Escuela de Ingeniería Electrónica
Licenciatura en Ingeniería Electrónica
EL 4314 Arquitectura de Computadoras I
I Semestre 2022

TEC | Tecnológico de Costa Rica

Proyecto 4

Protocolos de comunicación serie - I2C

Alejandro Antonio Araya Nuñez, Wilson Andres Cheung Li, Erick Doña Chaves, Carlos Enrique Elizondo Alfaro, Javier Ignacio Espinoza Rivera, Jose Isaias Gonzalez Gonzalez, Daniel Gonzalez Vargas, Marcelo Leandro Brenes, Nicolás Morúa Vindas, Denisse Cristina Solorzano Ruiz, Randall Josue Vargas Chaves, Diego Vilchez Villalobos

1. Introducción

En este documento se denota la implementación del protocolo de comunicación serial I2C por medio del uso de microcontroladores Raspberry Pi Pico los cuales son configurados de manera en que el microcontrolador maestro sea el encargado de enviar caracteres definidos en su terminal hacia a otro microcontrolador esclavo el cual desplegará la información recibida, además de la implementación del código desarrollado se muestran los resultados obtenidos de pruebas realizadas con el fin de corroborar el correcto funcionamiento de la comunicación en serie por medio de un osciloscopio donde se observan las tramas de datos.

2. Código Implementado

En esta sección, se detalla el codigo que se implemento para lograr la transmisión de datos de *master* a *slave* en lenguaje C como lo indican las instrucciones. Para estos códigos se hace uso de las librerías de stlib.h para la comunicación serial con la computadora, así como la librería de hardware/i2c.h para las funciones de comunicación I2C para rasberry Pi Pico, de las mismas se utilizan varias funciones como lo pueden ser la función i2c_init() que permite inicializar el estado del módulo de comunicación I2C en un estado conocido para ser utilizado. Tanto para master como para slave se definen los pines en los que se estara ejecutando el protocolo I2C, esto mediante la función gpio_set_function(). [1]

2.1. Codigo del Master

Para el código de master se define mediante la función 2c_set_slave_mode() el modo en el que opera el dispositivo slave o master, por otro lado se utiliza un ciclo en donde se verifica constantemente la disponibilidad del módulo de I2C para enviar datos mediante la función i2c_get_write_available(). Mediante la función i2c_write_blocking() se realiza la escritura de datos para cada byte en el buffer del I2C, por lo que para la palabra "hola"se necesita de un for de 4 ciclos para las 4 letras. Se muestra el código de master en detalle en el apéndice A.

2.2. Codigo del Slave

Para el slave se utiliza de misma forma la función i2c_set_slave_mode() para definir el modo en que trabajara el dispositivo slave en este caso. Se utiliza la función i2c_get_read_available() para conocer si hay algún byte disponible en el buffer I2C, con esto se realiza la lectura constante de cada byte recibido mediante un while. Se muestra el código de slave en detalle en el apéndice B.

3. Resultados Obtenidos

Como se puede observar en las siguientes figuras tomadas del osciloscopio, en la figura 2 se observa el bit en alto donde se marca el inicio de la transmisión de datos, seguido de la dirección en la cual se encuentra el *slave*, después de esto, el *slave* confirma con la señal de *acknowledge* que esta disponible para recibir datos. Posteriormente, el *master* envía en binario el código ASCII de la letra 'o' y el bit de parada de la transmisión.

Esto se puede observar de la misma forma para las figuras 1, 3 y 4 donde se transmiten de igual forma los caracteres 'h', 'l' y 'a' respectivamente.

En la figura 5 se muestra una captura de las terminales de ambas Pico tomadas del mismo puerto serial por medio de la conexión USB. El puerto COM7 corresponde al *master* que envía cada carácter en su código ASCII correspondiente, por ejemplo, en la figura se observa que primero envía un 104 que corresponde a la letra 'h', por otro lado en el COM5 se muestra la terminal para el esclavo que solamente va imprimiendo cada carácter que recibe

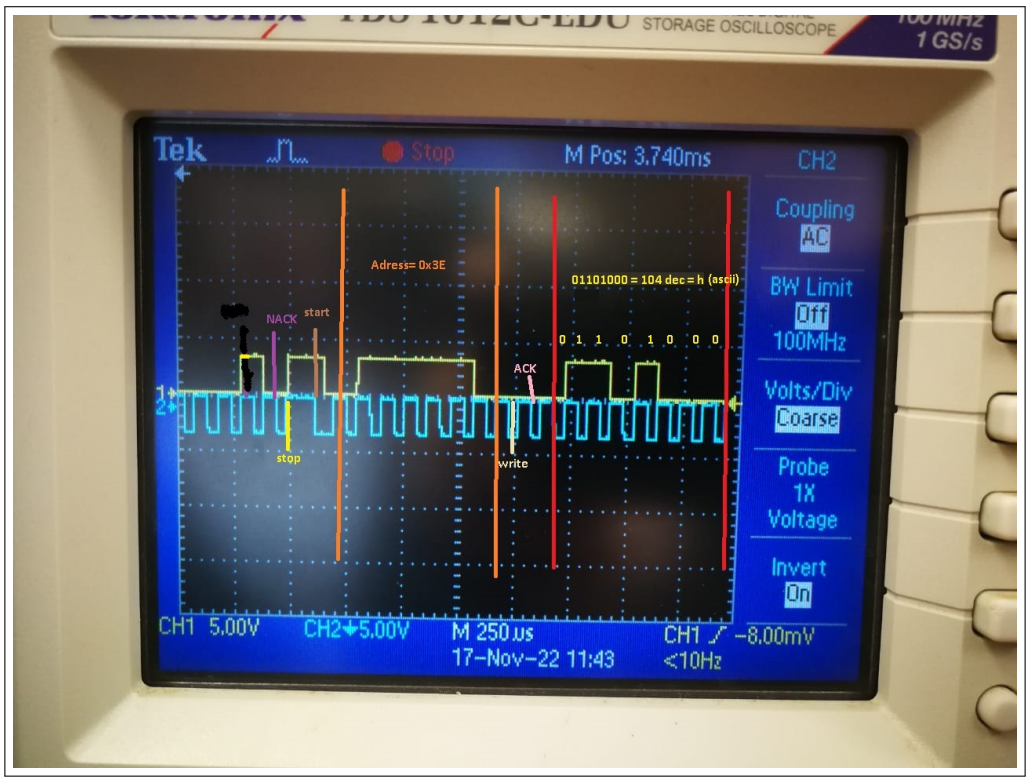

Figura 1: Gráfica de la trama de datos I2C para la transferencia del carácter *h*.

Figura 2: Gráfica de la trama de datos I2C para la transferencia del carácter *o*.

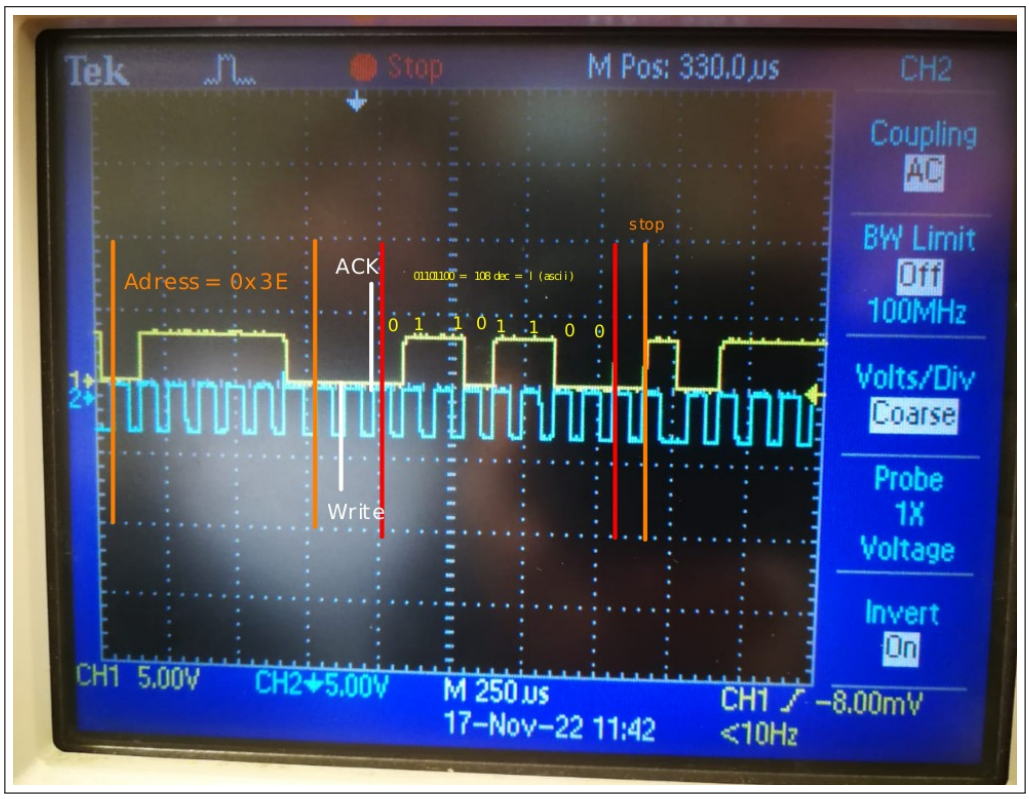

Figura 3: Gráfica de la trama de datos I2C para la transferencia del carácter *l*.

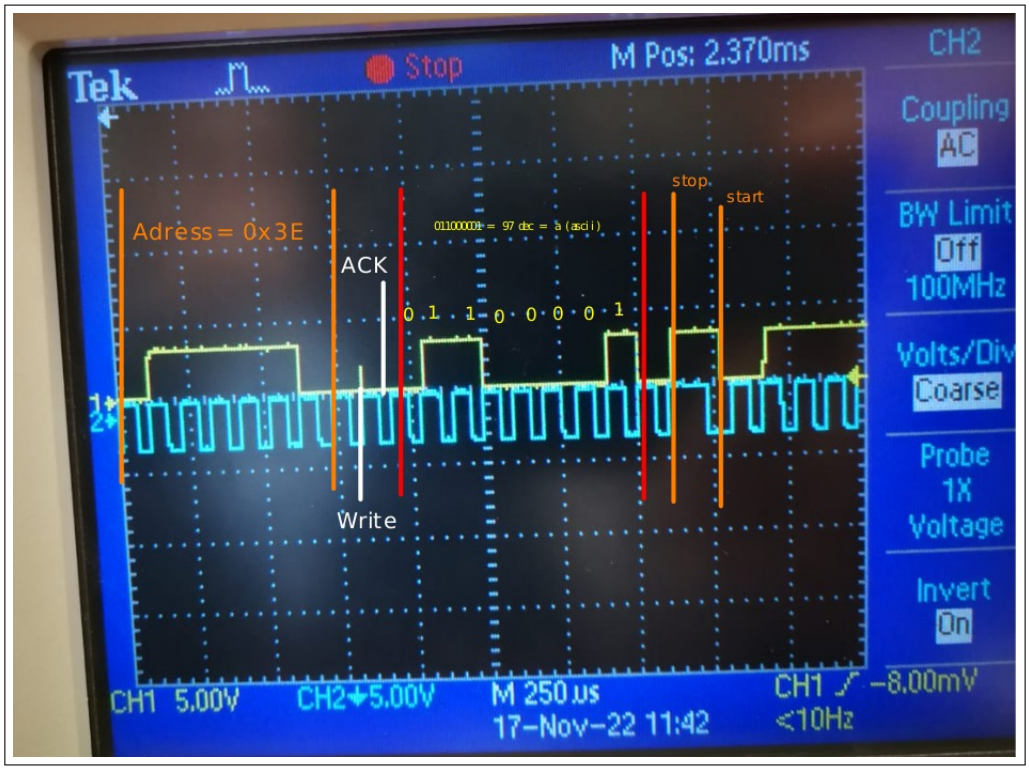

Figura 4: Gráfica de la trama de datos I2C para la transferencia del carácter *a*.

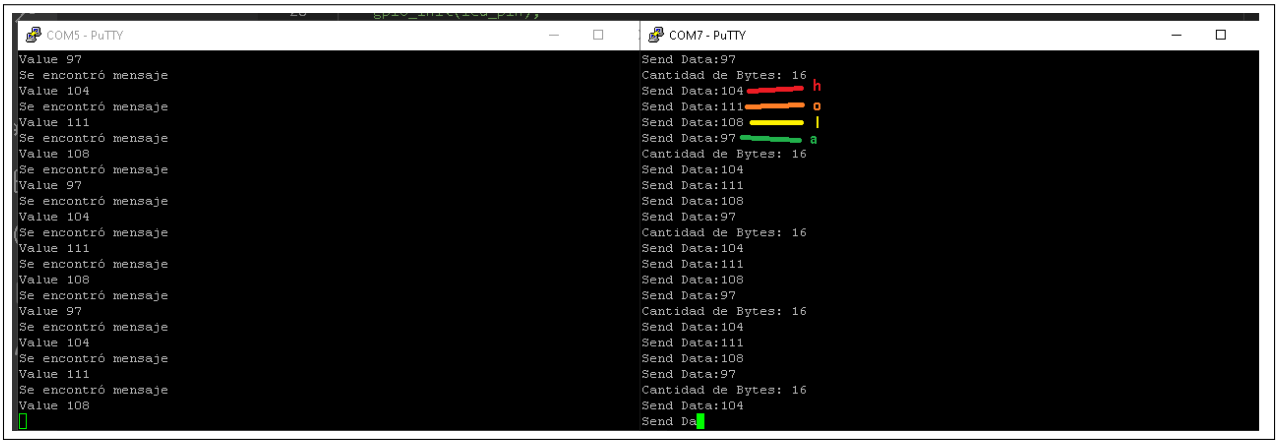

Figura 5: Captura de pantalla de las terminales seriales de ambas Raspberry Pi Pico

Finalmente, en la figura 6, se observa en el osciloscopio como se da la transmisión de datos de forma continua para enviar la palabra 'hola'.

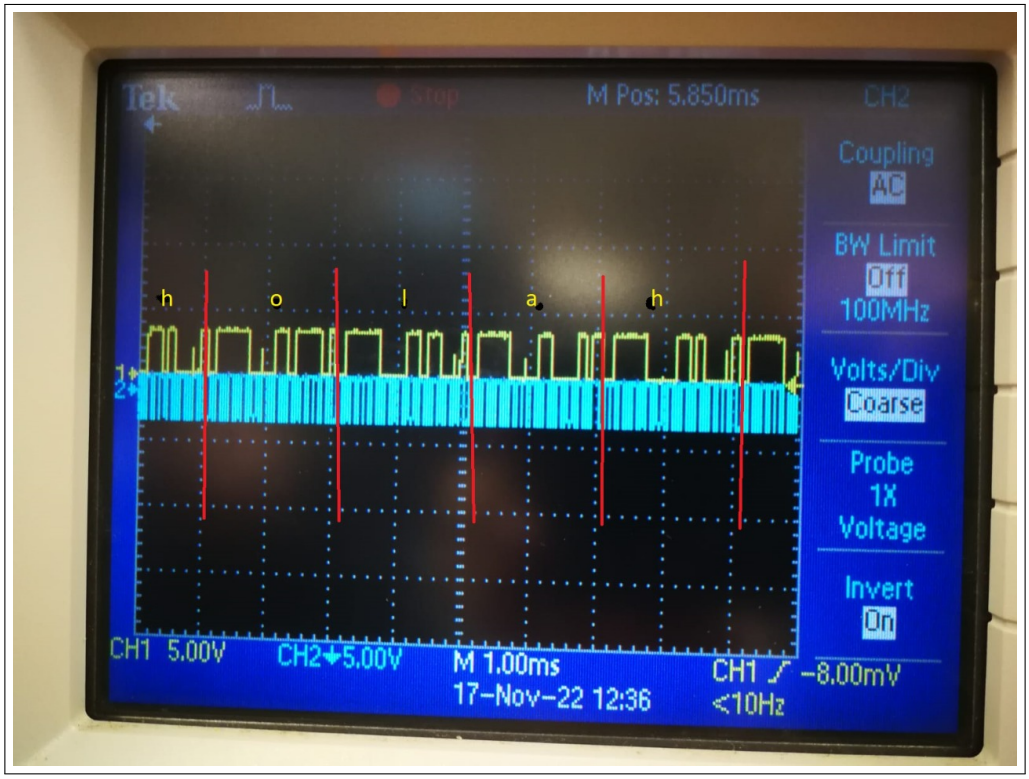

Figura 6: Gráfica de la trama de datos I2C para la transferencia de la palabra *hola*.

4. Conclusiones

- Al final se logro la transmision de datos desde el *master* hacia el *slave* de forma exitosa
- Se pudo observar en el osciloscopio los bits que se transmitian entre ambos dispositivos

Referencias

[1] Rasberry Pi Pico *"hardware_i2c"*, 2022. Disponible en: https://raspberrypi.github.io/pico-sdk-doxygen/group__hardware__i2c.html.

[2] Digikey.com. *Raspberry Pi Pico and RP2040 - C/C++ Part 1: Blink and VS Code*, 2022. Disponible en: https://www.digikey.com/en/maker/projects/raspberry-pi-pico-and-rp2040-cc-part-1-blink-and-vs-code/7102fb8bca95452e9df6150f39ae8422.

[3] Github.com *raspberrypi/pico-examples*, 2022. Disponible en: https://github.com/raspberrypi/pico-examples/tree/master/i2c.

[4] shawnhymel.com *How to Set Up Raspberry Pi Pico C/C++ Toolchain on Windows with VS Code*, 2021. Disponible en: https://shawnhymel.com/2096/how-to-set-up-raspberry-pi-pico-c-c-toolchain-on-windows-with-vs-code/.

A. Master.c

```
torder -cache:dl1 dl1:256:32:1:l/**
 * Raspberry Pi Pico - Voltmeter
 * Sends via i2c
 * Works with ADC pin 0
 * See: www.penguintutor.com/projects/pico
 */


#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <stdio.h>

//Dirección del I2C
#define I2C_ADDR 0x3E

int main() {
    stdio_init_all();
    // Inicia el I2C
    i2c_init(i2c0, 10000);
    //Configura la pico como master
    i2c_set_slave_mode(i2c0, false, I2C_ADDR);
    //Configura los pines correspondientes como I2C
    gpio_set_function(4, GPIO_FUNC_I2C);
    gpio_set_function(5, GPIO_FUNC_I2C);
    gpio_pull_up(4);
    gpio_pull_up(5);


    //Dato que se va a enviar
    char Palabra[] = "hola";
    uint8_t value;
```

```
    //Espera 20s antes que empiece el loop
    sleep_ms(20000);
    while (true) {
        //Obtiene e imprime la cantidad de bytes disponibles en el canal para escribir
        int bytes=i2c_get_write_available(i2c0);
        printf("Cantidad de Bytes: %d \n",bytes);
        //Ciclo para enviar cada carácter del mensaje
        for (int i=0; i<4;i++)
        {
            //Asigna a value el carácter correspondiente
            value=Palabra[i];
            //Verifica si se puede escrbir, en caso de que no, continua el ciclo
            if (i2c_get_write_available(i2c0) == 0) continue;
            //Escribe el carácter de value
            i2c_write_blocking(i2c0, 0x3E,&value, 1, true);
            //Imprime el carácter escrito
            printf ("Send Data:%d\r\n", value);
        }


    }
}
```

B. Slave.c

```c
#include "pico/stdlib.h"
#include "hardware/i2c.h"
#include <stdio.h>

//Dirección del I2C
#define I2C_ADDR 0x3E

int main() {
    stdio_init_all();
    // Inicia el I2C
    i2c_init(i2c0, 10000);
    //Configura la pico como esclavo
    i2c_set_slave_mode(i2c0, true, I2C_ADDR);
    //Configura los pines correspondientes como I2C
    gpio_set_function(4, GPIO_FUNC_I2C);
    gpio_set_function(5, GPIO_FUNC_I2C);
    gpio_pull_up(4);
    gpio_pull_up(5);

    //Variable donde se almacena el dato leido
    uint8_t  rxdata;
    //Espera 20s antes que empiece el loop
    sleep_ms(20000);
    while (true) {
        //Verifica si hay bytes en el canal para leer, en caso de que no, continua al siguiente ciclo
        if (i2c_get_read_available(i2c0) < 1) {
            continue;
        }
        printf("Se encontró mensaje\n");
        //Lectura del mensaje
        i2c_read_raw_blocking (i2c0, &rxdata, 1);
        //Muestra el mensaje en terminal
        printf ("Value %d\r\n", rxdata);
    }
}
```