Práctica No.3 Protocolos de Comunicación

Departamento Académico de Sistemas Digitales

Instituto Tecnológico Autónomo de México

Primavera 2019

1. Objetivos

- Entender las ventajas y desventajas de los protocolo I^2C, UART y SPI
- Utilizar los protocolos I^2C, UART y SPI para comunicar dispositivos
- Aprender a utilizar XBee para comunicación inalámbrica de dispositivos
- Aprender a utilizar un acelerómetro

2. Esbozo de la solución

En ésta práctica se utilizarán distintos protocolos de comunicación para la interacción de componentes:

I2C Comunicación serial síncrona maestro-esclavo. Se utilizará para comunicar dos Arduino.

SPI Comunicación serial síncrona maestro-esclavo. Se utilizará para comunicar el acelerómetro con la FPGA.

UART Comunicación asíncrona. Se utilizará para comunicar el Arduino con un Xbee utilizando un *shield*

Zigbee Comunicación inalámbrica utilizada para la comunicación entre Xbee. Se utilizará para comunicar un Arduino (con Xbee) con la computadora inalámbricamente.

2.1. Protocolo I^2C

Se utilizará el protocolo I^2C para comunicar dos microcontroladores: A B. Para esto, es necesario juntarse con otro equipo.

- A tiene conectado dos botones.
- B tiene conectado un motor mediante un puente H.
- El motor se mueve de acuerdo a la Tabla 1.

Botones	Acción del Motor
00	Freno pasivo
01	Rota en sentido de las manecillas
10	Rota en contrasentido de las manecillas
11	Freno activo

Tabla 1: Movimiento del motor

2.2. Procotolo SPI & acelerómetro

Se utilizará el acelerómetro de la tarjeta DE0-Nano para establecer la dirección de un motor. El acelerometro se comunica con la FPGA mediante el protocolo SPI. En el siguiente repositorio se encuentran los archivos iniciales: https://github.com/garygra/PM_pract_3. Es necesario leer la hoja de especificaciones del acelerómetro. Para leer los registros, es recomendable utilizar los LEDs de la tarjeta. A continuación se listan los pasos recomendados:

1. Establecer los valores adecuados para el componente *spi_3_wire_master* en *accel_rw.vhd*.
2. Leer el registro que contiene el *ID* del acelerómetro y verificar que el valor es correcto.
3. Determinar los valores adecuados:
   - Deshabilitar las funciones de *TAP* y *DOUBLE TAP*
   - Establecer *BANDWIDTH* de 200Hz.
   - Deshabilitar el modo *LINK*
   - Deshabilitar el modo *AUTOSLEEP*
   - Habilitar el modo *MEASURE*
   - Establecer el acelerómetro en modo *NORMAL*
4. Escribir los valores adecuados a los registros.
5. Leer de los registros para verificar la escritura correcta
6. Leer de los registros los valores para X, Y y Z, usar el *dip switch* para cambiar el valor leido.
7. Utilizando el valor leído de X, Y o Z, prender dos LEDs, cada LED indica la dirección del motor.
8. Reemplazar los LEDs por el motor utilizando un puente H.

2.3. Protocolo UART & XBee

1. Configurar los *XBee* utilizando *XCTU* con los parámetros apropiados.
2. Probar la comunicación de cada *XBee* desde XCTU.
3. Implementar el siguiente programa:
   - Se envian comandos desde XCTU al Arduino.
   - Al recibir un comando, se prende/apaga un LED.
   - Implementar tres comandos, cada uno asociado a un LED distinto.
   - Al presionar un boton, el Arduino le envía un mensaje al otro XBee.

3. Aspectos técnicos

3.1. Protocolo I^2C

El protocolo I^2C utiliza dos lineas para comunicarse: SDA y SCL. Para comunicar los microcontroladores, éstas deben conectarse.

3.1.1. Maestro

- Es necesario incluir el *header* de la librería Wire
- Wire.begin() - función que se debe llamar una vez
- Wire.requestFrom(X, N) - pedir N bytes al esclavo X
- Wire.beginTransmission(X) - iniciar transmisión al esclavo X
- Wire.endTransmission(X) - terminar transmisión al esclavo X
- Wire.available() - análoga a la función de *Serial*
- Wire.read() - análoga a la función de *Serial*
- Wire.write(data) - análoga a la función de *Serial*

3.1.2. Esclavo

- Es necesario incluir el *header* de la librería Wire
- Wire.begin(esclavo_num) - función que se debe llamar una vez con el número del esclavo
- Wire.onReceive(function) - registra la función que se va a llamar al recibir una transmisión
- Wire.onRequest(function) - registra la función que se va a llamar al tener una solicitud de transmisión
- Wire.available() - análoga a la función de *Serial*
- Wire.read() - análoga a la función de *Serial*
- Wire.write(data) - análoga a la función de *Serial*

3.1.3. Puente H

Para conectar el motor es necesario utilizar el puente H *L293*. En la hoja de especificaciones del puente H se especifica la forma de conectar el motor. Debido al alto consumo de corriente del motor, se debe alimentar utilizando las fuentes de voltajo, **NO** directamente desde el Arduino.

3.2. Acelerómetro & SPI

- Para determinar las señales y valores adecuados para el componente *spi_3_wire_master* se debe utilizar la *datasheet* del acelerómetro así como la referencia del módulo.
- El acelerómetro reporta valores a una frecuencia específica, por lo que la FPGA no puede leer valores más rápido de esta frecuencia. Es necesario establecer la lectura de los registros X, Y o Z a una frecuencia específica. Esto es, pasar del estado *IDLE* a *RD_PARAMS_SEND* a una frecuencia específica. Si esto no se hace, se van a reportar valores erroneos.

3.3. Configuración de XBee

Para comunicar XBee's, es necesario configurar los siguientes parámetros adecuadamente:

- CH: Todos los XBees deben estar en el mismo canal para comunicarse entre ellos.
- ID: Todos los XBees deben tener el mismo identificador para comunicarse entre ellos.
- DH + DL: Dirección destino, dividida en dos partes de 32 bits cada una, es la dirección del XBee al cual se le enviarán datos.
- MY: Establece una dirección fuente de 16 bits.
- SH + SL: Número serial, es la dirección de 64 bits única para cada XBee.
- BD: La comunicación entre todos los XBee se debe establecer a la misma tasa.

3.4. UART

Para implementar UART se va a utilizar la librería *SoftwareSerial* utilizando el shield para XBee.

1. Inlcuir la librería

```
#include <SoftwareSerial.h>
```

2. Declarar como variable global:

```
SoftwareSerial xbee_serial(RX, TX)
```

   Donde RX es el pin que recibe datos y TX el pin que transmite datos. Para el Arduino Mega, RX solamente puede ser: 10, 11, 12, 13, 14, 15, 50, 51, 52, 53, A8 (62), A9 (63), A10 (64), A11 (65), A12 (66), A13 (67), A14 (68), A15 (69).

3. Iniciar la comunicación serial en la función *setup*:

```
xbee_serial.begin(baud_rate);
```

4. Algunas de las funciones de *SoftwareSerial* que se pueden utilizar en el *loop* son:
   - available() - Devuelve el número de bytes disponibles en el buffer.
   - read() - Devuelve el byte recibido en RX.
   - write(data) - Envía los datos a transmitir al pin TX.

Referencias Recomendadas

Guía de usuario de XBee/XBee Pro S1 https://www.digi.com/resources/documentation/digidocs/PDFs/90000982.pdf

Librería Software Serial https://www.arduino.cc/en/Reference/SoftwareSerial

Comunicación Serial https://learn.sparkfun.com/tutorials/serial-communication

I²C https://www.i2c-bus.org/

SPI https://www.digikey.com/eewiki/pages/viewpage.action?pageId=27754638

Puente H http://www.ti.com/lit/ds/symlink/l293.pdf