Protocolos de Comunicación

Javier Montiel González C.U.159216, Andrea González Cardoso C.U. 157961,
Diego Villalvazo Suzer C.U. 155844

Resumen

~~La comunicación entre microcontroladores es una característica esencial para la elaboración de sistemas embebidos. Por otro lado, la comunicación inalámbrica es muy utilizada en diversos dispositivos y cuya ventaja principal es la ausencia de cables.~~ En esta práctica se buscó comparar los protocolos de comunicación I2C, UART y SPI, lo que también nos familiarizaría con estos. Además se intentó utilizar XBee para comunicación inalámbrica de dispositivos. Lo que resultó de mayor dificultad fue el establecer la comunicación SPI utilizando el acelerómetro de la DE0-Nano. La asignación de señales en VHDL fue un problema para la realización de la práctica. En términos de facilidad llegamos a la conclusión que el protocolo de comunicación que utilizaríamos sería el I2C.

1 Introducción

El propósito de la práctica fue familiarizarse con los protocolos de comunicación I2C, SPI, UART y Zigbee para posteriormente compararlos. Para ello se utilizó el arduino y el acelerómetro de la DE0-Nano.

Los sistemas embebidos son programas dedicados por lo que para realizar tareas complejas es necesario la comunicación entre distintos sistemas. La transmisión de información entre ellos es requerida para funcionalidades como el almacenamiento de datos en una sola memoria, asignación de tareas, entre otras.

Cada protocolo de comunicación requiere la sincronización de ciertas señales entre los microcontroladores para poder establecer un canal de transmisión o recepción y dependerá del protocolo las señales a asignar.

En el marco teórico se profundiza sobre las características de cada protocolo de comunicación y de cada uno de los componentes utilizados para la elaboración de la práctica.

2 Marco Teórico

Un protocolo de comunicación es un sistema de instrucciones que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información.

En la práctica se implementó el protocolo I2C (por su nombre en inglés *Inter-Integrated Circuit*). Se utiliza principalmente para comunicación entre diferentes partes de un circuito, por ejemplo, un controlador y circuitos periféricos integrados. Cabe mencionar que I2C es un protocolo síncrono. I2C usa solo 2 cables, uno para el reloj (SCL) y otro para el dato (SDA). Esto significa que el maestro y el esclavo envían datos por el mismo cable, el cual es controlado por el maestro, que crea la señal de reloj. I2C no utiliza selección de esclavo, sino direccionamiento. En la práctica, realizamos un protocolo de comunicación serial síncrona maestro-esclavo.

Por otro lado, UART (recepción-transmisión asíncrona universal) es uno de los protocolos serie más utilizados. La mayoría de los microcontroladores disponen de hardware UART. Usa una línea de datos simple para transmitir y otra para recibir datos. Comúnmente, 8 bits de datos son transmitidos de la siguiente forma: un bit de inicio, a nivel bajo, 8 bits de datos y un bit de parada a nivel alto.

UART se diferencia de SPI y I2C en que es asíncrono y los otros están sincronizados con señal de reloj. La velocidad de datos UART está limitado a 2Mbps

SPI es otro protocolo serie muy simple. Un maestro envía la señal de reloj, y tras cada pulso de reloj envía un bit al esclavo y recibe un bit de éste. Los nombres de las señales son por tanto SCK para el reloj, MOSI para el *Maestro Out Esclavo In*, y MISO para *Maestro In Esclavo Out*. Para controlar más de un esclavo es preciso utilizar SS

(selección de esclavo).

Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red llamado IEEE 802.15.4 para crear redes FAST POINT-TO-MULTIPOINT (punto a multipunto); o para redes PEER-TO-PEER (punto a punto). Fueron diseñados para aplicaciones que requieren de un alto tráfico de datos, baja latencia y una sincronización de comunicación predecible. XBee está basado en el protocolo Zigbee. En términos simples, los XBee son módulos inalámbricos fáciles de usar.

Zigbee es una alianza y un estándar de redes MESH de eficiencia energética y de costos. XBee utiliza el estándar Zigbee y lo agrega y envuelve en un pequeño empaque elegante.

El acelerómetro es un instrumento que sirve para medir la aceleración de movimiento de un vehículo. El acelerómetro de la DE0-Nano es el ADXL345, el cual es uno pequeño, delgado, de bajo poder y de 3-ejes con una alta resolucion (13-bit) de un peso en promedio de 16g. Los datos de salida están formateados como un complemento de dos bits de 16 bits y son accesibles a través de una interfaz digital SPI (3 o 4 cables) o I2C.

El puente H es un circuito electrónico que permite a un motor eléctrico DC girar en ambos sentidos, avanzar y retroceder.

Un puente H se construye con 4 interruptores (mecánicos o mediante transistores). Cuandos los interruptores S1 y S4 están cerrados (S2 y S3 abiertos) se aplica una tensión haciendo girar el motor en un sentido. Abriendo los interruptores S1 y S4 (cerrando S2 y S3), el voltaje se invierte, permitiendo el giro en sentido inverso del motor.

Un puente H no solo se usa para invertir el giro de un motor, también se puede usar para frenarlo de manera brusca, al hacer un corto entre los bornes del motor, o incluso puede usarse para permitir que el motor frene bajo su propia inercia, cuando desconectamos el motor de la fuente que lo alimenta.

3 DESARROLLO Y RESULTADOS

La práctica se dividió en 3 sesiones. En la primera sesión se implementó el protocolo de comunicación I2C para comunicar dos arduinos por lo que fue necesario juntarse con el equipo flam. A nosotros nos tocó ser el maestro en la comunicación serial síncrona maestro-esclavo. ~~A continuación se presenta el código que realizamos:~~

```
int pinButton1=14;
int pinButton2=15;
int button1State=0;
int button2State=0;
byte
arr[]={0b00000000,0b00000011,0b00000010,0b00000001};
int Mike=9;
#include <Wire.h>

void setup() {
   //Conexion al bus I2C
   Wire.begin();
   pinMode(pinButton1, INPUT);
   pinMode(pinButton2, INPUT);
}

void loop() {
  //pedimos 1 byte al esclavo
   Wire.requestFrom(Mike,1);
  //inicia transmision
   Wire.beginTransmission(Mike);

  //Lectura de los valores de los botones
  button1State= digitalRead(pinButton1);
  button2State= digitalRead(pinButton2);

  //Mandamos los estados
  if(button1State==0 && button2State==0){
   Wire.write(arr[1]);
  }
   else{
    if(button1State==0 && button2State==1){
     Wire.write(arr[2]);
    }else{
     if(button1State==1 && button2State==0){
      Wire.write(arr[3]);
     }else{
      Wire.write(arr[4]);
     }
    }
   }

  //Terminar la transmision
  Wire.endTransmission(Mike);
  button1State=0;
  button2State=0;
  delay(1000);
}
```

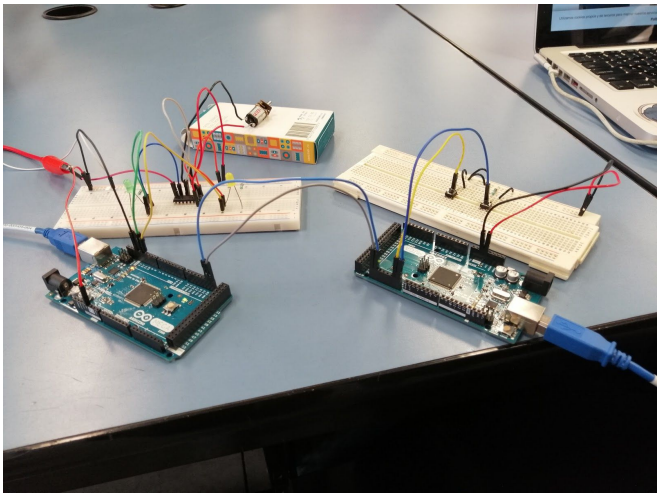

Figura 1. Dos Arduinos comunicados con el protocolo I2C.

Para la segunda y tercera sesión se armó el puente H y se intentó implementar el protocolo de comunicación SPI. Lo que se consiguió realizar fue la correcta asignación de las señales para el acelerómetro del DE0-Nano. ~~El siguiente código presenta dicha asignación:~~

Parte modificada del archivo spi 3 wire master.vhdl

```
port map
        (
                clock   => CLK,
                reset_n => reset_n,
                enable  => enable ,
                cpol    => '1',
                cpha    => '1',
                clk_div => 5,
                addr    => 0,
                rw      => rw_buffer,
                tx_cmd  => cmd_to_spi,
                tx_data => TX_IN,
                sclk    => I2C_SCLK,
                ss_n(0) => G_SENSOR_CS_N,
                sdio    => I2C_SDAT,
                busy    => busy,
                rx_data => RX_out
        );
```

~~Para la asignación de valores de BANDWIDTH y la habilitación y deshabilitación de LINK, AUTOSLEEP y MEASURE, al igual que establecer el acelerómetro en modo NORMAL, se modificó el código que se nos había proporcionado~~. Las partes modificadas del archivo intel_accel.vhd se muestran a continuación:

```
(...)
 WHEN WR_BW_RATE =>
        cmd_buffer <= BW_RATE
        rx_buffer <= "00001100";
        busy_buffer <= '1';
        if (next_wr = '1') then
                state <= WR_INT_ENABLE;
        end if;
(...)
WHEN WR_POWER_CONTROL =>
        cmd_buffer <= POWER_CONTROL;
        rx_buffer <= "00001000";
        busy_buffer <= '0';
        if (next_wr = '1') then
           state <= IDLE;
        end if;
```

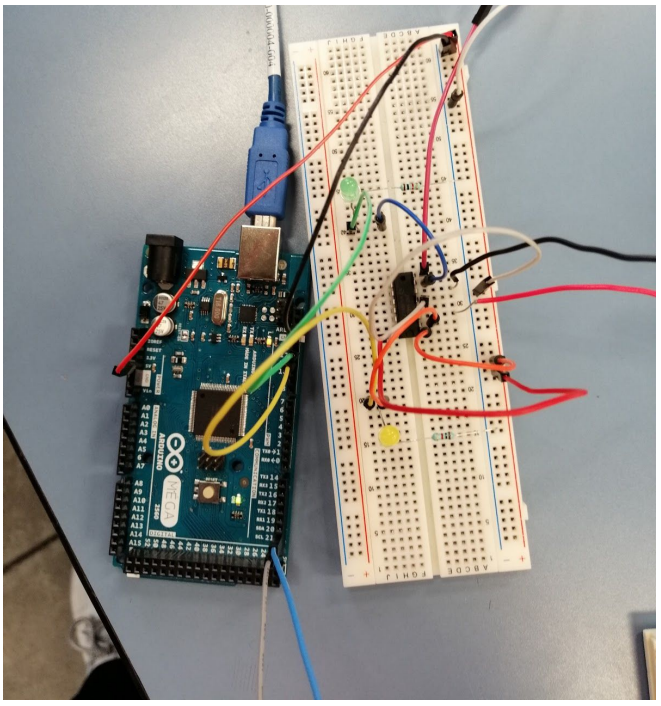

Figura 2. Circuito del puente H y el motor..

4 CONCLUSIONES

Javier: En esta práctica nos familiarizamos con diversos protocolos de comunicación. Cada protocolo posee distintas características que lo hacen idóneo para ciertas tareas. Por otro lado, la falta de conocimientos para realizar la segunda parte de la práctica fue un gran impedimento para su elaboración. En particular, la documentación del acelerómetro de la DE0-Nano era muy poco concisa y se requirió de varias horas para entenderla. Además, el cambiar del lenguaje C++ a VHDL provocó mayor confusión en la elaboración de los códigos. Considero que una sesión más hubiera sido suficiente para terminar la práctica.

Diego: Los protocolos de comunicación son una parte esencial para cualquier transferencia de datos. Durante la práctica logramos entender lo básico de los protocolos de comunicación. Yo creo que todos, incluso nosotros que somos estudiantes de ingeniería, damos por hecho todos los procesos que conlleva la comunicación entre sistemas, pero al elaborar esta práctica nos dimos cuenta de lo realmente complejos e importantes que son.

En mi opinión, fue una práctica con un nivel alto de dificultad ya que tuvimos que implementar diferentes protocolos que entre ellos fueron muy diferentes, entonces eso implicó mucha investigación de fondo, pero al final del día fue una experiencia gratificante ya que aprendimos mucho sobre los protocolos, sus usos e

implementación.

Andrea: Con esta práctica pudimos conocer los protocolos más comunes de comunicación entre micorcontroladores. Esto es importante ya que se puede entender el intercambio de datos, lo cual es un elemento importante en esta materia que servirá para el proyecto final.

Dada la complejidad de la práctica, se encontraron varias dificultades en la implementación, principalmente con el método I2C, lo cual dificultó el avance y terminación de la práctica. Aun así, pudimos poner en práctica algunos conceptos vistos en teoría, al igual que volver a poner en práctica lo aprendido el curso anterior, al tener que utilizar Quartus.

Bibliografía

[1] Protocolos de comunicación arduino. Consultado el 15 de marzo de 2019. Disponible en https://aprendiendoarduino.wordpress.com/2014/11/18/tema-6-comunicaciones-con-arduino-4/

[2] XBee. Consultado el 15 de marzo de 2019. Disponible en https://xbee.cl/que-es-xbee/

[3] Puente H. Consultado el 15 de marzo de 2019. Disponible en: https://www.ingmecafenix.com/electronica/puente-h-control-motores/

[4] Puente H. Consultado el 22 de febrero de 2019. Disponible en: http://www.ti.com/lit/ds/symlink/l293.pdf

[5] SPI 3-Wire Master. Consultado el 22 de febrero de 2019. Disponible en: https://www.digikey.com/eewiki/pages/viewpage.action?pageId=27754638

[6] I 2C - What's that? Consultado el 28 de febrero de 2019. Disponible en: https://www.i2c-bus.org/ 4

[7] Guía de usuario de XBee/XBee Pro S1. Consultado el 15 de Marzo de 2019. Disponible en: https://www.digi.com/resources/documentation/digidocs/PDFs/ 90000982.pdf

[8] SoftwareSerial Library. COnsultado el 22 de febrero de 2019. Disponible en: https://www.arduino.cc/en/Reference/SoftwareSerial

[9] Comunicación Serial. Consultado el 22 de febrero de 2019. Disponible en: https://learn.sparkfun.com/tutorials/serial-communication