Practica No.3 Protocolos de Comunicación

Marco A. Morales Aguirre, Jose Guadalupe Romero
Edgar Granados, J. Carlos Urteaga Reyesvera
Instituto Tecnológico Autónomo de México

2018

1 Objetivo

- Aprender el funcionamiento de los protocolos de comunicación más populares para sistemas integrados (*embedded systems*)
- Aprender el funcionamiento básico de XBee.

2 Problema

- Se requiere comunicar múltiples dispositivos para que intercambien información,

3 Esbozo de la solución

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 para tres LEDs.
   - Al presionar un boton, el Arduino le envía un mensaje al otro XBee.
4. Utilizar dos Arduinos A y B (juntarse con otro equipo) e implementar lo siguiente utilizando I^2C:
   - A tiene conectado un XBee.
   - B tiene conectado tres LEDs y un botón.
   - A recibe los comandos enviados desde XCTU y los envía a B mediante I^2C.
   - B recibe los comandos de A y realiza la acción adecuada.
   - Al presionar el botón, B envía a A un comando.
   - Al recibir un comando de B, A redirije el comando a la computadora y se despliega en XCTU.

3.1 Extra

1. (0.2 pts) Implementar un nodo de ROS que se le envíe comandos al Arduino y éste los retransmita a un XBee. Los comandos deben verse utilizando XCTU.
2. (0.8 pts) Implementar un nodo de ROS que le envíe comandos a un XBee y éste los retransmita al Arduino. Al recibir un comando, un LED conectado al Arduino cambia su estado. (Investigar cómo conectar ROS con XBee).

4 Aspectos técnicos

4.1 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.

4.2 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.

4.3 I^2C

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

4.3.1 Maestro

1. Incluir la librería *Wire*

```
#include <Wire.h>
```

2. En la función setup, inicializar *wire*

```
Wire.begin();
```

3. Dentro de *loop*, para pedirle N bytes al esclavo X:

```
Wire.requestFrom(X, N);
```

4. Las funciones available(), read() y write(data) son análogas a las de la librería *SoftwareSerial*.

5. Para utilizar write(data) es necesario iniciar la transmisión al esclavo X y terminarla:

```
Wire.beginTransmission(X);
Wire.write(data);
Wire.endTransmission();
```

4.3.2 Esclavo

1. Incluir la librería *Wire*

```
#include <Wire.H>
```

2. En la función *setup*, se debe unir al bus I^2C con la dirección *dir* y establecer las funciones que manejarán las interrupciones al recibir datos y al solicitarse datos.

```
Wire.begin(X);
Wire.onReceive(receive_function);
Wire.onRequest(request_function);
```

3. Algunos de los métodos que se pueden utilizar en las funciones que manejan cada interrupción son available(), read() y write(data), análogas a las de la librería *SoftwareSerial*.

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
- I^2C: https://www.i2c-bus.org/