

# Laboratorio 4

1<sup>st</sup> Luis David Nuñez Noguera  
*Universidad Tecnológica del Uruguay ITRSO*  
Fray Bentos, Uruguay  
luis.nunez@estudiantes.utec.edu.uy

2<sup>nd</sup> Tiago Corrales Martinez  
*Universidad Tecnológica del Uruguay ITRSO*  
Fray Bentos, Uruguay  
tiago.corrales@estudiantes.utec.edu.uy

**Index Terms**—SPI, I2C, Maestro, Esclavo, Gif, Giroscópio

**Abstract**—Este laboratorio aborda la implementación práctica de sistemas embebidos distribuidos utilizando el microcontrolador ATmega328P, aplicando distintos protocolos de comunicación, técnicas de control y manejo de sensores y actuadores. El trabajo se divide en cinco etapas experimentales. En la primera y segunda parte se desarrollaron arquitecturas maestro-esclavo empleando SPI e I2C, configurando el maestro para la lectura de sensores como el DHT11, potenciómetros y botones, y el esclavo para la activación de actuadores tales como motores, LEDs y buzzer, verificando la integridad de datos y la estabilidad del bus mediante pruebas sucesivas. En la tercera etapa se programó una matriz RGB WS2812 para reproducir animaciones generadas mediante tablas de colores y controladas vía UART, implementando rutinas de temporización estricta requeridas por el protocolo. En la cuarta parte se integró el sensor inercial MPU6050 para desplazar un LED dentro de la matriz RGB según la inclinación medida, incorporando zonas muertas, lógica de movimiento discreto y cambio de color mediante un botón físico. Finalmente, en la quinta etapa se construyó y teleoperó un robot de mini fútbol controlado a través de Bluetooth, implementando señales PWM para un servo y un esquema de control direccional para motores DC. Los resultados obtenidos demuestran el funcionamiento eficiente de cada subsistema y la correcta interacción entre hardware y software, evidenciando la aplicabilidad de los protocolos y técnicas utilizadas en sistemas robóticos y de instrumentación modernos.

## I. INTRODUCCIÓN

El presente informe corresponde al Laboratorio 4 de la Unidad Curricular Tecnologías de Microprocesamiento, perteneciente al cuarto semestre de la carrera de Ingeniería Mecatrónica. El propósito de este trabajo es diseñar, implementar y analizar diversos sistemas embebidos basados en el microcontrolador ATmega328P, integrando protocolos de comunicación digital, sensores inerciales, actuadores gráficos y módulos de control remoto orientados al desarrollo de plataformas distribuidas y robótica móvil.

El laboratorio se estructura en cinco partes principales, cada una enfocada en una aplicación específica: comunicación maestro-esclavo mediante SPI, comunicación maestro-esclavo mediante I2C, generación de animaciones en matrices LED RGB controladas por WS2812, control de movimiento basado en un giroscopio MPU6050 y la implementación de un robot de mini fútbol teleoperado mediante Bluetooth. Estas experiencias permiten aplicar de forma práctica conceptos de electrónica digital, manejo de buses de comunicación, programación en lenguaje C y control en tiempo real, fortaleciendo

el entendimiento de la interacción entre hardware y software en sistemas embebidos.

Asimismo, el enfoque del informe busca no solo validar el funcionamiento físico de los montajes desarrollados, sino también documentar de manera clara y estructurada el proceso experimental, siguiendo las pautas del formato IEEE. De esta forma, se promueve la capacidad de comunicar resultados técnicos de forma precisa, profesional y alineada con prácticas estándar de la ingeniería.

## II. OBJETIVOS

### OBJETIVO GENERAL

Diseñar e implementar arquitecturas de sistemas embebidos distribuidos y plataformas robóticas móviles basadas en el microcontrolador ATmega328P, integrando protocolos de comunicación estándar (SPI, I<sup>2</sup>C, UART), gestión de sensores inerciales y actuadores gráficos, con el fin de consolidar competencias en la sincronización de datos, optimización de recursos de memoria y control en tiempo real aplicados a la Ingeniería Mecatrónica.

### OBJETIVOS ESPECÍFICOS

- Comparar experimentalmente el desempeño y la complejidad de implementación entre arquitecturas maestro-esclavo basadas en protocolos SPI e I<sup>2</sup>C, analizando la integridad de datos y la eficiencia en la transmisión de comandos para sistemas de instrumentación distribuida.
- Desarrollar algoritmos de software no bloqueantes para la gestión de matrices LED RGB y el procesamiento de señales de sensores inerciales (MPU6050), logrando una interacción fluida entre la física del movimiento y la respuesta visual del sistema
- Integrar subsistemas electrónicos de potencia, sensado y control lógico en el diseño de un robot móvil competitivo, validando estrategias de navegación y autonomía energética bajo condiciones dinámicas de operación

## III. MATERIALES

En el desarrollo de las diferentes etapas del Laboratorio 4 se emplearon múltiples componentes electrónicos, dispositivos de comunicación y elementos de robótica móvil. A continuación, se detallan los materiales utilizados en cada una de las partes experimentales.

#### A. Parte A – Comunicación SPI (Maestro-Esclavo)

- **Dos Microcontroladores ATmega328P:** Configurados en arquitectura maestro-esclavo para el procesamiento distribuido de señales y control.
- **Sensor de temperatura y humedad DHT11:** Dispositivo de entrada principal conectado al microcontrolador maestro.
- **Sensores adicionales:** Potenciómetro y botón para la adquisición de datos analógicos y digitales.
- **Pantalla LCD:** Interfaz visual conectada al maestro para monitorear las lecturas y las acciones enviadas.
- **Actuadores varios:** Motores, LED y buzzer controlados por el microcontrolador esclavo.
- **Conexión SPI:** Cableado específico para las líneas MOSI, MISO, SCK y SS entre ambos microcontroladores.

#### B. Parte B – Comunicación I2C (Maestro-Esclavo)

- **Dos Microcontroladores ATmega328P:** Configurados para operar bajo el protocolo de comunicación I2C.
- **Sensores adicionales:** Potenciómetro y botón para la adquisición de datos analógicos y digitales.
- **Pantalla LCD:** Interfaz visual conectada al maestro para monitorear las lecturas y las acciones enviadas.
- **Actuadores varios:** Motores, LED y buzzer controlados por el microcontrolador esclavo.

#### C. Parte C – Animaciones en Matriz RGB

- **Microcontrolador ATmega328P:** Gestiona el almacenamiento de frames en memoria y el refresco de la visualización.
- **Matriz de LEDs RGB 8x8:** Dispositivo de visualización para reproducir las secuencias de animación diseñadas.
- **Interfaz UART:** Permite la recepción de comandos seriales desde un computador para conmutar entre animaciones.

#### D. Parte D – Control de Movimiento con Giroscopio

- **Microcontrolador ATmega328P.**
- **Módulo MPU6050:** Sensor con giroscopio y acelerómetro para detectar la inclinación física en los ejes X e Y.
- **Matriz de LEDs RGB:** Visualiza el desplazamiento del punto/objeto en tiempo real según la inclinación detectada.
- **Botón físico:** Interfaz de entrada para modificar el color del objeto visualizado.
- **Cableado y protoboard.**

#### E. Parte E – Robot de Mini Fútbol

- **Microcontrolador ATmega328P:** Cerebro principal del robot autónomo/semiautónomo.
- **Módulo de Comunicación (HC-05/RF):** Permite el control remoto del robot mediante Bluetooth.
- **Sensores de línea:** Módulos infrarrojos

- **Actuadores de tracción:** Motores DC y servo con sus respectivos drivers para el sistema de locomoción diferencial y mecanismo de chute.
- **Batería portátil:** Fuente de alimentación recargable para autonomía energética.
- **Chasis y estructura mecánica:** Diseño físico con dimensiones máximas de  $25 \times 25 \times 20$  cm y peso inferior a 1.5 kg.

### IV. MARCO TEÓRICO

Antes de detallar el procedimiento llevado a cabo, es necesario clarificar distintos conceptos importantes para la comprensión total de la práctica.

#### A. ATmega328p

Un microcontrolador puede definirse como un circuito integrado que incorpora en un solo dispositivo una unidad central de procesamiento (CPU), memoria volátil y no volátil, así como periféricos de entrada y salida programables. A diferencia de los microprocesadores de propósito general, los microcontroladores están diseñados para ejecutar tareas específicas de control en sistemas electrónicos.

Este tipo de dispositivos constituye el núcleo de los sistemas embebidos, los cuales se caracterizan por estar dedicados a una función concreta dentro de un sistema mayor. Dichos sistemas se encuentran presentes en una amplia variedad de aplicaciones cotidianas y de carácter industrial, tales como electrodomésticos, automóviles, dispositivos médicos, equipos de telecomunicaciones y sistemas de automatización. Su relevancia radica en que permiten el diseño de soluciones con bajo consumo energético, tamaño reducido, costo accesible y alta confiabilidad [1].

1) *Características básicas del ATmega328P:* El ATmega328P, fabricado por Microchip Technology (anteriormente Atmel), es uno de los microcontroladores más representativos de la arquitectura AVR de 8 bits. Es ampliamente conocido por ser el componente central de placas de desarrollo como la Arduino Uno, lo que lo ha convertido en un referente tanto en entornos educativos como en aplicaciones prototípicas y comerciales [2].

Entre sus principales características técnicas destacan:

- Arquitectura AVR RISC de 8 bits, que permite ejecutar la mayoría de instrucciones en un solo ciclo de reloj y optimizar el rendimiento mediante un pipeline sencillo.
- Memoria integrada, compuesta por 32 KB de memoria Flash programable, 2 KB de SRAM y 1 KB de EEPROM, lo que posibilita almacenar tanto el programa como datos temporales y permanentes.
- Puertos de entrada y salida digital (GPIO), con un total de 23 líneas configurables, capaces de manejar señales de entrada o salida y de activar resistencias pull-up internas.
- Módulos temporizadores y contadores, que incluyen dos de 8 bits y uno de 16 bits, con soporte para funciones de modulación por ancho de pulso (PWM).
- Interfaces de comunicación serial, como USART (síncrona y asíncrona), SPI (Serial Peripheral Interface)

y TWI (I<sup>2</sup>C), lo que facilita la conexión con otros microcontroladores o periféricos externos.

- Conversor analógico-digital (ADC) de 10 bits con hasta seis canales, que posibilita la digitalización de señales analógicas.
- Capacidad de operación hasta los 20 MHz y disponibilidad de múltiples modos de bajo consumo, lo cual lo hace adecuado para aplicaciones portátiles y autónomas.

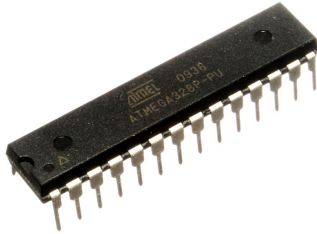


Fig. 1. Microcontrolador ATMEGA328P

### B. Comunicación Serial (USART)

La comunicación serial es un método de transmisión de datos en el cual la información se envía bit a bit a través de un canal de comunicación. Dentro de esta categoría, la interfaz UART (Universal Asynchronous Receiver and Transmitter) y su extensión USART (Universal Synchronous and Asynchronous Receiver and Transmitter) son ampliamente utilizadas en microcontroladores y sistemas embebidos [3].

En el modo asíncrono, la transmisión no requiere una señal de reloj compartida entre emisor y receptor. En su lugar, ambos dispositivos deben acordar previamente parámetros como la velocidad de transmisión (baud rate) y el formato de la trama de datos. La sincronización se logra mediante bits especiales que indican el inicio y fin de cada paquete transmitido.

Por otro lado, en el modo síncrono (propio de USART), se utiliza una línea de reloj adicional, lo cual permite mayor velocidad y confiabilidad en la comunicación. Sin embargo, el modo asíncrono UART sigue siendo el más utilizado debido a su simplicidad y menor requerimiento de pines.

1) *Formato de transmisión:* La transmisión de datos mediante UART/USART sigue un formato estructurado de trama, compuesto por los siguientes elementos:

- **Bit de inicio (Start bit):** Indica el comienzo de la transmisión; normalmente consiste en un nivel lógico bajo (0).
- **Bits de datos:** Generalmente entre 5 y 9 bits, aunque lo más común es el uso de 8 bits, lo que permite la transmisión de un byte por trama.
- **Bit de paridad (opcional):** Permite la detección básica de errores en la transmisión, pudiendo configurarse como par, impar o sin paridad.
- **Bit(s) de parada (Stop bit):** Señalizan el final de la trama; pueden configurarse en 1 o 2 bits.

- **Velocidad de transmisión (baud rate):** Corresponde al número de símbolos transmitidos por segundo. En UART, un baud equivale a un bit, por lo que baud rate y bit rate son equivalentes. Valores comunes son 9600, 19200 y 115200 baudios, dependiendo de la aplicación.

La correcta configuración de estos parámetros en ambos extremos es fundamental para garantizar una comunicación confiable y libre de errores.



Fig. 2. Comunicación UART

### C. Pantalla LCD 16x2 con interfaz I<sup>2</sup>C

Una pantalla LCD (Liquid Crystal Display) de 16x2 es un dispositivo de salida utilizado comúnmente en sistemas embebidos para la visualización de información alfanumérica. El número “16x2” indica que el módulo posee dos líneas capaces de mostrar hasta dieciséis caracteres cada una. Este tipo de pantallas se basa en la tecnología de cristal líquido, la cual controla la orientación de moléculas entre dos capas conductoras para modular la cantidad de luz transmitida y así formar los caracteres visibles.

El módulo LCD 16x2 requiere tradicionalmente múltiples conexiones paralelas para su control (generalmente 8 líneas de datos y varias de control). Sin embargo, mediante el uso de un expansor de bus I<sup>2</sup>C (como el PCF8574), es posible reducir significativamente el número de pines necesarios para su conexión, pasando de un esquema paralelo a una comunicación serial de dos líneas: SDA (Serial Data) y SCL (Serial Clock).

La interfaz I<sup>2</sup>C (Inter-Integrated Circuit) permite conectar múltiples dispositivos esclavos utilizando solo estas dos líneas, cada uno con una dirección única. En el caso del LCD 16x2, el módulo I<sup>2</sup>C actúa como un puente que traduce las señales seriales en comandos de control para el controlador interno del display (habitualmente el HD44780 o compatible). Esto simplifica el cableado, reduce el consumo de pines del microcontrolador y facilita la integración en aplicaciones de automatización o monitoreo.

El control del LCD incluye comandos para posicionar el cursor, limpiar la pantalla, activar o desactivar el retroiluminado, y escribir caracteres personalizados en la memoria CGRAM. Estos comandos pueden enviarse mediante bibliotecas específicas para el microcontrolador, como la librería `LiquidCrystal_I2C` en entornos de desarrollo Arduino, lo que permite una programación más sencilla y estructurada [5].



Fig. 3. Pantalla LCD con interfaz I2C

#### D. Tira de LEDs WS2812

La tira de LEDs WS2812 está compuesta por diodos emisores de luz RGB controlables individualmente mediante un único bus de datos digital. Cada módulo WS2812 integra un chip controlador que interpreta la señal y ajusta la intensidad de cada componente de color (rojo, verde y azul) mediante modulación por ancho de pulso (PWM). El protocolo de comunicación es de tipo unidireccional y de tiempo crítico, requiriendo precisión en la generación de los pulsos para codificar correctamente los valores de color de 24 bits por LED. Gracias a su facilidad de uso y a la posibilidad de controlar múltiples LEDs con un solo pin, se emplea ampliamente en proyectos de iluminación decorativa, interfaces visuales y sistemas embebidos interactivos [7].



Fig. 4. Matriz LED

#### E. Sensor de Temperatura y Humedad DHT11

El DHT11 es un sensor digital básico de bajo costo diseñado para medir temperatura y humedad relativa en el ambiente. Integra un sensor capacitivo para la humedad y un termistor para la temperatura, conectados a un microcontrolador de 8 bits de alto rendimiento que se encarga de realizar la conversión analógica-digital y enviar los datos mediante una señal digital serial por un único hilo (Single-Wire) [8].

Aunque su velocidad de muestreo es lenta (aproximadamente una lectura cada dos segundos), es ampliamente utilizado en aplicaciones académicas por su simplicidad. Sus rangos de operación típicos son de 20% a 90% de humedad relativa (con  $\pm 5\%$  de precisión) y de  $0^{\circ}\text{C}$  a  $50^{\circ}\text{C}$  de temperatura (con  $\pm 2^{\circ}\text{C}$  de precisión).

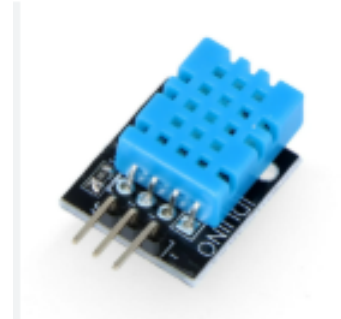


Fig. 5. Sensor DHT11

#### F. Arquitectura Maestro-Eslavo

El modelo maestro-esclavo es un paradigma de comunicación asimétrica utilizado en sistemas de control y redes de datos. En esta arquitectura, un dispositivo denominado "Maestro" (Master) tiene el control unilateral sobre el bus de comunicación o el proceso, siendo el responsable de iniciar las transferencias de datos y generar las señales de reloj (en protocolos síncronos). Por otro lado, uno o más dispositivos "Esclavos" (Slaves) permanecen a la espera de recibir comandos o solicitudes del maestro para actuar o enviar información [9].

Esta jerarquía es fundamental en sistemas embebidos distribuidos para evitar colisiones de datos en el canal de transmisión y mantener la sincronización de tareas críticas.

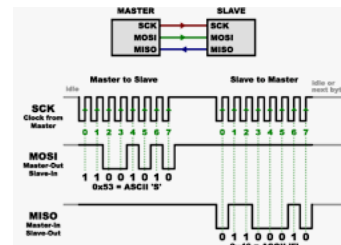


Fig. 6. Arquitectura Maestro-Eslavo

#### G. Protocolo de Comunicación SPI

La Interfaz Periférica Serial (SPI, por sus siglas en inglés) es un estándar de comunicación síncrona de alta velocidad desarrollado por Motorola, que opera en modo *full-duplex* (transmisión y recepción simultánea). A diferencia de otros protocolos, SPI utiliza una arquitectura de bus de cuatro líneas dedicadas [10]:

- **MOSI (Master Out Slave In):** Línea de datos del maestro al esclavo.

- **MISO (Master In Slave Out):** Línea de datos del esclavo al maestro.
- **SCK (Serial Clock):** Señal de reloj generada por el maestro para sincronizar la transmisión.
- **SS/CS (Slave Select / Chip Select):** Línea utilizada por el maestro para habilitar o deshabilitar a dispositivos esclavos específicos.

SPI es ideal para comunicaciones de corta distancia que requieren altas tasas de transferencia, como el control de pantallas TFT, memorias SD o comunicación entre microcontroladores, aunque tiene la desventaja de requerir más pines físicos a medida que aumenta el número de esclavos.

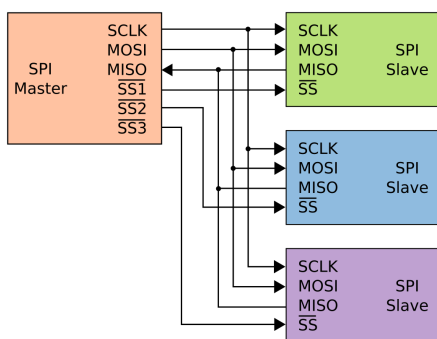


Fig. 7. Diagrama de conexión SPI

#### H. Protocolo de Comunicación I<sup>2</sup>C

El protocolo I<sup>2</sup>C (Inter-Integrated Circuit), desarrollado por Philips Semiconductors, es un bus de comunicación serial síncrono, *half-duplex* y multi-maestro. Su principal ventaja radica en la eficiencia del cableado, ya que permite conectar múltiples dispositivos (hasta 127 teóricamente) utilizando únicamente dos líneas con resistencias de pull-up [11]:

- **SDA (Serial Data):** Línea bidireccional para la transferencia de datos.
- **SCL (Serial Clock):** Línea para la señal de reloj.

A diferencia de SPI, I<sup>2</sup>C no utiliza líneas de selección físicas (SS), sino que emplea un sistema de direccionamiento por software donde cada esclavo posee una dirección única (generalmente de 7 bits). Esto simplifica el diseño del circuito impreso, aunque limita la velocidad de transmisión en comparación con SPI.

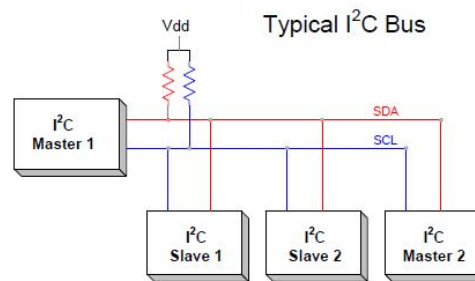


Fig. 8. Diagrama de conexión I<sup>2</sup>C

#### I. Giroscopio y Acelerómetro MPU6050

El MPU6050 es un dispositivo MEMS (Micro-Electro-Mechanical System) de 6 grados de libertad (6-DOF) que integra en un mismo chip un giroscopio de 3 ejes y un acelerómetro de 3 ejes. Además, cuenta con un procesador digital de movimiento (DMP) capaz de ejecutar algoritmos complejos de fusión de sensores para entregar datos de orientación precisos, reduciendo la carga computacional del microcontrolador principal [12].

El giroscopio mide la velocidad angular en grados por segundo ( $^{\circ}/s$ ), mientras que el acelerómetro mide la aceleración (incluyendo la gravedad) en fuerzas g. La combinación de ambos permite determinar la inclinación y orientación de un objeto en el espacio tridimensional, siendo fundamental para aplicaciones de robótica, drones y controladores de juegos. La comunicación con el microcontrolador se realiza típicamente mediante el bus I<sup>2</sup>C.

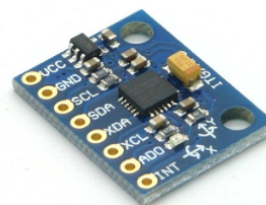


Fig. 9. Módulo MPU6050

#### J. Sensores Infrarrojos

Los sensores infrarrojos son componentes optoelectrónicos que emiten y detectan radiación en el espectro infrarrojo para determinar distancias o detectar la presencia de objetos. En aplicaciones de robótica móvil, como seguidores de línea o detección de obstáculos, se utilizan comúnmente en configuración reflectiva [13].

Estos módulos constan de un LED emisor de infrarrojos y un fototransistor o fotodiodo receptor. Cuando la luz emitida rebota en una superficie clara, el fototransistor conduce corriente; si la superficie es oscura (o no hay superficie), la luz es absorbida o dispersada, y el fototransistor no conduce. Esta variación de voltaje es interpretada por el microcontrolador como un estado lógico (0 o 1) o una señal analógica, permitiendo al robot distinguir líneas en el suelo o evitar colisiones.

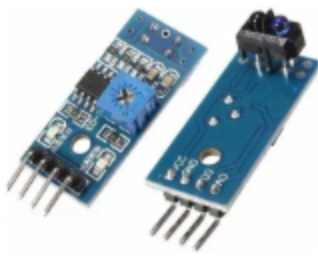


Fig. 10. Sensor infrarrojo

#### K. Módulo Bluetooth HC-05

El HC-05 es un módulo de comunicación inalámbrica que utiliza el estándar Bluetooth 2.0 + EDR (Enhanced Data Rate). Está diseñado para reemplazar conexiones seriales cableadas, funcionando bajo el perfil de puerto serial (SPP), lo que permite enviar y recibir datos de manera transparente hacia el microcontrolador a través de la interfaz UART [14].

Este módulo es versátil ya que puede configurarse tanto en modo maestro como en modo esclavo mediante comandos AT. En modo esclavo (configuración por defecto), permite que dispositivos como teléfonos inteligentes o computadoras se conecten a él para enviar comandos de control al sistema embebido, característica esencial para la teleoperación de robots móviles.

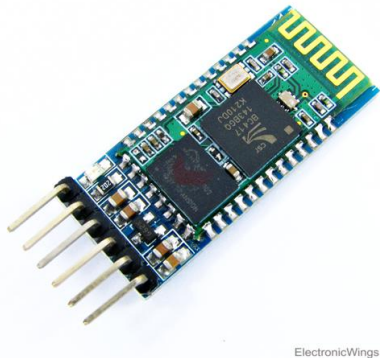


Fig. 11. Módulo Bluetooth HC-05

### V. PROCEDIMIENTO

A continuación se detallan los pasos seguidos para cumplir con cada apartado del laboratorio.

#### A. Parte A

#### B. Parte B

Para la realización de la Parte B del laboratorio, se implementó un sistema maestro-esclavo utilizando microcontroladores ATmega328P. El maestro se encargó de leer los sensores y controlar la visualización de datos en un LCD I2C, mientras que el esclavo se ocupó de activar los actuadores según los comandos recibidos.

En primer lugar, en el maestro se configuraron los periféricos necesarios: se inicializó el ADC para la lectura

del potenciómetro, se configuró un botón como entrada con resistencia pull-up interna, y se inicializó la comunicación I2C junto con la pantalla LCD mediante las librerías `twi.h` y `twi_lcd.h`. Se incluyó un retraso inicial de 500 ms para asegurar que la pantalla LCD estuviera lista antes de mostrar cualquier información.

El sensor DHT11 se configuró para la lectura de humedad y temperatura. Para ello, se envió una señal de inicio al sensor llevando su pin de datos a bajo durante 20 ms y luego a alto por 30  $\mu$ s, cambiando posteriormente el pin a entrada para recibir la respuesta del sensor. Se leyeron cinco bytes de información: humedad entera y decimal, temperatura entera y decimal, y checksum. La integridad de los datos se verificó comparando la suma de los primeros cuatro bytes con el checksum recibido.

Paralelamente, se leyeron los valores del potenciómetro y del botón. Se implementó una zona muerta alrededor del valor medio del potenciómetro para evitar que pequeñas variaciones hicieran oscilar el motor. Los valores de temperatura, humedad y potenciómetro se mostraron en la LCD en tiempo real, mientras que la pulsación del botón se utilizó para activar un buzzer conectado al esclavo.

La comunicación maestro-esclavo se realizó mediante el protocolo I2C. Según las lecturas, el maestro enviaba comandos discretos al esclavo: los comandos 'A' y 'B' controlaban un LED de alerta de humedad, los comandos 'C' y 'D' controlaban el motor DC, y el comando 'E' activaba el buzzer.

En el esclavo, se configuró la dirección I2C 0x10 y se habilitaron las interrupciones para recibir datos. Los pines correspondientes al LED, motor y buzzer se configuraron como salidas y se inicializaron apagados. Cada vez que llegaba un comando por I2C, la rutina de interrupción leía el dato y activaba el actuador correspondiente. Para el buzzer, se utilizó una bandera que se gestionaba en el bucle principal del esclavo para evitar bloquear la comunicación I2C, permitiendo así que el buzzer sonara brevemente cuando se presionaba el botón del maestro.

Durante todo el procedimiento se consideraron aspectos de seguridad y robustez: se implementó un retraso inicial para la LCD, se estableció una zona muerta para el control del motor, y se evitó bloquear el bus I2C con delays en la rutina de interrupción. Se utilizó únicamente la librería de I2C y de LCD para la visualización; no se implementó control proporcional de PWM en el esclavo, sino un control discreto de encendido y apagado para el motor y el buzzer.

En resumen, el maestro lee los sensores y evalúa las condiciones de humedad, potenciómetro y pulsación de botón, enviando comandos discretos al esclavo. El esclavo, mediante interrupciones, recibe estos comandos y activa los actuadores correspondientes, mostrando un sistema maestro-esclavo funcional que integra lectura de sensores, visualización y control de actuadores mediante comunicación I2C.

#### C. Parte C

Para este inciso de la práctica, en primer lugar, se determinaron las animaciones a mostrar en la matriz de leds. A

medida que la utilización del controlador WS2812B se hacía mas familiar, se plantearon la realización de animaciones más complejas. Finalmente, las animaciones elegidas fueron las siguientes:

La primera animacion se constituye de una moto, representada en la matriz como un rectángulo el cual ocupa dos leds encendidos de color rojo. Este vehiculo se encuentra transitando por una ruta desolada a altas velocidades, mientras que la vista sigue la secuencia de forma cenital.

Con respecto a la segunda animación, esta se conforma tan solamente de una figura geométrica de forma circular, simulando ser nuestro sol, en un fondo cambiante. Esta animación pretende mostrar el ciclo diario del sol, representándose esto con el cambio de intensidad de los colores tanto del fondo como de la figura, a la par que la misma se mueve de forma vertical.

En relación con el código desarrollado para la conclusión de este inciso, primeramente, luego de declarar variables, y llamar las librerías a utilizar, se incorpora una tabla de colores que representa la transición gradual del cielo para la animación del amanecer.

Posterior a esto, se configura la comunicación UART, y otra función, la cual permite leer caracteres sin bloquear la ejecución, lo que facilita cambiar entre las animaciones cuando llegan los caracteres '1' o '2'.

Debido a que el manejo de los LEDs WS2812 requiere precisión en los tiempos, se implementan rutinas que generan los pulsos exactos para representar bits altos y bajos utilizando instrucciones nop. Estas rutinas se integran en una función que envía un byte completo y en otra que se encarga de transmitir los tres bytes correspondientes a cada LED en el orden específico que el protocolo exige. Durante este envío se deshabilitan las interrupciones para evitar alteraciones en el tiempo de transmisión. Una función auxiliar permite modificar fácilmente el color RGB de cualquier LED dentro del arreglo general.

En la animación de la motocicleta se construye un escenario compuesto por una franja de pasto en los bordes, una carretera en la zona central y una línea discontinua que parece avanzar gracias al desplazamiento progresivo basado en un contador de cuadros. En la parte inferior se colorea un pequeño bloque rojo que representa la motocicleta. La animación cambia cada cuadro ajustando el patrón de la carretera.

La animación del amanecer utiliza la tabla de colores predefinida para determinar el tono del cielo según el avance del tiempo. A partir del mismo contador de cuadros se calcula también la posición del sol, que asciende a lo largo de la matriz y se representa mediante un grupo de LEDs cercanos, con un brillo derivado del promedio de los valores de color del cielo en ese momento.

Finalmente, con el flujo en la función principal, el programa alterna entre los efectos visuales a través de comandos recibidos por UART.

#### *D. Parte D*

Para la antepenúltima parte de este laboratorio se implementó un sistema de control de un LED dentro de una matriz RGB de 8x8 utilizando el giroscópio MPU6050 (9) como interfaz de usuario. El objetivo principal consistió en desplazar un LED activo dentro de la matriz según la inclinación física del giroscópio y permitir el cambio aleatorio de color al presionar un botón.

Se configuró el microcontrolador ATmega328P definiendo los pines de salida para el control de los LEDs WS2812 y el pin PD7 como entrada con resistencia de pull-up interna para el botón. A continuación, se inicializó el módulo de comunicación I2C (TWI) para establecer conexión con el sensor MPU6050. Como parte de la inicialización del sensor, se escribió en el registro de gestión de energía (0x6B) para desactivar el modo de suspensión y habilitar la adquisición de datos. Paralelamente, se configuró la comunicación UART para monitorear los valores crudos del sensor en tiempo real.

Se estableció un LED inicial activo en la posición (4,4) de la matriz, asignándole un color predeterminado. Para la actualización de la matriz, se implementaron funciones que transmiten los valores de rojo, verde y azul a través de la señal serial de los LEDs WS2812, asegurando que únicamente el LED activo muestre color mientras los demás permanecen apagados. Se incluyó un procedimiento de reinicio de señal para garantizar la actualización completa de la matriz.

Durante la ejecución, se realizaron lecturas cíclicas de los registros de aceleración X e Y del MPU6050 mediante I2C. Los valores de 16 bits con signo se reconstruyeron a partir de los bytes leídos. Se definió una zona muerta de  $\pm 2000$  unidades para filtrar pequeñas vibraciones o inclinaciones involuntarias que podrían generar movimientos no deseados del LED.

La lógica de movimiento se implementó evaluando los umbrales de inclinación de cada eje. Cuando la lectura de un eje excedió el límite positivo o negativo, se actualizó la coordenada correspondiente del LED, respetando los límites de la matriz (0–7). Para evitar movimientos continuos indeseados, se utilizó una variable de estado que requería que el sensor regresara a la zona neutral antes de permitir un nuevo desplazamiento.

El cambio de color del LED activo se implementó mediante la función rand(), inicializada con valores de temporizadores internos. Al presionar el botón PD7, se generaron nuevas componentes RGB para el LED activo, manteniendo su posición en la matriz. La actualización de la matriz se realizó de manera continua dentro del bucle principal del programa.

El bucle principal ejecutó de manera cíclica la lectura de los registros del sensor, la detección de presiones del botón y la actualización de la matriz, incluyendo retardos cortos para garantizar la correcta ejecución de cada paso sin movimientos múltiples no deseados.

#### *E. Parte E*

Para la última parte del laboratorio se implementó un sistema de control remoto de un robot móvil utilizando un microcontrolador ATmega328P. El robot fue armado físicamente

sobre una plantilla de MDF proporcionada, incorporando dos motores DC acoplados a ruedas para permitir el desplazamiento en todas las direcciones, y un servo montado para simular movimientos de golpe. La estructura proporcionada facilitó la colocación de los componentes electrónicos y mecánicos, asegurando una integración adecuada entre el chasis, los actuadores y el microcontrolador.

El objetivo principal consistió en controlar el movimiento del robot y el accionamiento del servo mediante comandos enviados por un módulo Bluetooth, utilizando la comunicación UART del microcontrolador. Este montaje permitió evaluar la correcta interacción entre hardware y software, incluyendo el control de motores, generación de señales PWM para el servo y la recepción de datos en tiempo real desde un dispositivo externo.

Lo que respecta al código del mismo se configuraron los pines PD2, PD3, PD4 y PD5 del microcontrolador ATmega328P como salidas digitales para el control del puente H de los motores DC. El pin PB1 se definió como salida para el control del servo mediante PWM utilizando el temporizador 1.

La inicialización del servo se realizó configurando el temporizador 1 en modo Fast PWM de 16 bits con TOP definido por ICR1. Se ajustaron los registros TCCR1A y TCCR1B para generar una señal de PWM con frecuencia adecuada para el servo, y se estableció el valor inicial de OCR1A correspondiente al ángulo de reposo.

Se implementaron funciones de control del servo para realizar movimientos específicos: `patearDER()` y `patearIZQ()`, que mueven el servo a posiciones predeterminadas para simular un golpe hacia la derecha o izquierda, retornando al ángulo de reposo después de un tiempo definido mediante retardos.

Para el control del movimiento del robot, se definieron funciones `adelante()`, `atras()`, `derecha()`, `izquierda()` y `parar()`. Estas funciones manipulan los estados de los pines de salida conectados al puente H para generar la dirección deseada en los motores, permitiendo movimiento en todas las direcciones y detención completa.

Se configuró la comunicación UART a 9600 bps permitiendo recibir comandos enviados por el módulo Bluetooth (12). El sistema imprime mensajes de estado a través de UART para monitoreo.

El programa principal ejecuta un bucle infinito en el que se verifica continuamente la recepción de datos por UART. Cuando se recibe un carácter, se interpreta mediante una estructura switch y se ejecuta la función correspondiente para mover el robot o accionar el servo, según el comando recibido (W, A, S, D, P, X, Y).

De esta manera, se logró implementar un sistema de control remoto del robot, con desplazamiento direccional mediante motores DC y acciones del servo para simular golpes, controlado de forma inalámbrica a través de UART y Bluetooth.

## VI. RESULTADOS

### A. Parte A

### B. Parte B

Durante la implementación del sistema maestro-esclavo, se obtuvieron los resultados esperados: el maestro logró leer correctamente los valores de humedad y temperatura del sensor DHT11, así como los valores del potenciómetro y la pulsación del botón. Estos datos se visualizaron en tiempo real en la pantalla LCD, mientras que el esclavo activaba los actuadores según los comandos recibidos por I2C.

La conexión entre las placas presentó ciertas dificultades iniciales, ya que fue necesario colocar resistencias pull-up adicionales en las líneas de datos y reloj del bus I2C para asegurar una comunicación estable entre maestro y esclavo. Este ajuste fue fundamental para evitar errores de transmisión y garantizar que los actuadores respondieran correctamente a los comandos enviados.

A pesar de la complejidad del montaje y de las correcciones necesarias, el sistema funcionó de manera satisfactoria, cumpliendo con los objetivos planteados. La lectura de sensores, el control del motor y la activación del buzzer se realizaron de forma confiable, logrando un resultado coherente con lo esperado. Aunque la puesta a punto requirió tiempo y paciencia, la experiencia permitió comprender mejor la integración de sensores, actuadores y comunicación I2C en un sistema maestro-esclavo, lo cual constituye un aprendizaje valioso para futuras aplicaciones.

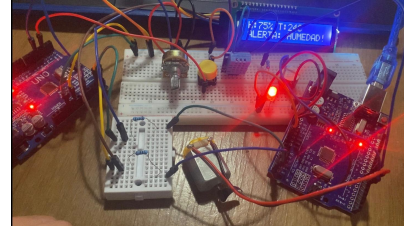


Fig. 12. Implementación maestro-esclavo con I<sup>2</sup>C

### C. Parte C

Como puede verse en las figuras 13 y ??, las animaciones corresponden perfectamente con las descritas antes de comenzar a desarrollarse el código, a su vez, las animaciones pueden ser intercaladas entre ellas a gusto haciendo uso de la comunicación USART, por lo que se concluye que los resultados de este apartado son plenamente satisfactorios.

### D. Parte D

En esta etapa se implementó el control de una matriz de LEDs RGB mediante un sensor de aceleración MPU6050. A diferencia de las secciones anteriores, esta parte no fue simulada digitalmente, sino implementada directamente en hardware físico. El sistema fue programado para encender un único LED de color variable dentro de la matriz, permitiendo desplazarlo en las cuatro direcciones mediante la inclinación del sensor.

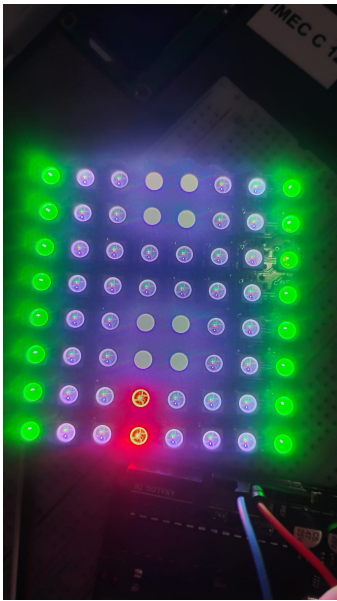


Fig. 13. Animacion 1 del inciso C

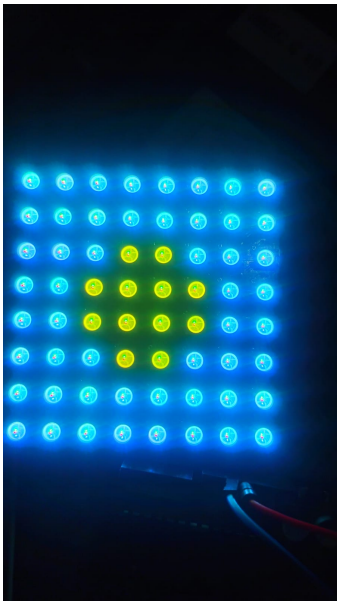


Fig. 14. Animacion 2 del inciso C

Se verificó la correcta lectura de los valores crudos del acelerómetro y la calibración de la zona muerta para filtrar pequeñas vibraciones involuntarias. Posteriormente, se adaptó el código de control de la matriz de LEDs para interpretar los valores del MPU6050 como desplazamientos del LED activo, asegurando que los límites de la matriz fueran respetados. El LED inicial se ubicó en la posición central (4,4) con un color predefinido, y fue posible desplazarlo en los ejes X e Y, incluyendo movimientos diagonales. Además, al presionar el botón físico, el color del LED activo se modificó de manera aleatoria, confirmando la correcta integración entre la lectura del sensor, la lógica de control y la actualización de los LEDs.

La siguiente imagen corresponde a un fragmento del video de la implementación funcional, mostrando la posición del LED activo y su cambio de color en respuesta a la interacción con el sensor y el botón.

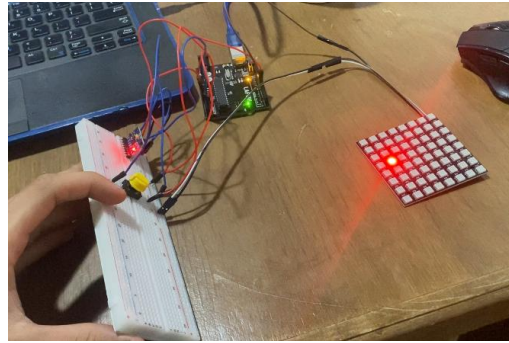


Fig. 15. Implementación matriz con sensor MPU6050

### E. Parte E

En esta etapa se implementó y probó el control remoto del robot móvil utilizando un microcontrolador ATmega328P y un módulo Bluetooth. A diferencia de secciones anteriores, no se utilizaron sensores infrarrojos, ya que el control se realiza completamente a través de la comunicación Bluetooth, delegando el manejo del robot en el usuario mediante comandos remotos.

Para la operación del robot se empleó la aplicación “Arduino Car”, que permite controlar el vehículo mediante una interfaz similar a la de una consola. Los comandos de la aplicación se transmiten al microcontrolador por UART y se interpretan para ejecutar movimientos direccionales y acciones del servo.

La alimentación del robot se realizó mediante un power bank, proporcionando la corriente necesaria tanto para los motores DC como para el servo y el microcontrolador. Durante las pruebas se detectó que el vehículo tendía a deslizarse hacia un lado debido a la distribución de peso, por lo que se agregó peso extra en la parte trasera del chasis para mejorar la estabilidad y la tracción durante el desplazamiento.

La implementación permitió el desplazamiento en todas las direcciones y el accionamiento del servo para simular golpes, con respuesta inmediata a los comandos enviados desde la aplicación. La siguiente figura muestra el robot en funcionamiento bajo control remoto mediante Bluetooth.

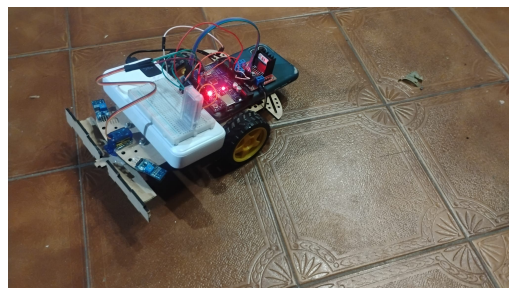


Fig. 16. Robot controlado mediante aplicación Bluetooth

## VII. CONCLUSIÓN

Como conclusi[on final, de esta práctica nos llevamos la capacidad integrar de manera práctica múltiples conceptos fundamentales de los sistemas embebidos, consolidando el uso de protocolos de comunicación, la gestión de sensores y actuadores, y la implementación de algoritmos de control en tiempo real.

En la Parte B se logró implementar correctamente la lectura de sensores, la visualización en LCD y el control de actuadores, destacando la necesidad de ajustes como el uso de resistencias pull-up para asegurar una comunicación estable.

En la Parte C se profundizó en el manejo de LEDs direccionables WS2812 y en la construcción de animaciones personalizadas, lo que implicó comprender la temporización estricta del protocolo y el diseño de estructuras de datos para manipular frames en memoria. El uso de UART para alternar animaciones demostró la utilidad de la comunicación serial como interfaz flexible.

En la Parte D se integró el sensor MPU6050 para controlar dinámicamente una matriz RGB, aplicando lectura de aceleraciones, zonas muertas y lógica de movimiento, reafirmando el rol de los sensores inerciales en aplicaciones interactivas y de control espacial.

Finalmente, en la Parte E se implementó un robot móvil controlado por Bluetooth, combinando tracción, servoactuación y comunicación inalámbrica, lo cual sintetiza la totalidad de los aprendizajes previos en un sistema funcional.

## VIII. BIBLIOGRAFÍA

### REFERENCES

- [1] W. Wolf, *Computers as Components: Principles of Embedded Computing System Design*, 3rd ed., Morgan Kaufmann, 2012.
- [2] Microchip Technology Inc., *ATmega328P: 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash*, Datasheet, 2023.
- [3] Microchip Technology Inc., *AVR306: Using the USART in AVR Microcontrollers*, Application Note, 2021.
- [4] J. García and M. Pérez, *Automatización y Control Industrial*, 2nd ed., Madrid: Alfaomega, 2020.
- [5] D. Flores, *Interfacing LCDs via I2C Bus: Using the PCF8574 with HD44780 Displays*, Texas Instruments Application Note, 2020.
- [6] Microchip Technology Inc., *AVR101: High Endurance EEPROM Storage in AVR Microcontrollers*, Application Note, 2021.
- [7] Worldsemi, "WS2812 Intelligent Control LED Datasheet," 2022. [Online]. Available: <https://cdn-shop.adafruit.com/datasheets/WS2812.pdf>
- [8] Aosong Electronics Co., Ltd., *Temperature and Humidity Module DHT11 Product Manual*, Datasheet, 2020. [Online]. Disponible en: <https://www.mouser.com/datasheet/2/758/DHT11-Technical-Data-Sheet-Translated-Version-1143054.pdf>
- [9] R. Kamal, *Embedded Systems: Architecture, Programming and Design*, 3rd ed., New York: McGraw-Hill Education, 2017.
- [10] Motorola Inc., *SPI Block Guide V04.01*, Application Note, 2004. [Online]. Disponible en: <https://www.nxp.com>
- [11] NXP Semiconductors, *I2C-bus specification and user manual*, Rev. 6, 2014. [Online]. Disponible en: <https://www.nxp.com/docs/en/user-guide/UM10204.pdf>
- [12] InvenSense Inc., *MPU-6000 and MPU-6050 Product Specification Revision 3.4*, Datasheet, 2013. [Online]. Disponible en: <https://invensense.tdk.com>
- [13] Vishay Semiconductors, *TCRT5000 Reflective Optical Sensor with Transistor Output*, Datasheet, 2021. [Online]. Disponible en: <https://www.vishay.com/docs/83760/tcrt5000.pdf>
- [14] ITEad Studio, *HC-05 Bluetooth to Serial Port Module*, Datasheet, 2010. [Online]. Disponible en: [https://www.itead.cc/wiki/HC-05\\_Bluetooth\\_Module](https://www.itead.cc/wiki/HC-05_Bluetooth_Module)