Sniffer I²C de bajo consumo

Ricardo Ercoli, Fausto Navadian, Gabriel Varela, Alfredo Solari, Julián Oreggioni
Universidad de la República, Montevideo, Uruguay
Email: {ricardo.ercoli, fausto.navadian, jorge.varela, asolari, juliano}@fing.edu.uy

***Resumen*—Se presenta el diseño e implementación de un sistema embebido capaz de capturar y analizar paquetes de datos en tránsito (Sniffer) que utilizan el protocolo de comunicación I²C. Se utiliza el microcontrolador MSP430G2553 de Texas Instruments donde dos pines configurados como entradas se conectan a las líneas de comunicación a espiar. El Sniffer se comunica vía UART con una PC, desde donde puede ser configurado y se accede a la información capturada. El sistema es capaz de identificar si la conexión física de los cables de datos y reloj es correcta. Se implementa una prueba de concepto donde se captura la información entre un reloj de tiempo real actuando como esclavo de un segundo MSP430G2553 funcionando como maestro. El sistema es compacto (no requiere hardware adicional), consume 33 μA en modo inactivo, y 5 mA en modo activo.**

***Palabras Claves*—Comunicación serial, I2C, bajo consumo**

I. Introducción

En la actualidad los sistemas embebidos cuentan con una gran variedad de sensores y periféricos que se comunican con un microcontrolador donde I²C es uno de los protocolos de comunicación más utilizados [1], [2]. Tanto para pruebas como para depuración de código es de gran utilidad conocer de manera no invasiva la información en tránsito de la comunicación entre un microcontrolador y sus periféricos [3].

Se encuentran en la literatura varios sniffers I²C, es decir, sistemas que pueden capturar y analizar paquetes en tránsito en una comunicación I²C entre dos dispositivos. Algunos están basados en FPGAs (Field Programmable Gate Arrays) [4], que ofrecen interesantes prestaciones a costa de un consumo excesivo. Por otro lado, existen sistemas implementados con microcontroladores [5], [6], muchos requieren hardware adicional, ya sea memoria [7], flip-flops [7], [8] o circuitos auxiliares [9] y ninguno es capaz de detectar si los cables están conectados correctamente.

Este trabajo presenta el diseño, la implementación y las pruebas de un *Sniffer I²C* compacto (no requiere hardware adicional), de bajo consumo, que puede detectar automáticamente si los cables están conectados correctamente.

II. Solución Propuesta

El sniffer se implementa en el microcontrolador MSP430G2553 de Texas Instruments operando a 16 MHz. El microcontrolador cuenta con variedad de modos de operación de bajo consumo, por ejemplo low power mode 3 (LPM3), y con periféricos internos que implementan los protocolos I²C y UART (Universal Asynchronous Receiver-Transmitter). El Sniffer se comunica vía UART con una PC, desde donde puede ser configurado y se accede a la información capturada.

Para implementar la captura de los datos se decide no usar periféricos I²C internos del microcontrolador, de modo tal de tener acceso total por software a la comunicación espiada. Por tanto, dos pines del microncontrolador configurados como entradas se conectan a los cables de la comunicación I²C, uno para señal de reloj (SCL) y otro para datos (SDA).

El software embebido se desarrolla en lenguaje C utilizando Code Composer Studio de Texas Instruments.

La implementación del *Sniffer I²C* se compone de cinco módulos *main*, *task*, *encolado*, *uart*, y *sniffer*. En la Fig. 1 se muestra cómo estos módulos interactúan entre ellos y con el hardware. El software desarrollado y su documentación está disponible en [10].

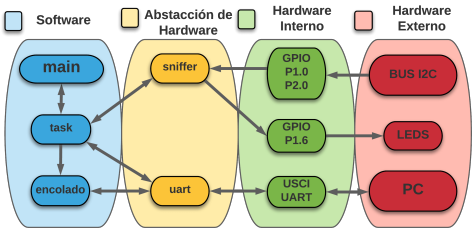

Figura 1. Diagrama de módulos del del software embebido del *Sniffer I²C*

El software embebido utiliza una arquitectura de Round-Robin con interrupciones [11]. En caso que no haya tareas pendientes, el microcontrolador pasa a LPM3. En *task* se implementan las funciones encargadas de procesar el llamado de cada bandera. En *encolado* se implementa una cola circular que almacena los datos que se transmitirán a la PC. En *uart* se implementa la comunicación UART entre el Sniffer y una PC. Finalmente, *sniffer* realiza la captura de la comunicación I²C, se basa en una máquina de estados la cual permite interpretar el protocolo I²C utilizando interrupciones.

La Fig. 2 presenta la máquina de estados, donde la transición entre estados se da mediante interrupciones. El primer estado es *STOP*, donde el Sniffer está inactivo en espera de la condición de *Start* del protocolo I²C. En *START* se configura el sistema para comenzar a guardar datos. El estado *1° L→H* señaliza que ocurre el primer flanco de subida de SCL y almacena el primer bit del byte transmitido; también reconoce la condición de *Stop*. El estado *1° H→L* configura el Sniffer para reconocer los bits restantes. En el estado *2°-8° L→H* se adquieren los últimos siete bits completando el byte transmitido. Finalmente, en el estado *9° L→H* se almacena el bit de *Ack* o *Nack* del protocolo I²C. Luego se retorna al estado *1° L→H* recorriendo la maquina de estados en forma cíclica hasta recibir una condición de *Stop*. Se observa que

esta implementación no es capaz de reconocer la condición de *Restart* debido a que el microcontrolador no posee la capacidad de reconocer interrupciones por flancos de subida y bajada al mismo tiempo. De esta forma solo es posible detectar condiciones de *Stop* o de *Start* y no ambas a la vez.

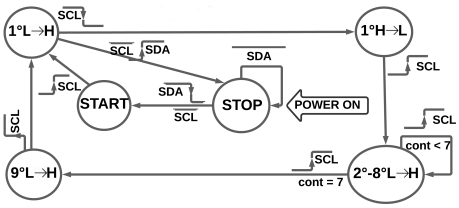

Figura 2. Máquina de estados implementada para el Sniffer

La Fig. 3 muestra el funcionamiento del Sniffer. El sistema responde a 5 comandos desde la PC partiendo de LPM3 (estado inactivo). Los comandos *START* y *STOP* sirven respectivamente para comenzar y frenar la captura de datos. *ADD* devuelve la dirección del último esclavo. *WHO* envía al PC "MASTER" o "SLAVE" indicando el último dispositivo que transmitió datos. *CHECK* comienza el modo de verificación de cables: se comprueba si el SCL y el SDA del Sniffer están conectados al SCL y SDA de la comunicación I²C respectivamente. Se pueden dar dos casos: que la conexión sea correcta, se prosigue de igual manera que con *START*; o que no lo sea, y se enciende un LED avisando que están mal conectados y se transmite el mensaje "WRONG". En este instante el sistema quedará a la espera del siguiente comando. El chequeo de los cables se realiza comparando la cantidad de interrupciones debidas a flancos en los puertos utilizados, sabiendo que las provocadas por la señal de reloj serán siempre mayores que las del puerto de datos. Un ejemplo de respuestas a estos comandos se puede observar en la Fig. 4.

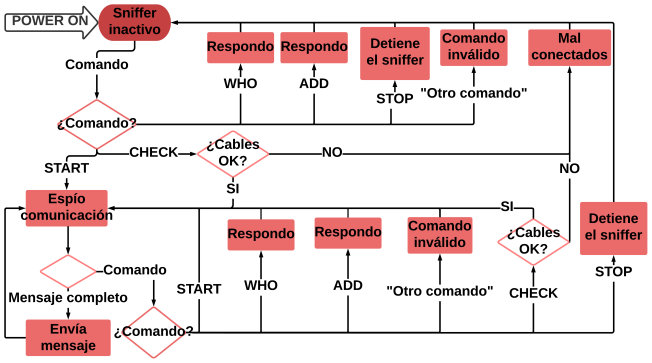

Figura 3. Diagrama de flujo del sistema

La solución propuesta presenta alta portabilidad pues la implementación se basa en software y lo único que depende del microcontrolador es el periférico utilizado para la comunicación con la PC.

III. Pruebas funcionales y Caracterización

Se desarrolla una prueba de concepto donde se captura información de una comunicación I²C entre un reloj de tiempo

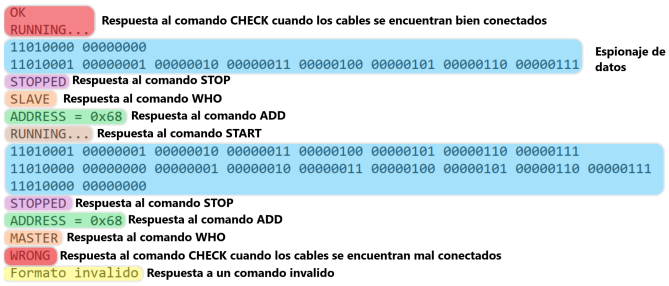

Figura 4. Respuesta del sistema a los comando.

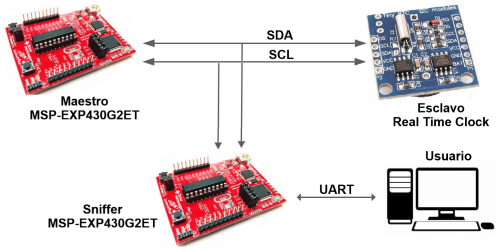

Figura 5. Diagrama funcional del sistema

real (RTC) DS1307 actuando como esclavo de un segundo MSP430G2553 funcionando como maestro (ver Fig. 5). Para la prueba se conecta el P1.0 y P2.0 del Sniffer a los pines P1.6 (SCL) y P1.7 (SDA) del maestro. Se implementa un programa en lenguaje C que realiza una transmisión y recepción I²C de forma periódica mediante interrupciones generando así una transferencia de información continua entre el MSP430G2553 y el RTC. La comunicación UART se configura en 9600 bps, 8 bits, sin bit de paridad y con 1 bit de parada.

Para probar la detección automática del conexionado de los cables, se colocaron los cables de forma correcta e incorrecta obteniendo una eficacia del 100 %. Se comprueba que el Sniffer tiene la capacidad de espiar comunicaciones de 100 kHz Si bien se estima que se podrían espiarse comunicaciones de 400 kHz, no se pudo probar debido a limitaciones en el RTC. Se observó que el Sniffer mientras está en LPM3 tiene un consumo de 33 μA. Por otro lado, se registraron picos de consumo de 5 mA cuando se encuentra activo, espiando una comunicación y transmitiendo a la PC. Finalmente, el uso de memoria RAM y FLASH registrado es de 178 y 860 bytes respectivamente, por lo que se podría utilizar un microcontrolador de menor memoria, o agregar procesamiento a los datos capturados.

IV. Conclusiones

Se implementa un Sniffer I²C con capacidad de discriminar las principales características de la comunicación: dirección del esclavo, si los paquetes enviados provienen del maestro o del esclavo, y los datos intercambiados. Asimismo, el sistema es capaz de identificar si la conexión física de los cables del protocolo I²C es correcta. Se desarrolla una interfaz para que un usuario configure y acceda a la información capturada por el Sniffer desde una PC. A futuro se agregará la capacidad de reconocer la condición de *Restart* y que si los cables están mal conectados se intercambien los puertos de manera automática.

REFERENCIAS

[1] J. Valdez and J. Becker, *Understanding the I2C Bus*, June 2015, Application Report, Texas Instruments. [En linea]. Disponible en: https://www.ti.com/lit/an/slva704/slva704.pdf

[2] J. H. Davies, *MSP430 microcontroller basics*. Elsevier, 2008.

[3] Infotecs. (2009, April) ¿Qué es un Sniffer? [En linea]. Disponible en: https://infotecs.mx/blog/que-es-un-sniffer.html

[4] M. Poppitz. (2008) FPGA Based Logic Analyzer. [En linea]. Disponible en: https://www.sump.org/projects/analyzer/

[5] Dangerous Prototypes. (2019) Bus Pirate. [En linea]. Disponible en: http://dangerousprototypes.com/docs/Bus_Pirate

[6] R. Kwiecień. (2008) I2C bus sniffer. [En linea]. Disponible en: http://en.radzio.dxp.pl/i2c-sniffer/

[7] Radio Locman. (2010) I2C Bus Sniffer on AVR. [En linea]. Disponible en: https://www.radiolocman.com/shem/schematics.html?di=64837

[8] A. Kalnoskas. (2014, October) How To Monitor I2C Communications Through RS232. [En linea]. Disponible en: https://www.microcontrollertips.com/monitor-i2c-communications-rs232/

[9] C. Grecu and C. Iordache, "Portable I2C monitor and debugger," in *2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)*. IEEE, 2015, pp. 131–134.

[10] R. Ercoli, F. Navadian, and G. Varela. (2021) Sniffer I2C (Analizador de paquetes en tránsito en comunicación I2C). Proyecto asignatura Sistemas Embebidos para Tiempo Real. Facultad de Ingeniería, UdelaR. Montevideo, Uruguay. [En linea]. Disponible en: https://gitlab.fing.edu.uy/ricardo.ercoli/proyecto-sisem

[11] D. E. Simon, *An embedded software primer*. Addison-Wesley Professional, 1999, vol. 1.