

Comunicación Xbee en modo API

Teoría de la Conmutación

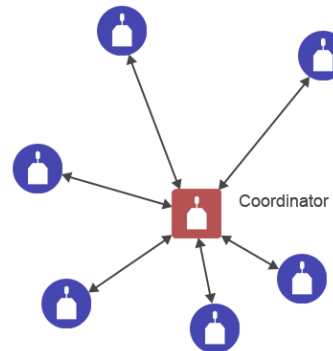
¿Qué es una trama API?

2

Las tramas api (Application Programming Interface) es una manera de transmitir los datos de una manera segura y estructurada. La información que envía y recibe cada xbee tiene que seguir una estructura específica y debe estar empaquetado en tramas (frames), que definen operaciones y eventos dentro del módulo.

Los Xbee son capaces de establecer conexiones de redes tipo “malla”; en este tipo de red existe un nodo coordinador, el cual se encarga de pasar los mensajes de un nodo a otro.

A diferencia del modo transparente, el modo API hace que el módulo XBee espere por una secuencia específica de bytes que le indican que tipo de operación deberá de realizar. Cada una de estas secuencias se le conoce como "frame" o también le podemos llamar paquete



API FRAME NAMES	API ID
AT Command	0x08
AT Command – Queue Parameter Value	0x09
ZigBee Transmit Request	0x10
Explicit Addressing ZigBee Command Frame	0x11
Remote Command Request	0x17
Create Source Route	0x21
AT Command Response	0x88
Modem Status	0x8A
ZigBee Transmit Status	0x8B
ZigBee Receive Packet (AO=0)	0x90
ZigBee Explicit Rx Indicator (AO=1)	0x91
ZigBee IO Data Sample Rx Indicator	0x92
Xbee Sensor Read Indicator (AO=0)	0x94
Node Identification Indicator (AO=0)	0x95
Remote Command Response	0x97
Over-the-Air Firmware Update Status	0xA0
Rote Record Indicator	0xA1

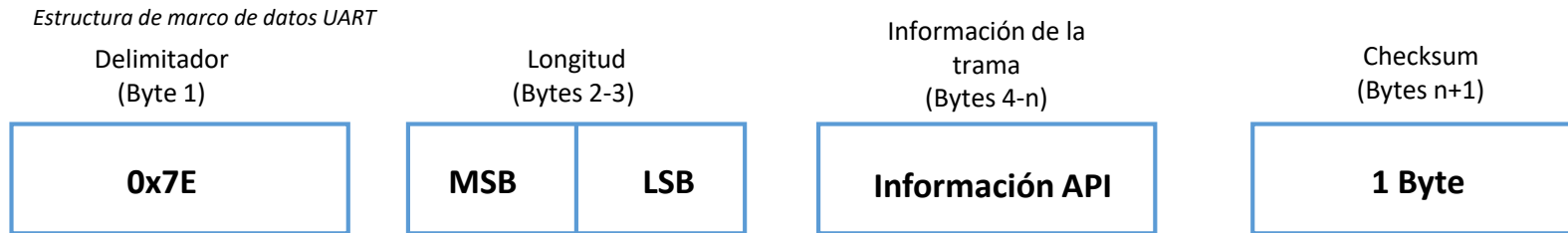
Fig.45 Diferentes tipos de trampas API en el protocolo ZIGBEE

Para comunicarnos entre dos XBee dentro de la red mallada necesitamos conocer la dirección del XBee destino al cual queremos enviar los datos. Cada XBee posee una dirección única de hardware asignada, de esta manera no hay dos módulos XBee con la misma dirección.



Estructura general de una trama api

Cuando este modo Api está habilitado (AP= 1), la estructura del marco de datos UART se define como :

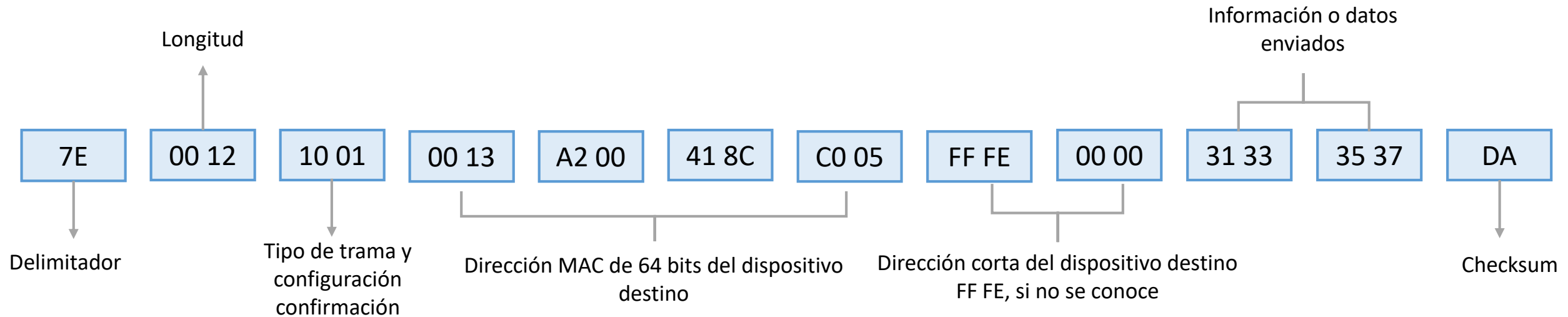


Cualquier dato recibido antes del delimitador se descarta silenciosamente. Si la trama no se recibe correctamente o si la suma de comprobación falla, el módulo responderá con una trama de estado del módulo que indica la naturaleza de la falla.

Trama API 10

A P I	Campos de la trama		Offset	Ejemplo	Descripción
	Delimitador		0	0X7E	Delimitador de inicio
	Longitud		MSB 1	0X00	Longitud de la trama (A partir de 10 y sin contar el checksum).
			LSB 2	0X16	
	Datos específicos de la estructura	Clase de estructura	3	0X10	Tipo de trama
		Estructura ID	4	0X01	Identifica la estructura de datos UART para el servidor con un posterior ACK (reconocimiento). Si establece en 0, no envía respuesta
		64-bit Dirección de destino	MSB 5	0X00	Establece en la dirección de 64 bits del dispositivo de destino
			6	0X13	
			7	0XA2	
			8	0X00	
			9	0X40	
			10	0X0A	
			11	0X01	
			12	0X27	
		16-bit Dirección de red de destino.	MSB 13	0XFF	Establece en la dirección de 16 bits del dispositivo de destino, si se conoce.
			LSB 14	0XFE	
		Broadcast Radius	15	0X00	Numero de saltos desde el coordinador al Router
		Opciones	16	0X00	Byte obligatorio en ceros
		RF DATA	17	0X54	Datos que se envían al dispositivo de destino.
			18	0X78	
			19	0X44	
			20	0X61	
			21	0X74	
			22	0X61	
			23	0X30	
			24	0X41	
	Checksum		25	0X13	Byte de verification de la trama enviada.

Ejemplo trama tipo 10



Contenido de la trama

8

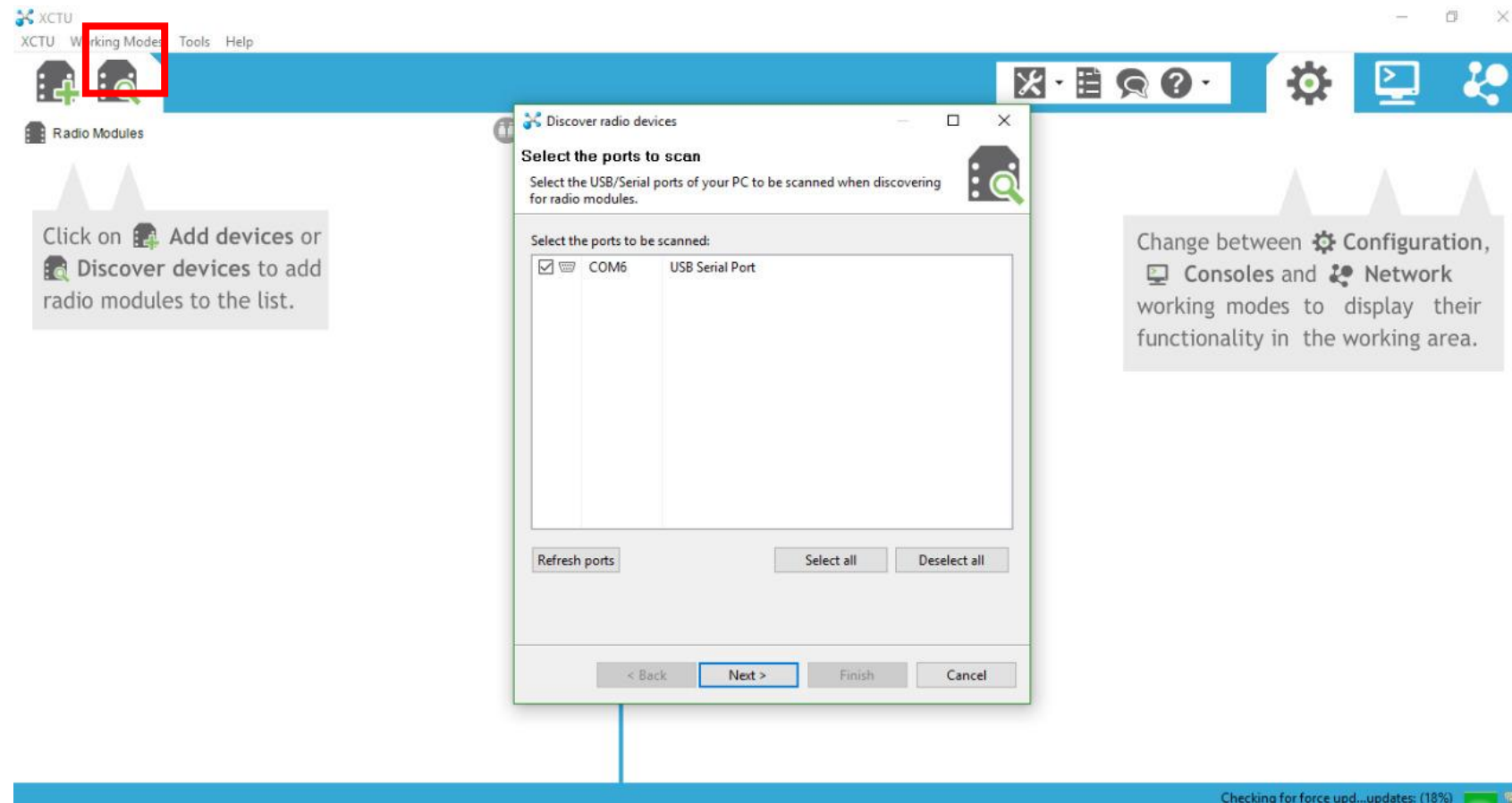
Analicemos el siguiente tipo de trama:

7E 00 12 10 01 00 13 A2 00 41 86 60 7F FF FE 00 00 30 33 35 37 FB

La trama es hexadecimal y significa lo siguiente:

- 7E es el delimitador de las tramas API, toda trama API debe empezar por 7E
- 00 12 son los dos bytes que representan la longitud de la trama, quiere decir que a partir de ahí el numero de datos es 12H, o sea 18 en decimal, sin contar el ultimo byte que es el algoritmo de verificación de que la información llego completa (Checksum)
- 10 01 Es el tipo de trama de API y esta configurado para recibir confirmación.
- 00 13 A2 00 41 8C C0 05 son la dirección o serial del módulo fuente, es decir el ROUTER que envía la información. Parte alta SH y baja SL respectivamente.
- FF FE Es la dirección corta o de 16 bits del módulo fuente asignada por el coordinador, como no se conoce se pone FF FE
- 00 00 Corresponde al numero de saltos desde el coordinador hasta el Router y el byte obligatorio.
- 31 33 35 37 Corresponde a la información enviada
- DA Corresponde al Checksum

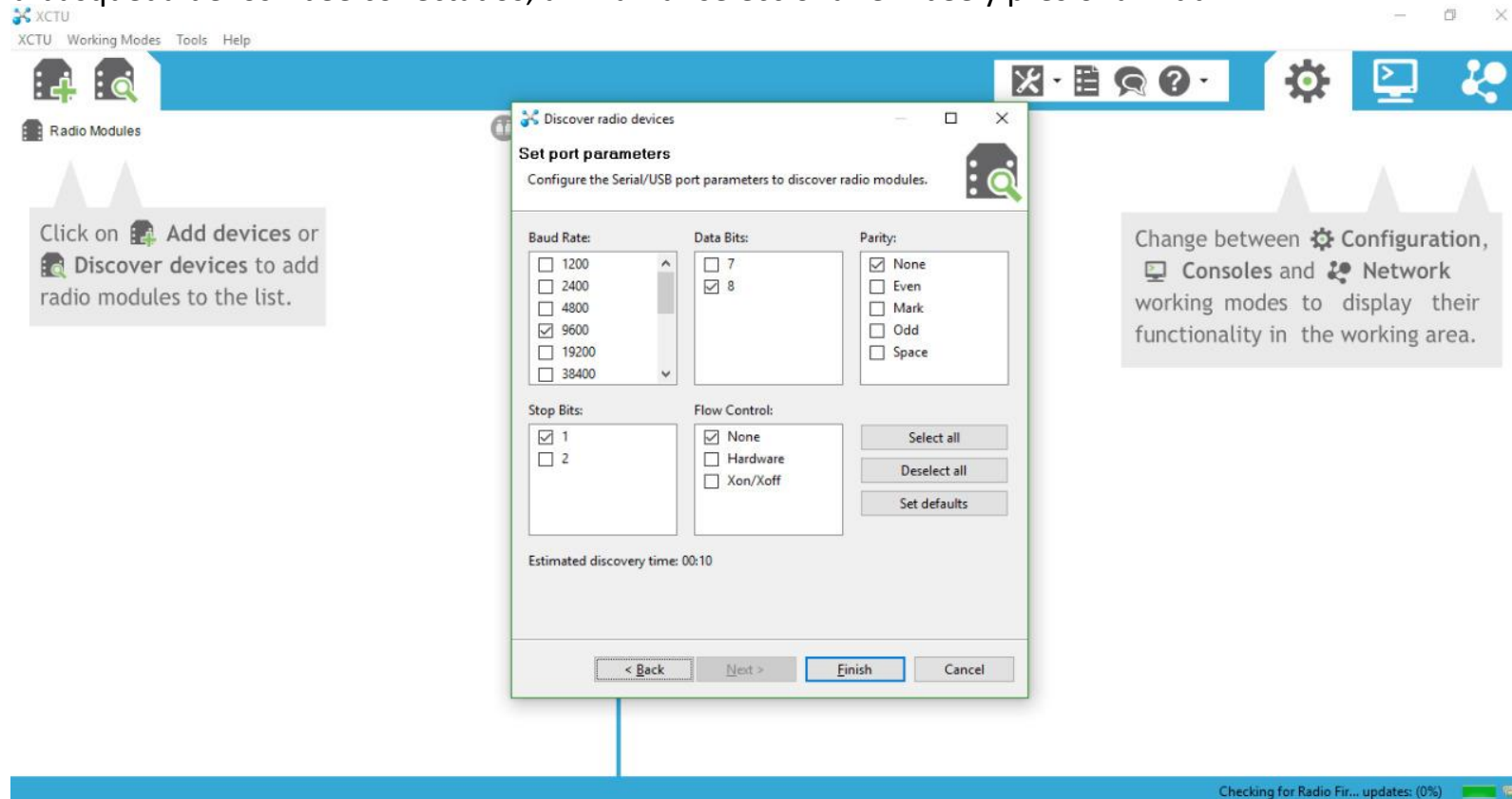
Presionamos el icono para buscar, seleccionamos el puerto y Next.



Ejemplo

12















Seleccionamos la velocidad de transmisión de 9600 Baudios y dejamos los demás parámetros iguales. Presionamos finalizar y empezará la búsqueda de los Xbee conectados, al finalizar seleccionar el Xbee y presionar Add.



Se configura el parámetro ID(Network ID) con un número de 4 dígitos con el que deberán estar configurados todos los Xbee conectados a nuestra trama API.

▼ **MAC/PHY**















Change MAC/PHY Settings

i	AF Available Frequencies	FFFFFFFFFFFFFFFF	
i	CM Channel Mask	<input type="text" value="00FFFFFFFFF7FFFF"/>	 
i	MF Minimum Frequencies	19	
i	HP Preamble ID	<input type="text" value="0"/> ID	 
i	ID Network ID	<input type="text" value="1226"/>	 
i	MT Broadcast Multi-Transmits	<input type="text" value="3"/>	 
i	PL TX Power Level	Highest [4] ▼	 
i	RR Unicast Retries	<input type="text" value="A"/> Retries	 

Configuramos el parámetro BD(Baud Rate) en 9600[3], y el parámetro AP(API Enable) con la opción "API Mode Without Escapes[1]".

Serial Interfacing

Change module interfacing options


















i	BD Baud Rate	9600 [3]		 
i	NB Parity	No Parity [0]		 
i	SB Stop Bits	One stop bit [0]		 
i	RO Packetization Timeout	3	* character times	 
i	FT Flow Control Threshold	13F	Bytes	 
i	AP API Enable	API Mode Without Escapes [1]		 
i	AO API Options	API Rx Indicator - 0x90 [0]		 

Verificamos que los parámetros DH(Destination Address High) y DL(Destination Address Low) estén configurados en 0.

Configuramos el parámetro NI(Node Identifier) con el nombre que queremos dar al Xbee.

▼ Addressing

Change Addressing Settings

i SH Serial Number High	13A200		
i SL Serial Number Low	417FBBED		
i DH Destination Address High	<input type="text" value="0"/>		 
i DL Destination Address Low	<input type="text" value="0"/>		 
i TO Transmit Options	<input type="text" value="C0"/>		 
i NI Node Identifier	<input type="text" value="Coordinador"/>		 
i NT Network Discovery Back-off	<input type="text" value="82"/> * 100 ms		 
i NO Network Discovery Options	<input type="text" value="0"/>		 
i CI Cluster ID	<input type="text" value="11"/>		 

Ejemplo en Arduino del emisor

```
void setup() {  
  Serial3.begin(9600);  
  Serial.begin(9600);  
}  
void loop() {  
  
  Serial3.write(byte(0x7E)); // Encabezado de la trama.  
  
  Serial3.write(byte(0x00));  
  Serial3.write(byte(0x13));  
  Serial3.write(byte(0x10));  
  Serial3.write(byte(0x01));  
  
  // Dirección MAC del XBee que recibirá la trama.
```

```
Serial3.write(byte(0x00));  
Serial3.write(byte(0x13));  
Serial3.write(byte(0xA2));  
Serial3.write(byte(0x00));  
Serial3.write(byte(0x40));  
Serial3.write(byte(0xB2));  
Serial3.write(byte(0xC5));  
Serial3.write(byte(0x76));  
Serial3.write(byte(0xFF));  
Serial3.write(byte(0xFE));  
Serial3.write(byte(0x00));  
Serial3.write(byte(0x00));
```

// Este es el mensaje.

```
Serial3.write(byte(0x65));  
Serial3.write(byte(0x61));  
Serial3.write(byte(0x66));  
Serial3.write(byte(0x69));  
Serial3.write(byte(0x74));
```

// Checksum.

Serial3.write(byte(0x06));

}

delay(10);

/* En caso de que el XBee receptor haya recibido la trama,

* enviará una trama de confirmación hacia este XBee,

* si eso ocurre, en el Serial3 habrá flujo de datos,

* en caso de que la trama finalice con el checksum correspondiente,

* se sabrá que el otro XBee recibió correctamente la trama.

*/

if (**Serial3.available** () > 0) {

String confirmacion = **Serial3.readString**();

if (confirmacion[confirmacion.**length**() - 1] == 'v') {

Serial.println("El mensaje ha llegado.");

} else {

Serial.println("El mensaje ha llegado.");

}

}

delay(3000);

Ejemplos en Arduino del receptor

17

```
int i;

void setup() {
  Serial.begin(9600);
  Serial3.begin(9600);
}

void loop() {

  if (Serial3.available() > 0) {

    (char)Serial3.read();
    String mensaje = Serial3.readString(); // Se obtiene todo el flujo de bytes que trae la trama.

    String nuevaCadena;

    /* El mensaje se empieza a leer desde el byte 15,
     * los bytes anteriores son información irrelevante
     * para el usuario
     */
```

```

for (i = 14; i < mensaje.length() - 1; i++) {

    nuevaCadena += mensaje[i];

}

Serial.println(nuevaCadena); // Se imprime el mensaje ya procesado.

/* Si el mensaje tiene al menos un byte,
 * significará que ha recibido correctamente la trama
 */
if (nuevaCadena.length() > 0) {
    // 7E 00 07 8B 01 FF FE 00 00 00 76 Trama API de confirmación.
    Serial3.write(byte(0x7E));
    Serial3.write(byte(0x00));
    Serial3.write(byte(0x07));
    Serial3.write(byte(0x8B));
    Serial3.write(byte(0x01));
    Serial3.write(byte(0xFF));
    Serial3.write(byte(0xFE));
    Serial3.write(byte(0x00));
    Serial3.write(byte(0x00));
    Serial3.write(byte(0x00));
    Serial3.write(byte(0x76));
}

}

}

```