Ramirez Velásquez, Tomas Alejandro.
@tomasramirez20 (GitHub)

Análisis de Tramas I²C con Analizador Lógico usando Raspberry Pi Pico

https://github.com/tomasramirez20/Practica-6-com.digitales
Est.tomas.ramirez@unimilitar.edu.co

Resumen

Este informe presenta el análisis experimental de tramas del protocolo I²C mediante el uso de un analizador lógico (Logic 2) y una Raspberry Pi Pico programada en MicroPython. Se identificaron visualmente los elementos fundamentales de una transacción I²C: condición de Start, dirección de 7 bits + bit R/W, ACK/NACK y condición de Stop. Se realizaron pruebas con un dispositivo OLED SSD1306 (0x3C), se provocaron respuestas ACK y NACK, se escanearon direcciones en el bus y se analizó el envío de comandos y datos hacia la pantalla OLED. Los resultados confirman el correcto funcionamiento del protocolo y la utilidad del analizador lógico para depurar comunicaciones seriales.

1. Introducción

El protocolo I²C (Inter-Integrated Circuit) es ampliamente utilizado para la comunicación entre circuitos integrados en sistemas embebidos. Su simplicidad y estructura de dos hilos (SDA y SCL) lo hacen ideal para conectar múltiples dispositivos con un bajo número de pines. Sin embargo, la depuración de problemas de comunicación requiere herramientas que permitan visualizar las señales en tiempo real. En este laboratorio se utilizó un analizador lógico para capturar y decodificar tramas I²C generadas por una Raspberry Pi Pico, con el fin de validar el comportamiento del protocolo en diferentes escenarios.

2. Marco Teórico

UMNG

2.1. Estructura de la trama I²C

Una transacción I²C incluye los siguientes elementos:

- Start: Transición de SDA de alto a bajo mientras SCL está en alto.
- Dirección + R/W: 7 bits de dirección seguidos de 1 bit de lectura/escritura (0: escritura, 1: lectura).
- ACK/NACK: Noveno bit donde el esclavo confirma (ACK = 0) o no (NACK = 1) la recepción.
- Datos: Bytes transmitidos después de la dirección, cada uno seguido de un ACK/NACK.
- Stop: Transición de SDA de bajo a alto mientras SCL está en alto.

2.2. Direccionamiento I²C

Las direcciones I²C son de 7 bits, lo que permite hasta 112 dispositivos en el bus (16 direcciones están reservadas). La dirección se envía primero con el bit más significativo (MSB).

3. Procedimiento

Configuración Hardware

- Raspberry Pi Pico con MicroPython.
- Pantalla OLED SSD1306 (dirección 0x3C).
- Analizador lógico (Saleae Logic 2) conectado a SCL (GP15) y SDA (GP14).
- Frecuencia de muestreo: 1 MS/s (bus I²C a 100 kHz).

Antes de seguir con la pruebas realizadas, es necesario aclarar que para cada una de estas existe un codigo, el cual se corre dentro de la interfaz de thonny en la rasberry, luego de haber guardado en la memoria de la ya mencionada, las dos librerias necesarias para el par de esclavos, en este caso la pantalla OLED y el sensor.

Los codigos se pueden encontrar dentro del repositorio "https://github.com/tomasramirez20/Practica-6-com.digitales", al igual que los archivos .sal con y sin marcadores. Generados en Logic 2.

Pruebas Realizadas

1. **Transacción básica con ACK**: Envío de un comando vacío a la dirección 0x3C.

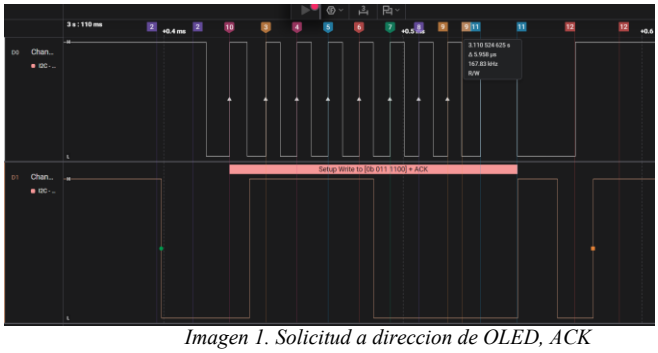

Imagen 1. Solicitud a direccion de OLED, ACK

En esta primera parte de la practica se provoca un "ACK", el cual confirma que la informacion esta siendo recibida, dentro de la imagen podemos reconocer la estrutura de I2C. Ya que solo se envia un byte de informacion seguido de un bit que confirma la escritura o lectura, tomando como referencia el maestro dentro de la comunicacion.

2. **Provocación de NACK**: Uso de una dirección incorrecta (0x3D).

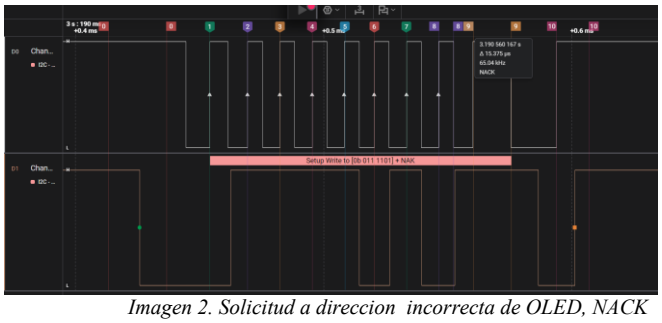

Imagen 2. Solicitud a direccion incorrecta de OLED, NACK

Anteriormente se confirma el envio del byte de informacion, ahora es necesario visualizar cuando el "NACK", el cual cumple la funcion inversa del ACK, esto lo provocamos al cambiar la direccion de destino "0x3c" por "0x3d". Dicho esto, se estima tambien que el bit de confirmacion (ACK o NACK) resulta en "1" cuando la direccion de destino en la comunicacion no existe.

3. **Escaneo de direcciones I²C**: Uso de `i2c.scan()` para detectar dispositivos.

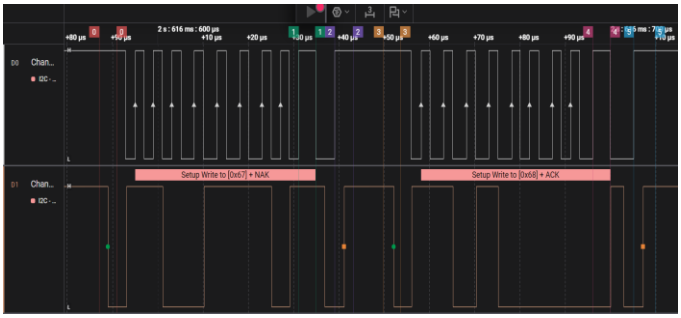

Imagen 3. Scaneo de direcciones

El scaneo envia solicitud a diferentes direcciones, cuando estas no estan asignadas a ninguna dispositivo esclavo dentro de la comunicacion, el maestro recibe luego del bit un NACK. Cuando reciba un ACK esa direccion de destino se imprime en la consola de la interfaz de Thonny, para este laboratorio como se encuentran dos dispositivos esclavos (sensor y pantalla OLED) , se imprimen dos direcciones.

En la imagen se muestra como luego de recibir un NACK una direccion que no esta asignada, recibe un ACK de la direccion de la pantalla OLED, la cual confirma que esta se le fue asignada.

4. **Análisis de comandos OLED**: Envío de comandos como encendido, apagado y contraste.

En este punto se procede a correr el codigo suministrado por el docente, el cual estable un menu dentro de la pantalla OLED, donde segun el comando que se envie resulta la visualizacion:

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

Imagen 4. Comandos para OLED

A diferencia con las pruebas anteriores, en esta se envia mayor informacion. Es decir, mayor numero de bytes esto puede conducir a una conclusion erronea, ya que anteriormente se presenta un byte el cual finaliza con el bit de lectura o escritura seguido por el de confirmacion. Como se estan enviando mas bytes, es logico pensar que por cada uno se presente la misma estructura. Pero, no es necesario ya que con solo poner el bit de lectura y escritura en el primer byte, se pueden agregar mas bytes con los bits de confirmacion y stop.

Para entender esto prosigamos con la captura de la opcion de animacion:

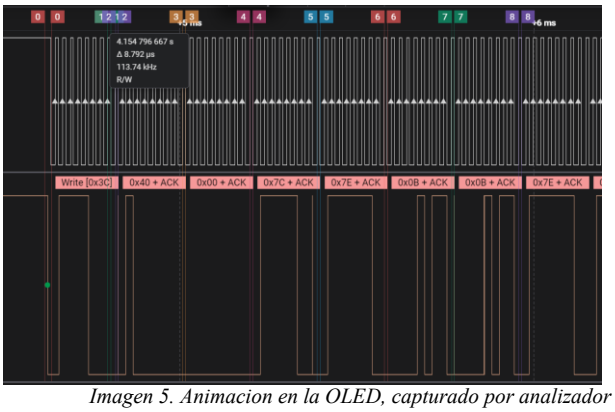

Imagen 5. Animacion en la OLED, capturado por analizador

Aqui se puede observar como solo se necesita un bit r/w para el resto de bytes de informacion, tambien como cada uno tiene su respectivo bit de confirmacion.

Esto pasa en los diferentes comandos ya que se envian una gran cantidad de bytes en el caso de encender y apagar la pantalla solo se envian 3 bytes, asi que podemos observar mejor el comportamiento de la estructura de comunicacion I2C.

Imagen 6. Captura de trama para encender la OLED

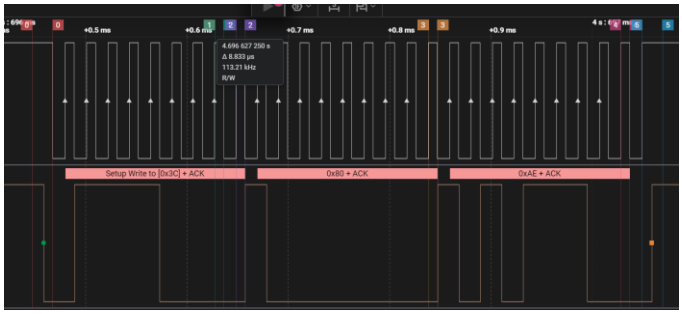

Imagen 7. Captura de trama para apagar la OLED

Como bien se menciono, en el primer byte se agrega el bit de r/w, esto se hace una sola vez sin importar el numero de bytes a enviar, lo que si se repite en cada byte es el bit de confirmacion "ACK".

4. Análisis de Resultados

1) Capturas con ACK y NACK

Elemento	Captura ACK	Captura NACK
Start detectado	Sí	Sí
Octeto enviado	0x78 (0x3C << 1)	0x7A (0x3D << 1)
Bit 9 (ACK/NACK)	ACK (0)	NACK (1)
Stop detectado	Sí	Sí
Frecuencia SCL (kHz)	100 kHz	100 kHz

5. Conclusiones

- El analizador lógico permitió visualizar y validar cada elemento de la trama I²C de manera efectiva.
- Se confirmó el correcto direccionamiento y respuesta de la pantalla OLED SSD1306.
- La provocación de NACK con una dirección incorrecta demostró el comportamiento esperado del protocolo.
- La herramienta `i2c.scan()` resultó eficaz para identificar dispositivos en el bus.
- Este método es reproducible y aplicable a otros dispositivos I²C en proyectos embebidos.

6. Referencias

[1] Raspberry Pi Ltd., "RP2040 Datasheet", 2021.

[2] Solomon Systech, "SSD1306 Datasheet", 2008.

[3] Saleae, "Logic 2 User Guide", 2023.

[4] MicroPython Documentation, "I2C Class", 2023.

[5] https://github.com/tomasramirez20/Practica-6-com.digitales