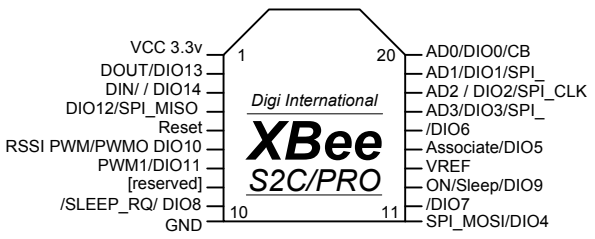
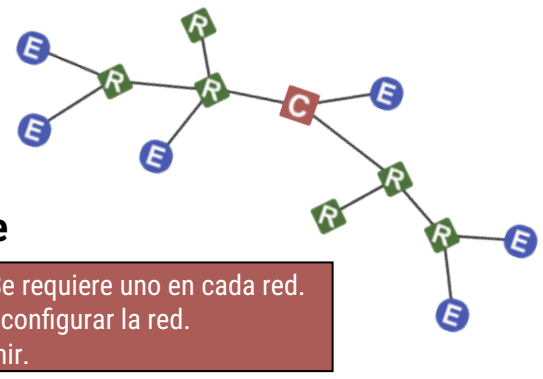
GUÍA DE REFERENCIA PARA **XBee S2C PRO**



XBee es un microcontrolador fabricado por digi, el cual utiliza el protocolo Zigbee. XBee utiliza 3.3V y tienen un espaciado entre pines menor que el utilizado en un protoboard. Dado lo anterior es recomendable utilizar una tarjeta o kit para que sea más fácil su uso.



Roles XBee

Coordinador: Se requiere uno en cada red.
Se encarga de configurar la red.
No puede dormir.

Router: Pueden existir multiples en una red. Pueden redirigir los mensajes a otros routers o End Devices. No pueden dormir.

End Device: Pueden existir muchos, no pueden redirigir mensajes.
Pueden dormir para ahorrar energía.

S2C

Especificaciones			
	Voltaje de operación: 2.1 – 3.6V Corriente de operación: 33mA@3.3V Rango en interiores: 60 metros Rango en línea vista: 1200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C

S2C PRO

Especificaciones			
	Voltaje de operación: 2.7 – 3.6V Corriente de operación: 31mA@3.3V Rango en interiores: 90 metros Rango en línea vista: 3200 metros Max valor lectura analógica: 1.2V	Pines I/O: 13 Entradas analógas: 4 *Configurable como red mesh* *Red auto reparable*	Firmware: Zigbee, Digimesh, 802.15.4 Velocidad de transmisión RF: 250kbps Frecuencia: 2.4GHz - 2.5GHz Temperatura operación: -40 a 85°C

Modos XBee

Transparente: Los dispositivos actúan como un reemplazo de cable serial. Cuando los datos RF son recibido, el dispositivo envía los datos a través del puerto serie. Utilice la interfaz de modo de comando AT para configurar los parámetros del dispositivo.
Comando: Basada en tramas, amplía el nivel en que una aplicación host puede interactuar con las capacidades de red del dispositivo. Cuando esta en modo API, el dispositivo contiene todos los datos que entran y salen en marcos que definen operaciones o eventos dentro del dispositivo.

Setup XBee

Conecta el Xbee a un adaptador TTL a Serial como un FTDI
Utiliza el software gratuito X-CTU para configurar el módulo XBee
Baud:9600 – FC: Hardware – Data Bits 8 – Parity: None – Stop Bits: 1

Ajustes Básicos

PAN ID: es la red a la cual se conectará el módulo. Si es 0, el XBee se asociará a cualquiera que esté disponible
DH/DL: Es la dirección del módulo de destino. Se utiliza para enviar información a un XBee en específico. Si se configura en 0 enviará datos solo al coordinador. Si se configura en 0x0000000000FFFF hará un broadcast (envío a todos los módulos de la red)

Ajustes Pin

Para poder trabajar con los pines como entradas/salidas en un XBee, debe estar configurado en modo API.
D0 - Configura el pin en 0 para comenzar a leer datos
IR - realiza una lectura del pin cada XX milisegundos

Trama API para envío de comandos AT remotos

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio – indica el comienzo del paquete de datos (frame)
1	0x00	Largo – Número de bytes (ChecksumByte# – 1 – 2)
2	0x10	
3	0x17	Tipo de mensaje - 0x17 significa que es solicitud de comando AT
4	0x01	Frame ID – secuencia del paquete
5	0x00	Dirección de destino de 64-bi t(número de serie)
6	0x13	MSB es el byte 5, LSB es el byte 12
7	0xA2	
8	0x00	0x0000000000000000 = Coordinador
9	0x40	0x000000000000FFFF = Broadcast
10	0x8B	
11	0x78	
12	0x4E	
13	0xFF	Dirección de la red de destino
14	0xFE	(configúralo como 0xFFFE para enviar un bodcast)
15	0x02	Opción del comando remoto (configuralo como 0x02 para aplicar los cambios)
16	0x44 (D)	Nombre del comando AT (Dos caracteres ASCII)
17	0x34 (2)	
18	0x05	Parámetro del comando
19	0x25	Checksum

Conexión con Arduino:
Arduino TX se conecta la RX de XBee (Data IN)
Arduino RX se conecta al TX de XBee (Data Out)

Integración con Arduino:
Los datos enviados utilizando Serial.print() saldrán por el puerto TX del Arduino, que estará conectado al RX del módulo XBee. Si XBee está en modo AT, se transmitirá inalámbricamente hacia el destino. Los datos recibidos en el módulo XBee serán enviados al puerto serial.

Ejemplo para Arduino: Lectura de un valor análogo utilizando modo API

```
// XBee remoto: AT, XBee base: API
if (Serial.available() >= 21) { // Nos aseguramos que ha llegado el mensaje completo
if (Serial.read() == 0x7E) { // 7E es el byte de inicio
  for (int i = 1; i<19; i++) { // descartamos los bytes hasta llegar los datos análogos
    byte discardByte = Serial.read();
}
int analogMSB = Serial.read(); // Lee el primer byte del dato análogo
analogLSB = Serial.read(); // Lee el segundo byte del dato análogo
int analogReading = analogLSB + (analogMSB * 256); }
 }
}
```

Ejemplo Arduino: Cambiar la configuración de un pin en un XBee remoto

```
// XBee remoto: AT, XBee base: API
Serial.write(0x7E); // byte de inicio
Serial.write((byte)0x0); // Largo MSB (siempre 0)
Serial.write(0x10); // Largo LSB
Serial.write(0x17); // 0x17 es el tipo de mensaje para enviar comandos AT
Serial.write((byte)0x0); // Frame ID (no solicitamos repuesta)
Serial.write((byte)00); // Envía los 64 bit de la dirección de destino
Serial.write((byte)00); // (Enviando 0x000000000000FFFF (broadcast))
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write((byte)00);
Serial.write(0xFF);
Serial.write(0xFF);
Serial.write(0xFF); // Red de destino
Serial.write(0xFE); // (enviar 0xFFFE si es desconocida)
Serial.write(0x02); // configurar 0x02 para aplicar los cambios
Serial.write('D'); // Comando AT : D1
Serial.write('1');
Serial.write(0x05); // Configura D1 para ser 5  (Digital Out HIGH)
long chexsum = 0x17 + 0xFF + 0xFF +0xFF + 0xFE + 0x02 + 'D' + '1' + 0x05;
Serial.write( 0xFF – (chexsum & 0xFF)); // Checksum
```

Trama API para recepción de datos I/O

Byte	Ejemplo	Descripción
0	0x7E	Byte de inicio - indica el comienzo del paquete de datos (frame)
1	0x00	
2	0x14	
3	0x92	Tipo de frame 0x92 indica que es un muestreo de las entradas del XBee
4	0x00	Dirección de origen de 64-bit (número de serie)
5	0x13	MSB es el byte 4, LSB es el byte 11
6	0xA2	
7	0x00	
8	0x40	
9	0x8B	
10	0x78	
11	0x4E	
12	0xA4	Dirección de 16-bit de la red de origen
13	0x02	
14	0x01	Opciones de recepción: 01 = Packet acknowledged 02 = Broadcast packet
15	0x01	Número de muestras. Siempre debe ser 1 dadas las limitaciones de XBee
16	0x00	Máscara para el canal digital, indica que pines estan configurados como DIO
17	0x30	
18	0x01	Máscara para el canal análogo, indica cuales pines están configurados como ADC
19	0x00	Lectura de los canales digitales. Estos dos bytes contienen los estados de
20	0x20	los pines configurados como DIO
21	0x02	Lectura del canal análogo.
22	0x0C	Cada canal entrega 2 bytes con el resultado de la lectura del ADC
23	0x20	Checksum (0xFF - la suma de todos los bytes desde el byte 3 a hasta el último)

Modo Sleep

End Device puede dormir para ahorrar energía. Un End Device que solo despierta cada 5 minutos para enviar datos puede solo estar despierto por 6 segundos en un día.

SM – 4 = Cyclic Sleep
SP - Sleep time (hasta 28 segundos)
SN - Número de ciclos sleep
ST – Tiempo que permanecerá despierto

Pin I/O Opciones

0 –Disabled
1 –N/A
2 –ADC
3 –Digital IN
4 –Digital OUT, LOW
5 –Digital OUT, HIGH

Máscara para canal digital

Primer Byte
n/a n/a n/a D12 D11 D10 n/a n/a
Segundo Byte
D7 D6 D5 D4 D3 D2 D1 D0
Ejemplo:
0x00 0x13 = 0000 0000 0000 1101
Pins D3, D2 y D0

Máscara para canal análoogo

(volt) n/a n/a n/a A3 A2 A1 A0
Ejemplo:
0x05 = 0000 0101 = Pin A2 and A0