Informe X – Análisis de Redes WiFi con Raspberry Pi Pico 2W

Dylan Ferney Vasquez Rojas
1401597
Comunicaciones Digitales

***Resumen*— En esta práctica se implementó un análisis de redes WiFi utilizando la Raspberry Pi Pico 2W, con el propósito de comprender el funcionamiento de la comunicación inalámbrica en la banda de 2.4 GHz mediante el uso del lenguaje MicroPython. Se estudiaron los conceptos teóricos fundamentales relacionados con la tecnología WiFi, incluyendo el BSSID (Basic Service Set Identifier) y el RSSI (Received Signal Strength Indicator), parámetros esenciales para la identificación y evaluación de redes inalámbricas.**

El desarrollo experimental comprendió la obtención y verificación de la dirección MAC del dispositivo, el escaneo de redes WiFi para registrar los puntos de acceso disponibles y analizar la variación del RSSI, así como la configuración del Pico W como punto de acceso (AP) para permitir la conexión de un cliente y el control remoto del LED integrado a través de una interfaz web local. También se realizaron pruebas de cambio de canal con el fin de observar la distribución espectral y la interferencia entre redes solapadas.

***Abstract*— This exercise involved analyzing WiFi networks using the Raspberry Pi Pico 2W to understand wireless communication in the 2.4 GHz band using the MicroPython programming language. Fundamental theoretical concepts related to WiFi technology were studied, including the BSSID (Basic Service Set Identifier) and RSSI (Received Signal Strength Indicator), essential parameters for identifying and evaluating wireless networks.**

The experimental process included obtaining and verifying the device's MAC address, scanning WiFi networks to record available access points and analyze RSSI variations, and configuring the Pico W as an access point (AP) to allow client connection and remote control of the integrated LED via a local web interface. Channel switching tests were also performed to observe spectral distribution and interference between overlapping networks.

I. Introducción

La tecnología WiFi se ha consolidado como uno de los principales medios de comunicación inalámbrica, permitiendo la interconexión de dispositivos dentro de redes locales mediante las bandas de 2.4 GHz y 5 GHz. En el ámbito de la electrónica y las telecomunicaciones, su integración en sistemas embebidos ha facilitado el desarrollo de aplicaciones IoT (Internet of Things) orientadas al control, monitoreo y transmisión de datos en tiempo real.

En esta práctica se utilizó la Raspberry Pi Pico 2W, un microcontrolador de bajo costo con capacidad WiFi integrada, con el fin de comprender los principios de operación de las redes inalámbricas y su aplicación en entornos prácticos. A través de scripts desarrollados en MicroPython, se realizaron procesos de escaneo de redes, identificación de parámetros de señal (RSSI, BSSID) y configuración de un punto de acceso (AP) que permitió la comunicación entre el microcontrolador y un dispositivo cliente mediante una interfaz web local.

El objetivo principal de la práctica fue analizar el comportamiento de la señal WiFi, el efecto del canal de transmisión y la capacidad del Pico 2W para establecer conexiones inalámbricas, reforzando los fundamentos teóricos y experimentales sobre comunicación inalámbrica y conectividad en sistemas embebidos.

DESARROLLO DE LA PRÁCTICA

El desarrollo de la práctica se realizó utilizando la Raspberry Pi Pico 2W y el entorno de programación MicroPython, siguiendo una secuencia de experimentos orientados al análisis del comportamiento de redes WiFi en la banda de 2.4 GHz.

I. Dirección MAC del dispositivo

Se ejecutó el programa mac_wifi.py para obtener la dirección MAC (Media Access Control) del módulo WiFi de la Raspberry Pi Pico 2W. Se comprobó que la dirección se mantiene constante entre reinicios, al ser un identificador único asignado al hardware del dispositivo.

Ilustración. 1. MAC Del Dispositivo RPP2W

a. La MAC del Pico (STA) identificada en el Procedimiento, ¿permanece constante entre reinicios? ¿Por qué?

Se ejecutó el programa mac_wifi.py para obtener la dirección MAC (Media Access Control) correspondiente a la

interfaz WiFi del modo estación (STA) de la Raspberry Pi Pico 2W. Este identificador se representa en formato hexadecimal (2C:CF:67:DD:9C:8E) y permite distinguir de forma única cada dispositivo dentro de una red.

Al reiniciar el microcontrolador en varias ocasiones, se observó que la dirección MAC permanece constante, debido a que está asignada físicamente al hardware del módulo WiFi por el fabricante (en este caso, Infineon para el chip CYW43439). Esto garantiza que cada dispositivo tenga un identificador único y permanente, facilitando su autenticación y gestión en redes inalámbricas.

II. Escaneo de redes Wi-Fi

Para esta parte del experimento se ejecutó el script scanner_wifi.py, encargado de escanear los puntos de acceso (AP) disponibles en el entorno y listar sus principales parámetros: canal, RSSI, BSSID y SSID. El proceso se mantuvo activo durante al menos 60 segundos, con actualizaciones periódicas cada 10 segundos, ordenando los resultados de mayor a menor intensidad de señal (RSSI).

```
Scanning for Wi-Fi access points...
CH  3 | RSSI -56 dBm | BSSID DE:74:EF:5B:B0:39 | SSID: OneScreen_7288
CH  6 | RSSI -58 dBm | BSSID 10:F0:68:77:21:2F | SSID: Recover.Me-372120
CH  2 | RSSI -58 dBm | BSSID D2:14:3D:5A:44:6C | SSID: DIRECT-hE-BRAVIA
CH 11 | RSSI -64 dBm | BSSID 28:37:37:47:86:36 | SSID: LABCOM
CH  8 | RSSI -73 dBm | BSSID C0:25:2F:44:F5:38 | SSID: <hidden>
CH  1 | RSSI -78 dBm | BSSID 1A:A4:EB:2F:29:1A | SSID: Armor X13
Waiting 10 seconds before next scan...

Scanning for Wi-Fi access points...
CH  3 | RSSI -56 dBm | BSSID DE:74:EF:5B:B0:39 | SSID: OneScreen_7288
CH 11 | RSSI -66 dBm | BSSID 28:37:37:47:86:36 | SSID: LABCOM
CH  1 | RSSI -68 dBm | BSSID 1A:A4:EB:2F:29:1A | SSID: Armor X13
CH  6 | RSSI -73 dBm | BSSID 10:F0:68:77:21:2F | SSID: Recover.Me-372120
CH  8 | RSSI -75 dBm | BSSID C0:25:2F:44:F5:38 | SSID: <hidden>
Waiting 10 seconds before next scan...

Scanning for Wi-Fi access points...
CH  3 | RSSI -55 dBm | BSSID DE:74:EF:5B:B0:39 | SSID: OneScreen_7288
CH  2 | RSSI -56 dBm | BSSID D2:14:3D:5A:44:6C | SSID: DIRECT-hE-BRAVIA
CH  6 | RSSI -62 dBm | BSSID 10:F0:68:77:21:2F | SSID: Recover.Me-372120
CH 11 | RSSI -63 dBm | BSSID 28:37:37:47:86:36 | SSID: LABCOM
CH  6 | RSSI -65 dBm | BSSID 10:F0:68:77:21:21 | SSID: UMMG-PRIVR-CLL100
CH  6 | RSSI -66 dBm | BSSID 10:F0:68:77:21:22 | SSID: LABORATORIOS
CH  6 | RSSI -66 dBm | BSSID 10:F0:68:77:21:20 | SSID: UMMG-PUBR-CLL100
CH  1 | RSSI -71 dBm | BSSID 1A:A4:EB:2F:29:1A | SSID: Armor X13
CH  8 | RSSI -71 dBm | BSSID C0:25:2F:44:F5:38 | SSID: <hidden>
CH  5 | RSSI -78 dBm | BSSID 10:F0:68:77:20:C2 | SSID: LABORATORIOS
CH  5 | RSSI -78 dBm | BSSID 10:F0:68:77:20:C0 | SSID: UMMG-PUBR-CLL100
CH  5 | RSSI -80 dBm | BSSID 10:F0:68:77:20:C1 | SSID: UMMG-PRIVR-CLL100
CH  1 | RSSI -83 dBm | BSSID 5C:DF:89:6A:5D:11 | SSID: UMMG-PRIVR-CLL100
CH  1 | RSSI -86 dBm | BSSID 5C:DF:89:6A:5D:10 | SSID: UMMG-PUBR-CLL100
Waiting 10 seconds before next scan...
```

Ilustración. 2. Datos De Puntos De Escaneo.

Durante las mediciones, se identificaron los tres puntos de acceso con mayor potencia de señal en cada iteración. Los valores de RSSI fluctuaron entre mediciones, lo que se atribuye a interferencias electromagnéticas, movimiento de personas u objetos, reflexiones de señal y variaciones en la sensibilidad del receptor.

Iteración (s)	Canal	SSID	RSSI (dBm)
0	3	OneScreen_7288	-63
	11	LABCOM	-63
	6	Recover.Me-372120	-65
10	3	OneScreen_7288	-58
	6	Recover.Me-372120	-58
	11	LABCOM	-61
20	3	OneScreen_7288	-56
	6	Recover.Me-372120	-58
	2	DIRECT-hE-BRAVIA	-58
30	3	OneScreen_7288	-56
	11	LABCOM	-66
	1	Armor X13	-68

Ilustración. 3. AP con mayor RSSI por cada iteración.

(Tabla completa en el repositorio GitHub)

a. <u>Compare los canales de los tres AP con mayor RSSI. ¿Hay solapamiento (1,6,11, etc en 2.4GHz)? ¿Qué implicaciones tiene?</u>

Durante el escaneo se identificó que los canales 3, 6 y 11 fueron los más utilizados por los puntos de acceso cercanos. Se evidenció un solapamiento espectral entre los canales 2, 3, 5, 6 y 8, lo cual puede generar interferencias y reducción del rendimiento de la red. Esto confirma la importancia de emplear los canales 1, 6 y 11, que son los únicos no solapados en la banda de 2.4 GHz.

b. <u>¿ El RSSI fluctúa entre iteraciones ?. ¿ Cuales podrian ser las causas ?.</u>

Asimismo, se observó que el RSSI varía en cada iteración, presentando fluctuaciones de hasta 10 dB. Dichas variaciones se deben a cambios en la propagación de la señal, interferencias de otros dispositivos Wi-Fi o Bluetooth, movimiento de objetos o personas y reflexiones en las superficies del entorno.

III. Creación de un punto de acceso inalámbrico

En esta fase de la práctica se configuró la Raspberry Pi Pico 2W como punto de acceso (Access Point, AP) con el fin de permitir la conexión directa de un cliente (PC o teléfono móvil) y controlar el LED integrado mediante una interfaz web local.

Para ello, se cargaron en el microcontrolador los archivos APWifipico.py e index.html, los cuales implementan la funcionalidad de servidor web y la página de control. Al renombrar el archivo Python como main.py, el programa se ejecutó automáticamente al energizar el dispositivo.

En el código se editó el bloque de configuración para asignar un nombre único al punto de acceso y establecer los parámetros de conexión:

SSID = "PicoW-AP-Grupo 4"
PASSWORD = "12345678"
CHANNEL = 6

Una vez ejecutado el programa, el Pico 2W generó su propia red WiFi, permitiendo que el cliente se conectara a través del SSID configurado.

Ilustración. 4.IP, CANAL y HTTP de la página web del LED

Al acceder desde el navegador web a la dirección http://192.168.4.1, se mostró la página de control (index.html) con botones para encender o apagar el LED mediante solicitudes HTTP GET.

Ilustración. 5. Visualización de la página web del LED

El control remoto del LED se realizó de forma instantánea, evidenciando la correcta comunicación entre el servidor embebido y el cliente. Este proceso permitió comprender el funcionamiento del modo AP, la gestión de solicitudes web en sistemas embebidos y la aplicación práctica de la comunicación cliente-servidor

Finalmente, se ingresó en el navegador la ruta http://192.168.4.1/state, donde el microcontrolador devolvió un mensaje en formato JSON, indicando el estado actual del LED ({"led": "on"} o {"led": "off"}). Este intercambio de información evidenció la capacidad del Pico 2W para actuar como servidor web embebido, procesar solicitudes y enviar respuestas estructuradas, lo que constituye la base funcional de un sistema IoT de control y monitoreo local.

Ilustración. 6. State con el Led apagado.

Ilustración. 7. State con el Led Encendido.

IV. Cambio de canal

Para verificar la asignación del canal de transmisión del punto de acceso, se modificó el parámetro CHANNEL en el archivo APWifipico.py, configurándolo inicialmente en el canal 6. Posteriormente, se utilizó la aplicación WiFi Analyzer desde un dispositivo móvil para corroborar la posición espectral del punto de acceso generado por la Raspberry Pi Pico 2W.

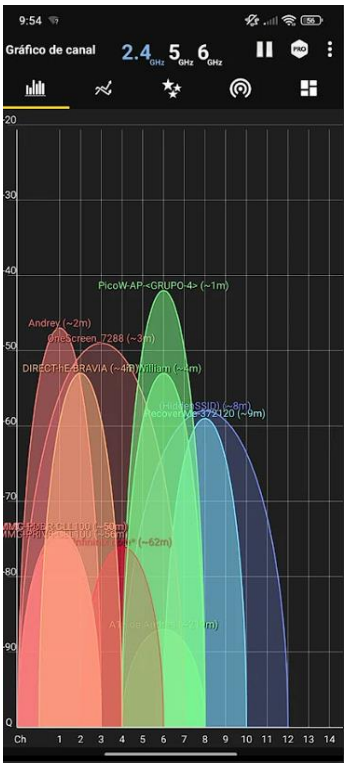

Ilustración. 8. Gráfico de canales WiFi donde se observa el AP PicoW-API-GRUPO4 centrado en el canal 6.

En el gráfico de canales mostrado en la Iustración 8, se puede observar que la red “PicoW-AP-GRUPO4” se encuentra centrada en el canal 6, confirmando la correcta configuración del parámetro de frecuencia.

Posteriormente, el canal fue cambiado al canal 10 para observar la nueva asignación de frecuencia. La verificación se realizó empleando una segunda Raspberry Pi Pico 2W ejecutando el programa scanner_wifi.py, además del monitoreo en tiempo real con WiFi Analyzer desde un dispositivo móvil. En ambos casos se confirmó que el punto de acceso cambió efectivamente de canal, evidenciando la flexibilidad del módulo WiFi integrado en la Pico 2W para operar en cualquier canal de la banda de 2.4 GHz.

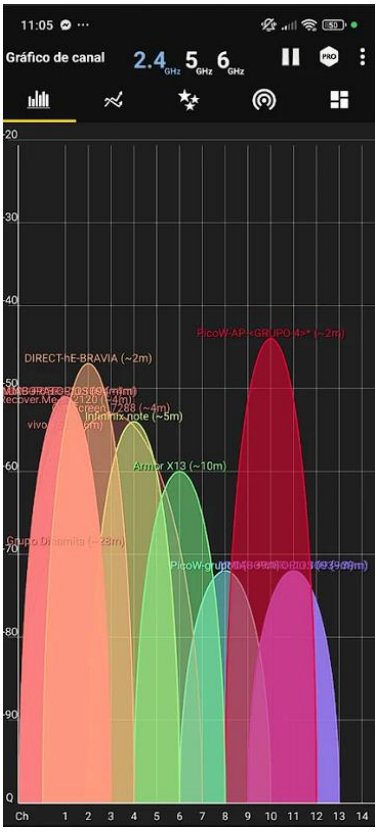

Ilustración. 9. Gráfico de canales WiFi donde se observa el AP PicoW-API-GRUPO4 centrado en el canal 10.

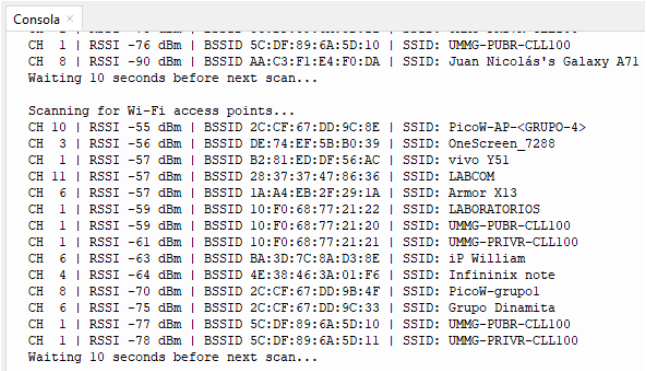

Ilustración. 10. Canales WiFi donde se observa el AP PicoW-API-GRUPO4 en el canal 10.

a. <u>¿Por que razón no hay solapamiento en los canales 1,6 y 11 en 2.4GHz)?</u>

Durante el análisis espectral se observó que al utilizar canales intermedios (como el 10), se generan zonas de solapamiento con redes adyacentes, lo cual puede incrementar la interferencia y reducir la intensidad de señal efectiva (RSSI). Por esta razón, se concluye que los canales 1, 6 y 11 son los más adecuados para evitar interferencias en entornos con múltiples puntos de acceso activos.

V. Actividades complementarias

- ADC

Como extensión de la práctica, se modificaron los programas APWifipico.py e index.html con el propósito de integrar la lectura del conversor analógico–digital (ADC0) de la Raspberry Pi Pico 2W, permitiendo visualizar en tiempo real el valor analógico desde la interfaz web local.

En el código principal, se incorporó el objeto adc0 = ADC(0) correspondiente al canal GPIO26, y se añadió una nueva ruta en el servidor web, /adc, que retorna el valor de conversión en formato JSON mediante el método adc0.read_u16(), el cual entrega valores de 0 a 65535. De esta forma, cada solicitud HTTP al endpoint /adc envía una respuesta con el valor actual del voltaje analógico leído, por ejemplo:

{“adc”: 32768}

En la interfaz index.html, se implementó un bloque adicional en el cuerpo del documento para mostrar el valor leído, junto con un script en JavaScript que consulta periódicamente (cada 2 segundos) la ruta /adc usando el método fetch(). Este valor se actualiza dinámicamente en pantalla dentro de un elemento identificado como adc-val.

La integración entre el servidor y la interfaz permitió monitorear de forma continua el estado del canal ADC mientras el dispositivo permanecía en modo punto de acceso (AP). Con esto, se comprobó la capacidad del Pico W para ejecutar simultáneamente funciones de servidor web embebido, control digital (LED ON/OFF) y adquisición de señales analógicas.

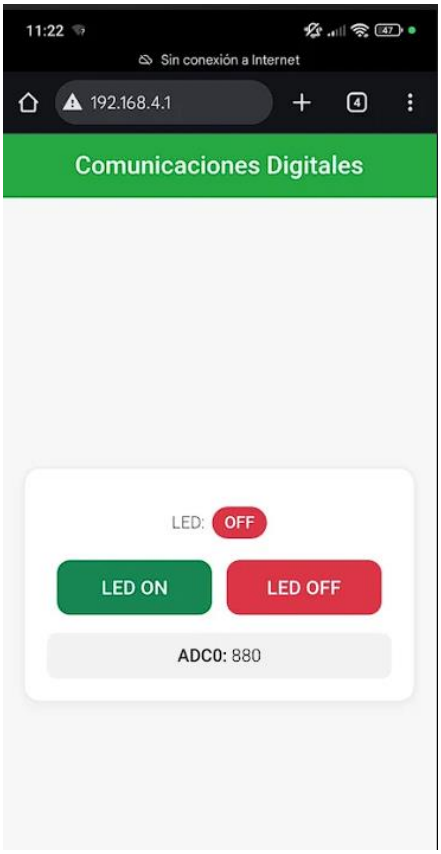

Ilustración. 11. Pagina Web con el ADC 0 incluido.

- Analisis de la variación del RSSI respecto a la distancia.

Con el fin de analizar el comportamiento de la potencia de señal recibida (RSSI) en función de la distancia, se diseñó un código en MicroPython que permite realizar mediciones automáticas del nivel de señal de un punto de acceso determinado. El código, mostrado en la Figura 2, se ejecutó en la Raspberry Pi Pico 2W configurada en modo estación (STA_IF).

En el script se definieron una lista de distancias medidas entre 1 y 10 metros, y para cada punto se realizaron 10 mediciones consecutivas del RSSI utilizando el método wlan.status('rssi'). Los valores obtenidos se promediaron y se almacenaron en un archivo CSV (rssi.csv) dentro del sistema de archivos del microcontrolador, registrando la distancia y el valor promedio en decibelios-milivatios (dBm).

El proceso permitió observar que, a medida que aumenta la distancia entre el punto de acceso (AP) —en este caso un teléfono celular configurado como hotspot— y el receptor (Pico 2W), el valor del RSSI disminuye de forma casi logarítmica, en concordancia con el modelo de pérdida de trayectoria de propagación en espacio libre.

A partir de los datos registrados, se generó la gráfica del RSSI frente a la distancia, obteniéndose una curva descendente cuyo comportamiento puede aproximarse mediante la ecuación:

$$RSSI(d) = RSSI_0 - 10n\log_{10}(d)$$

Ilustración. 12. Ecuación .

donde RSSI0 es la potencia recibida a un metro y n es el exponente de pérdida de trayectoria (típicamente entre 2 y 4 en interiores).

El alcance máximo medido fue el punto donde el RSSI descendió por debajo de -85 dBm, valor a partir del cual se observó inestabilidad en la conexión. Este experimento permitió relacionar los principios teóricos de propagación electromagnética con la práctica de medición de señales WiFi, validando la influencia de la distancia y los obstáculos sobre la calidad de la comunicación inalámbrica.

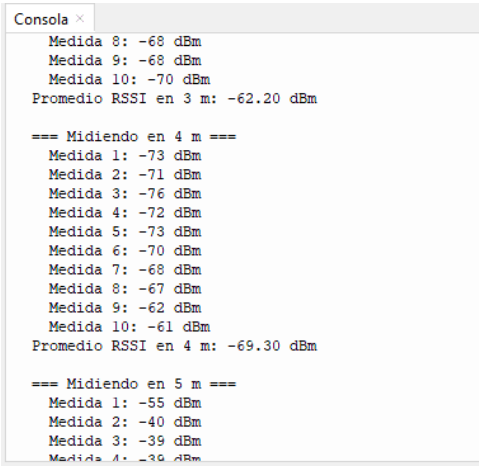

Ilustración. 13. Medidas tomadas con su respectiva potencia.

Posterior a la ejecución del programa, se descargó el archivo rssi.csv generado por la Raspberry Pi Pico 2W. En dicho archivo se registraron los valores promedio de RSSI (en dBm) correspondientes a cada una de las distancias medidas entre 1 y 10 metros.

	A	B
1	Distancia_m	RSSI_dBm
2	1	-41.8
3	2	-48.9
4	3	-62.2
5	4	-69.3
6	5	-42.1
7	6	-60.5
8	7	-58.8
9	8	-63.1
10	9	-64.7
11	10	-52.1

Ilustración. 14. Promedio de cada medida en su respectiva distancia.

(Todas las evidencias completas estan adjuntas al repositorio de GitHub, tanto el excel como los codigos python

y html.)

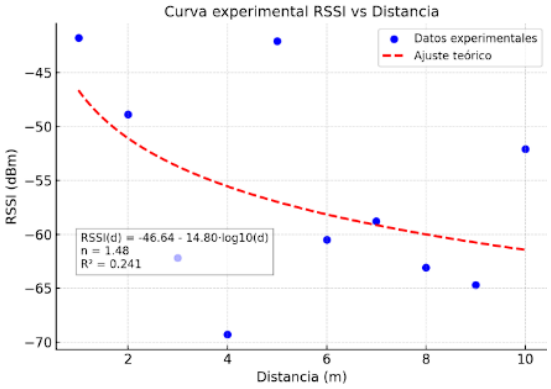

Ilustración. 15. Grafica del RSSI teórico y experimental.

CONCLUSIONES

[1] La práctica permitió comprender el funcionamiento del módulo WiFi integrado en la Raspberry Pi Pico 2W, logrando establecer y analizar conexiones inalámbricas tanto en modo estación (STA) como en punto de acceso (AP) mediante el uso de MicroPython.

[2] El escaneo de redes demostró la variabilidad del RSSI en función del entorno y la interferencia entre canales, resaltando la importancia de emplear canales no solapados (1, 6 y 11) para optimizar la calidad de la comunicación.

[3] La configuración del servidor web embebido permitió implementar un sistema de control remoto del LED y de monitoreo analógico (ADC), evidenciando la capacidad del Pico 2W para ejecutar tareas simultáneas de adquisición, procesamiento y transmisión de datos.

[4] Las pruebas de cambio de canal confirmaron que la selección adecuada de frecuencia minimiza interferencias, mejorando la estabilidad del enlace inalámbrico y validando las observaciones realizadas con herramientas externas como WiFi Analyzer.

[5] El análisis del RSSI en función de la distancia mostró una disminución logarítmica de la potencia recibida, coherente con el modelo de propagación en espacio libre, permitiendo estimar experimentalmente el alcance máximo y el exponente de pérdida del entorno.

REFERENCIAS

[1] MicroPython, machine — functions related to the hardware, Documentación oficial, [En línea]. Disponible en: https://docs.micropython.org/en/latest/library/machine.html. [Accedido: 14-ago-2025].

[2] Raspberry Pi Foundation, Getting started with MicroPython on Raspberry Pi Pico, Documentación oficial, [En línea]. Disponible en: https://www.raspberrypi.com/documentation/microcontrollers/micropython.html. [Accedido: 14-ago-2025].

[3] MathWorks, *Communications Toolbox User's Guide*, MATLAB R2008a Documentation. Disponible en: https://www.mathworks.com/help/com

[4] ANSI/TIA, TIA-232-F: Interface Between Data Terminal Equipment and Data Circuit-Terminating Equipment Employing Serial Binary Data Interchange, Telecommunications Industry Association, 1997.

[5] A. S. Tanenbaum and D. Wetherall, Computer Networks, 5th ed. Boston, MA, USA: Pearson, 2011.

[6] Vasquez Rojas, D. (s.f.). *dylanrojas04 - Overview*. GitHub. https://github.com/dylanrojas04

[7] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, Jan. 1962.

[8] J. Rugeles, Repositorio de prácticas de comunicación digital – I²C con Raspberry Pi Pico, Universidad Militar Nueva Granada. [En línea]. Disponible en: https://github.com/jrugeles/I2C