Instituto Universitario Aeronáutico

Especialización en Sistemas Embebidos

Entornos Inalámbricos

Trabajos Prácticos

Alumno:
Luis Alberto Guanuco
Santiago Nicolás Nolasco
Sebastian Aguero
Franco Bocalon

Docentes:
Jose Duclox

Septiembre 2016

1. Introducción

Se realizarán ejercitaciones sobre plataformas XBee con el objetivo de entender los *Entornos Inalámbricos*. Se trabajará con los modos *AT* y *API* a fin de comprar las ventajas y desventajas de estos. Estos ejemplos prácticos servirán, además, alcanzar la resolución del *Trabajo Final*.

1.1. Módulos XBee S2C

Para el establecimiento de una red basado en los estándares 802.15.4 se plantea como requisito la disponibilidad de módulos que implementen dicho estándar. En nuestro caso se utilizarán los módulos *XBee S2C*[1]. Las principales características de estos dispositivos son:

- Velocidades de datos
  - RF 250 Kbps
  - Comunicación serial hasta 1 Mbps
- Alcances
  - *Indoor* 60 metros
  - *Outdoor* 1200 metros
- Potencia de transmisión 3.1 mW (+5 dBm). En modo *Boost* 6.3 mW (+8 dBm)
- Sensibilidad de recepción -100 dBm. EN modo *Boost* -102 dBm
- Banda de frecuencia 2.4 GHz (16 canales)
- 15 puertos digitales I/O (4 entradas analógicas)
- Alimentación 2.1V a 3.6V. Consumos:
  - En la transmisión 33 mA @ 3.3 VDC. En modo *Boost*45 mA
  - En la recepción 28 mA @ 3.3 VDC. En modo *Boost* 31 mA

Los puntos anteriores solo describen las principales características de los módulos XBee 2SC. Para conocer más en detalle se puede acceder a la hoja de datos disponible en el sitio web del fabricante [1].

1.2. Plataformas adicionales

Se utilizará *hardware* adicional que permita establecer la comunicación de los módulos XBee con una computadora. Sí bien el Laboratorio de Sistemas embebidos pone a disposición las plataformas XBoard[2] se necesita adaptar la comunicación serial del módulo XBee con la la computadora mediante una puerto USB. Para esto se desarrolló la placa *XBee-MCP Adapter* (Figura 1). Sobre esta placa se montarán dos módulos, una

[1] http://www.digi.com/support/productdetail?pid=4838
[2] http://www.cika.com/soporte/Information/Digi-RF/XBee/

es el módulo XBee y el otro es la placa *MCP2200 Breakout Module*[3]. Este último proporciona una comunicación serial USB-UART. Además se agregaron dos LEDs conectados a los puertos `AD1/DIO1` y `AD2/DIO2`. Opcionalmente se puede conectar un potenciómetro al conector `K1` que se encuentra mapeado al puerto analógico de la XBee, `AD0/DIO0`.

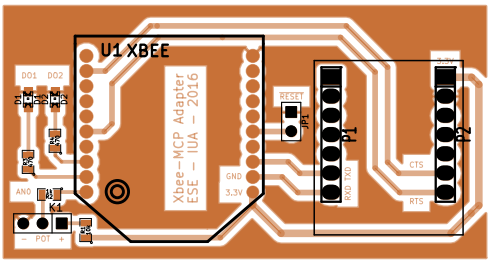

Figura 1: Placa XBee-MCP Adapter.

La Xboard, anteriormente mencionada, es una plataforma desarrollada por la empresa Cika Electrónica SRL. La XBoard permite interconectar los módulos XBee con dispositivos externos [2]. En la Figura 2 se presenta una vista superior de la XBoard. Se puede ver que esta placa tiene un conector hembra con las señales de comunicación serial, niveles de tensión y otras más. Para conectar la XBoard a una computadora se utiliza la placa MCP2200 Breakout Module. Al no requerir componentes externos, se utilizan simplemente cables que realicen la interconexión de las señales de comunicación serial.

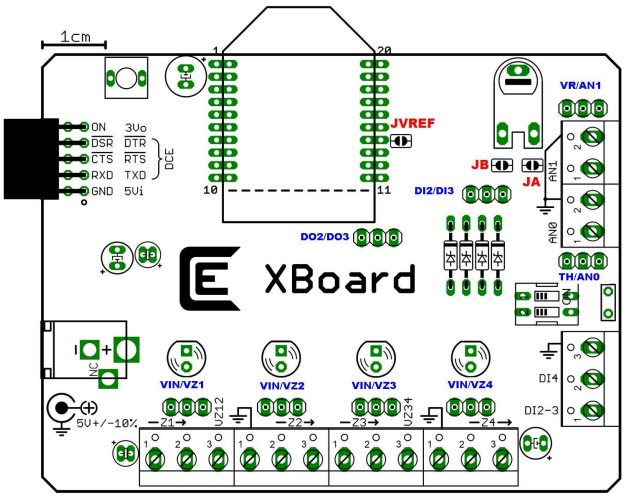

Figura 2: Placa XBoard.

Todas las características y configuraciones de las diferentes plataformas comerciales se encuentran en la bibliografía referenciada al final de este documento.

Se desarrollarán tres tipos de ejercitaciones. La primera sobre los alcances de RF que proporcionan estos dispositivos. La segunda parte sobre el manejo de los módulos en modo AT. Y por último se trabajará en modo API con los módulos sobre una red ZigBee

[3] http://www.microchip.com/DevelopmentTools/ProductDetails.aspx?PartNO=adm00393

completa. Se documentarán todos las implementaciones y por último se presentará una conclusión sobre lo hecho.

2. Análisis de enlaces

Completar con lo de santiago.

3. Ejercitación en modo AT

El set de comandos AT, también conocidos como set de comandos Hayes, fue originalmente desarrollado para usar con los modem de Hayes en la década de 1980. Muchos modems modernos todavía utilizan este estándar. EL término *comando AT* viene de usar los caracteres ASCII para notificar al *host* que un le sigue un comando.

En el caso de los módulos XBee desarrollados por Digi, implementa un set de comandos propieratrios para interactuar con los módulos de Digi a través de una comunicación serial. Basado en el set de comandos AT, Dii usa los caracteres `AT` antes de cada comando enviado al modem. Un ejemplo de un comando utilizado por el radio Digi XBee podría ser `ATCH`. Este comando es usado para leer o setear el canal que un radio XBeee es configurado[3].

3.1. Configuración de puertos GPIO

La configuración de los diferentes canales de propósitos generales se puede realizar utilizando el software XCTU en modo gráfico pero el objetivo es realizarlo mediante el set de comandos AT.

3.1.1. Comando ATIS

El comando `ATIS` fuerza la lectura de todos los canales habilitados. Analógicos como digitales. En nuestro caso tenemos la siguiente salida a la petición.

```
1 +++
2 OK
3 ATIS
4 01
5 0006
6 01
7 0000
8 0219
```

La respuesta que entrega el módulo comienza en la línea **4** hasta la **8**. El primer byte `01` es la cantidad de muestra recibidas. En la línea **5** se tiene configuración de los canales digitales y la línea **6** canales analógicos. Para entender se debe analizar a nivel de bits cada una de las respuestas. Recuerde que los valores representados están en hexadecimal. En el primer se tiene $0006_{HEX} = 000000000000110_{BIN}$" por lo los canales `DIO1` y `DIO2` se encuentran habilitados y configurados como salidas digitales. Mientras que para los canales analógicos tenemos $01_{HEX} = 0001_{BIN}$ solo el puerto `AD0` habilitado. Para terminar el análisis de la respuesta al comando `ATIS`, las líneas **7** y **8** son el estado de los canales digitales y analógicos respectivamente. Como se mencionó anteriormente, puede

visualizarse la configuración de los canales de I/O en forma gráfica desde el software XCTU. En la Figura 3 se puede ver la misma información que proporciona el comando `ATIS`.

Figura 3: Configuración de los GPIO en modo gráfico.

Para dar comienzo con la ejercitación pedida por los docentes, se deshabilitarán todos los canales de forma tal que el módulo quede sin ningún GPIO en uso. Sí la configuración es tal como la descrita anteriormente, se deben aplicar los comandos siguientes.

```
+++OK
ATD00
OK
ATD10
OK
ATD20
OK
ATAC
OK
ATWR
OK
ATIS
ERROR
```

Para deshabilitar un canal digital/analógico se debe simplemente pasar como último argumento el valor `0`. Esto se aplica a los canales `DIO0`, `DIO1` y `DIO2`. Los comandos que le siguen son para aplicar los cambios y liberar los *buffers* (`ATAC`) y escribir la memoria no-volatil del módulo (`ATWR`). Al finalizar los cambios se envía el comando `ATIS` y se tiene como respuesta `ERROR` esto no se debe a un problema de comunicación sino que al no existir ningún canal habilitado, no puede ofrecer información alguna.

3.1.2. Configurar dos canales analógicos

4. Ejercitación en modo API

En la Sección **??** se listó los requerimientos del proyecto. Una tarea del RTOS debía ser la encargada de realizar la lectura de una señal analógica cada 100 Hz. Las características para esta tarea se definen en el lenguaje OIL.

Declaración del objeto `Analogic` en el lenguaje OIL

```
TASK Analogic {
    PRIORITY = 1;
    ACTIVATION = 1;
    STACK = 512;
    TYPE = EXTENDED;
    SCHEDULE = FULL;
    RESOURCE = POSIXR;
    EVENT = POSIXE;
    AUTOSTART = TRUE {
```

```
10          APPMODE = AppMode1;
11          ALARMTIME = 1;
12          CYCLETIME = 1;
13      }
14  }
15
16  RESOURCE = POSIXR;
17  EVENT = POSIXE;
18
19  ALARM AnalogicAlarm {
20      COUNTER = SoftwareCounter;
21      ACTION = ACTIVATETASK {
22          TASK = Analogic;
23      }
24  }
```

La periodicidad del evento se configura en el archivo `main.c` del proyecto. En la línea **20** se define la base de tiempo será un contador generado por el OS es decir `SoftwareCounter`.

Código de la tarea `Analogic` en el `main.c`

```
1   TASK(Analogic)
2   {
3      uint16_t hr_ciaaDac;
4      uint8_t outputs;
5
6      /* Read ADC. */
7      ciaaPOSIX_read(fd_adc, &hr_ciaaDac, sizeof(hr_ciaaDac));
8
9      /* Signal gain. */
10     hr_ciaaDac >>= 0;
11
12     /* Write DAC */
13     ciaaPOSIX_write(fd_dac, &hr_ciaaDac, sizeof(hr_ciaaDac));
14
15     TerminateTask();
16  }
```

5. Conclusiones

La implementación de este RTOS tan específico y novedoso, por lo menos para nosotros, nos ha permitido ampliar la concepción de sistemas operativos en tiempo real. A la vez que se ha podido interactuar con profesionales de la región que están en pleno desarrollo y aportes para mejorar el código utilizado (freeOSEK).

Referencias

[1] Digi International, *XBee®/XBee-PRO ZigBee® RF Module – User Guide*. 2016.

[2] Cika Electrónica SRL., *XBoard [-W]*. http://www.cika.com/soporte/Information/Digi-RF/XBee/XBoard_doc.pdf.

[3] Digi International, *Knowledge Base – The AT Command Set*. http://knowledge.digi.com/articles/Knowledge_Base_Article/The-AT-Command-Set.