200 baudios.

La precisión del baud rate es esencial: una diferencia excesiva entre emisor y receptor puede causar errores de lectura o pérdida de datos. Para garantizar esta sincronización, el microcontrolador calcula el registro UBRR en función de su frecuencia de reloj y el baud rate deseado, ajustando internamente el temporizado del módulo UART.

La elección de la velocidad depende del equilibrio entre confiabilidad y velocidad: baud rates bajos ofrecen mayor estabilidad ante ruido eléctrico, mientras que velocidades altas permiten transmitir grandes cantidades de datos a costa de mayor sensibilidad a errores.

Envío y Recepción de Comandos

El módulo UART del ATmega328P permite tanto transmitir como recibir datos de manera independiente, lo que facilita la interacción con terminales seriales, módulos inalámbricos o sistemas embebidos conectados. El envío de información se realiza escribiendo un byte en el registro UDR0, el cual es transmitido cuando el bit UDRE0 indica que el buffer está vacío. La transmisión de cadenas completas se implementa mediante funciones que recorren una secuencia de caracteres y los envían uno por uno.

La recepción se realiza a través del bit RXC0, que indica cuándo un nuevo byte ha llegado al buffer de recepción. Al leer el registro UDR0, el sistema obtiene el dato recibido y lo interpreta como un carácter, un comando o parte de un paquete estructurado. Esta técnica permite implementar interfaces interactivas basadas en texto, donde el usuario puede enviar instrucciones como 'P', '1', 'S' o comandos completos para controlar modos de operación, solicitar datos o activar funciones específicas del sistema.

En sistemas embebidos más avanzados, la recepción puede manejarse de forma no bloqueante, verificando periódicamente el estado del buffer, o mediante interrupciones UART, permitiendo capturar comandos en tiempo real sin detener el funcionamiento general del programa. Esta capacidad es fundamental en aplicaciones donde se controla un motor, un sensor o un actuador mientras se mantienen abiertos canales de comunicación, como ocurrió en varios problemas del laboratorio.

SPI

SPI es un protocolo de comunicación serial síncrona y de alta velocidad, diseñado para conectar un microcontrolador con uno o varios periféricos (memorias, sensores, displays, etc.). A diferencia de UART, en SPI existe un reloj generado por el maestro, lo que permite una transmisión estable y rápida.

El bus SPI está compuesto por cuatro líneas principales:

- **MOSI (Master Out Slave In): Línea por la cual el maestro envía datos hacia el esclavo. Cuando el ATmega328P actúa como maestro, transmite a través de MOSI; cuando actúa como esclavo, recibe por esta línea.**
- **MISO (Master In Slave Out): Línea por la cual el esclavo envía datos hacia el maestro.En operación,**

mientras el maestro transmite en MOSI, puede recibir simultáneamente información por MISO, permitiendo comunicación full-duplex.

- **SCK (Serial Clock): Señal de reloj generada por el maestro. Su función es sincronizar el envío de cada bit, permitiendo que todos los dispositivos conectados en el bus operen al mismo ritmo.**
- **SS (Slave Select): Línea utilizada por el maestro para seleccionar cuál esclavo desea activar.**

Nivel bajo (0) → El esclavo está habilitado.

Nivel alto (1) → El esclavo queda inactivo.

En sistemas con múltiples periféricos SPI, cada esclavo requiere su propia línea SS o un decodificador externo.

El modo de operación depende del rol del microcontrolador:

En el protocolo SPI, la diferencia fundamental entre maestro y esclavo radica en el control de la comunicación.

El maestro es quien inicia y coordina toda transferencia: genera la señal de reloj en la línea SCK, define la velocidad de transmisión y selecciona a cada esclavo mediante la línea SS, que activa en nivel bajo para habilitarlo. Además, se encarga de enviar los datos por MOSI y de leer simultáneamente la respuesta del esclavo en MISO.

Por otro lado, el esclavo adopta un rol pasivo y dependiente del maestro. No genera reloj ni decide cuándo comienza la comunicación; simplemente responde cuando su línea SS es activada. El esclavo coloca los datos en MISO siguiendo el ritmo impuesto por el maestro y lee la información recibida en MOSI según las transiciones del reloj.

En conjunto, el maestro actúa como controlador central del bus, mientras que los esclavos operan únicamente cuando son seleccionados, garantizando un sistema determinista y sincronizado bajo un único origen de reloj.

Además el SPI permite transferencias full-duplex, lo que significa que el envío y la recepción ocurren simultáneamente. Su simplicidad y velocidad lo vuelven ideal para dispositivos como sensores rápidos, pantallas gráficas o memorias externas.

I²C (Inter-Integrated Circuit)

I²C es un protocolo serial síncrono diseñado para la comunicación de múltiples dispositivos utilizando únicamente dos líneas:

- **SDA (Serial Data):** Envio y recepcion de datos
- **SCL (Serial Clock): Reloj generado por el maestro el cual sincroniza cada bit transmitido**

Además, ambas líneas funcionan en modo open-drain, por lo que requiere resistencias pull-up que mantienen el bus en nivel alto cuando ningún dispositivo está transmitiendo.

Líneas SDA/SCL y Direcciones

El protocolo I²C funciona utilizando solo dos líneas compartidas: SDA, por donde se transmiten los datos, y SCL, que suministra la señal de reloj que sincroniza toda la comunicación. Ambos conductores operan en modo open-drain, por lo que requieren resistencias pull-up externas para mantener el nivel alto cuando ningún dispositivo transmite.

Cada periférico conectado al bus posee una dirección única, lo que permite que múltiples esclavos convivan en el mismo par de líneas. El maestro inicia toda transferencia enviando la condición START seguida de la dirección del dispositivo objetivo, junto con un bit que indica si la operación será de lectura o escritura.

Comunicación Maestro–Esclavo

En I²C, el maestro controla completamente el flujo de comunicación: genera la señal de reloj SCL, define el inicio (START) y el fin de la transmisión (STOP), y determina el sentido de la operación.

El esclavo, en cambio, permanece pasivo hasta que su dirección es enviada por el maestro, momento en el que responde mediante un bit de reconocimiento (ACK). Durante la transferencia, el maestro emite los pulsos de reloj mientras intercambia datos con el esclavo por la línea SDA, y la comunicación finaliza únicamente cuando el maestro genera la condición STOP.

Gracias a esta estructura, I²C permite conectar múltiples dispositivos —como el MPU6050, LCD con módulo I²C o memorias EEPROM— de forma sencilla y con un uso mínimo de pines.

LCD

Las pantallas LCD alfanuméricas, típicamente de 16×2 o 20×4 caracteres, se utilizan ampliamente en sistemas embebidos para mostrar mensajes, estados del sistema y datos provenientes de sensores. Funcionan mediante la manipulación de cristales líquidos que modifican la luz polarizada cuando se aplica una tensión eléctrica, permitiendo generar caracteres predefinidos en una matriz interna.

Para interactuar con el microcontrolador, el LCD dispone de líneas de datos, líneas de control (RS, RW, E) y un potenciómetro externo para ajustar el contraste.

El pin RS selecciona si se envía un comando o un dato, RW define el sentido de lectura/escritura (generalmente se mantiene en escritura), y E actúa como señal de habilitación, indicando al módulo cuándo capturar la información.

El LCD trabaja internamente con un conjunto de instrucciones estándar, como limpiar pantalla, mover cursor, escribir caracteres y configurar el modo de operación. Estas instrucciones se ejecutan mediante secuencias temporizadas, por lo que se emplean retardos o verificaciones del "busy flag" para asegurar que el controlador procese cada comando correctamente.

Controlador HD44780

Los LCD alfanuméricos más comunes están basados en el controlador HD44780, un circuito integrado que gestiona toda la lógica interna de escritura y actualización del display.

Este controlador incluye:

- **Memoria DDRAM: Almacena los caracteres que se mostrarán en pantalla**
- CGROM: Donde residen los caracteres ASCII estándar.
- CGRAM: Una memoria de gráficos para crear caracteres personalizados (por ejemplo)
- Un decodificador, el cual toma cada byte enviado desde el microcontrolador y lo convierte en la forma visible del carácter.

El HD44780 interpreta una seria de instrucciones como:

Clean Display (0x01): borra completamente el contenido del LCD y restablece la memoria DDRAM, eliminando todos los caracteres visibles. Tras ejecutarse, el cursor vuelve a la posición inicial (fila 1, columna 1) y el display deja de mostrar cualquier desplazamiento previo. Es una de las instrucciones más lentas del controlador, requiriendo alrededor de 1.5 ms para completarse.

Return Home (0x02): La instrucción Return Home reposiciona el cursor en la dirección base (0x00 en DDRAM) sin eliminar el contenido en pantalla. También restablece cualquier desplazamiento del display aplicado previamente. Al igual que el borrado, requiere un tiempo de ejecución elevado, cercano a 1.5 ms.

Entry Mode Set (0x04-0x07): Esta instrucción configura el comportamiento del cursor tras escribir un carácter. Permite seleccionar si el cursor se incrementa hacia la derecha o se decrementa hacia la izquierda, y si al escribir se desplaza todo el display automáticamente. Su configuración más común es incrementar el cursor sin desplazar el contenido (0x06), lo que facilita la escritura secuencial.

Display ON/OFF Control (0x08-0x0F): Este comando controla la visibilidad general del LCD. Permite encender o apagar la pantalla, activar o desactivar el cursor subrayado, y habilitar o no el parpadeo del cursor. El modo más utilizado es display encendido, cursor apagado y sin parpadeo (0x0C), lo que evita distracciones visuales durante la lectura.

Cursor or Display Shift (0x10-0x1F): La instrucción Shift desplaza el cursor o el contenido completo del display sin modificar la memoria interna del LCD. Puede mover el cursor a izquierda o derecha, o desplazar el texto completo como si fuera un ”scroll”. Es útil para animaciones o para recorrer textos largos que exceden el ancho del display.

Function Set (0x20-0x3F): Este comando define la configuración básica del LCD, como el tamaño del bus de datos (4 u 8 bits), el número de líneas (1 o 2) y el tipo de fuente (5×8 o 5×10). La configuración típica en proyectos embebidos es modo 4 bits, dos líneas y matriz de 5×8 (0x28), ya que reduce pines sin comprometer funcionalidad.

Set CGRAM Address (0x40-0x7F): Esta instrucción establece la dirección inicial de la CGRAM, memoria donde se almacenan caracteres personalizados. Tras enviarla, cualquier dato posterior se interpreta como patrones de 5×8 puntos que forman nuevos símbolos. Permite crear íconos propios, flechas, barras de progreso o gráficos simples.

Set DDRAM Address (0x80-0xFF): El comando Set DDRAM Address mueve el cursor a una posición específica del display. Cada fila del LCD tiene direcciones internas asignadas, y esta instrucción permite saltar directamente a cualquier columna y fila. Es fundamental para ubicar texto en posiciones exactas sin escribir caracteres de relleno.

Modos 4/8 bits y Funciones Básicas

El controlador HD44780 permite operar el LCD en modo 8 bits, donde se utiliza un bus completo de ocho líneas de datos (D0–D7). Este modo permite enviar cada instrucción o dato en un solo ciclo, lo que simplifica el protocolo y reduce el tiempo necesario para la transferencia. Al trabajar en 8 bits, el microcontrolador transmite directamente el byte completo hacia el LCD, por lo que el flujo de comunicación resulta más directo y menos propenso a errores de sincronización. Sin embargo, este modo requiere una cantidad mayor de pines, lo cual puede convertirse en una limitación en sistemas embebidos donde los recursos de E/S son reducidos o están asignados a múltiples periféricos.

En contraste, el modo 4 bits permite operar el LCD utilizando solo cuatro líneas de datos (D4–D7), dividiendo cada instrucción en dos nibbles (parte alta y parte baja). La transferencia consiste en enviar primero los bits altos del byte y luego los bajos, cada uno acompañado de un pulso en la línea Enable. Aunque el proceso implica más ciclos de comunicación y una secuencia de inicialización específica, reduce significativamente el consumo de pines del microcontrolador. Por esta razón, el modo 4 bits es el más utilizado en aplicaciones de robótica, sistemas embebidos educativos y proyectos donde se requiere integrar varios sensores y actuadores sin sacrificar funcionalidad.

PWM

La modulación por ancho de pulso (PWM) consiste en variar el ciclo de trabajo (duty cycle) de una señal digital periódica para controlar la cantidad de energía entregada a un dispositivo.

Ciclo de trabajo y control de velocidad/brillo.

El ciclo de trabajo expresa el porcentaje de tiempo en que la señal permanece en nivel alto durante un periodo, lo que permite regular de manera precisa variables continuas como brillo, velocidad o potencia sin necesidad de convertir digital–analógico. Un PWM del 20% entrega menos energía que uno del 80%, aun manteniendo la misma frecuencia. Esto permite implementar controles eficientes, estables y de bajo consumo energético, aprovechando los temporizadores del microcontrolador para generar señales confiables a frecuencias constantes.

Uso con motores y LEDs.

El uso de PWM se extiende ampliamente en motores y LEDs debido a su capacidad para controlar velocidad y luminosidad sin generar pérdidas significativas. En motores de corriente continua, el PWM permite ajustar la velocidad de giro modulando la energía promedio aplicada al motor, y mediante un puente H es posible complementar este control con inversión de dirección. En el caso de los LEDs, la modulación por ancho de pulso actúa como un control de brillo perceptual: ciclos de trabajo pequeños producen luz tenue, mientras que ciclos altos generan iluminación intensa, sin alterar la tensión nominal del diodo. Ambos casos se benefician de la estabilidad, precisión y eficiencia del PWM, convirtiéndolo en un componente fundamental dentro de los sistemas embebidos.

Matriz RGB + WS2812B

Los WS2812B son LEDs RGB inteligentes que integran en un mismo encapsulado el diodo tricolor y un pequeño controlador interno. Su característica más distintiva es el protocolo de un solo cable, a través del cual se envían las

órdenes necesarias para definir el color y la intensidad de cada LED.

LED inteligente y protocolo de un solo cable.

El microcontrolador transmite una secuencia de pulsos digitales con tiempos muy precisos que representan bits '0' y '1'; cada WS2812B recibe el primer dato del flujo, lo almacena en su registro interno y reenvía el resto de la cadena al siguiente LED. Esta arquitectura permite controlar grandes cantidades de LEDs usando únicamente una línea de datos, lo que reduce de forma significativa el consumo de pines y facilita el diseño de matrices RGB direccionables en sistemas embebidos.

Frames y animaciones RGB.

La representación visual de la matriz se organiza mediante frames o cuadros, que consisten en arreglos de valores RGB donde cada posición almacena el color correspondiente a un LED dentro de la grilla. Para crear animaciones, el sistema actualiza estos frames de manera secuencial, modificando colores, intensidades o desplazamientos entre actualizaciones. El microcontrolador envía continuamente los datos al arreglo WS2812B mediante rutinas de temporización precisa, logrando efectos como desplazamiento de puntos luminosos, cambios aleatorios de color, patrones dinámicos y transiciones suaves. Gracias a esta estructura basada en buffers y actualización periódica, las matrices RGB permiten generar animaciones complejas con mínima carga de hardware y total control desde software.

MPU6050

El MPU6050 es un módulo que integra en un mismo chip un acelerómetro de 3 ejes y un giroscopio de 3 ejes, permitiendo medir tanto aceleraciones lineales como velocidades angulares del dispositivo.

Acelerómetro/giroscopio

El acelerómetro registra variaciones de movimiento y también la componente de la gravedad, lo que permite inferir la orientación estática del sensor. Por su parte, el giroscopio detecta rotaciones rápidas o cambios bruscos de dirección, lo que complementa las mediciones del acelerómetro, especialmente en situaciones donde existe vibración o movimiento dinámico. La combinación de ambos sensores, denominada unidad de medición inercial (IMU), proporciona información precisa sobre el estado espacial del sistema, siendo ampliamente utilizada en robótica, estabilización, videojuegos, controladores gestuales y sistemas de navegación.

Lectura de inclinación por I2C.

El MPU6050 se comunica con el microcontrolador mediante el bus I²C, utilizando únicamente las líneas SDA y SCL para el intercambio de datos. Durante la lectura de inclinación, el microcontrolador inicia la comunicación enviando la dirección del dispositivo y consultando los registros internos donde el MPU6050 almacena las mediciones crudas del acelerómetro y del giroscopio. A partir de estas lecturas se calcula la inclinación mediante relaciones trigonométricas (por ejemplo, usando atan2 para obtener el ángulo de pitch y roll), filtrado complementario o fusionando ambos sensores para lograr una estimación estable. Gracias a este protocolo y al formato estructurado de sus registros, el MPU6050 permite obtener información de orientación en tiempo real con un consumo mínimo de pines y una excelente precisión para aplicaciones embebidas.

LDR (Light Dependent Resistor)

El LDR es un sensor cuyo principio de funcionamiento se basa en la variación de su resistencia según la cantidad de luz incidente. A mayor iluminación, menor resistencia; a menor iluminación, mayor resistencia. Para obtener su valor, se utiliza un divisor resistivo conectado al ADC del microcontrolador, lo que permite convertir la tensión resultante (0–5 V) en un valor digital de 10 bits (0–1023). De esta forma es posible medir niveles de luz, detectar objetos por reflexión o realizar mediciones ambientales básicas.

Potenciómetro

El potenciómetro funciona como un divisor de tensión variable, donde la posición mecánica de su eje determina la salida analógica entre 0 y 5 V. Su principio de operación consiste en deslizar un cursor sobre una pista resistiva, variando así la relación entre los terminales extremos y la terminal central. La lectura se realiza mediante el ADC, permitiendo obtener un valor proporcional a la posición física. Es uno de los métodos más simples para ingresar referencias analógicas, calibrar parámetros o medir posiciones angulares cuando es acoplado a un eje mecánico.

HC-SR04 (Sensor Ultrasónico de Distancia)

El HC-SR04 mide distancias utilizando ultrasonido. Su funcionamiento se basa en emitir un pulso de alta frecuencia desde el pin TRIG y luego medir, mediante el pin ECHO, el tiempo que tarda en regresar la onda reflejada por un objeto. El microcontrolador mide este tiempo usando un timer interno, y mediante la relación distancia = tiempo × velocidad del sonido / 2, obtiene la distancia en centímetros. La lectura no usa ADC, sino mediciones de tiempo precisas en entradas digitales, lo que lo hace ideal para aplicaciones de proximidad o mapeo básico

Botón / Pulsador

El botón es un sensor digital mecánico que opera mediante el contacto eléctrico entre dos terminales. Su principio es simple: cuando se presiona, cierra el circuito (nivel bajo o alto según la configuración); cuando no se presiona, lo abre. Para su lectura, el microcontrolador utiliza una entrada digital, frecuentemente acompañada de una resistencia pull-up o pull-down para evitar estados flotantes. En software se suele implementar antirrebote para evitar lecturas falsas causadas por el ruido mecánico del contacto.

DHT11 (Temperatura y Humedad)

El DHT11 es un sensor digital que mide temperatura y humedad mediante dos elementos internos: un termistor NTC para la temperatura y un sensor capacitivo cuya capacitancia varía con la humedad del aire. El módulo integra además un pequeño procesador que convierte esas señales analógicas en datos digitales listos para transmitir.

La lectura no utiliza ADC, sino un protocolo digital de un solo hilo, en el que el microcontrolador inicia la comunicación con un pulso de arranque y el sensor responde enviando los datos mediante pulsos cronometrados. La duración de

cada pulso representa un bit, e incluye un checksum para validar la transmisión. Por su simplicidad y bajo costo, el DHT11 se utiliza en sistemas básicos de monitoreo ambiental, proporcionando mediciones rápidas con un uso mínimo de pines.

Motores DC

Los motores de corriente continua (DC) permiten convertir energía eléctrica en movimiento rotacional gracias a la interacción entre un campo magnético fijo y una bobina alimentada por corriente. Su velocidad y sentido de giro dependen directamente del voltaje aplicado y de la polaridad de la alimentación, lo que los convierte en actuadores esenciales dentro de sistemas embebidos y proyectos mecatrónicos.

Control de velocidad y dirección

El control de velocidad se logra modificando la potencia entregada al motor, regulando así la rapidez con la que gira el eje. En sistemas digitales como el ATmega328P, esta regulación se realiza mediante modulación por ancho de pulso (PWM), generando una señal cuadrada cuyo ciclo de trabajo determina el promedio de energía aplicada. Un duty cycle mayor aumenta la velocidad, mientras que valores bajos permiten giros lentos y controlados.

Para controlar la dirección, se invierte la polaridad aplicada al motor, estableciendo así giro horario o antihorario. Esta inversión no puede hacerse directamente desde el microcontrolador, ya que requiere manejar corrientes mayores y evitar cortocircuitos entre ramas internas.

Uso de PWM y puente H

El control completo del motor se obtiene combinando PWM con un puente H, un circuito de cuatro transistores que permite aplicar tensión al motor en ambos sentidos. El microcontrolador genera la señal PWM en un pin específico (por ejemplo, OC1A del ATmega328P), mientras que otros pines digitales controlan las entradas del puente, definiendo si el giro será hacia adelante, hacia atrás o si el motor estará detenido.

Este esquema permite implementar funciones como aceleración progresiva, frenado suave, rampas de velocidad y cambios de sentido seguros mediante deadtime, evitando activaciones simultáneas que puedan dañar el puente. Por su simplicidad y eficiencia, esta arquitectura es estándar en robots móviles, mecanismos automáticos y controladores de posición o velocidad.

Comunicación Bluetooth

La comunicación Bluetooth permite establecer un enlace inalámbrico de corto alcance entre el microcontrolador y un dispositivo externo, como un teléfono móvil o una computadora. Este tipo de conexión se utiliza para enviar y recibir datos de forma sencilla, manteniendo un consumo energético reducido y una interfaz transparente para el usuario final.

Módulo HC-05 y UART

El módulo HC-05 funciona como un enlace Bluetooth clásico que se comunica con el microcontrolador mediante UART, actuando como un puente serial inalámbrico. Para utilizarlo, se configura el ATmega328P con un baud rate compatible con el módulo (comúnmente 9600 bps) y se conectan las líneas TX y RX de forma cruzada.

El HC-05 recibe los datos desde el microcontrolador y los retransmite por Bluetooth hacia el dispositivo emparejado, y viceversa, permitiendo establecer un canal bidireccional sin necesidad de implementar un protocolo complejo. Esto facilita tareas como monitorear variables, controlar actuadores o ajustar parámetros del sistema en tiempo real.

Control remoto por comandos

El control remoto mediante Bluetooth se realiza enviando comandos desde un dispositivo externo hacia el microcontrolador. Cada comando recibido por UART puede interpretarse como una instrucción, permitiendo activar funciones, cambiar modos de operación o ajustar valores internos del sistema.

Este esquema resulta especialmente útil en aplicaciones de robótica, domótica o control de motores, donde se requiere modificar el comportamiento del sistema sin conexión física. El microcontrolador procesa cada comando recibido y ejecuta las acciones correspondientes, manteniendo una interfaz simple y eficiente basada en texto o caracteres.

III. METODOLOGÍA

Problema A

La elaboración del sistema se basó en la implementación progresiva de una arquitectura maestro–esclavo utilizando dos microcontroladores ATmega328P conectados mediante comunicación SPI, con el objetivo de permitir que un dispositivo se encargará de la adquisición de datos y la supervisión del sistema, mientras que el segundo ejecutará las acciones solicitadas mediante el control de diversos actuadores. El diseño se estructuró en tres etapas principales: lectura y procesamiento de sensores en el microcontrolador maestro, envío de comandos a través del bus SPI y ejecución de acciones en el microcontrolador esclavo, siguiendo una metodología incremental que permitió validar cada módulo antes de su integración final.

Como punto de partida, se configuró el microcontrolador maestro para gestionar los sensores asignados: un DHT11 como sensor principal y al menos dos sensores adicionales, tales como un LDR, un potenciómetro o botones digitales. Cada uno de estos dispositivos se leyó utilizando el periférico correspondiente (ADC para señales analógicas, lectura digital para pulsadores, y una rutina temporizada para el DHT11), garantizando estabilidad y coherencia en las lecturas. Paralelamente, se implementó una pantalla LCD conectada al maestro para mostrar en tiempo real las mediciones obtenidas y las acciones enviadas al microcontrolador esclavo. Esta etapa permitió validar la adquisición y visualización de datos, asegurando que el maestro mantuviera un ciclo de captura estable y sin bloqueos.

La segunda etapa consistió en la implementación de la comunicación SPI entre ambos microcontroladores. Para ello, se configuró el maestro en modo SPI Master, habilitando las líneas MOSI, SCK y SS, y estableciendo la velocidad adecuada del reloj, así como el modo de operación (CPOL y CPHA) según el estándar requerido. El esclavo fue configurado

en modo SPI Slave, habilitando la interrupción correspondiente para reaccionar inmediatamente a cada byte recibido. Se definió un protocolo de comandos sencillo, donde el maestro enviaba códigos específicos según los eventos detectados en los sensores: activación de LEDs, encendido de motores, variación de intensidad por PWM o activación de un buzzer, entre otros. Esta etapa permitió confirmar la integridad del intercambio de datos, verificando que cada comando fuera recibido sin errores y con latencia mínima.

La tercera etapa se centró en la implementación del microcontrolador esclavo, encargado de interpretar los comandos provenientes del maestro y ejecutar las acciones correspondientes sobre los actuadores asignados. Las salidas incluyen dispositivos como motores DC, LEDs RGB, buzzer o sistemas PWM para controlar intensidad o velocidad. Para garantizar un comportamiento seguro y estable, se establecieron rutinas modulares de activación que permiten encender, apagar o modificar el estado de los actuadores según el comando recibido, evitando conflictos de hardware y asegurando la correcta transición entre estados. Además, se realizaron pruebas individuales para validar cada actuador antes de integrarlo en la máquina de estados general del esclavo.

Una vez completadas las tres etapas, se integró el sistema en una arquitectura maestro–esclavo funcional. El maestro se encargó de sensar, procesar y decidir, enviando órdenes al esclavo en tiempo real, mientras que el esclavo ejecutaba las acciones y mantenía una respuesta coherente frente a cada instrucción. Finalmente, se validó el funcionamiento tanto en simulación como en la implementación física, demostrando que la comunicación SPI permitía un control distribuido, eficiente y coordinado entre sensores, procesamiento y actuadores.

Problema B

El desarrollo del sistema se centró en construir un módulo de adquisición y visualización de datos basado en el protocolo I²C del ATmega328P, integrando sensores digitales y analógicos para conformar una plataforma de monitoreo unificada. Para ello, se adoptó un enfoque gradual que permitió verificar primero la comunicación en el bus I²C, luego la lectura y procesamiento de cada sensor involucrado y finalmente la visualización completa de la información en una pantalla LCD con interfaz I²C. Esta organización progresiva del diseño facilitó la depuración de cada subsistema antes de su integración final.

Como primera etapa, se procedió a habilitar el periférico TWI del microcontrolador para establecer la comunicación maestro–esclavo en el bus I²C. Se configuró la frecuencia de operación ajustando los registros TWBR y TWSR según las recomendaciones del fabricante y se implementaron las rutinas fundamentales: condición de inicio (START), envío de dirección, transferencia de datos y reconocimiento (ACK/NACK). Con estas funciones básicas se realizó la lectura del sensor digital principal —como el MPU6050 cuando el proyecto lo requiere— solicitando sus registros internos de acelerómetro y giroscopio y verificando la integridad de cada respuesta. Esta etapa permitió confirmar que el bus operara de manera estable y que las direcciones de los esclavos fueran correctamente reconocidas por el microcontrolador.

En la segunda etapa, se integraron otros sensores necesarios para completar la funcionalidad del sistema. Para señales analógicas, como las provenientes de un LDR o un potenciómetro, se configuró el ADC interno en modo de referencia AVcc y un prescaler adecuado para asegurar lecturas estables. En el caso de señales digitales, como botones o selectores, se emplearon resistencias internas pull-up para evitar oscilaciones por ruido. Todas las medidas fueron procesadas mediante funciones que permiten transformar los valores obtenidos en variables interpretables —como luz ambiental en porcentaje, lectura de ángulo, nivel de humedad o cualquier otra magnitud requerida por el diseño— asegurando una adquisición homogénea de datos provenientes de múltiples fuentes.

La tercera etapa consistió en la implementación del sistema de visualización utilizando una pantalla LCD controlada mediante un expansor I²C basado en el PCF8574. Se programó la secuencia de inicialización del controlador HD44780 en modo de 4 bits, enviando los nibbles altos y bajos a través del bus I²C con los tiempos adecuados de habilitación. Sobre esta base se construyeron funciones de alto nivel para limpiar la pantalla, posicionar el cursor y escribir mensajes, permitiendo mostrar simultáneamente valores de sensores, estados del sistema o indicadores de diagnóstico. Esta etapa confirmó la estabilidad del sistema bajo carga, asegurando que la comunicación I²C pudiera coexistir con la lectura del ADC sin generar interferencias.

Finalmente, se integraron todas las etapas en un bucle principal donde el microcontrolador realiza una lectura periódica de los sensores, procesa los datos obtenidos y actualiza la pantalla LCD de forma continua. El sistema fue validado mediante pruebas individuales y completas, verificando que la comunicación I²C se mantuviera estable aun en presencia de múltiples dispositivos en el bus y que la visualización mostrara información clara, coherente y actualizada en tiempo real.

Problema C

La elaboración del sistema se centró en el diseño de un motor de animaciones para una matriz RGB 16×16 basada en LEDs WS2812B, controlada por el microcontrolador ATmega328P en lenguaje C. El objetivo principal consistió en generar patrones gráficos predefinidos —como un corazón y una pokebola— y permitir su selección dinámica mediante comandos enviados por UART, manteniendo al mismo tiempo un control seguro del brillo y del consumo de corriente de la matriz. Para alcanzar esto, se siguió una metodología incremental que permitió validar primero la base de tiempo y el driver de LEDs, y luego incorporar la lógica de animación y la interfaz de usuario por comunicación serial.

Como punto de partida, se implementó una base de tiempo en milisegundos utilizando el Timer0 del ATmega328P. Se configuró el temporizador en modo normal con un prescaler de 64, habilitando la interrupción por overflow (TIMSK0) para incrementar en cada desbordamiento la variable global millis_count. Sobre esta base se definió la función millis(), la

cual devuelve un contador de milisegundos protegido mediante deshabilitación temporal de interrupciones (cli()/sei()) para garantizar lecturas consistentes. Esta rutina se utilizó más adelante como referencia temporal para espaciar los fotogramas de las animaciones sin recurrir a retardos bloqueantes.

En paralelo, se integró el driver de la matriz WS2812B y se diseñó un mecanismo de control global de brillo. Se declaró una variable brillo con un valor reducido (por ejemplo, 40 sobre 255) para limitar la corriente demandada por los 256 LEDs y se definió la macro ESCALA(x) para escalar las componentes de color antes de enviarlas al LED. La función setRGB_scaled() encapsula esta operación y llama internamente a ws2812_setRGB(), permitiendo modificar el brillo general sin reescribir los datos de cada patrón. Como prueba inicial del hardware y del pipeline de datos, se programó la rutina test_colores_auto(), que recorre toda la matriz con una secuencia de colores básicos (rojo, verde, azul y blanco), manteniendo cada estado durante un intervalo definido por millis(). Esta etapa permitió verificar el correcto funcionamiento de la cadena WS2812B, la visibilidad de los colores y el efecto real del factor de brillo.

Una vez validado el encendido básico de la matriz, se procedió a la organización de los fotogramas de las animaciones. Para optimizar el uso de memoria RAM, se almacenaron los patrones de corazón y pokebola en arreglos constantes ubicados en memoria de programa (Flash) mediante la palabra clave PROGMEM. Cada imagen se representó como un arreglo de 256 elementos ([256][3]), donde cada entrada contiene las componentes RGB de un píxel. Se agruparon estos fotogramas en arreglos de punteros (const uint8_t (*corazon[6])[3], const uint8_t (*pokebola[5])[3]) para facilitar la selección del frame activo sin copiar datos a RAM. Esta estrategia permitió manejar múltiples fotogramas de alta resolución en una matriz relativamente grande sin saturar la SRAM del microcontrolador.

Dado que el cableado físico de la matriz WS2812B sigue un esquema en "zigzag", se implementó la función zigzag_index() para mapear coordenadas lógicas (fila, columna) a índices lineales coherentes con el orden real de los LEDs. La rutina mostrar_frame_zigzag() recorre cada fila y columna de la imagen de origen, calcula el índice de destino mediante zigzag_index(), lee los valores de color desde Flash usando pgm_read_byte() y llama a setRGB_scaled() para actualizar el LED correspondiente. Al finalizar el barrido de todos los píxeles, la función invoca ws2812_show() para enviar el frame completo a la matriz. De este modo, se separa el concepto de coordenadas lógicas del detalle de cableado físico, simplificando la definición de patrones gráficos.

Sobre esta infraestructura se construyó el motor de animación propiamente dicho. Se definieron variables globales para llevar el seguimiento del fotograma actual (fotograma), la dirección de avance (direccion) y el instante del último cambio de frame (ultimo_fotograma). En el lazo principal, se verifica periódicamente si ha transcurrido más de 100 ms desde el último fotograma utilizando la función millis(). Cuando se cumple esta condición, se selecciona el conjunto de imágenes según el valor de animacion_actual (corazón o pokebola), se llama a mostrar_frame_zigzag() con el fotograma correspondiente y se actualiza fotograma sumando o restando según la variable direccion. Al alcanzar el último frame, la dirección se invierte para generar un efecto de vaivén (ida y vuelta), evitando saltos bruscos entre el primer y el último fotograma. Esta lógica permite obtener animaciones suaves y cíclicas sin bloquear el programa con retardos fijos.

Finalmente, se incorporó la interfaz de control mediante UART para seleccionar las animaciones en tiempo real. En la función main(), se inicializa la UART mediante UART_INICIAR(103) (configurando así una velocidad estándar, por ejemplo 9600 baudios) y se envía un menú de ayuda al terminal serial describiendo los comandos disponibles: '0' para ejecutar el test RGB automático, '1' para activar la animación de corazón y '2' para la animación de pokebola. Dentro del bucle principal, se consulta de forma no bloqueante si hay datos disponibles con UART_DISPONIBLE(); en caso afirmativo, se lee el carácter mediante UART_LEER() y se actualiza la variable animacion_actual, reiniciando fotograma y direccion cuando corresponde. El comando '0' ejecuta la rutina test_colores_auto(), mientras que los comandos '1' y '2' seleccionan la secuencia de fotogramas deseada e informan el cambio por UART con UART_IMPRIMIR().

Con esta metodología, el Problema C integra de forma coherente el manejo avanzado de una matriz WS2812B, el uso de memoria de programa para almacenar gráficos, la gestión de una base de tiempo por interrupciones y la interacción con el usuario mediante comunicación serial, dando lugar a un sistema de animaciones RGB flexible, modular y fácilmente extensible a nuevos patrones.

Problema D

Para resolver el Problema D se procedió a implementar un sistema capaz de controlar el desplazamiento de un LED dentro de una matriz WS2812B de 16×16 en función de las lecturas del acelerómetro MPU6050, integrando además la generación de colores aleatorios y la detección de una entrada por pulsador. El desarrollo se llevó a cabo siguiendo las siguientes etapas:

Configurar el hardware de comunicación y la matriz LED, comenzando por la inicialización del bus TWI (I^2C) y la puesta en marcha del sensor MPU6050 mediante la escritura del registro de energía correspondiente. Paralelamente, se inicializó la línea de datos de la matriz WS2812B estableciendo el pin de salida y realizando el pulso de arranque requerido por este tipo de dispositivos.

Implementar el protocolo de temporización para los LEDs WS2812B, diseñando rutinas específicas para enviar bits y bytes respetando los tiempos estrictos definidos por el estándar. Para ello se generaron retardos con instrucciones NOP, permitiendo utilizar el microcontrolador AVR sin periféricos adicionales. A partir de este mecanismo se construyó una función para enviar la trama completa de los 256 LEDs y otra para asignar valores RGB a posiciones individuales dentro del arreglo.

Definir el mapeo físico en zig-zag de la matriz, mediante una función que traduce coordenadas cartesianas (x, y) a índices lineales según la disposición serpenteada de los LEDs. Esta conversión resultó fundamental para asegurar que el LED activo se visualice en la posición correcta independientemente del sentido de cableado interno.

Asignar el color inicial del LED mediante una rutina de generación aleatoria, la cual calculó valores pseudoaleatorios para los tres canales RGB. Se utilizó un valor fijo como semilla para asegurar repetitibilidad durante las pruebas.

Leer continuamente las aceleraciones del MPU6050, utilizando funciones de lectura de 16 bits que permiten obtener los valores de los ejes X e Y. Dichas lecturas se compararon con umbrales definidos para determinar la dirección del movimiento dentro de la matriz. Si la aceleración superaba el valor establecido, la posición del LED se desplazaba una unidad en el eje correspondiente, limitando el movimiento a los bordes de la matriz.

Incorporar la lectura del pulsador como mecanismo de cambio de color, supervisando el estado del pin de entrada asociado. Al detectarse una pulsación sostenida, se generaba un nuevo color aleatorio y se aplicaba un breve proceso de anti-rebote temporal para evitar detecciones múltiples.

Actualizar la matriz en cada iteración del bucle principal, limpiando primero todos los LEDs, posicionando el LED activo con su color actual y enviando nuevamente toda la trama a la matriz WS2812B. Finalmente, se incorporó un retardo fijo para estabilizar el ritmo de actualización del sistema.

Mediante esta metodología fue posible implementar un sistema interactivo que responde en tiempo real a la inclinación física del módulo y a la acción del pulsador, actualizando dinámicamente la posición y el color del LED sobre la matriz RGB.

Problema E Para resolver el Problema E se procedió a desarrollar un sistema de control remoto por Bluetooth capaz de gestionar múltiples actuadores —motores DC, servos, buzzer pasivo y una luz LED— a través del módulo HC-05, integrando además el procesamiento de comandos estructurados y la operación simultánea de periféricos basados en interrupciones y temporizadores. En primera instancia, se configuró la interfaz UART en modo no bloqueante, estableciendo la velocidad de comunicación en 9600 baudios y creando un buffer circular para almacenar caracteres recibidos hasta la detección de un salto de línea, momento en el cual la cadena completa se enviaba a una función encargada de interpretar el comando. Seguidamente, se implementó el módulo encargado del control de los motores DC, para lo cual se inicializó el Timer1 en modo Fast-PWM de 8 bits y se configuraron los pines del puente H para permitir el control de dirección y velocidad en ambos motores mediante valores positivos, negativos o nulos enviados desde la aplicación remota.

A continuación, se desarrolló la lógica de accionamiento del buzzer pasivo utilizando el Timer2 en modo CTC, generando una señal cuadrada periódica a través de interrupciones del comparador para producir un tono audible. Paralelamente, se preparó un canal digital independiente en PD6 destinado a encender o apagar una luz LED bajo la instrucción correspondiente. En una etapa posterior, se implementó el subsistema de servomotores, el cual requirió el uso del Timer0 para generar una base de tiempo precisa de 20 ms mediante interrupciones periódicas. Dentro del manejador de interrupciones se controlaron las ventanas de temporización de ambos servos (izquierdo y derecho), ajustando de forma dinámica el ancho de pulso en función del ángulo deseado. Este mecanismo permitió además la implementación de acciones predefinidas como la "patada" en cada servo, combinando cambios temporales de ángulo con retardos calibrados.

Una vez configurados todos los módulos de hardware, se diseñó el procesador de comandos encargado de interpretar cadenas recibidas vía Bluetooth siguiendo un formato estandarizado. Esta función incorporó casos específicos para encender y apagar la luz, activar o desactivar la bocina, ejecutar movimientos de servos y modificar los valores de velocidad de los motores. Para ello se emplearon funciones como `strstr()` y `atoi()` para localizar parámetros dentro de la cadena y convertirlos a valores numéricos válidos, aplicando además recortes a límites máximos y mínimos para evitar saturación en los actuadores. Finalmente, se integró todo el sistema en un ciclo principal que se encargó únicamente de verificar continuamente la recepción de nuevos comandos, delegando todo el trabajo crítico a las interrupciones de temporizadores y a las funciones de control previamente desarrolladas.

Mediante esta metodología fue posible implementar un sistema modular, robusto y completamente controlable de manera remota, en el que cada actuador opera en paralelo sin interferencias, gracias a la correcta gestión de interrupciones, temporización precisa y procesamiento eficiente de comandos.

IV. RESULTADOS

Problema A

En el Ejercicio 1 se implementó un sistema maestro–esclavo basado en dos microcontroladores ATmega328P interconectados mediante el bus SPI. El microcontrolador maestro se encargó de la adquisición de datos: lectura periódica del sensor de temperatura DHT11, conversión ADC de un potenciómetro de 10 k y muestreo del estado de un pulsador digital. En paralelo, las mediciones se mostraron en una pantalla LCD 16×2 controlada por I²C, lo que permitió supervisar en tiempo real la temperatura ambiente, el valor del potenciómetro y el estado lógico del pulsador mientras se verificaba el envío de información hacia el esclavo.

En el extremo esclavo, los datos recibidos por SPI se tradujeron en acciones concretas sobre los actuadores conectados. El primer byte, asociado al valor del potenciómetro, se utilizó como referencia de ciclo de trabajo para generar una señal PWM en el pin OC0A (PD6), controlando el brillo de un LED: al girar el potenciómetro hacia uno de sus extremos el LED permanecía prácticamente apagado, mientras que en el extremo opuesto alcanzaba su máxima luminosidad, observándose variaciones suaves y continuas en los niveles intermedios. El segundo byte, correspondiente a la condición

de temperatura, permitió encender o apagar un LED indicador en PD7 según si la lectura del DHT11 superaba el umbral configurado, evidenciando una respuesta inmediata ante cambios térmicos. Finalmente, el tercer byte controló un buzzer en PD5 que se activaba únicamente cuando el pulsador del maestro estaba presionado, replicando de forma remota el estado de la entrada digital.

Durante las pruebas se comprobó la correcta sincronización maestro–esclavo: cada actualización del ciclo de muestreo en el maestro se reflejó de forma consistente en las salidas del esclavo. El sistema mantuvo un funcionamiento estable, confirmando que la arquitectura distribuida planteada para el Ejercicio 1 permite desacoplar de forma efectiva el sensado y la interfaz de usuario (maestro) del control de los actuadores (esclavo), aprovechando el protocolo SPI para transmitir de manera eficiente la información necesaria para el control del LED PWM, el LED de temperatura y el buzzer.

Problema B

En el Problema B se mantuvo la misma funcionalidad general del sistema que en el Ejercicio 1, pero reemplazando el enlace SPI por una comunicación I²C entre los dos microcontroladores ATmega328P. El nodo maestro continuó siendo el encargado del sensado y la interfaz de usuario: lectura periódica del sensor de temperatura DHT11, conversión ADC del potenciómetro de 10 k y detección del estado del pulsador digital, además de la actualización de la pantalla LCD 16×2 a través del bus I²C. Sobre ese mismo bus se incorporó el microcontrolador esclavo, configurado con una dirección propia, lo que permitió que el maestro enviara en una sola transacción los tres datos de interés (valor del potenciómetro escalado para PWM, valor de temperatura y valor del pulsador) reutilizando las líneas SDA y SCL ya empleadas por el LCD.

Del lado del esclavo, la recepción de los tres bytes por I²C se tradujo nuevamente en acciones directas sobre los actuadores: el primer byte se utilizó para ajustar el ciclo de trabajo de una salida PWM sobre el pin OC0A (PD6), controlando el brillo de un LED de forma proporcional a la posición del potenciómetro; el segundo byte determinó el encendido o apagado de un LED en PD7 en función de si la temperatura medida por el DHT11 superaba el umbral fijado; y el tercer byte activó un buzzer en PD5 únicamente cuando el pulsador en el maestro se encontraba presionado. En las pruebas realizadas, la respuesta del esclavo fue inmediata y coherente con los valores mostrados en el LCD del maestro, sin comportamientos erráticos ni falsas activaciones.

En comparación con la solución basada en SPI, la implementación con I²C resultó más sencilla desde el punto de vista del cableado y de la ampliación del sistema, ya que el esclavo compartió el mismo bus que el módulo LCD y sólo fue necesario definir correctamente su dirección de esclavo y el formato de los mensajes. La reducción en la cantidad de líneas de interconexión y la buena estabilidad observada durante las pruebas confirmaron que, para este tipo de aplicación con velocidades de actualización bajas y varios dispositivos periféricos, el uso de I²C representa una alternativa simple y eficaz para distribuir funciones entre dos microcontroladores, manteniendo el mismo comportamiento funcional alcanzado previamente con SPI.

Problema C

Como se mostró tanto en la simulación como en las pruebas físicas realizadas con la matriz WS2812B de 16×16, el código desarrollado para la reproducción de animaciones logró su cometido de manera satisfactoria. Se consiguió implementar correctamente el manejo de fotogramas almacenados en memoria Flash, así como la visualización fluida de las secuencias correspondientes al corazón y a la pokebola, permitiendo alternar entre ellas mediante la comunicación UART sin presentar fallas críticas durante su funcionamiento.

Siguiendo lo descrito en la metodología, el primer paso consistió en configurar la comunicación UART y preparar el sistema para recibir instrucciones que seleccionaran la animación a mostrar. Durante las primeras pruebas se detectó que algunos fotogramas no estaban siendo interpretados correctamente, lo que generaba pequeñas distorsiones en ciertos bordes de las imágenes. Este problema se debió principalmente al tamaño y organización de las matrices, ya que la cantidad de datos almacenados incrementaba el riesgo de errores en los índices o en la lectura desde PROGMEM. Tras varias revisiones se ajustó la estructura de los arreglos y se corrigieron pequeños desplazamientos, logrando que todos los LED recibieran los valores RGB adecuados.

En la segunda etapa se verificó la correcta implementación del mapeo en zigzag. En las primeras ejecuciones se observó que algunos fotogramas parecían invertidos horizontalmente o presentaban líneas fuera de lugar. Esto ocurrió porque el cálculo del índice físico no coincidía exactamente con el orden de cableado real de la matriz. Una vez identificada la falla, se ajustó la función encargada de generar los índices para cada LED, lo que permitió que las animaciones se alinearan correctamente con la topología física de la matriz.

Posteriormente, se evaluó la eficiencia del sistema de temporización encargado de generar el efecto de animación tipo "ping-pong". La lógica utilizada para avanzar y retroceder entre los fotogramas funcionó correctamente, y el intervalo de 100 ms elegido permitió obtener una animación visualmente fluida sin parpadeos ni interrupciones. Los cambios de dirección entre el último y primer fotograma se realizaron de forma estable, garantizando una transición continua y agradable a la vista.

En cuanto a la comunicación UART, inicialmente se detectó que algunos mensajes no coincidían con la animación seleccionada, particularmente cuando se enviaban comandos repetidos en poco tiempo. Esto se debió a que el sistema no reiniciaba correctamente los índices y la dirección del fotograma al cambiar de animación. Una vez corregido este detalle, las tres opciones disponibles (test RGB, animación de corazón y animación de pokebola) respondieron inmediatamente y sin errores, logrando así un control confiable del sistema a través de la terminal.

Al integrar todas las funciones en un único programa se realizaron pruebas extensivas, verificando que los módulos de lectura de fotogramas, temporización, mapeo zigzag y

comunicación UART pudieran coexistir sin interferencias. Tras realizar ajustes finales en el orden de ejecución y llamadas a funciones, el código quedó funcionando de manera estable, mostrando las animaciones seleccionadas con precisión y sin pérdida de información.

Como resultado final, se obtuvo un sistema completamente funcional capaz de reproducir de forma clara y estable animaciones RGB complejas sobre una matriz de 256 LEDs, respondiendo de manera inmediata a los comandos enviados por UART y manteniendo un rendimiento fluido incluso con grandes cantidades de datos almacenados en memoria. El proyecto cumplió con los objetivos planteados y demostró un correcto manejo de estructuras, temporización, visualización y comunicación serial en un entorno de microcontroladores AVR.

Problema D

Como se evidenció durante las pruebas realizadas tanto en simulación como en el prototipo físico, el código desarrollado para el control del LED desplazable en la matriz WS2812B mediante el sensor MPU6050 funcionó correctamente y logró cumplir su objetivo. Se consiguió implementar un sistema interactivo que responde de forma fluida a los movimientos del módulo y que permite modificar el color del LED mediante un pulsador, integrando adecuadamente lectura sensorial, control espacial, animación y generación aleatoria de colores.

Siguiendo lo trabajado en la metodología, la primera etapa consistió en la configuración del bus TWI para establecer comunicación con el MPU6050. Durante las pruebas iniciales se presentó un problema: en ocasiones los valores retornados por el acelerómetro parecían congelarse o no correspondían con la inclinación real del módulo. Este inconveniente se debía a retrasos en la espera del bit TWINT y a la falta de un margen mayor de estabilización después de la escritura del registro de energía. Tras ajustar los tiempos y verificar la secuencia de lecturas, el sensor comenzó a responder correctamente, brindando valores estables para ambos ejes.

Posteriormente se trabajó con la matriz WS2812B de 16×16. Uno de los primeros obstáculos fue la temporización estricta del protocolo, ya que pequeñas variaciones en la duración de las señales provocaban que parte de la matriz mostrara colores incorrectos o que algunos LEDs quedaran apagados. Luego de ajustar las rutinas de envío de bits y bytes utilizando instrucciones NOP precisas, el sistema logró enviar la trama completa sin errores perceptibles, estabilizando la representación visual del LED en movimiento. A continuación, se verificó el funcionamiento de la conversión de coordenadas (x, y) al índice lineal siguiendo el mapeo zigzag propio del cableado. En las primeras pruebas, el LED parecía "saltar" entre posiciones incorrectas, especialmente al cambiar de filas pares a impares. Esto se debió a un cálculo invertido en las columnas durante el zigzagpero después de corregir este detalle el LED se desplazó de forma fiel a la inclinación física del módulo.

En cuanto al movimiento controlado por acelerómetro, los resultados fueron positivos. Los umbrales establecidos permitieron evitar movimientos bruscos o erráticos, consiguiendo que el LED avanzara una posición por vez solo cuando la inclinación era suficientemente marcada. Esto permitió un control más fino y predecible del desplazamiento dentro de la matriz. También se comprobó el correcto funcionamiento del sistema de límites, impidiendo que el LED saliera del rango de 0 a 15 tanto en el eje X como en Y, lo que aportó estabilidad general al comportamiento del sistema.

En la prueba del pulsador, se detectó inicialmente un comportamiento inconsistente: el LED cambiaba de color múltiples veces con una sola pulsación. Este problema se debía a la ausencia de un mecanismo de anti-rebote. Tras implementar un tiempo de espera y verificación del estado estable del pin, el cambio de color comenzó a producirse únicamente cuando el usuario realmente pulsaba el botón, dando lugar a una interacción confiable y sin falsos disparos.

Por último, al integrar todas las funciones —lectura del MPU6050, mapeo zigzag, actualización de la matriz y pulsador— el sistema funcionó de manera estable. El LED se movió correctamente en respuesta a la inclinación física, el color se mantuvo consistente y solo cambió cuando se presionó el botón, y la matriz respondió adecuadamente a las actualizaciones de cada ciclo. El retardo final permitió suavizar la actualización visual, evitando cambios excesivamente rápidos sin comprometer la capacidad de respuesta.

Como resultado final, se obtuvo un sistema completamente funcional y robusto, capaz de interpretar las lecturas del acelerómetro en tiempo real, desplazar un LED dentro de una matriz WS2812B respetando su cableado físico, y permitir la interacción del usuario mediante un pulsador para el cambio de color. El proyecto cumplió plenamente con los objetivos planteados y demostró un correcto dominio del control sensorial, temporización, comunicación I²C y manejo de matrices RGB en un entorno de microcontroladores AVR.

Problema E Como se pudo comprobar durante las pruebas de funcionamiento tanto en simulación como en ejecución física, el código desarrollado para el Problema E cumplió con éxito los objetivos planteados, logrando un sistema completamente controlable vía Bluetooth capaz de gestionar motores, servos, buzzer pasivo y una luz LED sin presentar fallas críticas. El sistema respondió adecuadamente a los comandos enviados desde la aplicación remota, mostrando un comportamiento estable y coordinado entre todos los periféricos, a pesar de la complejidad asociada al manejo simultáneo de múltiples temporizadores, interrupciones y módulos independientes.

Siguiendo lo implementado en la metodología, la primera etapa consistió en la validación de la comunicación UART en modo no bloqueante. Durante las pruebas iniciales se detectó que algunas cadenas llegaban incompletas o con saltos de línea irregulares, lo cual ocasionaba que ciertos comandos no se interpretaran correctamente. Este problema se solucionó ajustando el manejo del buffer y asegurando que los caracteres de control (`\n` / `\r`) fueran tratados correctamente antes de llamar a la función de procesamiento. Una vez corregido este detalle, la comunicación se estabilizó y el sistema comenzó a reconocer los comandos de forma inmediata y sin pérdidas.

En la etapa correspondiente al control de motores, los resultados fueron satisfactorios. El puente H respondió adecuadamente tanto para giros hacia adelante como hacia atrás, y la modulación por PWM mediante el Timer1 permitió regular la velocidad de manera progresiva. Un inconveniente menor surgió cuando ciertos valores fuera de rango producían saturación o respuestas inesperadas; sin embargo, tras aplicar límites y normalización de valores el problema quedó resuelto, permitiendo un control preciso y seguro del movimiento.

Con respecto al buzzer pasivo, se verificó que el Timer2 generara correctamente la señal cuadrada a través de interrupciones. Un problema inicial fue que la bocina permanecía activa aun después de enviar el comando de apagado debido a que la interrupción no se deshabilitaba correctamente. Tras revisar la configuración del comparador del Timer2 se corrigió este comportamiento, logrando que la bocina reaccionara inmediatamente a los comandos "H:1" y "H:0" sin ningún tipo de residuo sonoro.

En las pruebas de los servomotores se observaron resultados positivos. Los pulsos generados mediante Timer0 lograron mover ambos servos con estabilidad y sin vibraciones notorias. La función de "patada" también funcionó adecuadamente, aunque inicialmente presentó un pequeño retraso por un conflicto entre los retardos y las interrupciones activas. Ajustando los retardos y asegurando que el sistema marcara correctamente el estado de ocupación del servo ("servos_busy"), la animación de patada pasó a ejecutarse con fluidez y sin interferencias con otros comandos del sistema.

La luz LED en PD6 también respondió de forma correcta a los comandos "LED:1" y "LED:0", sin retrasos perceptibles y sin afectar el funcionamiento de los demás periféricos. Esta etapa fue sencilla, pero permitió confirmar que los comandos remotos podían controlar adecuadamente tanto periféricos simples como complejos.

Finalmente, se integraron todos los módulos en un único sistema, y se realizaron pruebas enviando secuencias rápidas y combinadas de comandos (motores + servos + buzzer + LED) con el fin de detectar posibles conflictos. El sistema se comportó de manera estable, manteniendo la ejecución de las interrupciones sin interferencias y reaccionando a cada comando en tiempo real. La arquitectura modular desarrollada permitió que cada componente funcionara de forma independiente, garantizando fluidez incluso bajo carga elevada de comandos.

Como resultado final, se obtuvo un sistema completamente funcional, robusto y de respuesta inmediata, capaz de controlar múltiples actuadores mediante comunicación Bluetooth sin degradación del desempeño. El proyecto demostró un uso adecuado del microcontrolador, del manejo de interrupciones y de técnicas de control distribuido, cumpliendo plenamente con los requisitos planteados para el Problema E.

V. CONCLUSIÓN

El desarrollo y ejecución de los proyectos correspondientes al Laboratorio 4 permitió integrar de manera completa los conocimientos adquiridos sobre sistemas embebidos, consolidando un entendimiento más profundo del microcontrolador ATmega328P y de su interacción con periféricos avanzados mediante los protocolos UART, SPI e I²C.

Implementar cada uno de los sistemas propuestos permitió aplicar técnicas fundamentales de comunicación serial, sensado, procesamiento digital y control de actuadores, demostrando la versatilidad del microcontrolador frente a tareas de distinta complejidad. Resolver los distintos problemas del laboratorio permitió abordar desafíos específicos en la lectura de sensores, la actualización de pantallas LCD, el manejo de matrices RGB, el control de motores mediante PWM y el intercambio de datos con módulos externos como el MPU6050 o el HC-05, requiriendo aplicar estrategias modulares de programación y validación progresiva de cada componente.

En conjunto, completar este laboratorio permitió afianzar la arquitectura interna del ATmega328P y las metodologías de programación en lenguaje C orientadas a sistemas embebidos, integrando sensores, actuadores y canales de comunicación para construir soluciones robustas, funcionales y coherentes con los objetivos planteados.

A pesar de que el laboratorio se desarrolló con éxito y los problemas fueron resueltos de manera competente, es necesario reconocer que surgieron algunas dificultades durante la implementación de ciertos sistemas. En particular, en el problema relacionado con la matriz RGB, la matriz entregada presentaba un tamaño mayor al previsto, lo que generó inconvenientes en el suministro de voltaje y provocó un nivel de brillo considerablemente reducido. Proceder a realizar múltiples pruebas para determinar el origen de la falla permitió identificar que el inconveniente no se debía al código ni al control de los LEDs, sino al consumo de corriente requerido por la matriz. Finalmente, solucionar el problema incrementando adecuadamente la capacidad de alimentación permitió restaurar la iluminación correcta y garantizar el funcionamiento esperado del sistema.

También, al trabajar con el sensor de temperatura mediante comunicación SPI, surgen dificultades en la obtención de las lecturas, las cuales se presentaban inestables, erráticas o fijadas en cero debido a una configuración incorrecta del modo SPI, en particular los parámetros CPOL y CPHA. Reconocer que cada dispositivo SPI requiere un modo específico de operación permitió comprender que, al seleccionar un modo distinto al indicado en la hoja de datos, el microcontrolador procede a muestrear los datos en el flanco incorrecto del reloj, lo que provoca que el sensor envíe información corrupta o que directamente no responda. Este tipo de desajuste suele generar valores de temperatura incoherentes o completamente fijos, conduciendo inicialmente a sospechar de un posible daño en el sensor, cuando en realidad el problema radica únicamente en la configuración inapropiada del modo de comunicación. Una vez ajustados correctamente los parámetros CPOL y CPHA según las especificaciones del fabricante, lograr restablecer la lectura estable y verificar el funcionamiento normal del módulo SPI.

References

VI. BIBLIOGRAFÍA

[1] Microchip Technology Inc., *ATmega328P – 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*, Chandler, AZ, USA: Microchip Technology Inc., 2020. [Online]. Available: https://www.microchip.com/en-us/product/ATmega328P

[2] Arduino, *Arduino UNO Rev3 Technical Specifications*, Ivrea, Italia: Arduino AG, 2024. [Online]. Available: https://docs.arduino.cc/hardware/uno-rev3

[3] Adafruit Industries, *WS2812 Intelligent Control LED Datasheet*, New York, NY, USA: Adafruit Industries, 2019. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/WS2812.pdf

[4] TowerPro, *MG996R Metal Gear Servo Motor Specifications*, Taipei, Taiwán: TowerPro Co. Ltd., 2018. [Online]. Available: https://www.towerpro.com.tw/

[5] SparkFun Electronics, *LDR (Photoresistor) Sensor Hookup Guide*, Boulder, CO, USA: SparkFun Electronics, 2022. [Online]. Available: https://learn.sparkfun.com/tutorials/photocell-hookup-guide

[6] Microchip Technology Inc., *Microchip Studio for AVR® and SAM Devices – Integrated Development Environment*, Chandler, AZ, USA: Microchip Technology Inc., 2023. [Online]. Available: https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

[7] M. Pont, *Embedded C*, 2nd ed., Boston, MA, USA: Addison-Wesley, 2002.

[8] M. Banzi and M. Shiloh, *Getting Started with Arduino*, 3rd ed., Sebastopol, CA, USA: Maker Media, 2022.

[9] Datasheet de los componentes HC-05, HC-SR04, MPU-6000, HD44780U.

VII. APÉNDICE

Apéndice A – Repositorio GitHub

Con el fin de garantizar la trazabilidad y disponibilidad de los códigos desarrollados en este laboratorio, se utilizó un repositorio en GitHub como plataforma de control de versiones y colaboración. En él se documentan los programas en lenguaje C implementados para cada problemática, así como las evidencias complementarias (fotos, videos y diagramas) y las simulaciones correspondientes.

El repositorio está disponible en el siguiente enlace:

REPOSITORIO

Apéndice B – Hoja de Cálculo de Animaciones RGB

Para el diseño estructurado de las animaciones utilizadas en la matriz LED RGB del Problema C, se desarrolló una planilla de cálculo destinada a la representación visual y numérica de cada cuadro de la animación. En dicha planilla, cada celda corresponde a un LED individual y contiene su valor RGB en formato decimal (R, G, B), permitiendo diseñar, editar y verificar los patrones de color antes de su implementación en el microcontrolador.

La planilla incluye:

- Hoja 1: Diseño visual por celdas de las animaciones(frame a frame).
- Hoja 2: Tabla de valores RGB exportados para su conversión a arreglos en lenguaje C.

Dado que el volumen de datos excede el formato óptimo para incluir directamente en el informe, se proporciona acceso al archivo completo mediante el siguiente enlace:

HOJA DE CÁLCULO