salesianos
DOMINGO SAVIO

Grado Superior de Desarrollo de Aplicaciones Multiplataforma

2021-2022

Proyecto Final de Grado

SpotGuard

Yago Villar Gurruchaga

Tutor

Sergio Sánchez Crespo

Agradecimientos

A la innumerable lista de docentes, voluntarios e integrantes de la comunidad salesiana que, con su cariño, experiencia e implicación, han logrado que me encuentre donde me encuentro ahora, con la bolsa llena de herramientas y el privilegio de decidir donde las empleo. El sentimiento de haber subido juntos cada peldaño de esta etapa permanecerá conmigo imperecederamente.

Índice general

Introducción

1.1. Descripción

SpotGuard es un sistema de control remoto para dispositivos Digi Xbee, una gama de chips de radiofrecuencia de la marca Digi International S.A. El sistema pretende facilitar el uso de estos dispositivos a través de conceptos simplificados con respecto a todas las funcionalidades que ofrece esta tecnología. Por lo tanto, el enfoque para este proyecto se centra en la simplicidad, practicad y función didáctica de los diferentes elementos que lo componen. En su fase temprana, SpotGuard ofrece una interfaz con todo lo necesario para configurar e interactuar con una red de pruebas con hasta 3 dispositivos Xbee trabajando en modo transparente.

1.1.1. Características

A continuación se listan las características principales de la versión actual del sistema:

- Emplea 3 dispositivos XBee S2C, uno de ellos en modo Coordinador para la gestión de la comunicación.
- Ofrece una interfaz realizada en C# para el control remoto de la comunicación entre los dispositivos XBee. Esta interfaz tiene como principales objetivos:
  - Permitir el análisis del envío y recepción de los diferentes datos.
  - Activar perfiles de comportamiento automático de los diodos.
  - Realizar pruebas de comunicación.
  - Guiar al usuario para comprender e interactuar con el funcionamiento de los dispositivos XBee.
- Se requiere de un equipo compatible con Windows y la versión 4.8 de .NET Framework para operar con la librería SerialPort de .NET, esta es necesaria para la comunicación entre el programa y el dispositivo coordinador a través del puerto USB.

1.1.2. Recursos empleados

Los recursos necesarios para llevar a cabo este proyecto no van más allá de los dispositivos XBee S2C y los elementos de circuitería pertinentes. No obstante, para este caso se han empleado placas de desarrollo especificas para proyectos formativos. Para el montaje de este sistema se ha empleado lo siguiente:

- 2 dispositivos XBee S2C.
- 2 placas de pruebas XBee Grove Development.
- 1 dispositivo XKB2-Z7T-WZM (Fusión de placa Grove Development y un XBee S2C).
- 2 Resistencias 220 Ω
- 3 Resistencias de 10 KΩ
- 3 Diodos led rojos.
- 3 Protoboard.
- Visual Studio 2019 o anterior y la librería SerialPort de Microsoft compatible con versiones anteriores a la 4.8 de .NET Framework.
- 2 Baterías portátiles de 3.3v, un mayor voltaje puede dañar los dispositivos.

1.2. Desarrollo

Durante el desarrollo de SpotGuard, el proyecto a pasado por diversas fases y enfoques.

1.2.1. Primera fase

Se puso en marcha la investigación y recolección de requisitos para el sistema de iluminación vehicular que figura en el ANEXO II de este documento. Durante esta fase se determinaron los recursos iniciales para comenzar un prototipo con un dispositivo XBee 3, que más adelante se conectaría por Bluetooth a través de una aplicación móvil a otro par de dispositivos luminosos.

1.2.2. Segunda fase

En la segunda fase del proyecto, se detectaron inconvenientes con respecto a los XBee y sus prolongados plazos de entrega, no llegando ninguno antes de un mes. Por lo que la actuación fue trabajar en el desarrollo teórico del sistema vehicular, en la aplicación Xamarin que iba a soportar la comunicación vía Bluetooth y en cualquier aspecto que el equipo pudiere avanzar mientras se producía la llegada del envío.

1.2.3. Tercera fase

En la tercera fase del proyecto se terminaron de definir los handicaps[1] del planteamiento inicial:

- El tiempo de envío de los XBee superaba el mes de entrega en todas las tiendas on-line de componentes electrónicos.
- La placa de pruebas adquirida estaba mal escogida. Por desconocimiento adquirimos una que permitía configurar el dispositivo pero no permitía sacar tomas de E/S sin necesidad de realizar labor de micro soldadura.
- El chip XBee llegó un mes y dos semanas después de pedirlo, con el añadido de que la versión no era la escogida, en lugar del XBee 3, llegó un XBee 2, incompatible con Bluetooth y por lo tanto, incompatible con la solución diseñada.

La fase 3 terminó de dejar claro que debíamos plantear el proyecto de otra manera. Llegados este punto habíamos consumido más de la mitad del tiempo disponible antes de la entrega. por lo que la prioridad paso a ser sacar a delante un desarrollo viable.

1.2.4. Cuarta fase

Gracias a los recursos de facilitados por el colegio, para la cuarta fase conseguimos un Kit Digimesh que trae consigo 3 dispositivos S2C, que aunque siguen sin ser la versión compatible con el planteamiento inicial, nos valían para realizar una demostración sobre el funcionamiento de las redes Zigbee y sentar las bases estructurales sobre cual iba a ser el planteamiento de SpotGuard. Juntamos los los conocimientos adquiridos en el desarrollo de la versión anterior y los recursos que nos fueron facilitados para realizar una primera versión de un sistema de balizamiento con SpotGuard. Una pequeña red que permita entender los principios básicos que hacen funcionar a estos dispositivos y lo potenciales para el mercado que los hace su escalabilidad, sencillez y adaptabilidad al Internet de las cosas.

1.2.5. Quinta Fase

La quinta fase ha consistido en una meticulosa definición de objetivos y la ejecución de los mismos sin exceder el tiempo estipulado. Aun siendo la fase más costosa de todas, pues ha requerido de jornadas intensivas de trabajo, ha sido también la más productiva, satisfactoria y gratificante. Como resultado de esta última fase: Un prototipo funcional de una red inalámbrica montada con dispositivos XBee S2C y el resumen de todo lo que hemos podido aprender durante nuestra formación en esta tecnología.

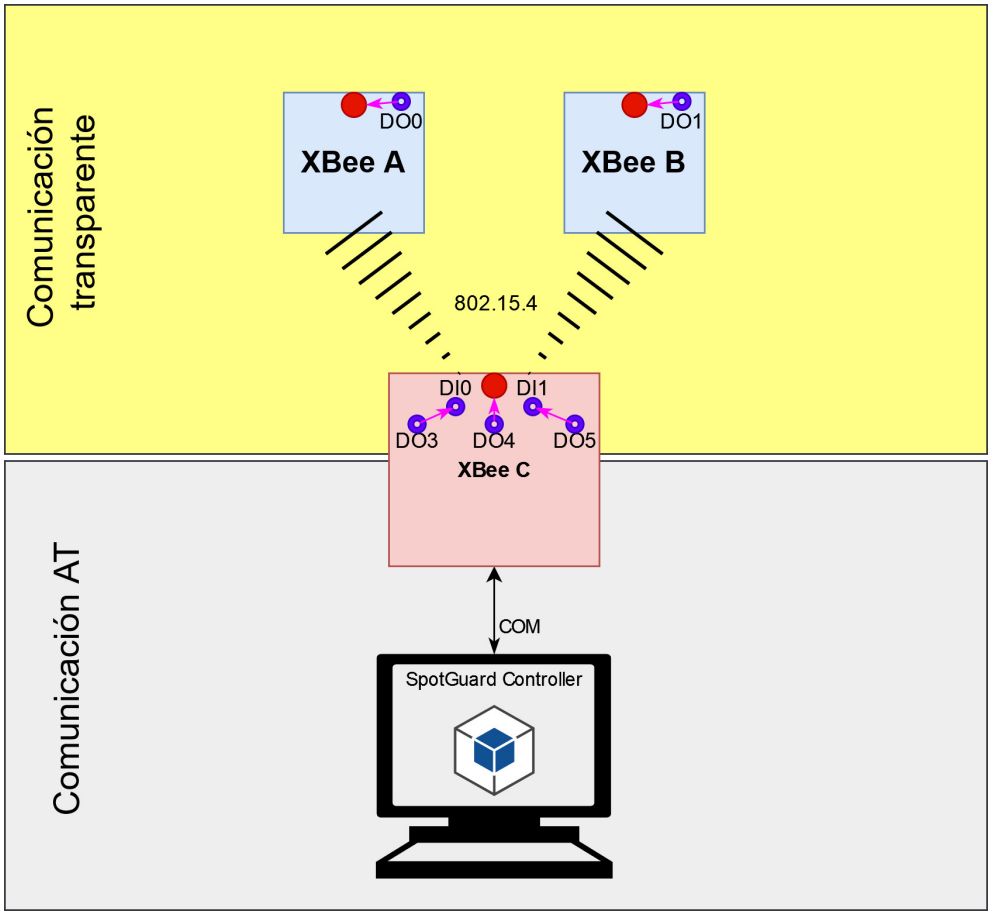
[1] Condición o circunstancia en la cual se advierte una desventaja de una cosa en relación con otra u otras.

Memoria

2.1. Descripción de los elementos

Sobre está sección se describen los principales elementos que conforman SpotGuard. Pretende ser un vistazo general, pero más detallado, de lo que a grandes rasgos compone SpotGuard. Por lo que la descripción técnica de cada uno de ellos comienza a partir de esta sección.
El sistema se compone de 3 dispositivos XBee configurados en la misma red. Cada uno de ellos tiene un diodo led de control y esta acompañado del circuito electrónico que corresponde a su función. Para interactuar con los leds de control de cada dispositivo, se emplea una interfaz gráfica que traduce las ordenes a comandos comprensibles por los dispositivos gracias a la clase XBeeSerialController.

Figura 2.1
Esquema conceptual SportGuard fase temprana

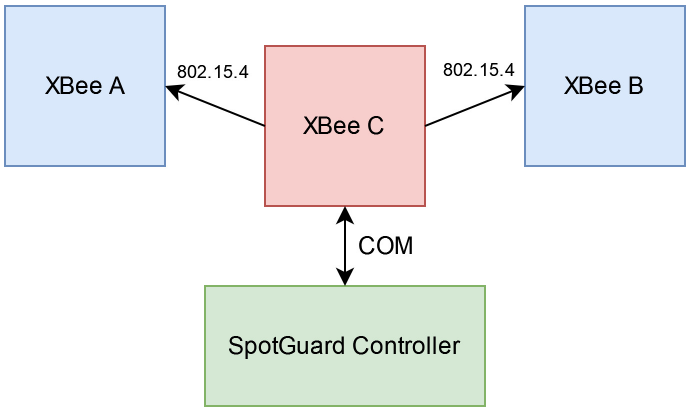


2.1.1. Red de dispositivos XBee S2C

Para la fase temprana del proyecto, se emplea una topología de estrella que involucra 3 dispositivos XBee S2C. Las características de esta red son:

- 1 dispositivo en modo controlador.
- 2 dispositivos en modo end-point.
- Todos los dispositivos trabajan bajo la misma red.
- La comunicación se produce de manera unidireccional del XBee coordinador hacia los XBee dispositivo final.
- El XBee coordinador se comunica bidireccional-mente a través del puerto serie conectado al ordenador que ejecute la interfaz de uso.
- Los datos son enviados en broadcast a toda la red e interpretados de manera diferente en función a la configuración de cada dispositivo.

Figura 2.2
Esquema red XBee



2.1.2. Circuito electrónico

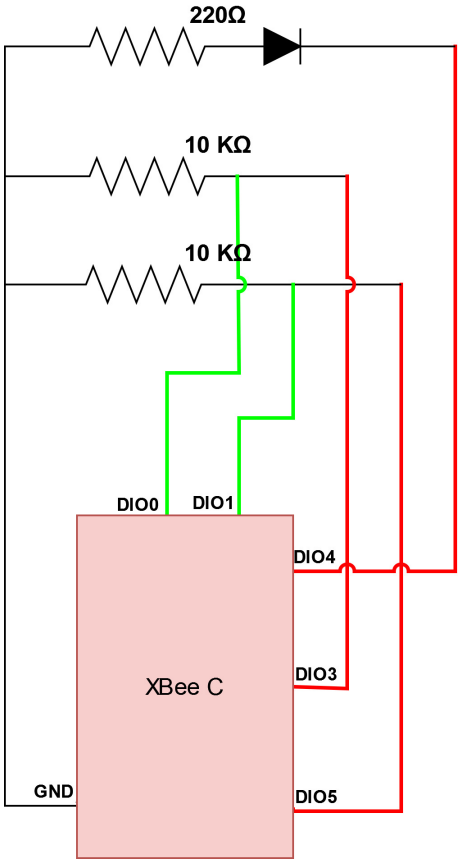
Sobre cada dispositivo XBee se monta un circuito electrónico diferente. A continuación se define el circuito para cada dispositivo y se describe su propósito.

Dispositivo coordinador

Este dispositivo será el encargado que recibir los comandos de configuración y compartir con el resto de dispositivos la misma. Para ello se emplearan los DIO 0, 1, 3, 4 y 5 de la siguiente manera:

- DIO0: Entrada digital. Detecta el cambio de estado del DIO3.
- DIO1: Entrada digital. Detecta el cambio de estado del DIO5.
- DIO3: Salida digital. Envía su estado al DIO0.
- DIO5: Salida digital. Envía su estado al DIO1.
- DIO4: Salida digital. Modifica su estado para suministrar corriente diodo led.

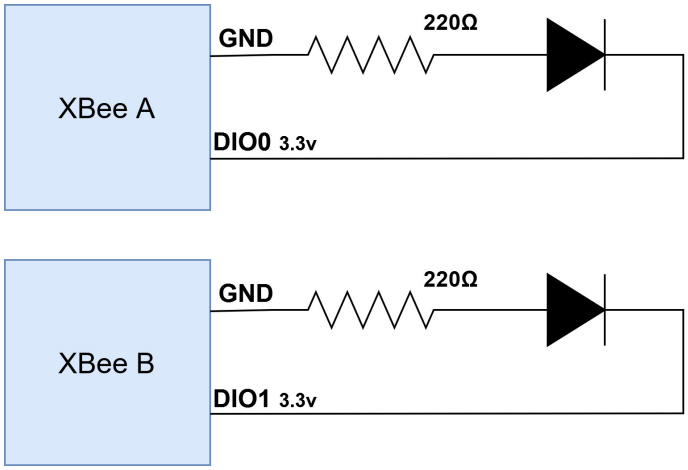
Figura 2.3
Esquema electrónico dispositivo coordinador



Dispositivos finales

Los dispositivos finales comparten el mismo esquema electrónico. Un único led, alimentado por uno de los DIO, el cual cambia de estado en función a la información compartida por el coordinador. Se emplea un DIO distinto para cada dispositivo final, en este caso el dispositivo final A emplea el DIO0 y el dispositivo final B emplea el DIO1.

Figura 2.4
Esquema electrónico para el dispositivo coordinador



2.1.3. SpotGuard Controller

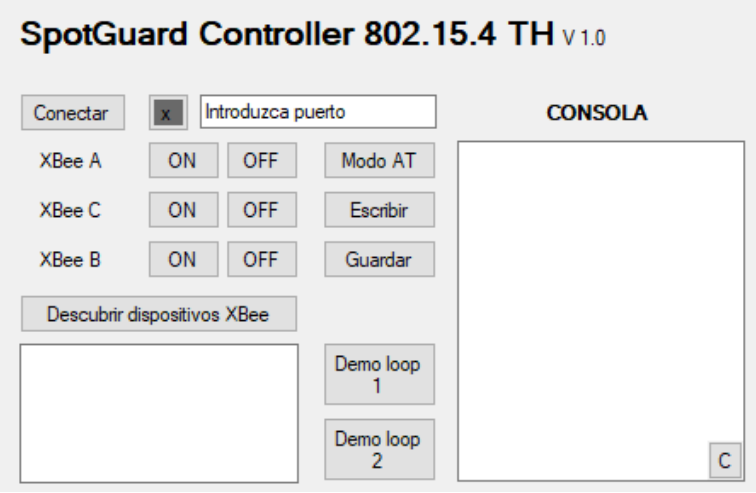
Se trata de un programa realizado con C# basado en formularios de Windows. Este programa se compone por un lado de un formulario de Windows que brinda la interfaz con todos los controles necesarios para interactuar con el dispositivo controlador y a su vez con el resto de dispositivos de la red. Por el otro, emplea la clase XBeeSerialController desarrollada para este proyecto.

Interfaz gráfica

La interfaz plantea un diseño sencillo, con los controles pertinentes habilitados e indicaciones sobre el propósito de cada botón. Desde esta interfaz se puede gestionar:

- El rastreo de dispositivos XBee conectados al equipo.
- En envío de los comandos principales al dispositivo coordinador.
- El estado de los diodos.
- El estado de la comunicación con el puerto serie.
- El envío, recepción y visualización de la información que gestiona el coordinador.

Figura 2.5
Interfaz gráfica SpotGuard Controller



Clase XBeeSerialController

Se trata de una clase que actúa como intermediaria entre la interacción con la interfaz gráfica y el dispositivo coordinador. Siendo esta la encargada de traducir las ordenes enviadas desde el formulario a comandos AT hexadecimales comprensibles por el dispositivo XBee.

2.2. Dispositivos Digi XBee

La información contenida en esta sección se centra en los aspectos técnicos y de configuración de los dispositivos XBee, así como su aplicación practica sobre SpotGuard. No obstante, el contenido se presenta resumido, tratando principalmente aquellos puntos directamente relacionados con el proyecto. Se recomienda acceder a la biblioteca de documentación de Digi, presente en la bibliografía de esta memoria, para consultar cualquier otro aspecto que pueda quedar al margen de la presentación [6].

2.2.1. Descripción

Los dispositivos XBee son una gama de chips de la marca DIGI[1] cuyo objetivo es el de proporcionar redes Point To Multipoint[2] a través de las cuales intercambiar datos entre diferentes dispositivos. Estos dispositivos emplean el protocolo ZigBee[1], un protocolo basado en el estándar IEEE 802.15.4[3] y tiene su origen en 1998, cuando un conjunto de empresas, la ZigBee Aliance (En la actualidad la Connectivity Standards Alliance[8]), se asociaron para definir un estándar de comunicaciones complementario al Wifi y el Bluetooth. Este protocolo determina la forma en la que los dispositivos realizan la operaciones sobre la capa física y el acceso a los datos. Los dispositivos XBee deben su estructura de configuración al protocolo ZigBee [11]. Algunas de las ventajas por las que destacan sobre el resto de dispositivos de radio difusión digital:

- Bajo consumo energético.
- Permite la realización de topologías de red en malla.
- Fácil integración con otros dispositivos y sistemas.
- Seguro, los datos pueden viajar cifrados con clave asimétrica AES 128 bits.

[1] Empresa dedicada al diseño, desarrollo, implementación y comercialización de sistemas hardware y programas software en especial de sistemas embebidos.[5]

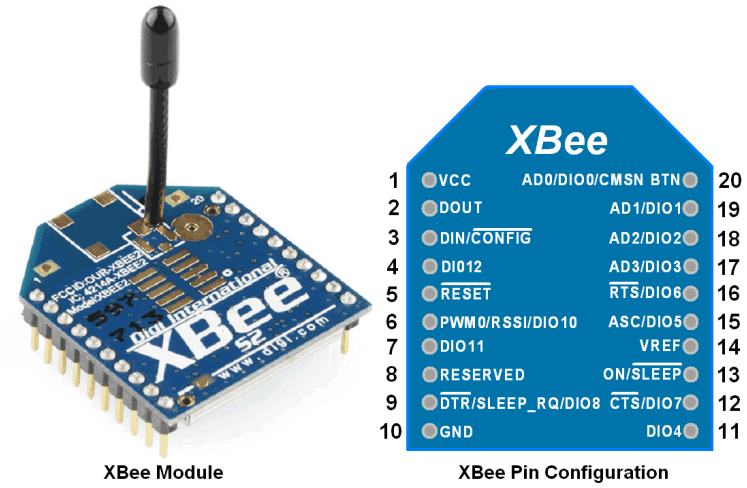
[2] En el contexto de las redes, se refiere a la dinámica de comunicación uno a muchos

[3] Estándar que define el nivel físico y el control de acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos[10]

2.2.2. Estructura física

Los dispositivos se presentan como un chip montado sobre una placa electrónica de 2,4 x 2,7cm en el caso del S2C. Estos poseen 20 tomas o pines con diferentes propósitos.

Figura 2.6
Conexiones físicas XBee



En la siguiente tabla se describe el propósito de cada tipo de toma que posee el dispositivo:

Nombre	Propósito
VCC	Alimentación eléctrica.
DOUT	Salida de datos UART.
DIN/CONFIG	Entrada de datos UART, por defecto la configuración se realiza a través de este pin.
RESET	Restaura la configuración por defecto.
RSSI	Indica la intensidad de la señal de radio.
PWMO	Modulación de la amplitud de pulsos.
DIOX	Entradas/salidas digitales.
ADX	Entradas analógicas.
DTR/SLEEP	Control del modo sleep.
GND	Toma a tierra.
CTS	Limpia los datos de la sesión actual (Clear to send).
ASC	Para asociar un indicador.

Tabla 2.1
Proposito de las tomas de un XBee

2.2.3. Versiones del dispositivo

A continuación se describen las características principales de las diferentes series, pero cabe destacar que de cada serie descienden diferentes versiones del dispositivo con diferentes propósitos. Para conocer las variaciones de cada dispositivos o las especificaciones técnicas de los mismos, se recomienda consultar la guía de dispositivos XBee referencia en la bibliografía de este documento.

XBee Serie 1

Xbee 802.15.4 fue el primer XBee fabricado. Permite la comunicación por RF sin necesidad de configuración del dispositivo. Destaca por ser uno de los dispositivos con menor consumo de reposo en el mercado (1uA). No posee memoria programable por lo que las funcionalidades disponibles son reducidas. Se podrían remplazar los dos extremos de una conexión por puerto serie cableada por dispositivos XBee S1.

Figura 2.7
XBee Serie 1

XBee Serie 2

Los dispositivos S2C ya incorporan un microprocesador de 50MHz y una memoria de hasta 2KB de RAM. Esto permite programar los módulos para comportarse como una red en forma de maya e interactuar de manera más compleja con las funcionalidades del firmware. Los dispositivos que se emplean en SpotGuard son todos XBee S2c.

Figura 2.8
XBee Serie 2

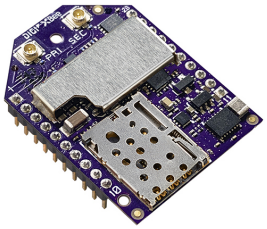
XBee Serie 3

Los dispositivos XBee 3 son muy similares a los de la serie 2. Están más enfocados en el despliegue ágil de soluciones comerciales. Según la versión son compatibles con Wifi y Bluetooth, lo que permite comunicarse y configurar el dispositivo desde sistemas compatibles con estas tecnologías inalámbricas, por ejemplo, un smartphone.

Figura 2.9
XBee Serie 3

Existen versiones de XBee 3 como el módem inteligente XBee 3 Cellular LTE que permiten trabajar con una tarjeta sim y redes de telefonía móvil.

Figura 2.10
XBee 3 Cellular LTE



2.2.4. Especificaciones técnicas XBee S2C

A continuación se muestra una tabla con las especificaciones técnicas mas relevantes del modelo S2C de XBee empleado para este proyecto. Para consultar el resto de especificaciones existe una referencia al respecto en la bibliografía de este documento.

Característica	Valor
Rango	60m.
Corriente de transmisión	33mA a 3,3v.
Velocidad de comunicación	250Kbps
Alimentación de entrada	3,3v
Banda de frecuencias	2,4GHz
Memoria	2KB RAM
Frecuencia del procesador	50,33MHz

Tabla 2.2
Especificaciones técnicas XBee S2C

2.2.5. Topologías

Los dispositivos XBee permiten crear diferentes topologías de red. Para ello, los dispositivos pueden actuar como:

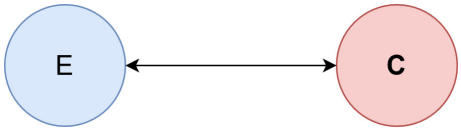
- Coordinadores: Es necesario un único dispositivo coordinador para gestionar la comunicación de toda la red. No puede haber más de uno.
- Enrutadores: Permiten la redirección de tramas y por consecuencia la expansión del radio de acción de la red. No existe limite de enrutadores.
- Dispositivos finales: Intercambian datos con el nodo padre al que estén conectados. Siempre deben conectarse a un dispositivo coordinador o enrutador.

Las topologías compatibles con XBee:

Punto a punto

La composición de red más simple. Un dispositivo coordinador y otro final conectados entre si.

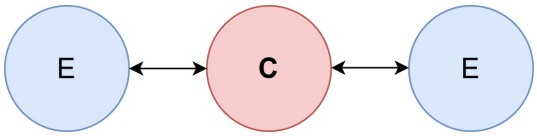
Figura 2.11
Topología punto a punto



Estrella

Un dispositivo coordinador que comunica dos o más dispositivos finales entre si. Es la topología que emplea SpotGuard

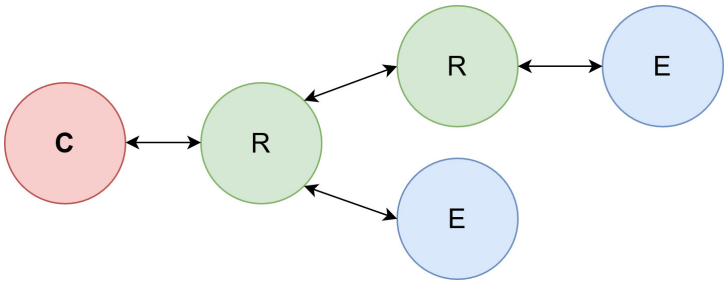
Figura 2.12
Topología de estrella



Árbol

El dispositivo coordinador se comunica con dos o más dispositivos finales a través de al menos un dispositivo enrutador.

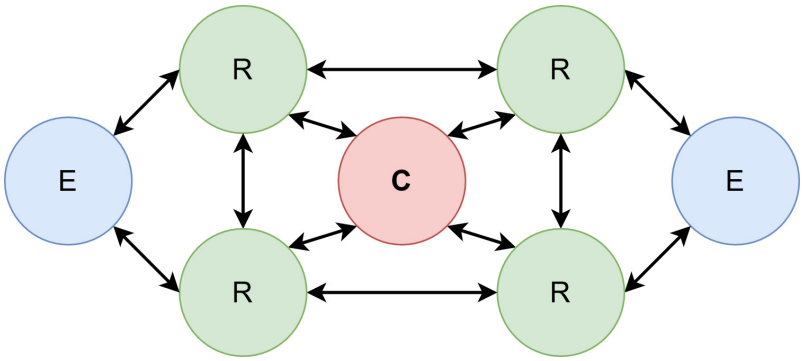
Figura 2.13
Topología de árbol



Malla

El dispositivo coordinador se comunica con los dispositivos a través de diferentes caminos gracias a la existencia de dos o más dispositivos enrutadores.

Figura 2.14
Topología de malla



2.2.6. Configuración del dispositivo XBee S2C

Los dispositivos XBee poseen una gran variedad de parámetros y funcionalidades configurables. Para configurar un XBee se requiere del envío de una serie de comandos AT[4] hexadecimales a través de el puerto paralelo, para ello se puede emplear un ordenador con windows y cualquier programa que proporcione una consola para enviar datos a los puertos paralelos, como TeraTerm.[5]

Los modos de comunicación

Se comprenden 3 modos de comunicación:

- **Modo transparente:** El dispositivo envía la exactamente la misma información que recibe a todos los dispositivos que se encuentren como destino. Esto no significa que conozca varios destinos, significa que el destino puede estar representado en varios sitios. El funcionamiento se podría asemejar a las comunicaciones broadcast[6]. Envía valor a valor toda la información que le llega. Este es el modo de comunicación que emplea la fase temprana de SpotGuard.
- **Modo API:** En el modo API el dispositivo envía tramas compuestas en lugar de replicar los datos que le llegan. Esto permite asignar destinos a las tramas y realizar un direccionamiento unitario [7] entre los diferentes dispositivos de la red, inclusive permite enviar datos de mayor complejidad en base64. Recomendable para proyectos que impliquen Arduino o RaspBerryPI. Al no emplearse en la fase temprana de este proyecto, la tabla de construcción de tramas queda recogida en el manual de uso XBee, el cual esta disponible en las referencias de la bibliografía de este documento.
- **Modo AT:** Este modo se emplea independientemente del transparente y el API. Mientras que el resto de modos pertenecen a la configuración del dispositivo, el modo AT se emplea para habilitar la escucha activa de comandos (ATención). Al activar este modo se interrumpe cualquier comunicación que este llevando a cabo el dispositivo y los datos que entren por el puerto de comunicaciones pasan a interpretarse en modo AT, lo que significa que ciertos conjuntos de datos se interpretaran como ordenes que implicaran modificaciones en la configuración y respuestas de verificación en el puertos serie.

Parámetros de configuración

Los parámetros de configuración del dispositivo son numerosos y varían en función a el tipo de dispositivo y la versión del mismo. Para la configuración inicial del dispositivo es importante tener seleccionar adecuadamente:

- **Familia del dispositivo:** En el caso de los XBee S2C, XB24C.
- **Set de funciones correspondiente al tipo de dispositivo:** Algunos dispositivos XBee pueden actuar como versiones anteriores o simplificadas de productos de la misma familia. En el caso del S2C, es compatible con el set de funciones de DigiMesh 2.4[3], pero para este proyecto se emplea el set correspondiente al 802.15.4 SMT S2 (S2C).
- **Versión del firmware:** En este caso es una selección dentro de un control de versiones. Se recomienda tener instalada siempre la última versión. SpotGuard emplea la versión 2003 del firmware.

[4] Abreviatura de Attention.

[5] Tera Term[9] es un programa de comunicaciones gratuito. Entre otras prestaciones, soporta emulación VT100, conexiones telnet y seriales.

[6] En redes, el envío de un mismo paquete a todos los nodos de la red, sin importar el identificador de cada dispositivo final

Con el firmware que actualmente ejecutan los XBee S2C, se dispone de más de 70 parámetros[4] de configuración. A continuación se muestra una tabla con los involucrados en el proyecto.

Parámetro	**Descripción**
CH	Número de canal para enviar o recibir datos.
ID	Nombre de la red, determinado por un número.
CE	Valor booleano que determina si el dispositivo actúa como coordinador.
DL	Dirección 16bits de destino (low).
MY	Dirección 16bits del origen.
NI	Identificador de nodo, determinado por una cadena de texto.
D0-7	Representan las tomas de E/S disponibles en el dispositivo. Configurables como entradas o salidas tanto digitales como analógicas.
IU	Valor booleano que determina si se permite la salida de datos por los DIO.
IC	Representa una máscara hexadecimal para enviar el estado de los DIO en una única trama.
RP	Valore hexadecimal que determina el tiempo de refresco de la comunicación.

Tabla 2.3
Tabla de parámetros empleados.

2.2.7. Configuración de cada dispositivo

Para la configuración inicial de los dispositivos se ha empleado la aplicación de escritorio de DIGI, XCTU. Esta aplicación facilita la configuración de los módulos a través de un formulario y la realización de pruebas con una consola y varias herramientas. De los 3 dispositivos (A, B y C siendo C el controlador) estas son las configuraciones empleadas:

Parámetro	**Valor**
CH	C
ID	2022
CE	Activado
DL	5678
MY	1234
NI	MASTER
IU	Activado
IC	3 (Dios 0 y 1)
RP	5 (Cada 1/2 segundo)
D0	DI (Entrada digital)
D1	DI (Entrada digital)

Tabla 2.4
Tabla de configuración del dispositivo C.

Parámetro	Valor
CH	C
ID	2022
CE	Activado
DL	5678
MY	1234
NI	B
IU	Activado
D0	DO (Salida digital)

Tabla 2.5
Tabla de configuración del dispositivo A.

Parámetro	Valor
CH	C
ID	2022
CE	Activado
DL	5678
MY	1234
NI	B
IU	Activado
D1	DO (Salida digital)

Tabla 2.6
Tabla de configuración del dispositivo B.

2.2.8. Comandos AT empleados

Sobre esta configuración, para interactuar con los diferentes diodos led, se envían los comandos AT correspondientes desde el puerto serie abierto por el controlador de SpotGuard. A continuación se muestra una tabla con los comandos AT empleados para gestionar el comportamiento de los diodos. La correspondencia de los comandos ASCII con los valores hexadecimales que se envían por el puerto se puede consultar en el **ANEXO I** de este documento. De igual manera, para consultar el resto de comandos AT disponibles, quedan recogidos en el manual de usuario diferenciado[4] en la bibliografía de este documento.

ASCII	Propósito
+++	Iniciar comunicación por linea de comandos con el dispositivo.
AT	.ᴬttention"devuelve un eco para comprobar que existe comunicación.
ATID	Devuelve el PAN ID configurado en el dispositivo.
ATWR	Escribe los cambios para que se conserven aun después de apagar el módulo.
ATMY	Devuelve el ID del dispositivo.
ATD(pin)...ATD(mode)	Establece la configuración de los pines para enviar menor o mayor voltaje
ATRE	Reinicia la configuración
ATCN	Cierra la comunicación por linea de comandos.

2.2.9. Otras aplicaciones de los XBee

Los dispositivos XBee amplían el abanico de aplicaciones practicas conforme la ciencia del IoT avanza. La integración de tecnologías inalámbricas de bajo consumo esta a la orden del día. Algunas de las área en las que se enfocan los dispositivos XBee:

- Sistemas de comunicación para aeronaves.[2]
- Sistemas de control remoto de drones o vehículos RF.
- Entornos con espectáculos de luces o platós de grabación.
- Sistemas de balizamiento para la monitorización de activos de cualquier índole.

Existe una propuesta de proyecto llamada SpotGuard Lighting. Esta desarrollada por este mismo equipo y propone un sistema de iluminación vehicular inteligente basado en la estructura del actual desarrollo. Se puede consultar la propuesta de proyecto en el **ANEXO II** de este documento.

2.3. Controlador SpotGuard

El controlador de SpotGuard es una aplicación desarrollada en C que se compone de una interfaz WindowsForms[7] y una clase, XBeeSerial controller que permite la integración de tantos dispositivos XBee se desee conectar. Mientras que la interfaz proporciona de un vistazo un panel con todos los controles necesarios para interactuar con la red de pruebas presente en este proyecto, la clase XBeeSerialController desarrollada por el equipo para ser consumida desde SpotGuard brinda un acceso ágil y simplificado a las funcionalidades del modo transparente de estos dispositivos.
Desglose de funciones: La clase XBeeSerialController parte del concepto de que un XBee conectado al equipo equivale a un puerto de comunicaciones ocupado, por lo que desde su construcción, un objeto de este tipo necesita recibir un puerto existente en la máquina y para que este se pueda utilizar debe estar ocupado por un dispositivo y que este se encuentre con la conexión al puerto cerrada, lo que significa que cualquier programa del equipo con permisos podrá abrir el puerto y leer o escribir datos sobre el. La clase XBeeSerialController se compone únicamente de un objeto SerialPort "puertoActivo"que representa el puerto serie al que esta asociado el dispositivo y una variable booleana "status"que representar si al inicializar el puerto la conexión ha sido exitosa, de lo contrario esta variable guarda un false marcando ese dispositivo como no funcional. A continuación se realiza un desglose de las funciones presentes en la clase y su propósito. Para consultar el código fuente de la mismas, se puede encontrar ordenado por orden de mención a las funciones en el ANEXO III.

- Constructor:
  1. Recibe el nombre del puerto en formato cadena.
  2. Inicializa el puerto con el nombre recibido.
  3. Por defecto marca el status a true.
  4. Configura el puerto para trabajar a la misma frecuencia que el dispositivo XBee.
  5. Realiza un intento de abrir el puerto, si la conexión al puerto falla, la variable status pasa a false y se considera al objeto fallido en su creación.
- Función testSaludo: Envía un comando AT de saludo y devuelve true si el dispositivo XBee responde con la cadena de confirmación 'OK'.
- Función getStatus: Devuelve el status de la conexión con el puerto asignado.
- Función abrirPuerto: Abre el puerto asignado al objeto.
- Función cerrarPuerto: Cierra el puerto asignado al objeto.
- Función leerPuerto: Vuelca el contenido del puerto a la memoria y lo devuelve en forma de cadena.
- Función enviarSeñal: Posee la traducción a bytes hexadecimales de los comandos AT con los que se quiere operar el dispositivo. Es recomendable emplear colecciones de Byte a ingresar directamente los caracteres que conforman cada comando, pues estos contienen caracteres invisibles (y $\mathring{)}$ que generan muchos conflictos al ingresarlos directamente en formato ASCII desde el teclado. Esta función:

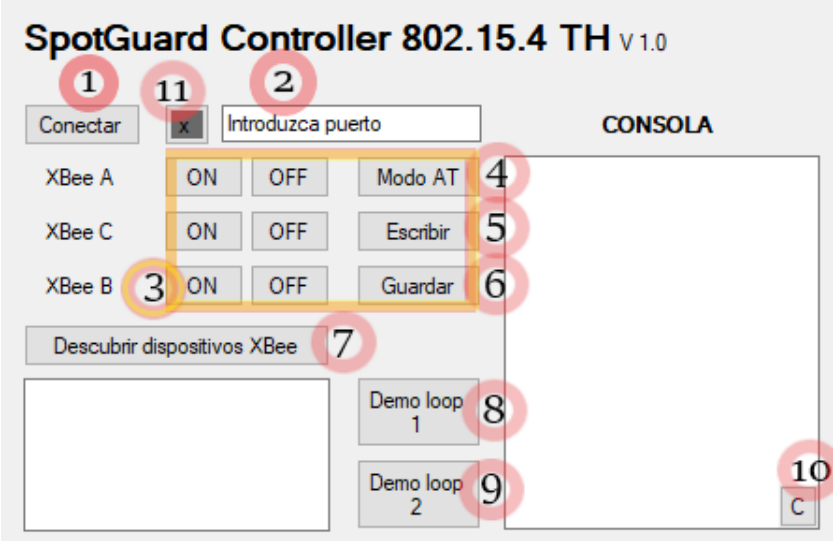
[7] Librería de .NET para el desarrollo de interfaces gráficas.

1. Recibe dos enteros. El primero, .ªction", indica el tipo de acción a realizar, siendo 20, 21, 22 las relacionadas con la comunicación básica (saludo inicial, escritura, guardado) y acciones 0 y 1 apagado y encendido de los diodos. El segundo entero, "dio"hace referencia al DIO sobre el que realizar la acción. Para las acciones 20, 21 y 22 no se tiene en cuenta esta variable.
2. Determina cual de las señales debe enviar en función a los datos que han llegado.
3. Envía una señal de escritura al dispositivo con la información de configuración pertinente.

2.3.1. Interacción con el usuario

Al usuario se le presenta lo que pretende ser una interfaz limpia e intuitiva, donde los controles que todavía no tienen propósito, están deshabilitados. Nos hemos centrado también desarrollar test funcionales que sirvan para entender los aspectos de esta tecnología.

Figura 2.15
Interfaz del controlador de SpotGuard

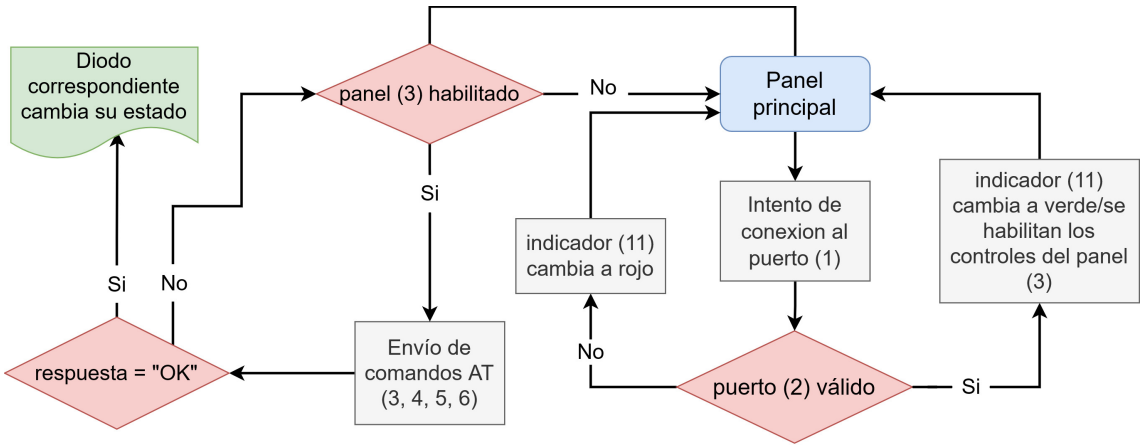


1. Intenta conectar al el puerto serie descrito en el input del puerto (2)
2. Texto de entrada donde se introduce el puerto como cadena.
3. Panel de encendido y apagado de los diodos.
4. Envía el saludo inicial para entrar en modo comunicación AT.
5. Escribe la configuración en el dispositivo.
6. Guarda los cambios en el dispositivo y termina la comunicación AT.
7. Realiza un sondeo de los puertos serie en busca de dispositivos XBee.
8. Demo variable 1.
9. Demo variable 2.
10. Limpieza de consola.
11. Indicador de conexión fructífera.

Diagrama de flujo interacción con la interfaz

A continuación se muestra un diagrama que define las posibles interacciones del usuario con el programa , siendo el punto de partida el "Panel principalz donde la numeración entre paréntesis hace referencia a las anotaciones presentes en la figura 2.15.

Figura 2.16
Diagrama de flujo interacción con la interfaz



Resumen y conclusión

Este proyecto no pretende ser un producto comercial, si no una obra didáctica con la que aprender acerca de las aplicaciones practicas del IoT partiendo de las redes Zigbee como ejemplo. Ha sido muy interesante realizar un proyecto de estas características. Por un lado, se ha vivido de primera mano lo abrumador y desalentador de meterse a aprender algo totalmente nuevo. Así mismo y en mayor proporción, se ha disfrutado y disfruta de la sensación de recompensa, fortaleza y seguridad que brinda verse más preparado y capaz que antes.

Índice de figuras

Índice de tablas

Bibliografía

[1] CSA. *Connectivity Standards Alliance*. 2022. URL: https://csa-iot.org/ (visitado 01-01-2022).

[2] DIGI. *Digi Drone Paths*. 2022. URL: https://www.digi.com/resources/project-gallery/drone-telemetry-system (visitado 01-01-2022).

[3] DIGI. *DigiMesh 2.4*. 2022. URL: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-digimesh-2-4#specifications (visitado 01-01-2022).

[4] DIGI. *Guia de usuario XBee S2B*. 2022. URL: https://www.digi.com/resources/documentation/digidocs/pdfs/90000976.pdf (visitado 01-01-2022).

[5] Einforma. *DIGI INTERNATIONAL SPAIN SOCIEDAD ANONIMA*. 2022. URL: https://www.einforma.com/informacion-empresa/digi-international-spain (visitado 01-01-2022).

[6] ELECTROALL. *Nociones generales XBee*. 2022. URL: https://www.youtube.com/watch?v=4I5laCU1JTg&ab_channel=ELECTROALL (visitado 01-01-2022).

[7] Engineersgarage. *Uso del modo API*. 2022. URL: https://www.engineersgarage.com/controlling-led-light-over-zigbee-api-mode-iot-part-38/ (visitado 01-01-2022).

[8] LedBox. *Protocolo ZigBee*. 2022. URL: https://blog.ledbox.es/que-es-el-protocolo-zigbee/ (visitado 01-01-2022).

[9] OSDN. *Tera Term Home*. 2022. URL: https://ttssh2.osdn.jp/index.html.en (visitado 01-01-2022).

[10] Wikipedia. *Protocolo IEEE 802.15.4*. 2022. URL: https://es.wikipedia.org/wiki/IEEE_802.15.4 (visitado 01-01-2022).

[11] Wikipedia. *Zigbee*. 2022. URL: https://es.wikipedia.org/wiki/Zigbee#Cronolog%C3%ADa (visitado 01-01-2022).

Anexo I

3.1. Correspondencia de valores en comandos AT

Figura 3.1
Imagen de la biblioteca de señales presente en SpotGuard.

```
{
    //Señales                   //HEX                          //Ación            //ASCII
    byte[] saludo =     { 0x2B, 0x2B, 0x2B };                  //Act 20           +++
    byte[] escribir =   { 0x41, 0x54, 0x57, 0x52, 0x0D };      //Act 21           ATWR
    byte[] guardar =    { 0x41, 0x54, 0x43, 0x4E, 0x0D };      //Act 22           ATCN
    byte[] d1_h =       { 0x41, 0x54, 0x44, 0x31, 0x35, 0x0D }; //Act 1 Dio 1     ATD15
    byte[] d1_l =       { 0x41, 0x54, 0x44, 0x31, 0x34, 0x0D }; //Act 0 Dio 1     ATD14
    byte[] d0_h =       { 0x41, 0x54, 0x44, 0x30, 0x35, 0x0D }; //Act 1 Dio 1     ATD05
    byte[] d0_l =       { 0x41, 0x54, 0x44, 0x30, 0x34, 0x0D }; //Act 0 Dio 1     ATD04
    byte[] d3_h =       { 0x41, 0X54, 0X44, 0x33, 0x35, 0x0D }; //Act 1 Dio 3     ATD35
    byte[] d3_l =       { 0x41, 0X54, 0X44, 0x33, 0x34, 0x0D }; //Act 0 Dio 3     ATD34
    byte[] d4_h =       { 0x41, 0X54, 0X44, 0x34, 0x35, 0x0D }; //Act 1 Dio 4     ATD45
    byte[] d4_l =       { 0x41, 0X54, 0X44, 0x34, 0x34, 0x0D }; //Act 0 Dio 4     ATD44
    byte[] d5_h =       { 0x41, 0X54, 0X44, 0x35, 0x35, 0x0D }; //Act 1 Dio 5     ATD55
    byte[] d5_l =       { 0x41, 0X54, 0X44, 0x35, 0x34, 0x0D }; //Act 0 Dio 5     ATD54
```

Anexo II

3.2. Introducción a SpotGuard Ligthing

SpotGuard Ligthing consistirá en un sistema de iluminación vehicular conformado por 3 módulos luminiscentes XBee S3C, integrando uno de ellos, a modo de coordinador, en una centralita electrónica Arduino BLE, que a su vez se comunica con un dispositivo móvil vía Bluetooth para la interacción con el sistema de luces. Su propósito es asegurar la visibilidad de vehículos, motorizados o no, de pequeño tamaño.

3.2.1. Requisitos y características

- Los módulos deben ser fácilmente potables y acoplables a vehículos de pequeño tamaño, tales como: Bicicletas, motos, patinetes eléctricos, patines...
- La conexión entre los módulos debe ser inalámbrica, sin cables a la vista.
- La cantidad de luz emitida se corresponderá con la de la homologación de un vehículo convencional dando sentido a su uso, también durante el día.
- Existirá una gestión inteligente del comportamiento de los módulos, una aplicación móvil se encargará de la gestión del mismo permitiendo al usuario:
  - Apagar y encender el sistema.
  - Activar perfiles de comportamiento automático de las luces.
  - Conocer el estado de las baterías de los distintos módulos.
- El planteamiento de la estructura del código debe permitir la implementación de más módulos en el futuro.
- El usuario podrá controlar manualmente el comportamiento de los módulos, pero también existirá una parte automatizada.
- Los módulos emplearan baterías de litio recargables.
- Los módulos se podrán acoplar en manillares, cuadros, muñecas, piernas o cintura.

3.2.2. Requisitos prácticos

- Se empleará el lenguaje C# para la gestión de la comunicación entre los diferentes nodos.
- Se empleará un dispositivo Arduino BLE como microcontrolador intermediario entre el XBee coordinador y el ordenador o smartphone que lo configure.
- Materiales básicos:
  - Sensor inercial “10 DOF IMU”
  - Centralita de conexión (DigiXBee 3)
  - 3 Diodos led rojos.

- 4 Baterías de iones de litio (1600mA).
- Kit de electrónica (Estaño, soldados, cables, adaptadores).
- Sensor de ultrasonidos.

- Se emplean formularios Xamarin para la integración de la interfaz gráfica.

3.3. Elementos del Sistema

Sobre está sección se enumeran los principales elementos que conforman SpotGuard Ligthing. También se proporciona una pequeña introducción al propósito de cada uno de ellos pretendiendo dejar claro cuales son los actores principales del sistema.

3.3.1. Dispositivo de ejecución objetivo

El dispositivo encargado de llevar a cabo la ejecución del programa deberá cumplir los siguientes requisitos hardware:

- Sensor giroscópico.
- Acelerómetro.
- Módulo bluetoot compatible con el protocolo IEEE 802.15.4
- Sistema operativo Android, Windows o IOs

SpotGuard esta principalmente pensado para ejecutarse en teléfonos móviles inteligentes los cuales en su gran mayoría incorporan el hardware necesario para el funcionamiento del sistema. Esto hace cómodo y versátil el uso del dispositivo, pues un teléfono móvil es algo común y accesible por la mayoría de usuarios.

3.3.2. Comunicador Digi Xbee3

Se trata del componente protagonista en este proyecto. El módulo de radio Digi Xbee3 permite comunicar dispositivos compatibles con el protocolo 802.15.4 empleando Bluetooth. Este elemento permite a SpotGuard establecer una comunicación entre el dispositivo de ejecución objetivo y el conjunto de diodos led.

3.3.3. Diodos led

Responsables de emitir las señales luminosas. Conectados a las tomas de entrada y salida del comunicador, emiten secuencias de parpadeos predefinidas y enviadas desde la ejecución del programa en el dispositivo de ejecución objetivo.

Anexo III

3.4. Código fuente de las funciones

Sobre este ANEXO III quedan todas las imágenes referentes al código de las funciones contenidas en la clase XBeeSerialController.

Figura 3.2
Constructor del objeto

```
private SerialPort puertoActivo;
bool status;
2 referencias
public XBeeSerialController(string puerto)
{
    puertoActivo = new SerialPort(puerto);
    status = true;
    puertoActivo.BaudRate = 9600;
    puertoActivo.Parity = Parity.None;
    puertoActivo.DataBits = 8;
    puertoActivo.StopBits = StopBits.One;
    try
    {
        puertoActivo.Open();
    } catch (Exception e)
    {
        status = false;
    }
}
```

Figura 3.3
Función testSaludo.

```
public bool testSaludo()
{
    string r = "";
    if (status)
    {
        enviarSeñal(20, 0);
        Thread.Sleep(1250);
        r = leerPuerto();
        puertoActivo.Close();
        if (r == "OK\r")
        {
            return true;
        } else
        {
            return false;
        }
    } else
    {
        return false;
    }
}
```

Figura 3.4
Función getStatus.

```
11 referencias
public bool getStatus()
{
    return status;
}
```

Figura 3.5
Función abrirPuerto.

```
public void abrirPuerto()
{
    if (!puertoActivo.IsOpen)
    {
        puertoActivo.Open();
    }
}
```

Figura 3.6
Función cerrarPuerto.

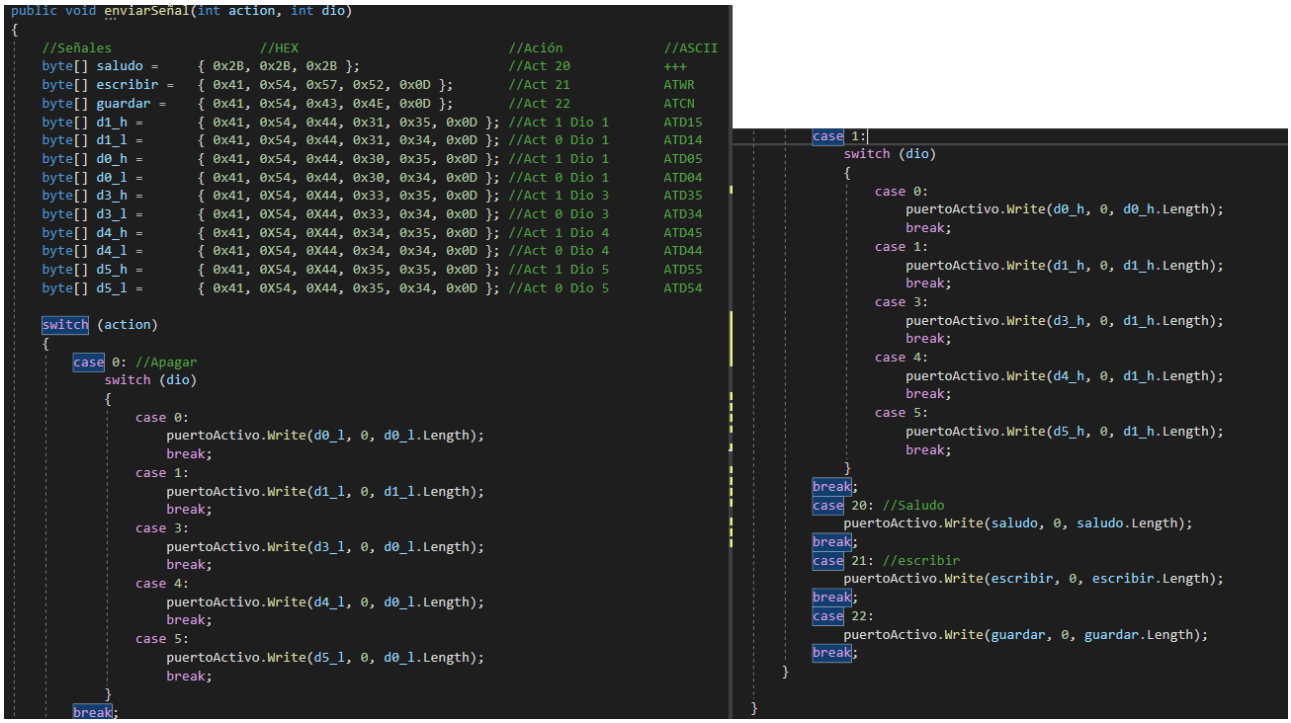
```
public void cerrarPuerto()
{
    if (puertoActivo.IsOpen)
    {
        puertoActivo.Close();
    }
}
```

Figura 3.7
Función leerPuerto.

```
{
    //Señales                 //HEX                          //Ación          //ASCII
    byte[] saludo =     { 0x2B, 0x2B, 0x2B };                //Act 20         +++
    byte[] escribir =   { 0x41, 0x54, 0x57, 0x52, 0x0D };    //Act 21         ATWR
    byte[] guardar =    { 0x41, 0x54, 0x43, 0x4E, 0x0D };    //Act 22         ATCN
    byte[] d1_h =       { 0x41, 0x54, 0x44, 0x31, 0x35, 0x0D }; //Act 1 Dio 1    ATD15
    byte[] d1_l =       { 0x41, 0x54, 0x44, 0x31, 0x34, 0x0D }; //Act 0 Dio 1    ATD14
    byte[] d0_h =       { 0x41, 0x54, 0x44, 0x30, 0x35, 0x0D }; //Act 1 Dio 1    ATD05
    byte[] d0_l =       { 0x41, 0x54, 0x44, 0x30, 0x34, 0x0D }; //Act 0 Dio 1    ATD04
    byte[] d3_h =       { 0x41, 0X54, 0X44, 0x33, 0x35, 0x0D }; //Act 1 Dio 3    ATD35
    byte[] d3_l =       { 0x41, 0X54, 0X44, 0x33, 0x34, 0x0D }; //Act 0 Dio 3    ATD34
    byte[] d4_h =       { 0x41, 0X54, 0X44, 0x34, 0x35, 0x0D }; //Act 1 Dio 4    ATD45
    byte[] d4_l =       { 0x41, 0X54, 0X44, 0x34, 0x34, 0x0D }; //Act 0 Dio 4    ATD44
    byte[] d5_h =       { 0x41, 0X54, 0X44, 0x35, 0x35, 0x0D }; //Act 1 Dio 5    ATD55
    byte[] d5_l =       { 0x41, 0X54, 0X44, 0x35, 0x34, 0x0D }; //Act 0 Dio 5    ATD54
```

Figura 3.8

Función enviar señal



```
public void enviarSeñal(int action, int dio)
{
    //Señales                   //HEX                           //Ación             //ASCII
    byte[] saludo =     { 0x2B, 0x2B, 0x2B };                   //Act 20            +++
    byte[] escribir =   { 0x41, 0x54, 0x57, 0x52, 0x0D };       //Act 21            ATWR
    byte[] guardar =    { 0x41, 0x54, 0x43, 0x4E, 0x0D };       //Act 22            ATCN
    byte[] d1_h =       { 0x41, 0x54, 0x44, 0x31, 0x35, 0x0D }; //Act 1 Dio 1       ATD15
    byte[] d1_l =       { 0x41, 0x54, 0x44, 0x31, 0x34, 0x0D }; //Act 0 Dio 1       ATD14
    byte[] d0_h =       { 0x41, 0x54, 0x44, 0x30, 0x35, 0x0D }; //Act 1 Dio 1       ATD05
    byte[] d0_l =       { 0x41, 0x54, 0x44, 0x30, 0x34, 0x0D }; //Act 0 Dio 1       ATD04
    byte[] d3_h =       { 0x41, 0X54, 0X44, 0x33, 0x35, 0x0D }; //Act 1 Dio 3       ATD35
    byte[] d3_l =       { 0x41, 0X54, 0X44, 0x33, 0x34, 0x0D }; //Act 0 Dio 3       ATD34
    byte[] d4_h =       { 0x41, 0X54, 0X44, 0x34, 0x35, 0x0D }; //Act 1 Dio 4       ATD45
    byte[] d4_l =       { 0x41, 0X54, 0X44, 0x34, 0x34, 0x0D }; //Act 0 Dio 4       ATD44
    byte[] d5_h =       { 0x41, 0X54, 0X44, 0x35, 0x35, 0x0D }; //Act 1 Dio 5       ATD55
    byte[] d5_l =       { 0x41, 0X54, 0X44, 0x35, 0x34, 0x0D }; //Act 0 Dio 5       ATD54

    switch (action)
    {
        case 0: //Apagar
            switch (dio)
            {
                case 0:
                    puertoActivo.Write(d0_l, 0, d0_l.Length);
                    break;
                case 1:
                    puertoActivo.Write(d1_l, 0, d1_l.Length);
                    break;
                case 3:
                    puertoActivo.Write(d3_l, 0, d0_l.Length);
                    break;
                case 4:
                    puertoActivo.Write(d4_l, 0, d0_l.Length);
                    break;
                case 5:
                    puertoActivo.Write(d5_l, 0, d0_l.Length);
                    break;
            }
        break;
        case 1:
            switch (dio)
            {
                case 0:
                    puertoActivo.Write(d0_h, 0, d0_h.Length);
                    break;
                case 1:
                    puertoActivo.Write(d1_h, 0, d1_h.Length);
                    break;
                case 3:
                    puertoActivo.Write(d3_h, 0, d1_h.Length);
                    break;
                case 4:
                    puertoActivo.Write(d4_h, 0, d1_h.Length);
                    break;
                case 5:
                    puertoActivo.Write(d5_h, 0, d1_h.Length);
                    break;
            }
        break;
        case 20: //Saludo
            puertoActivo.Write(saludo, 0, saludo.Length);
        break;
        case 21: //escribir
            puertoActivo.Write(escribir, 0, escribir.Length);
        break;
        case 22:
            puertoActivo.Write(guardar, 0, guardar.Length);
        break;
    }
}
```