Reconocimiento de tramas I^2C con Analizador Lógico

jose.rugeles@unimilitar.edu.co

Propósito

Identificar visualmente la estructura de una transacción I^2C (Start, dirección 7 bits + R/W, bit de ACK/NACK y Stop) usando una Raspberry Pi Pico (RP2040) con MicroPython y un analizador lógico (Logic 2).

1. ¿Cómo identificar los bits de la trama I^2C?

1. Conexiones y preparación

- **Plataforma:** Raspberry Pi Pico / Pico W (RP2040) con MicroPython.
- **Dispositivo I^2C bajo prueba:** Pantalla OLED SSD1306 (`ADDR=0x3C`).
- **Pines usados en el script:** I^2C1 → `SCL=GP15`, `SDA=GP14`.
- **Analizador lógico:** Conecte `CH0` a `SCL` y `CH1` a `SDA`. Une **GND** del analizador con **GND** de la placa.
- **Frecuencia de muestreo:** configure el analizador a $\geq 10\times$ la frecuencia del bus (p.ej., si el bus está a 100 kHz, muestrear a 1 MS/s o más).
- Active el **decodificador I^2C** en Logic 2 para ver etiquetas como *Start*, *Write to 0x3C*, *ACK*, *Stop*.

2. Elementos de la trama y cómo verlos en la señal

1. **Bus en reposo (IDLE):** SCL y SDA en alto (pull-ups).
2. **Bit de Start:** transición **SDA: alto→bajo** *mientras* SCL se mantiene alto. Marca el inicio de la transacción.
3. **Dirección (7 bits) + R/W:** se transmiten 8 pulsos de SCL. El maestro establece cada bit en SDA antes del flanco de subida de SCL.
   ADDR = $0x3C$ = `0b 011 1100` (MSB primero) y R/W = 0 indica escritura. El octeto completo enviado es `0x78` = $(0x3C \ll 1) \,|\, 0$.
4. **ACK/NACK (9. bit):** en el noveno pulso, el maestro *libera* SDA.
   - **ACK** → el esclavo fuerza SDA a `0`.
   - **NACK** → SDA queda en `1` (alto).

5. **Bit de Stop:** transición **SDA: bajo→alto** *mientras* SCL está alto. Finaliza la transacción.

2. Pruebas ACK - NACK

A. Código de prueba (ACK esperado)

Cargue y ejecute el siguiente script en la Pico (MicroPython). El código está disponible en la URL: https://github.com/jrugeles/I2C

Listing 1: i2c_ping_0x3c_min.py

```
# i2c_ping_0x3c_min.py  (MicroPython RP2040)
import machine, time

i2c  = machine.I2C(1, scl=machine.Pin(15), sda=machine.Pin(14), 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)
    # Si tu firmware no permite buffer vacío, descomenta la siguiente línea:
    # i2c.writeto(ADDR, b"\x00", True)  # ver s 0x3C(W), ACK, 0x00, ACK, STOP
```

1. Configure la adquisición en Logic 2 (decodificador I^2C habilitado).
2. Inicie captura y, dentro del segundo de espera del script, presione *Run*.
3. Identifique visualmente: *Start* → octeto (0x78) → *ACK* → *Stop*.
4. Con cursores, mida f_{SCL} y anote el valor.
5. **Guarde** una captura con anotaciones (flechas o notas sobre Start, bits de dirección, R/W, ACK y Stop).

B. Provocar un NACK (dirección incorrecta)

Modifique el valor de `ADDR` a una dirección no presente, por ejemplo `0x3D`:

```
ADDR = 0x3D  # direccion inexistente para provocar NACK
```

Tarea 2 (captura con NACK):

1. Repita el procedimiento de captura.

2. En el 9. bit verifique que SDA permanece alto (**NACK**).
3. **Guarde** la captura con una nota que diga "NACK en bit 9".

C. Comparación y análisis

1. Compare las dos cronogramas (ACK vs. NACK). ¿Qué cambia y qué se mantiene?
2. Complete la tabla:

Elemento	Captura ACK	Captura NACK
Start detectado (SDA↓ con SCL alto)		
Octeto enviado (0x78) indicado por el decodificador		
Bit 9 (ACK=0 / NACK=1)		
Stop detectado (SDA↑ con SCL alto)		
Frecuencia medida de SCL	_____	_____

3. Explique por qué con dirección incorrecta el esclavo no responde con ACK.

3. Prueba: descubrir la `ADDR` correcta

Objetivo

Usar el método `i2c.scan()` de MicroPython para detectar automáticamente la(s) dirección(es) I^2C presentes en el bus y verificar, con el analizador lógico, el momento en que un dispositivo responde con **ACK** (por ejemplo, la OLED en `0x3C`).

Código de escaneo

Listing 2: scan_i2c_addr.py

```
# Digital Communication UMNG
# jose.rugeles@unimilitar.edu.co
# SCAN I2C ADDR- Raspberry Pi Pico

import machine

# Create I2C object
i2c = machine.I2C(1, scl=machine.Pin(15), sda=machine.Pin(14))

# Print out any addresses found
devices = i2c.scan()

if devices:
    for d in devices:
        print(hex(d))
```

Qué hace el escaneo

- `i2c.scan()` recorre el rango de direcciones de 7 bits (`0x08`–`0x77`; `0x00`–`0x07` y `0x78`–`0x7F` están reservadas) y, para cada una, envía *START + (ADDR≪1 | W) + ACK? + STOP*.
- Devuelve una lista con las direcciones que respondieron con **ACK**. Si tu OLED está conectada, deberías ver `0x3c`.

Procedimiento

1. Conecte el analizador lógico: **CH0** a `SCL (GP15)`, **CH1** a `SDA (GP14)` y **GND** común. Activa el decodificador I^2C en Logic 2.
2. Configure la tasa de muestreo del analizador a $\geq 10 \times f_{\mathrm{SCL}}$ (p. ej., 1 MS/s para 100 kHz).
3. Inicie la captura en Logic 2 y ejecuta el script `scan_i2c_addr.py` en la Pico.
4. Observe en la consola las direcciones detectadas (en hexadecimal). Analice la(s) dirección(es).
5. En la traza del analizador, identifique:
   a) el **Start**,
   b) los intentos de *"Write to 0xXX"* (uno por dirección sondeada),
   c) el **ACK** para la dirección real (p. ej., *"Write to 0x3C ack"*),
   d) el **Stop**.

¿ Qué debería ver ?

- En consola: al menos `0x3c`. Si hay más dispositivos, aparecerán más direcciones.
- En Logic 2: muchos sondeos con *NACK* y, en la(s) dirección(es) válidas, **ACK** en el 9. bit.

4. Analisis de códigos hex para la pantalla OLED SSD1306

Objetivo

Analizar el proceso de comunicación entre el microcontrolador y la pantalla Oled mediante el protocolo I^2C, comprobando con el analizador lógico los bytes transmitidos (*control byte*, comando/datos y ACK).

Procedimiento

Ejecute el programa *OLED_ demo._ menu.py*. Utilice el analizador lógico para analizar los comandos enviados a la pantalla OLED para cada una de las opciones. En la tabla puede ver un resumen de algunos de los comandos empleados.

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

4.1. Referencia rápida: comandos del menú y páginas en la hoja de datos

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 (I^2C, Fig. 8-8/8-9)
Enviar DATO RAW	`0x40` (Co=0, D/C#=1)	20–22 (I^2C, Fig. 8-8/8-9)
Dirección I^2C	`0x3C/0x3D` (SA0)	20 (Sec. 8.1.5.2)