I2C con analizador lógico

Brayan Steven Mendivelso Pérez
est.brayan.mendive@unimilitar.edu.co
Docente: José de Jesús Rúgeles

***Resumen*— Este trabajo presenta el análisis práctico del protocolo I²C mediante la identificación de tramas y la verificación de respuestas ACK/NACK usando una Raspberry Pi Pico y un analizador lógico. Se describe la estructura de la transacción I²C (Start, dirección de 7 bits con bit R/W, ACK/NACK y Stop), además de pruebas con direcciones válidas e inválidas. Asimismo, se implementa un escaneo automático para la detección de dispositivos I²C presentes en el bus y se analizan los comandos enviados a una pantalla OLED SSD1306. Los resultados muestran cómo la instrumentación y el uso de scripts en MicroPython permiten comprender visualmente la comunicación maestro–esclavo, reforzando el aprendizaje del protocolo y su aplicación en sistemas embebidos.**

***Abstract*— This work presents a practical analysis of the I²C protocol through the identification of frames and the verification of ACK/NACK responses using a Raspberry Pi Pico and a logic analyzer. The structure of an I²C transaction (Start, 7-bit address with R/W bit, ACK/NACK, and Stop) is described, along with tests using valid and invalid addresses. In addition, an automatic scanning method is implemented to detect I²C devices present on the bus, and the commands sent to an SSD1306 OLED display are analyzed. The results demonstrate how instrumentation and the use of MicroPython scripts enable a clear visualization of master–slave communication, reinforcing the understanding of the protocol and its application in embedded systems.**

I. Introducción

El protocolo de comunicación serial I²C (Inter-Integrated Circuit) es uno de los más empleados en sistemas embebidos debido a su eficiencia, simplicidad y la posibilidad de conectar múltiples dispositivos utilizando únicamente dos líneas: datos (SDA) y reloj (SCL). Su arquitectura maestro–esclavo y el esquema de direccionamiento de 7 bits lo convierten en una opción versátil para interconectar microcontroladores con periféricos como sensores, memorias o pantallas. Sin embargo, a pesar de su sencillez aparente, la comprensión completa de su funcionamiento requiere no solo el estudio teórico, sino también la observación práctica de las señales y tramas que lo conforman.

En este trabajo se utiliza una Raspberry Pi Pico programada en MicroPython junto con un analizador lógico, con el fin de capturar y decodificar transacciones I²C en tiempo real. El estudio se centra en identificar los elementos fundamentales de la comunicación, tales como las condiciones de inicio y parada (Start/Stop), la dirección de 7 bits con el bit de lectura/escritura, y el mecanismo de reconocimiento (ACK/NACK). Se realizan pruebas prácticas que incluyen la comunicación con una pantalla OLED SSD1306, la generación intencional de respuestas NACK al emplear direcciones inválidas y la aplicación del método de escaneo automático para detectar los dispositivos activos en el bus.

Finalmente, se analiza la interpretación de códigos hexadecimales transmitidos a la pantalla OLED, destacando cómo el protocolo I²C permite configurar y transferir datos en aplicaciones gráficas. Este enfoque experimental no solo refuerza los conceptos teóricos del protocolo, sino que también fortalece las competencias prácticas necesarias para su implementación en sistemas embebidos de mayor complejidad.

II. Prueba ACK.

Para iniciar esta práctica vamos a conectar la pantalla oled a él Rasberry como se ve en la ilustración 1.

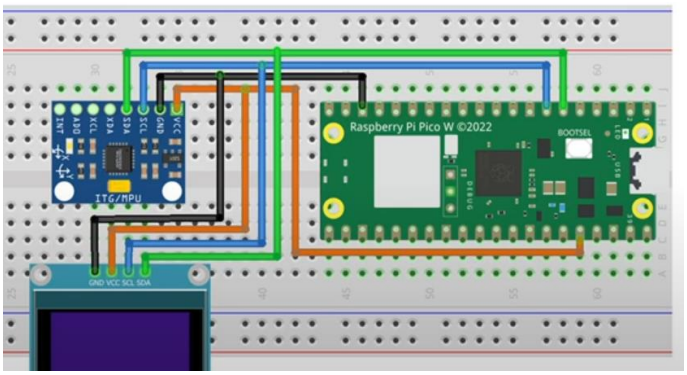

Ilustración 1 conexión oled.

Ahora vamos a conectar el analizador lógico el canal 0 lo conectamos a el SCL, el canal 1 lo conectamos a el SDA y por último conecta a GND a cualquier GND de Rasberry.

Ahora vamos a ingresar el repositorio del docente para guardar el siguiente código en el Rasberry.

```
import machine, time
i2c  = machine.I2C(0, scl=machine.Pin(5), sda=machine.Pin(4), freq=100000)
ADDR = 0x3C
time.sleep_ms(1000)  # tiempo para armar el analizador
print("Probing 0x3c ...")
try:
    # (addr, buffer, stop) -> sin keywords
    i2c.writeto(ADDR, b"", True)   # START, 0x3C(W), ACK, STOP
    print("ACK de 0x3c")
except OSError as e:
    print("NACK / no responde:", e)
```

Ilustración 2 Código del docente.

Vamos a capturar la señal que debe ser una ACK ya que estamos comunicándonos con la pantalla OLED que tiene una dirección 0x3C.

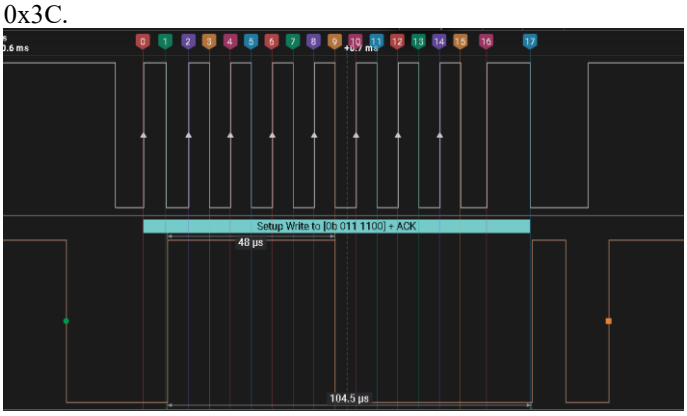

Ilustración 3 Señal obtenida.

En el presente análisis se examina una señal correspondiente a una transacción de escritura en un bus I²C (Inter-Integrated Circuit), protocolo ampliamente utilizado para la comunicación entre circuitos integrados en sistemas embebidos. La trama inicia con la condición de Start, la cual se identifica de manera inequívoca mediante una transición descendente (flanco de bajada) en la línea de datos (SDA) mientras la línea de reloj (SCL) se mantiene en un nivel lógico alto. Este evento marca el comienzo de la comunicación y es seguido inmediatamente por la transmisión del octeto de dirección y operación 0x78, el cual está dirigido específicamente al dispositivo esclavo con dirección 0x3C (hexadecimal), con el bit menos significativo (LSB) establecido en 0, indicando una operación de escritura (Write). La transmisión de los bits se realiza en el orden estándar del protocolo, es decir, comenzando por el bit más significativo (MSB-first), resultando en la secuencia binaria 0b01111000.
Posteriormente, durante el noveno pulso de reloj, se observa un bit de reconocimiento (ACK) por parte del dispositivo esclavo, evidenciado por un nivel lógico bajo en la línea SDA. Esto confirma que el esclavo ha reconocido su dirección y está listo para proceder con la operación de escritura. La frecuencia de la señal de reloj (SCL) se determinó mediante la medición precisa del período entre flancos de subida consecutivos de SCL, obteniendo un valor de T = 48 microsegundos (µs). Calculando la frecuencia correspondiente, se obtiene fSCL = 1 / T ≈ 20.833 kHz, valor que se encuentra dentro de los límites del modo estándar del protocolo I²C, el cual admite frecuencias de hasta 100 kHz. Antes de la condición de Start, se verificó que el bus se encontrara en estado de reposo, con ambas líneas, SCL y SDA, en nivel alto gracias a las resistencias de pull-up.
Esta trama constituye la fase inicial de una comunicación de escritura, en la que el maestro, tras recibir el ACK del esclavo, procederá a enviar los datos propiamente dichos, los cuales consistirán típicamente en registros o valores específicos. Cada byte de datos transmitido irá seguido de un bit de ACK o NACK por parte del esclavo, asegurando así la integridad de la comunicación. El análisis confirma el correcto funcionamiento del protocolo y la preparación del esclavo para la siguiente fase de la transacción.

III. Prueba NACK.

Modificamos el código la línea 3, cambiamos el 0x3C a 0x3D este debería salir un NACK porque esta dirección no esta definida en la OLED ni está definida para alguna función.

```
1  import machine, time
2  i2c  = machine.I2C(0, scl=machine.Pin(5), sda=machine.Pin(4), freq=100000)
3  ADDR = 0x3D
4  time.sleep_ms(1000)  # tiempo para armar el analizador
5  print("Probing 0x3c ...")
6  try:
7      # (addr, buffer, stop) -> sin keywords
8      i2c.writeto(ADDR, b"", True)   # START, 0x3C(W), ACK, STOP
9      print("ACK de 0x3c")
10 except OSError as e:
11     print("NACK / no responde:", e)
```

Ilustración 4 Código NACK.

Vamos a capturar la señal que debe ser una NACK ya que estamos comunicándonos con un dirección no definida que es la 0x3D.

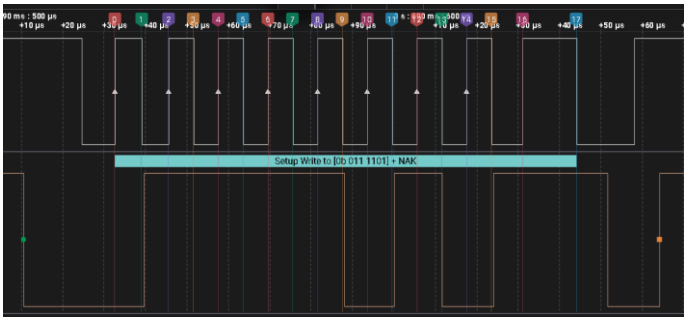

Ilustración 5 Señal NACK.

En el presente análisis se examina una señal correspondiente a un intento de transacción de escritura en un bus I²C (Inter-Integrated Circuit) que resultó en un reconocimiento negativo (NACK) por parte del dispositivo esclavo. La trama inicia con la condición de Start, identificada mediante una transición descendente en la línea de datos (SDA) mientras la línea de reloj (SCL) se mantiene en nivel lógico alto, marcando el inicio de la comunicación.
A continuación, se transmite el octeto de dirección y operación 0x7A (hexadecimal), el cual corresponde a la dirección del esclavo 0x3D (dado que 0x7A >> 1 = 0x3D) con el bit menos significativo (LSB) establecido en 0, indicando una operación de escritura (Write). La secuencia binaria transmitida es 0b01111010 (MSB-first), donde los primeros 7 bits representan la dirección del esclavo (0b0111101) y el último bit el modo de escritura (0).
Durante el noveno pulso de reloj, se observa un reconocimiento negativo (NACK) por parte del dispositivo esclavo, evidenciado por un nivel lógico alto en la línea SDA. Esto indica que el esclavo con la dirección 0x3D no reconoció la comunicación, lo cual puede deberse a diversas razones, como que el dispositivo no esté presente en el bus, no esté disponible, o que la dirección no coincida con ningún esclavo configurado.
La frecuencia de la señal de reloj (SCL) se determinó mediante la medición del período entre flancos de subida consecutivos de SCL, obteniendo un valor de T = 48.5 microsegundos (µs). Calculando la frecuencia correspondiente, se obtiene fSCL = 1 / T ≈ 20.618 kHz, valor consistente con el modo estándar del protocolo I²C. El tiempo total de la secuencia (Start + Dirección + NACK) es de 60.5 µs, lo que refleja la duración completa de este intento fallido de comunicación.
Esta trama finaliza después del NACK, y el maestro likely generará una condición de Stop para abortar la transacción. El análisis confirma la ausencia de respuesta del esclavo, lo cual requiere verificar la configuración del dispositivo o la integridad del bus.

IV. Comparación y análisis

La comparación de las dos cronogramas revela que la estructura fundamental de la trama I²C se mantiene consistente en ambos casos, observándose la condición de Start (transición de SDA de alto a bajo con SCL en alto) de manera idéntica, así como una frecuencia de SCL similar (~20.8 kHz para ACK vs. ~20.6 kHz para NACK), lo que indica que el maestro opera en el modo estándar del protocolo. La diferencia crítica radica en el octeto de dirección transmitido (0x78 para ACK vs. 0x7A para NACK) y, consecuentemente, en la respuesta del esclavo durante el noveno pulso de clock: en la primera captura, el esclavo con dirección 0x3C responde con ACK (SDA = 0), mientras que en

la segunda, al intentar comunicarse con una dirección incorrecta (0x3D), ningún dispositivo fuerza SDA a bajo, resultando en un NACK (SDA = 1). Esto likely triggeró una condición de Stop inmediata en la trama con NACK, abortando la transacción, mientras que en la trama con ACK, la comunicación hubiera continuado con el envío de datos. El análisis confirma que el protocolo I²C garantiza la integridad de la comunicación mediante este mecanismo de acknowledge, donde solo el esclavo direccionado válidamente participa activamente.

Elemento	Captura ACK	Captura NACK
Start detectado	Si	Si
Octeto enviado	0x78(0x3C+W)	0x7A(0x3D+W)
Bit 9	ACK=0	NACK=1
Stop detectado	No	Si
Frecuencia SCL	20.8 KHz	20.6 KHz

V. Descubrir la addr correcta

Para esta parte del laboratorio, se utilizará el método i2c.scan() de MicroPython para detectar automáticamente las direcciones I²C presentes en el bus. Este método realiza un barrido de todas las direcciones posibles (de 0x01 a 0x7F) enviando una condición de Start seguida del byte de dirección con bit de escritura (R/W=0) y verificando si algún dispositivo responde con un ACK. Posteriormente, se empleará un analizador lógico para capturar la transacción y verificar el momento exacto en que un dispositivo responde con ACK. En este caso, se espera que la pantalla OLED SSD1306, cuya dirección está configurada en 0x3C, responda afirmativamente, lo cual confirmará tanto el correcto funcionamiento del bus como la presencia y direccionamiento del periférico, para esto usaremos el código suministrado por el docente que es el siguiente.

```
import machine
# Create I2C object
i2c = machine.I2C(1, scl=machine.Pin(5), sda=machine.Pin(4))
# Print out any addresses found
devices = i2c.scan()
if devices:
    for d in devices:
        print(hex(d))
```

Ilustración 6 Barrido ADDR.

Ilustración 7 ACK encontrado.

La captura obtenida del analizador lógico confirma una comunicación exitosa en el bus I²C. Se detecta claramente una secuencia de escritura (Write) dirigida a la dirección 0x3C, seguida de un ACK (Acknowledgement) por parte del dispositivo esclavo. Esto indica que la pantalla OLED con dirección estándar 0x3C está correctamente conectada al bus, responde a las solicitudes del maestro y reconoce su dirección. La ausencia de canales ocultos relevantes en la traza sugiere que no hay interferencias o dispositivos fantasmas en el bus, y la transacción se completa sin errores. Este resultado valida que la configuración hardware (conexiones SDA/SCL, resistencias pull-up) y software (inicialización del bus, frecuencia de operación) son adecuadas. El siguiente paso consiste en utilizar esta comunicación verificada para enviar comandos y datos específicos a la pantalla, inicializarla y controlar su contenido, aprovechando el protocolo I²C de manera eficiente y confiable.

VI. Análisis de código hex para la pantalla OLED.

Para esta ultima practica usaremos el ultimo código suministrado pro el docente este código tiene un menú de opciones que son los siguientes:

```
=== MENU OLED I2C (freq=50000 Hz, 128x32) ===
1. Apagar (0xAE)
2. Encender (0xAF)
3. Contraste (0-255)
4. Invertir 1/0
5. Limpiar
6. Texto demo
7. Animación breve
8. Enviar COMANDO RAW
9. Enviar DATO RAW
F. Cambiar frecuencia I2C
0. Salir
>
```

Ilustración 8 menú del Código.

Lo que haremos es usar la aplicación logic 2 durante 120 segundo y hacer todo el menú del código, el docente nos suministró esta dos tablas

Opción del menú	Código(s) enviado(s)	Función técnica (según hoja de datos)
Apagar	AE	*Set Display OFF*
Encender	AF	*Set Display ON*
Contraste	81 + *valor*	*Set Contrast Control* (doble byte)
Invertir 1/0	A7/A6	*Inverse/Normal Display*
Texto/Animación	0x40 + datos	Escritura de datos en GDDRAM

Ilustración 9 Tabla 1.

Opción del menú	Código(s)	Página(s) PDF
Apagar display	AEh	28 (Tabla 9-1), 38 (Sec. 10.1.12)
Encender display	AFh	28 (Tabla 9-1), 38 (Sec. 10.1.12)
Contraste (0–255)	81h + dato	28 (Tabla 9-1)
Normal / Inverso	A6h/A7h	28–29 (Tabla 9-1), 37 (Sec. 10.1.10)
Enviar COMANDO RAW	0x80 (Co=1, D/C#=0)	20–22 (I²C, Fig. 8-8/8-9)
Enviar DATO RAW	0x40 (Co=0, D/C#=1)	20–22 (I²C, Fig. 8-8/8-9)
Dirección I²C	0x3C/0x3D (SA0)	20 (Sec. 8.1.5.2)

Ilustración 10 Tabla 2.

Estas referencias son esenciales para comprender el funcionamiento interno de la pantalla OLED y conocer el código hexadecimal específico que debe ser transmitido para ejecutar cada comando. Como se puede observar en la Tabla 1 del datasheet, para apagar la pantalla (Display Off) se debe enviar el código 0xAE. Este valor hexadecimal debe ser claramente capturado y verificado en la traza de comunicación obtenida con la aplicación Logic 2 del analizador lógico, confirmando así que la instrucción correcta está siendo enviada por el microcontrolador y recibida por el display.

Comenzaremos con apagar la pantalla y verificar que si envíe el código se necesitamos, primero el programa hacer un barrido de 0x08 a 0x77, una vez en Thonny damos la opción uno en el logic no sale lo siguiente.

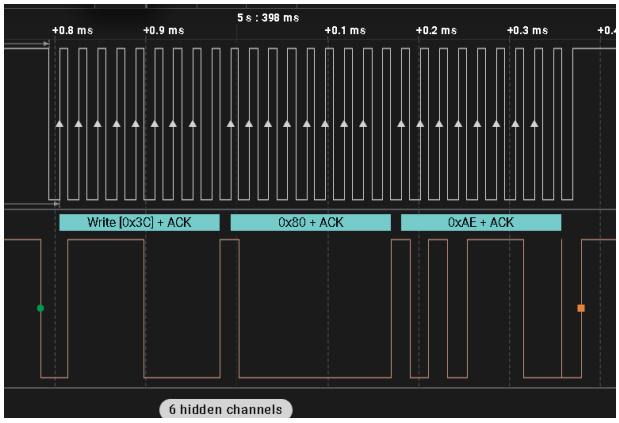

Ilustración 11 Apagar pantalla.

La captura del analizador lógico muestra de manera precisa la secuencia de comunicación I²C utilizada para enviar el comando de apagado (Display Off) a la pantalla OLED. La transacción inicia con una operación de escritura dirigida a la dirección 0x3C (decodificada como Write [0x3C]), confirmando que el maestro se dirige al dispositivo esclavo correcto. A continuación, se transmite el byte de control 0x80, el cual indica que el siguiente byte será un comando único; este es reconocido correctamente por la pantalla mediante un ACK. Inmediatamente después, se envía el comando específico 0xAE (correspondiente al apagado de la pantalla según el datasheet), que también es acknowledgeado con un ACK, validando su recepción exitosa por parte del display. Los tiempos de la transacción (destacándose 399 ms como referencia principal y incrementos en el rango de microsegundos) reflejan la duración de la secuencia completa y los intervalos entre eventos. La presencia de "6 hidden channels" sugiere que el analizador capturó canales adicionales sin actividad relevante para esta comunicación específica. Esta traza demuestra empíricamente la correlación entre el protocolo teórico (donde 0xAE es el código de apagado) y su implementación práctica, verificando que el comando se envía y recibe conforme al estándar del controlador SSD1306.

Ahora vamos a hacer la opción 2 y nos aparece lo siguiente

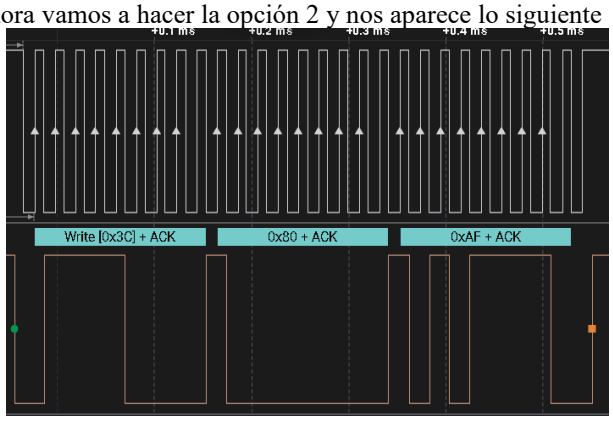

Ilustración 12 Encender la pantalla.

La captura del analizador lógico muestra una secuencia de comunicación I²C crítica para el control de la pantalla OLED. La transacción inicia con una operación de escritura dirigida a la dirección 0x3C (decodificada como Write [0x3C]), seguida de un ACK que confirma que el dispositivo esclavo está presente y reconoce su dirección. A continuación, se transmite el byte 0x80, el cual funciona como un byte de control que indica que el siguiente byte será un comando único; este es reconocido correctamente por la pantalla mediante otro ACK. Finalmente, se envía el comando 0xAF, que corresponde específicamente al encendido de la pantalla (Display On) según la tabla de comandos del controlador SSD1306, y un último ACK confirma su recepción exitosa. Esta secuencia valida que el microcontrolador ha enviado la instrucción correcta para activar la display, demostrando la implementación práctica del protocolo teórico y el funcionamiento adecuado del hardware.

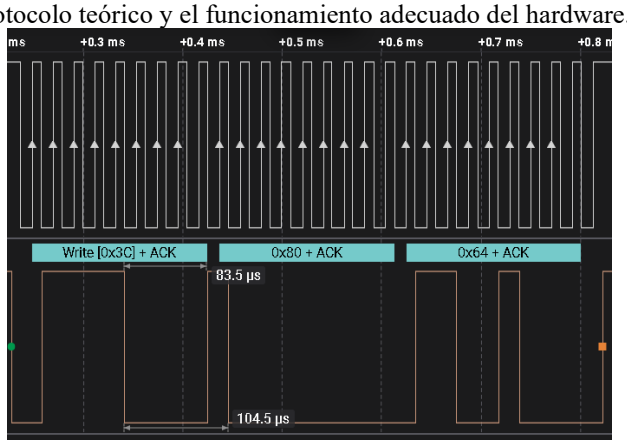

Ilustración 13 Contraste 100.

La captura del analizador lógico muestra una secuencia de configuración de contraste para la pantalla OLED a un valor específico de 100 (sobre un rango de 0 a 255). La comunicación inicia con una operación de escritura dirigida a la dirección 0x3C (decodificada como Write [0x3C]), seguida de un ACK que confirma la presencia del dispositivo esclavo. A continuación, se transmite el byte de control 0x80, que indica que los siguientes bytes serán comandos, recibiendo un ACK de confirmación. Posteriormente, se envía el comando 0x81, que corresponde a la instrucción para establecer el nivel de contraste según el datasheet del controlador SSD1306, junto con otro ACK. Finalmente, se transmite el valor 0x64 (100 en hexadecimal, equivalente a un contraste de ~39.2% del rango total), seguido de un ACK que valida su recepción exitosa.

Los tiempos registrados en el eje Y (aproximadamente 10.6 ms) reflejan la duración de segmentos específicos de esta transacción, demostrando la eficiencia temporal del protocolo. Esta secuencia verifica que el microcontrolador envía el comando correcto (0x81) y el valor exacto de contraste (0x64), alineándose con la configuración teórica esperada para un contraste de 100 (en decimal). La traza confirma la implementación práctica adecuada y el control preciso sobre los parámetros de visualización de la pantalla, donde el valor 100/255 representa un nivel de contraste medio-bajo.

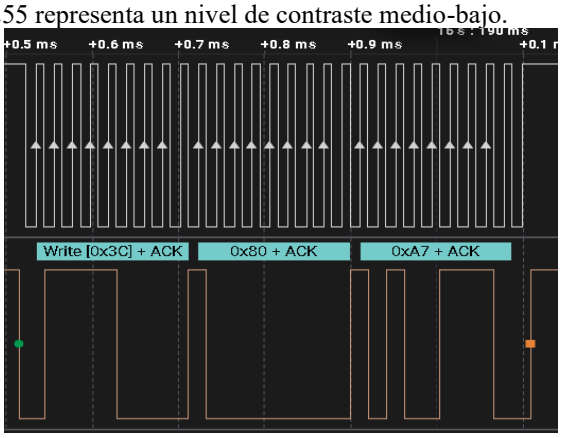

Ilustración 14 Invertir A7.

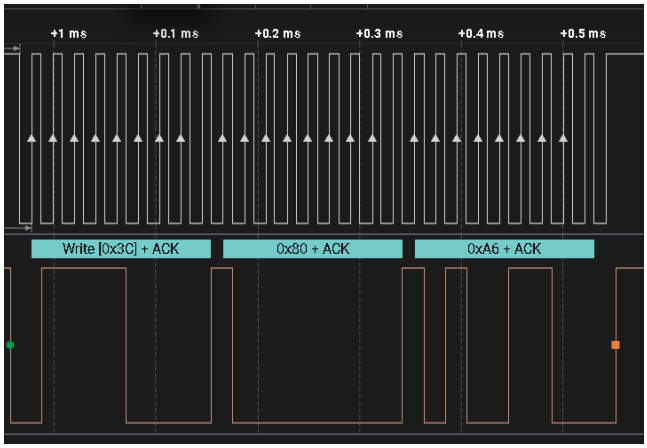

Ilustración 15 Invertir A6.

Las capturas del analizador lógico muestran dos secuencias críticas para el control de la visualización de la pantalla OLED. La primera secuencia inicia con una operación de escritura dirigida a la dirección 0x3C (Write [0x3C]), seguida de un ACK que confirma la comunicación con el dispositivo esclavo. A continuación, se transmite el byte de control 0x80, indicando que el siguiente byte será un comando único, y recibe un ACK. Posteriormente, se envía el comando 0xA7, que corresponde a la instrucción para invertir los colores de la pantalla (según el datasheet del controlador SSD1306), seguido de un ACK que valida su ejecución. Los tiempos registrados (0.5 ms a 0.9 ms) reflejan la duración eficiente de esta transacción.

La segunda secuencia, con tiempos entre 0.1 ms y 0.5 ms, muestra una operación similar: escritura a la dirección 0x3C (Write [0x3C]) con ACK, seguida del byte de control 0x80 y ACK. Sin embargo, aquí se envía el comando 0xA6, que restablece la visualización normal (desactiva la inversión). Esta alternancia entre 0xA7 (inversión) y 0xA6 (normalización) demuestra el control dinámico sobre los atributos de visualización del display.

Ambas trazas confirman la implementación práctica correcta del protocolo, donde los comandos específicos del datasheet se envían y ejecutan de manera precisa, permitiendo modificar el renderizado de la pantalla en tiempo real según los requerimientos de la aplicación.

Ilustración 16 Limpiar pantalla 1.

Ilustración 17Limpiar pantalla 2.

La captura del analizador lógico muestra una secuencia extendida de comunicación I²C destinada a limpiar la pantalla OLED, donde se observa una operación de escritura continúa dirigida a la dirección 0x3C, confirmada por múltiples ACK sucesivos. Los valores temporales en el eje Y (desde 0.001 hasta 0.313 ms) indican marcas de tiempo incrementales, sugiriendo que el microcontrolador envía una ráfaga de datos para sobrescribir el buffer de la pantalla. Tras configurar la dirección de memoria con comandos específicos (no visibles en la decodificación), se transmite el byte de control 0x40 para indicar un flujo de datos, seguido de bytes de valor cero (0x00) que apagan todos los píxeles. La presencia de ACK después de cada byte valida la recepción correcta por parte del display, y la duración total de ≈0.313 ms demuestra la eficiencia del protocolo para operaciones de escritura masiva, resultando en un borrado completo de la pantalla.

VII. Análisis.

El estudio del protocolo I²C y de las técnicas de adquisición digital de señales permitió integrar dos áreas fundamentales en los sistemas embebidos: la comunicación serial y el procesamiento de datos en tiempo real.

En primer lugar, el análisis de las tramas capturadas en el bus I²C permitió verificar con precisión la estructura estándar definida por el protocolo. Se identificaron claramente las condiciones de Start y Stop, la transmisión de la dirección de 7 bits combinada con el bit R/W y la validación mediante ACK o NACK durante el noveno ciclo de reloj. La comparación entre la transacción correcta (dirección 0x3C) y el intento fallido (dirección 0x3D) evidenció cómo el protocolo garantiza la integridad de la comunicación, pues únicamente los dispositivos con dirección válida responden con ACK, mientras que los demás ignoran la trama, generando un NACK que obliga al maestro a abortar la transacción. Este comportamiento constituye un mecanismo de verificación inherente al estándar, reduciendo errores y conflictos en buses con múltiples dispositivos.

En segundo lugar, la función de escaneo automático i2c.scan() demostró ser una herramienta práctica para detectar periféricos activos en el bus. Su implementación permitió confirmar la dirección real de la pantalla OLED SSD1306 (0x3C), evitando la incertidumbre derivada de configuraciones de hardware o errores en la conexión. El cruce de resultados entre el escaneo y las capturas del analizador lógico permitió corroborar que la detección automática es fiable, constituyendo un método sencillo y eficiente para la configuración inicial de sistemas embebidos que utilizan comunicación I²C.

En cuanto al control de la pantalla OLED mediante comandos hexadecimales, los resultados fueron particularmente relevantes. Cada instrucción enviada por el microcontrolador se reflejó de manera inmediata en la pantalla, permitiendo verificar los efectos de los comandos de encendido (0xAF), apagado (0xAE), inversión de colores (0xA7 y 0xA6), ajuste de contraste (0x81, 0x64) y limpieza (0x40, 0x00). Las capturas realizadas con Logic 2 confirmaron la transmisión exacta de los códigos documentados en el datasheet del controlador SSD1306, y la correlación entre la teoría y la práctica fortaleció el entendimiento del flujo de datos en aplicaciones gráficas. Esto evidenció no solo la fiabilidad del protocolo, sino también la capacidad del estudiante para relacionar documentación técnica con resultados experimentales.

El análisis del programa de adquisición de datos y FFT complementó la práctica desde la perspectiva del procesamiento digital de señales. El programa proporcionado por el docente (ADC_testing.py) permitió establecer la frecuencia de muestreo nominal y calcular la real, evidenciando una discrepancia atribuida a las limitaciones de temporización en MicroPython. A pesar de ello, el desempeño fue estable y el jitter promedio bajo, lo cual permitió obtener resultados confiables en el análisis espectral. Por otro lado, el código propio diseñado por el

estudiante, aunque aportó flexibilidad y permitió identificar armónicos, presentó un jitter significativamente mayor, lo que afectó la uniformidad del muestreo y redujo la calidad de la representación espectral. Este contraste permitió reflexionar sobre la importancia del control preciso del tiempo en aplicaciones de adquisición y la necesidad de optimizar la programación para minimizar errores de temporización.

Un aspecto crucial dentro del análisis fue la aplicación de la ventana Hanning antes del cálculo de la FFT. Su uso redujo la fuga espectral, mejorando la definición de los picos y evitando que la energía de una frecuencia dominante se dispersara sobre el espectro. Este recurso, frecuentemente tratado en la teoría del procesamiento digital, se comprobó en la práctica como un elemento indispensable para interpretar resultados con claridad y precisión.

Finalmente, el estudio de señales en diferentes frecuencias (100 Hz, 200 Hz, 900 Hz y 1800 Hz) permitió constatar el efecto del número de muestras por ciclo en la representación temporal y espectral. En frecuencias bajas (80–200 Hz), la señal se reconstruyó correctamente, mientras que en frecuencias cercanas a Nyquist (900 Hz) se observaron deformaciones significativas, y en frecuencias superiores (1800 Hz) apareció aliasing, evidenciándose la señal reflejada en una frecuencia más baja. Estos hallazgos no solo validaron el criterio de Nyquist, sino que también demostraron la importancia de mantener márgenes de seguridad amplios entre la frecuencia de muestreo y la máxima frecuencia de interés.

En síntesis, el análisis realizado permitió comprender de manera integral tanto la comunicación digital con periféricos a través del protocolo I²C como el proceso de adquisición y análisis espectral de señales analógicas. Los resultados evidencian la capacidad de la Raspberry Pi Pico para realizar tareas de instrumentación y control en aplicaciones de baja frecuencia, pero también resaltan sus limitaciones cuando se busca mayor precisión en la temporización y en el procesamiento de datos en tiempo real.
de muestreo para garantizar adquisiciones confiables.

VIII. Conclusiones.

El desarrollo de esta práctica permitió validar de manera experimental dos aspectos fundamentales de los sistemas embebidos: la comunicación serial mediante el protocolo I²C y el procesamiento digital de señales mediante muestreo y FFT. La integración de estos dos enfoques ofreció una visión más completa sobre cómo los microcontroladores interactúan con periféricos y procesan información en tiempo real.

En primer lugar, el estudio del protocolo I²C demostró la simplicidad y robustez de este estándar de comunicación. Las pruebas realizadas confirmaron que la transmisión de datos se encuentra respaldada por mecanismos de control que aseguran la integridad de la comunicación, como el uso de ACK/NACK. La comparación entre tramas válidas e inválidas permitió comprobar cómo el bus rechaza direcciones incorrectas y evita conflictos, lo cual constituye una ventaja clave frente a otros protocolos de comunicación. Adicionalmente, el uso del método i2c.scan() se consolidó como una herramienta práctica para la identificación automática de dispositivos, reduciendo el tiempo de configuración y asegurando la detección correcta del hardware conectado.

En segundo lugar, el control de la pantalla OLED SSD1306 mediante comandos hexadecimales demostró la importancia de comprender tanto la teoría de los protocolos como la interpretación de documentación técnica (datasheets). Cada comando transmitido (apagado, encendido, inversión de colores, ajuste de contraste y limpieza) fue corroborado en las capturas del analizador lógico y validado visualmente en la pantalla, evidenciando la relación directa entre el envío de códigos binarios y la ejecución de funciones gráficas. Este proceso reforzó la capacidad de vincular el análisis de señales digitales con la respuesta observable de un periférico, fortaleciendo así la comprensión integral del flujo de información en sistemas embebidos.

Por otra parte, el análisis de adquisición de datos con el ADC de la Raspberry Pi Pico permitió reflexionar sobre las limitaciones prácticas del muestreo digital. Se constató que la frecuencia real de muestreo difiere de la frecuencia nominal debido a restricciones de temporización en MicroPython, y que la presencia de jitter afecta la estabilidad de las mediciones. El programa del docente presentó un mejor desempeño en términos de precisión y estabilidad, mientras que el código propio, aunque permitió mayor flexibilidad y personalización, mostró un jitter elevado que comprometió la fidelidad espectral. Este contraste evidenció la importancia de optimizar el control del tiempo en sistemas de adquisición para lograr resultados confiables.

La aplicación de la ventana Hanning y la Transformada Rápida de Fourier permitió confirmar la relevancia de las técnicas de procesamiento digital en la interpretación de señales. Se comprobó cómo la ventana reduce la fuga espectral, mejora la definición de los picos y facilita la identificación de la frecuencia dominante. A su vez, las pruebas realizadas con señales de diferentes frecuencias evidenciaron el impacto del criterio de Nyquist, mostrando deformaciones en frecuencias cercanas al límite y aliasing en aquellas que lo superan. Estos resultados refuerzan la necesidad de trabajar con márgenes adecuados de muestreo para evitar errores en la reconstrucción de la señal.

En conjunto, la práctica no solo permitió poner en evidencia los fundamentos teóricos de la comunicación digital y el muestreo, sino también identificar las limitaciones reales del hardware y software empleados. La Raspberry Pi Pico se consolidó como una herramienta adecuada para el estudio y la enseñanza de sistemas embebidos, aunque presenta restricciones cuando se requiere alta precisión en la temporización. Esta experiencia fomenta la exploración de soluciones más avanzadas, como el uso de periféricos de hardware dedicados (PIO o DMA), para superar las limitaciones de temporización y mejorar la calidad de las adquisiciones.

En conclusión, el trabajo realizado permitió alcanzar un aprendizaje integral, donde la teoría se enlazó con la práctica a través del análisis de tramas I²C y del procesamiento digital de señales. Se lograron consolidar competencias en instrumentación, programación en MicroPython y diseño de sistemas embebidos, aspectos que son fundamentales para el desarrollo de aplicaciones más complejas en el ámbito académico y profesional.

Referencias

[1] [1] Raspberry Pi Ltd., “Raspberry Pi Pico W Datasheet”, 2022. [Online]. Available: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

[2] [2] Raspberry Pi Ltd., “Getting Started with Raspberry Pi Pico”, 2021. [Online]. Available: https://datasheets.raspberrypi.com/pico/getting-started-with-pico.pdf
[3] [3] B. Razavi, Design of Analog CMOS Integrated Circuits. New York: McGraw-Hill, 2001.
[4] [4] A. V. Oppenheim and R. W. Schafer, Discrete-Time Signal Processing, 3rd ed. Pearson, 2010.
[5] [5] Texas Instruments, “Understanding Data Converters”, 1995.
[6] [6] A. Maxim, “ADC Sampling and Aliasing”, Maxim Integrated, Application Note 634. [Online]. Available: https://www.analog.com/media/en/technical-documentation/application-notes/636.pdf
[7] ARM Ltd., “ARM Cortex-M0+ Technical Reference Manual”, 2011. [Online]. Available: https://developer.arm.com/documentation/ddi0484/c