Práctica 3: Laboratorio de Principios de Mecatrónica

Rebeca Baños García, Leonardo García Osuna

Resumen—En esta práctica se conocieron las ventajas y desventajas de los protocolos I2C, UART y SPI para comunicar microprocesadores con sensores, en el caso del acelerómetro, y con otros MCUs, como en el caso del motor. Además, se aprendió a configurar el radio típico de DigiKey, el XBee, para mandar señales de control –*cualquier mensaje*– vía 2.4GHz. Por último se aprendió a utilizar el acelerómetro embebido en la FPGA DE0-NANO por medio del protocolo SPI.

1. INTRODUCCIÓN

Esta práctica fue muy importante ya que tuvimos una mejor experiencia y más práctica acerca de dispositivos que se comunican entre ellos. Para lograr que la comunicación deseada fuera el resultado, debimos conocer las diferencias entre los protocolos I2C, UART y SPI y utilizar cada uno en las comunicaciones específicas y que funcionara correctamente.

Es imprescindible que un dispositivo –microprocesador, sensor, actuador– pueda comunicarse de alguna manera con otros de la misma naturaleza para poder interactuar de manera coordinada y correcta en cualquier aplicación. Sin los protocolos adecuados, se corre el riesgo de que los mensajes no lleguen adecuadamente, entre ruido al bus, la información no llegue a la velocidad adecuada, o simplemente se altere o pierda parte de ella.

Si bien hay otros protocolos, como RS232 o RS485, se exploran en ésta práctica los más extendidos y de uso general.

2. MARCO TEÓRICO

Un puerto serial es un módulo de comunicación digital para un sistema embebido. Es decir, permite la comunicación entre dos dispositivos digitales.

- **I2C:** El protocolo I2C se puede encontrar en pantallas OLED, sensores de presión barométrica y módulos Giroscopio o Acelerómetros, por mencionar algunos ejemplos. El circuito integrado I2C es un protocolo de comunicación serial que comunica diferentes partes de un circuito. El protocolo I2C utiliza solo dos vías o cables de comunicación que son el maetsro y el esclavo; la transferencia de datos siempre inicia por el maestro y el esclavo reacciona.
- **UART:** Universal Asynchronous Receiver-Transmitter es lo que representan sus siglas en inglés. El UART es el dispositivo que controla los puertos y dispositivos serie. El controlador del UART es el componente clave del subsistema de comunicaciones series de una computadora. El UART contiene un generador de paridad, registros de corrimiento, oscilador variable, verificadores de las tres condiciones y lógica de control. De acuerdo al tipo de conexión, una UART tiene separadas las lineas de trasmisión y recepción; esta caracteristica le perite poder operar los tres modos de comunicación asíncrona que existen. Los modos de comunicación son:
  - FULL-DUPLEX: Significa que puede transmitir y recibir simultáneamente.
  - HALF-DUPLEX: Sólo transmite o sólo recibe.
  - SIMPLEX: Sólo se dedica a transmitir información binaria.
- **SPI:** Serial Peripheral Interface por sus siglas en inglés, es un estándar de comunicaciones, usado principalmente para la transferenia de información entre circuitos integrados en equipos electrónicos. El SPI es un protocolo síncrono que trabaja en modo full duplex para recbir y transmitir información, permitiendo que dos dispositivos pueden comunicarse entre sí al mismo tiempo utilizando canales diferentes o líneas diferentes en el mismo cable.

3. DESARROLLO

Para la primera parte de la práctica se utilizó el protocolo I2C para comunicar dos microcontroladores. Para esto fue necesario ocupar un segundo arduino. Primero cableamos dos botones a una protoboard y los conectamos al arduino, lo que llamamos el microcontrolador A que hace el papel de maestro. Posteriormente conectamos a otro arduino un puente H y un motor, a este mircocontrolador lo llamamos B por lo que jugara el papel de esclavo. Una vez teniendo el cableado, escribimos el código necesario para que al presionar algún botón del microcontrolador A, el motor conectado al microcontrolador B frenara pasivamente, rotara en sentido de las manecillas, rotara en contrasentido de las manecillas o que frenara activamente.

Para la segunda parte de la práctica, utilizamos el acelerómetro de la tarjeta DE0-Nano. Para comunicar el acelerómetro lo comunicamos con la FPGA mediante el protocolo SPI. La base del código necesario se obtuvo a partir de

un repositorio, y solamente editamos los valores necesarios para que el acelerómetro funcionara de manera adecuada. A cada registro de los valores X, Y y Z le asignamos los valores de los dip switch para poder intercambiar entre ellos y que la FPGA responda de acuerdo al eje que la movemos. Por último, para el motor escogimos trabajar con el eje Y y conectamos el motor con el puente H a la FPGA y configuramos el código para que al mover la FPGA el motor cambiara de dirección.

Para la última parte utilizamos el protocolo UART para comunicar las XBees. Primero configuramos ambos Xbees para que estuvieran en el mismo canal y con mismos identificadores y se detectaran entre ellos. Posteriormente escribimos el código en Arduino para que al escribir algo en un XBee, el otro lo detectara inalámbricamente y reaccionara. Finalmente conectamos el XBee conectado al Arduino a una protoboard con 3 LEDs y dos botones y programamos diferentes comandos para que al presionar un botón se encendiera un LED, al presionar otro se encendiera otro y finalmente al presionar embos se encendiera el tercero.

4. RESULTADOS

Para la primera parte de la práctica, el código que utilizabamos al inicio parecía estar correcto, sin embargo nuestro motor no giraba de la manera en la que queríamos, por lo que checamos la parte cableada. Fue ahí donde nos percatamos que el verdadero error fue un falso en uno de los dos botones que estabamos utilizando, al arreglar esto notamos que nuestro código funcionaba bien ya que nuetsro motor giraba en las direcciones que le indicamos.

Para la tercera parte tuvimos complicaciones al configurar inicialmente los XBees, ya que creíamos haberlos puesto a ambos en el canal D, sin embargo uno estaba en el canal C, por lo que la comunicación entre ellos era errónea. Al detectar este error simplemente cambiamos los dos XBees al canal C y así logramos que se detectaran y que la comunicación funcionara. Para encender los LEDs primero probamos que la comunicación fuera correcta, es decir, que al escribir algo en el XBee conectado al Arduino, el otro XBee lo leyera correctamente.

La segunda parte de la práctica fue la más complicada, ya que tuvimos que leer las específicaciones del acelerómetro varias veces para asegurarnos de asignar el valor a los rehistros del código correctamente. A pesar de hacer esto, tuvimos complicaciones también en otras partes del código, por lo que tuvimos que cambiar todo a 8 bits, ya que antes etsaba definido a 16 bits. Por último, la FPGA que estabamos usando no leía correctamente el código que le escribimos ya que no hacía nada, al conectar otra FPGA (provista por Edgar) comprobamos que nuestro código era correcto ya que funcionó bien a la primera. Una vez teniendo esta parte funcionando, el conectar el motor fue sencillo y el resultado esperado lo obtuvimos a la primera.

5. CONCLUSIONES

- **Rebeca:** Ha sido la práctica más costosa a lo largo del curso ya que tuvimos que acudir a varios manuales e información de como funcionaba cada uno de los protocolos de comunicación y así poder usarlos correctamente. Fue mucho tiempo invertido en leerlos pero finalmente comprendimos todos los pasos y así logramos los resultados esperados.
- **Leonardo:** Es muy importante conocer con una profundidad adecuada los protocolos de comunicación para los sistemas embebidos y microprocesadores; esto permite tomar mejores decisiones a la hora de integrar sistemas, dependiendo de los requerimientos del mismo. Realizar ésta práctica me ayudó a ver que se pueden simplificar ciertos procesos que a veces al integrar librerías o dependencias preexistentes, vuelven innecesariamente más complejo el proceso.

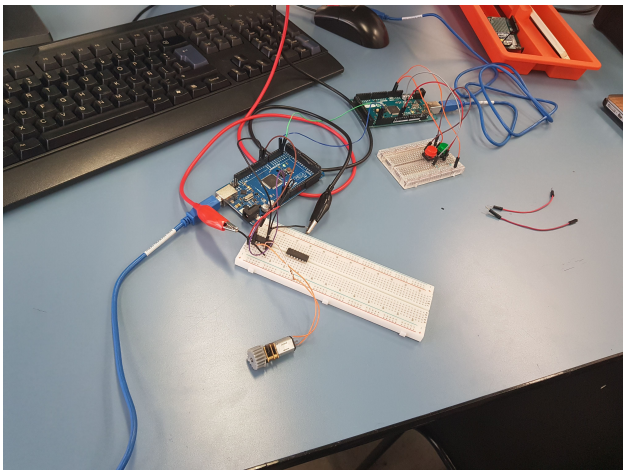

Figura 1. Parte 1: protocolo I2C.

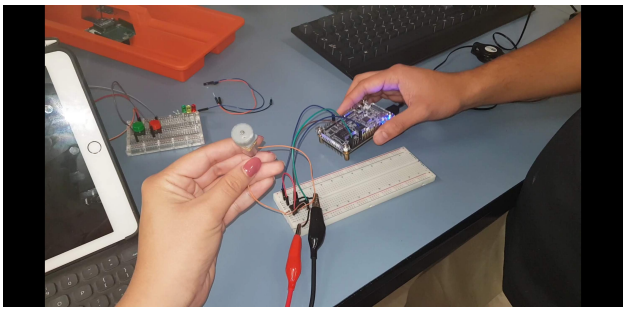

Figura 2. Parte 2: protocolo SPI.

6. ROL O PAPEL

- **Rebeca:** Consultaba manuales necesarios, escribía partes del código y verificaba los cableados necesarios.
- **Leonardo:** Código, debugging y alambrado de los circuitos.

7. FUENTES CONSULTADAS

- **Puerto Serial:** https://hetpro-store.com/TUTORIALES/puerto-serial/
- **I2C:** https://teslabem.com/nivel-intermedio/fundamentos-del-protocolo-i2c-aprende/ https://es.wikipedia.org/wiki/I%C2%B2C#Sistema_de_bus
- **UART:** https://hetpro-store.com/TUTORIALES/puerto-serial/

- **SPI:** http://panamahitek.com/como-funciona-el-protocolo-spi/ https://es.wikipedia.org/wiki/Serial_Peripheral_Interface