Trabajo de Fin de Grado

Aplicación centralizada para la gestión de una red domótica basada en XBee

Centralized application for home automation network management based on XBee

Adrián Estévez Expósito

La Laguna, 5 de julio de 2016

D. **Alberto Hamilton Castro**, con N.I.F. 43.773.884-P profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Aplicación centralizada para la gestión de una red domótica basada en XBee"

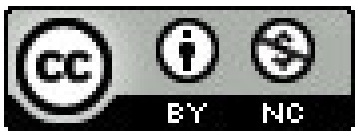
ha sido realizada bajo su dirección por D. **Adrián Estévez Expósito**, con N.I.F. 43.838.690-T.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 5 de julio de 2016.

Agradecimientos

Agradezco a mi tutor Alberto Hamilton Castro sus consejos y apoyo durante todo el desarrollo de este proyecto, así como a mi familia y amigos por todo su apoyo incondicional.

Licencia



Resumen

El objetivo del proyecto ha sido implementar una aplicación que fuera capaz de comunicarse con una red de sensores que utilizan un protocolo de comunicación inalámbrica. Tras analizar esta red de sensores y todos sus componentes, se pretende que la aplicación diseñada sea capaz de realizar una configuración flexible de la red, así como poder enviar órdenes de manera remota que el controlador de la red interprete, para así poder actuar en consecuencia.

Para llevar a cabo el desarrollo de la aplicación, se construyó previamente una red de sensores conectados a diversos dispositivos de entrada y salida mediante los cuales comprobar su funcionamiento. Además, es necesario indicar que la aplicación constará de dos versiones: la primera está destinada a ser localizada en un servidor remoto, de forma que un administrador pueda manipular la red de dispositivos mencionada mediante su uso; la segunda, en cambio, está destinada a ser manipulada por el usuario final. Ambas versiones se comunicarán entre ellas mediante un protocolo apoyado en UDP que ha sido desarrollado para la ocasión.

Se pretende que la aplicación se englobe dentro del campo de la "Internet de las Cosas" (IoT), de manera que una versión final posibilite la manipulación remota de diversos dispositivos conectados a una red. De esta forma, se podría destinar a diversos campos, como pudiera ser el de la domótica, entre otros.

Palabras clave: Red de sensores, protocolo de comunicación inalámbrica, servidor, UDP, Internet de las Cosas (IoT), domótica.

Abstract

The main purpose of this Project has been to implement an app that would be able to communicate with a network of sensors which uses a wireless communication protocol. After analyzing this network of sensors and all of its components, we wanted the designed application to be able to make a flexible configuration in the network, as well as being able to send commands remotely that the network's controller would be able to understand. This way, the network would be able to act according to these commands.

To carry out the development of the application, it was previously built a network of sensors connected to many input and output devices that were used to check its performance. In addition, it's necessary to say that the app will have two versions: the first one would be located in a remote server, letting an admin to use it to handle with the mentioned network of sensors; besides, the other one would be the version that the final user would be able to interact. Both versions will communicate each other using a communication protocol supported by UDP, which has been designed for this project.

We want the app to be included in the field of the "Internet of Things" (IoT), so that a final version of the app would let to handle remotely many devices connected to a network. This way, it would be possible to use the app in many other fields, as could be the home automation, among other fields.

Keywords: *Network of sensors,* wireless communication protocol*, server, UDP, Internet of Things (IoT), home automation.*

Índice General

Índice de figuras

Índice de tablas

Capítulo 1. Introducción

1.1 Ubicación tecnológica

Hoy en día, cualquier individuo de la sociedad actual admitiría que el uso de diversas tecnologías facilita notablemente la vida cotidiana, tanto en entornos personales como profesionales. Esto se pone de manifiesto ante el caso de la interconexión de sistemas empotrados, y, en el caso que nos ocupa, facilitar un mayor y más flexible control de múltiples dispositivos manipulables. Debido a estas ventajas, como resultado se está produciendo una mayor demanda de este tipo de dispositivos.

La comunicación entre distintos tipos de dispositivos cotidianos como teléfonos o cámaras es a día de hoy un hecho. Es por esto que el siguiente requisito en estos dispositivos comunes será la posibilidad de comunicarse con otros objetos que hasta la fecha no suelen poseer de estas capacidades de comunicación. El protocolo de comunicación inalámbrica ZigBee es uno de los sistemas que pueden dar respuesta a esta problemática.

A continuación se detallarán algunos datos de interés sobre varios de los conceptos mencionados relacionados con el proyecto, para posteriormente poder situar con mayor facilidad los datos presentados en el mismo.

1.1.1 Internet de las Cosas

Si de por sí el campo de la informática es de origen reciente en cuanto al desarrollo de conocimiento se refiere, el denominado Internet de las Cosas apenas tiene una década de vida desde que se nombró por primera vez. [1]

Internet de las Cosas, término acuñado por Kevin Ashton el 12 de julio de 2009, se refiere a la interconexión de todos los objetos cotidianos con la red. Ashton expone los beneficios que se podrían obtener al conectar todos los objetos que nos rodean a internet, logrando, entre otras ventajas, poder obtener mayor información de cualquiera de ellos en cualquier momento. Las previsiones vaticinan que, para el año 2020, existirán 30000-50000 millones de dispositivos conectados a internet.

Todo esto ofrece la posibilidad de que, en un futuro no muy lejano, la Internet de las Cosas mejore la calidad de vida de la sociedad, ahorrando energía y haciendo la vida más sencilla, incluso dentro del ámbito doméstico.

1.1.2 Protocolos de comunicación inalámbrica

Existen múltiples protocolos de comunicación que no requieren de una red cableada para su implementación, los cuales facilitan mucho la conexión de los dispositivos para así poder manipularlos de manera más flexible y efectiva. En este proyecto hemos utilizado el protocolo ZigBee, el cual se incluye dentro del grupo de las WPAN (Wireless Personal Area Network) o Redes Inalámbricas de Área Personal. Por tanto, pasaremos a detallar algunas alternativas al protocolo ZigBee dentro de este tipo de redes. (El protocolo ZigBee será detallado en profundidad en el apartado 2.2-Protocolos ZigBee)

Una de las principales alternativas a ZigBee es el protocolo Bluetooth. Este sobrepasa a ZigBee en velocidad de transferencia, posibilidades de interconexión, capacidad para evitar las interferencias en frecuencia y estandarización en perfiles de aplicación. Bluetooth resulta ideal para transferencia de archivos e intercambio de datos, y más adecuado para la comunicación de ordenadores con periféricos. En resumen, se puede determinar que si la aplicación necesita de forma predominante alguna de las siguientes características, se debe elegir un dispositivo Bluetooth:

Imprescindible si se necesita:

- Intercambio de datos en tiempo real (audio o video streaming)
- Calidad de servicio garantizada

Opción más adecuada si se necesita:

- Identificación de compatibilidad entre dispositivos o disponibilidad de servicios por parte de los sensores.
- Interoperabilidad garantizada entre dispositivos similares de distintos fabricantes.

Por otro lado, ZigBee aventaja a Bluetooth en cuanto a duración de la batería de los dispositivos y es más robusto en caso de fallos de sensores en la red. ZigBee es la mejor alternativa para aplicaciones que necesiten la transferencia de pequeños paquetes sin una periodicidad definida a través de redes malladas. En resumen, se puede determinar que si la aplicación necesita de forma predominante alguna de las siguientes características, se debe elegir un dispositivo ZigBee:

Imprescindible si se necesita:

- Capacidad para la interconexión simultánea de cientos de dispositivos.
- Enrutamiento multisalto de mensajes hacia un dispositivo localizado más allá del rango de transmisión directa del dispositivo transmisor.

Opción más adecuada si se necesita:

- Monitorización de múltiples dispositivos.

Como apreciaremos más adelante, el protocolo ZigBee se ajusta perfectamente a los requisitos exigidos por el proyecto: es por este motivo por el que hemos utilizado esta tecnología, para ser más concretos, mediante el uso de dispositivos XBee.

Los módulos XBee son soluciones integradas que brindan un medio inalámbrico para la comunicación e interconexión entre dispositivos. Estos módulos utilizan el protocolo de red IEEE 802.15.4, y fueron diseñados para aplicaciones que requieren de una baja latencia y una sincronización de comunicación predecible, en la que el escaso ancho de banda comparado con otras tecnologías no suponga un impedimento fundamental. Aunque estos módulos están basados en el protocolo ZigBee, XBee es propiedad de la compañía Digi. En términos simples, los XBee son módulos inalámbricos fáciles de usar.

El mercado de ambos estándares (ZigBee y Bluetooth) está creciendo de forma significativa en los últimos años, presentando ambas opciones diferentes, pero en parte solapados, dominios de aplicación. A pesar de haberse descrito solo estas dos opciones, se deben mencionar otras tecnologías que también pueden coincidir parcialmente en cuanto a su dominio de aplicación, como pueden ser Ultra Wide Band, Wi-Fi o la red de telefonía móvil. En la figura 1.1 [3] apreciaremos mejor dicho solape entre estas tecnologías.

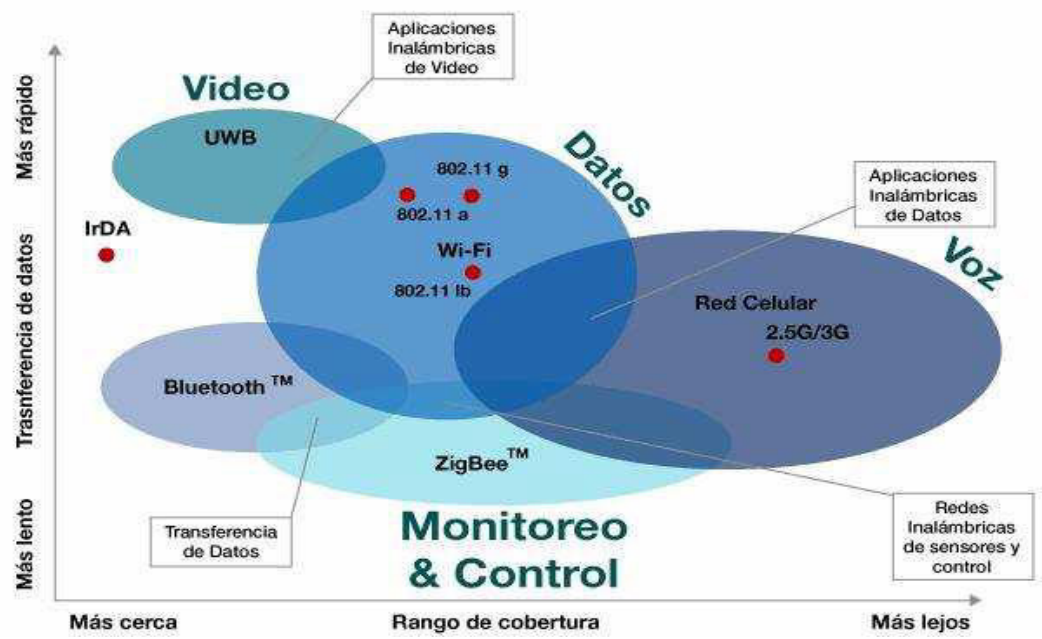


Figura 1.1 Ámbitos de aplicación de distintas tecnologías

A continuación, realizaremos una comparativa general entre estas tecnologías en la tabla 1.1 [3]:

Estándar	ZigBee	Bluetooh 3.0	Ultra Wide Band	Wi-Fi
Especificación IEEE	802.15.4	802.15.1	802.15.3ª	802.11a/b/g
Banda de frecuencias	868/915 MHz; 2.4 GHz	2.4 GHz	3.1-10.6 GHz	2.4 GHz; 5 GHz
Tasa de transferencia máxima	250 Kb/s	1-24 Mb/s	110 Mb/s	54 Mb/s
Alcance	70m	100m	10m	100m
Número de canales	1/10; 16	79	(1-15)	14
Ancho de banda del canal	0.3/0.6 MHz; 2 MHz	1 MHz	500 MHz-7.5 GHz	22MHz
Tipo de modulación	BPSK (+ASK), O-QPSK	GFSK	BPSK, QPSK	BPSK, QPSK, COFDM, CCK, M-QAM
Ensanchamiento	DSSS	FHSS	DS-UWM, MB-OFDM	DSS, CCK, OFDM
Mecanismo de coexistencia	Selección Dinámica de Frecuencia	Salto de Frecuencia Adaptativo	Salto de Frecuencia Adaptativo	Selección Dinámica de Frecuencia, Control de Potencia Transmitida (802.11h)
Número máximo de sensores	> 65000	8 (para una piconet)	8	---

Tabla 1.1: Comparativa de tecnologías inalámbricas

1.1.3 Domótica

La domótica se define como un conjunto de sistemas que permiten automatizar las distintas instalaciones de una vivienda, aportando a sus inquilinos comodidad, seguridad y comunicación.

Los sistemas domóticos se comunican y conectan entre sí realizando tomas de medidas relevantes para alcanzar las características anteriores y actuando en función es estas medidas. Normalmente tiene un nodo central que coordina todas las operaciones que realiza la instalación, y que recibe datos del entorno mediante sensores.

Podemos agrupar los principales servicios que una instalación de este tipo proporciona en cinco tipos [2]:

- Ahorro energético.
- Comodidad y confort.
- Sistemas de seguridad.
- Comunicaciones.
- Accesibilidad.

Un sistema de instalación domótica extendido en la actualidad es KNX, un estándar abierto para el control de casa y edificios. Cualquier fabricante puede

integrar sus productos en el sistema y, además, la red KNX se puede conectar a otros sistemas a través de las pasarelas adecuadas.

Por otra parte, todos los elementos que intervienen en la instalación utilizan un protocolo común para comunicarse.

El medio de transmisión por excelencia de KNX es TP1 (Par trenzado), es decir, mediante un bus de control independiente.

Se pueden destacar dos ventajas de este tipo de redes domóticas:

- Tiene una arquitectura distribuida.
- Todos los elementos son configurables mediante un software único.

El principal inconveniente a día de hoy de las instalaciones domóticas es que puede conllevar un gasto económico y energético considerable, que no todas las familias pueden permitirse. Sin embargo, recientemente se ha comenzado a extender el uso de sistemas y protocolos de comunicación como ZigBee, el cual es utilizado en este proyecto, y que permiten realizar instalaciones de baja complejidad, coste y consumo, soslayando sensiblemente los inconvenientes mencionados.

Una finalidad de la aplicación desarrollada en el presente proyecto podría estar destinada a realizar un sistema domótico utilizando las ventajas del protocolo ZigBee mediante el uso de dispositivos XBee, permitiendo realizar acciones como el encendido y apagado automático de diversos dispositivos, la manipulación de interfaces remotas de control de sistemas domésticos o el guardado de configuraciones personalizadas de cualquier parámetro modificable destinados a diferentes perfiles de uso.

1.2 Objetivos

El principal propósito de este proyecto ha sido implementar una aplicación que pueda comunicarse con una red de sensores que utilizan el protocolo de comunicación inalámbrica ZigBee, de forma que pueda realizarse una configuración flexible de la red y, además, poder manipular de diversas formas los dispositivos conectados a la red.

Para ser más precisos, en este proyecto se ha pretendido completar los siguientes objetivos específicos:

- Estudio y análisis del material proporcionado, tanto de los diversos sensores y dispositivos que conforman la red construida, como del programa que se comunica con el controlador de la misma.
- Flexibilización de conexión de dispositivos y aproximación a un nivel más cercano al usuario del programa de comunicación.

- Implementación de protocolos de uso flexibles de pulsadores y dispositivos analógicos, de forma que sea posible la creación de sistemas distribuidos. El protocolo debe permitir que un evento ocurrido en un módulo pueda producir una acción en otro módulo remoto al primero, de forma que estas operaciones sean totalmente transparentes para el usuario final.
- Diseño y creación de reglas de comunicación a implementar.
- Diseño e implementación de una interfaz de control destinada al usuario final.
- Diferenciación entre versiones cliente-servidor, siendo necesario que la versión del servidor, conectada al nodo coordinador, permanezca en un estado demonizado.

1.3 Metodología

Dado que se pretende realizar el proyecto de forma que se vayan solucionando las problemáticas que surjan sobre la marcha, partiendo de un pequeño guion inicial, consistente en realizar primero tareas de análisis y prueba sobre el código proporcionado para posteriormente ir construyendo nuevo código sobre el mismo, se ha optado por seguir una metodología ágil.

Siendo más específicos, dentro de las metodologías ágiles, hemos optado por seguir la metodología de Desarrollo Basado en Funcionalidades. Partiendo de una lista de funcionalidades requeridas proporcionadas por el cliente (en este caso, el tutor del proyecto), se ha planificado y diseñado diferentes formas de implementar las mismas dentro del proyecto. El cliente en todo momento ha podido comprobar el estado de las funcionalidades añadidas en diversos prototipos construidos para satisfacer cada funcionalidad, y así sugerir mejoras y correcciones que se deban tener en cuenta, sin necesidad de tener que visualizar continuamente el conjunto global del programa.

La planificación del trabajo ha consistido en múltiples fases a seguir, llevadas a cabo siempre en un total de aproximadamente 15 horas semanales. Estas fases han sido:

1. Documentación, análisis y prueba del código proporcionado. Partiendo de los datos recabados, creación de un programa simple, de consola, pero funcional, de manipulación de dispositivos.
2. Planificación, análisis y contraste de diversas maneras de llevar a cabo el proyecto final, buscando posibles herramientas que pudieran facilitar diversos aspectos en la construcción de la aplicación.
3. Diseño y construcción de una versión con interfaz gráfica del programa de manipulación por consola, realizando las mismas funcionalidades que el anterior.

4. Diseño y construcción de la versión final de la aplicación, con todos los casos de uso planteados, así como diferenciando las versiones de servidor-cliente de la misma.

1.4 Entorno de trabajo

Para poder llevar a cabo la realización del proyecto de forma satisfactoria, uno de los aspectos más importantes fue escoger las herramientas de programación y los diversos protocolos que se utilizarían para satisfacer las diversas problemáticas que fueron surgiendo. Es conveniente indicar además que el sistema en el que instalar las herramientas utilizadas ha de estar convenientemente actualizado, para evitar así problemas de compatibilidades.

En la tabla 1.2 mostraremos las diversas herramientas que se analizaron, diferenciando la escogida para realizar la aplicación.

Tipo de herramienta	Opción escogida	Alternativas analizadas
Lenguaje de programación a utilizar	Python 2.7	C++, Python 3.0
Interfaz	PyQt 4	WxPython, Tkinter, interfaz web.
Sistema de comunicación cliente-servidor	Comunicación por UDP	Señales Posix-IPC

Tabla 1.2: Herramientas tratadas

Capítulo 2.
Conocimientos previos

En el capítulo anterior se introdujeron aspectos relevantes del ámbito general en los que se engloba este proyecto, para así comenzar a crear una mejor perspectiva del trabajo concreto que se ha realizado en el mismo. A pesar de esto, también es necesario presentar conceptos más concretos directamente relacionados, para conocer así los dispositivos y herramientas que se utilizaron para desarrollar la aplicación final implementada.

A continuación se incluyen descripciones detalladas de varios de los elementos utilizados en el transcurso de este proyecto.

2.1 Protocolos ZigBee

El protocolo de comunicación ZigBee [4] se basa en el estándar IEEE 802.15.4 de comunicación, el cual define el nivel OSI 1 (capa física) y el control de acceso al medio (MAC) de la capa de enlace de datos para redes inalámbricas de área personal con bajas tasas de transmisión de datos. El protocolo ZigBee, partiendo de dicho estándar, desarrolla las capas superiores del modelo OSI y permite la creación de redes inalámbricas en configuración de malla con un bajo consumo de energía.

Este protocolo fue concebido a finales del pasado milenio, en 1998, cuando diversos instaladores y empresas del sector concluyeron que tanto el protocolo Wi-Fi como el Bluetooth no podrían emplearse en varias aplicaciones. A raíz de estas necesidades, se fundó la ZigBee Alliance, una organización abierta sin ánimo de lucro que agrupaba inicialmente a 25 empresas y compañías, llegando en la actualidad a más de 400, entre las que se incluyen empresas de renombre como Philips, Schneider Electric o Texas Instrument. Esta organización creó y desarrolló el protocolo, específicamente diseñado para las necesidades particulares de redes inalámbricas de bajo coste y baja potencia. Para usos no comerciales, la especificación ZigBee es gratuita para el público general, y cualquier empresa que entre al nivel más básico de la alianza, tiene el derecho a acceder a especificaciones todavía no lanzadas al público y crear y comercializar productos usando este protocolo.

Las radios toleran de forma nativa redes en estrella o árbol, así como topología en malla. Esto permite al sistema la transmisión de datos entre largas distancias sin necesidad de un controlador central capaz de conectarse a todos los miembros de la

red. El protocolo opera en varias bandas de frecuencia de radio ISM (reservadas para el uso industrial, científico y médico), dependiendo del lugar geográfico y de las necesidades de velocidad de transmisión. A su vez, cada banda tiene varios canales distintos habilitados para su uso. Todos estos datos de la capa física del protocolo se definen en el estándar IEEE, el cual ha sido objeto de varias reformas para mejorar estas características.

De esta forma, se pueden tener transmisiones con velocidades desde 20 kb/s a 868 MHz hasta 250 kb/s en la banda de 2.4 GHz.

Una red ZigBee cuenta siempre con hasta tres tipos de dispositivos:

- Un único coordinador que se encarga de crear la red, administrar las direcciones y llevar a cabo funciones de seguridad y mantenimiento de la red.
- Routers, nodos con plenas capacidades de funcionamiento. Pueden unirse a redes existentes, enviar y recibir información o reenviarla hacia otros destinos, permitiendo que se pueda establecer una comunicación fiable entre otros dos puntos que estén a demasiada distancia como para conectarse directamente. Un mensaje se va transmitiendo de uno a otro hasta llegar al destino final. Para cumplir con estas funciones, los nodos que sean routers deben estar siempre encendidos y disponibles, por lo que suelen estar directamente alimentados mediante una toma de corriente.
- Dispositivo final, es un nodo router con funciones reducidas. Estos dispositivos actúan como terminaciones de una red, y sus capacidades incluyen unirse a redes y enviar o recibir información. Como no necesitan estar continuamente en funcionamiento, requieren un hardware más barato y con menor consumo energético. Para esto, los dispositivos finales pueden entrar periódicamente en modo de ahorro de energía o modo sueño. Todo dispositivo final necesita un router o coordinador como dispositivo padre, el cual ayuda a los dispositivos a unirse a las redes y almacenar los mensajes dirigidos a ellos en caso de que estén en estado de hibernación.

El direccionamiento de dispositivos consta de dos direcciones:

- Dirección de 64 bits: cada nodo posee una dirección permanente de 64 bits única y fijada de fábrica, la cual identifica completamente al dispositivo.
- Dirección de red de 16 bits: se asigna a un nodo cuando este se une a una red. El protocolo requiere que los datos se envíen a esta dirección, lo que obliga a conocerla antes de poder comunicarse. Esta dirección es única para cada nodo de la red, pero es variable, puede cambiar si se produce una de las siguientes situaciones:

a) Si un dispositivo final no se puede comunicar con su padre, necesita abandonar la red y reconectarse para encontrar un nuevo padre.

b) Si el tipo de dispositivo cambia de router a dispositivo final o viceversa, este debe desconectarse de la red y volverse a unir como un dispositivo del nuevo tipo.

A pesar de esto, el protocolo incluye la funcionalidad para emitir mensajes en modo broadcast, de forma que cualquier dispositivo que esté conectado reciba el mismo mensaje. También permite realizar un multicast o direccionado a un grupo.

Del mismo modo, cada red o PAN (Personal Area Network) creada tendrá su propia dirección para diferenciarla de otras redes. Todas estas direcciones, junto con el canal de frecuencia correcto, permiten identificarla y comunicarse con cualquier elemento miembro de una PAN en un conjunto de varias redes coexistiendo en un mismo entorno, sin interferir las unas con las otras.

Por último, es importante señalar que el protocolo que emplean los módulos XBee para comunicarse entre sí dentro de las redes que crean es siempre el mismo.

En la figura 2.1 se muestra el módulo coordinador que se ha utilizado en la red creada para el proyecto:

Figura 2.1: Nodo coordinador de la red

2.2 Dispositivos XBee

Los módulos XBee [5] son la opción escogida para realizar la implementación de una red de sensores y dispositivos actuadores conectados entre sí. Estos módulos son perfectos para cubrir las necesidades del proyecto, ya que como se describirá más detalladamente a continuación, cuentan con la capacidad de recoger directamente información de sensores tanto digitales como analógicos, y transmitirla a otros nodos sin necesidad de emplear microcontroladores adicionales.

Así mismo, también poseen salidas digitales y de PWM que les permiten realizar actuaciones básicas por sí solos.

Las ventajas que aportan estas características son:

- Alto rendimiento con bajo coste. El rango de alcance en entornos interiores o urbanos es de 40 metros, alcanzando los 120 metros en exteriores y con línea de visión directa. El XBee PRO tiene un rango mayor (unos 100 metro en interior y hasta 1.6 Kilómetros en exterior y con línea directa de visión).
- Bajo consumo energético. El módulo XBee tiene un consumo máximo de corriente de 40 miliAmperios a 3.3 Voltios. Si no está ejecutando ninguna tarea pero está encendido, su consumo es de 15 miliAmperios, pero si se configura para entrar en modo hibernación, el consumo en dicho modo es menor de 1 microAmperio. El módulo PRO tiene un consumo mayor de energía debido a sus mejoradas prestaciones.
- Creación avanzada de redes y alta seguridad. El sistema de interconexión cuenta con procesos de reintegro y confirmación de recepción, múltiples direcciones disponibles, admite topologías punto a punto, estrella y malla, y tiene la capacidad de auto-rutear, auto-reparar y tolerar fallos.
- Fácil de usar. Cuenta con dos modos de funcionamiento (transparente y API) para enviar órdenes AT y configuraciones. Tiene un amplio catálogo de comandos y cuenta con un software gratuito de configuración y testeo de los dispositivos (Aunque en este proyecto se haya desarrollado una aplicación específica para realizar estas operaciones en vez de utilizar este software gratuito). Además, funciona con frecuencias estandarizadas y se ha manufacturado bajo el estándar ISO 9001:2000.

Los dispositivos que se han empleado en este trabajo serán unas radios XBee ZNet 2.5 OEM RF Module [7], anteriormente conocida como Serie 2. Se trata de un pequeño circuito integrado de 2.5 x 2.8 centímetros, con 20 pines de inserción y tamaño de paso de 2 milímetros. Implementa el protocolo ZigBee y se emplea principalmente para proyectos con la necesidad específica de una red inalámbrica sensorial de bajo consumo y coste. Opera en la banda de frecuencias ISM de 2.4 GHz y se alimenta mediante una fuente de tensión continua comprendida entre los 2.1 y los 3.6 Voltios. Su funcionamiento nominal es a 3.3 Voltios.

Figura 2.2 Módulo XBee utilizado

A continuación, en la tabla 2.1 [6][7] detallaremos las funciones de cada pin de este módulo XBee:

Pin	Nombre	Dirección	Descripción
1	VCC	---	Fuente de alimentación
2	DOUT	Salida	Salida de datos UART
3	DIN / CONFIG	Entrada	Entrada de datos UART
4	DIO12	Entrada/Salida	E/S digital 12
5	RESET	Entrada	Reseteo del módulo (pulso min. 200 ns)
6	PWM0 / RSSI / DIO10	Entrada/Salida	PWM salida 0 / Indicador fuera señal RSSI / E/S digital 10
7	PWM / DIO11	Entrada/Salida	E/S digital 11
8	[reservado]	---	No conectar
9	DTR / SLEEP_RQ / DIO8	Entrada/Salida	Control de hibernación por pin / E/S digital 8
10	GND	---	Tierra
11	DIO4	Entrada/Salida	E/S digital 4
12	CTS / DIO7	Entrada/Salida	Control de flujo Clear-to-send / E/S digital 7
13	ON / SLEEP / DIO9	Salida	Indicador de estado del módulo / E/S digital 9
14	[reservado]	---	No conectar
15	Associate / DIO5	Entrada/Salida	Indicador asociado / E/S digital 5
16	RTS / DIO6	Entrada/Salida	Control de flujo Request-to-send / E/S digital 6
17	AD3 / DIO3	Entrada/Salida	Entrada analógica 3 / E/S digital 3
18	AD2 / DIO2	Entrada/Salida	Entrada analógica 2 / E/S digital 2
19	AD1 / DIO1	Entrada/Salida	Entrada analógica 1 / E/S digital 1
20	AD0 / DIO0 / Commissioning Button	Entrada/Salida	Entrada analógica 0 / E/S digital 0 / Botón de comisionado

Tabla 2.1: Distribución de pines del XBee PRO ZNet 2.5

Estos dispositivos requieren de 2 conexiones para poder funcionar: VCC y GND. Además, los módulos XBee ZNet 2.5 pueden comunicarse con otro dispositivo a través de un puerto serial asíncrono (UART), compatible en voltaje y lógica, mediante los pines DIN y DOUT.

Para lograr una comunicación efectiva de la aplicación desarrollada en el proyecto con el módulo XBee que hace las veces de nodo coordinador de la red de sensores, se han utilizado diversos comandos AT que soportan estos módulos[7]. De esta forma, haciendo uso de la comunicación API, la aplicación envía diversos tipos de comandos que el módulo interpreta. Estos comandos son los siguientes:

- Comandos de configuración de la red de sensores:

Estos comandos inicializan diversos parámetros necesarios para crear la red de sensores, y se utilizan al arrancar el motor de comandos principal de la aplicación desarrollada, el cual se analizará con más detalle posteriormente.

Comando	Descripción
SH	Lee los 32 bits superiores de la dirección de 64 bits propia de cada módulo.
SL	Lee los 32 bits inferiores de la dirección de 64 bits propia de cada módulo.
VR	Lee la versión del firmware de cada módulo.
AI	Lee la información relativa a la última petición de unión de cada módulo a la red de sensores.
OP	Lee el identificador de la red PAN al que está asociado cada módulo.
CH	Lee el número del canal de comunicación usado para la transmisión entre módulos de cada uno de ellos.
NI	Almacena la cadena identificadora de cada módulo.
ND	Descubre y toma información de todos los módulos detectados.

Tabla 2.2: Comandos de configuración de la red

- Comandos de muestreo:

Estos comandos son útiles para obtener el estado actual de los diversos dispositivos conectados a los módulos, y se utilizan para obtener el valor actual de los dispositivos de entrada, digitales o analógicos, ya sea cuando se produzca un cambio de estado o de manera continua cada intervalo de tiempo indicado.

Comando	Descripción
IS	Realiza una sola lectura de todas las líneas digitales y analógicas activadas en el módulo en el que se activa el comando.
IR	Realiza un muestreo periódico de las líneas digitales y analógicas activadas. Para activarse el muestreo periódico, debe especificarse un intervalo de tiempo en milisegundos junto al comando. Para desactivarlo, basta con establecer este intervalo a 0.
IC	Establece/devuelve los pines donde se monitorean cambios de estado digitales. Si un pin está configurado como de entrada digital, el comando lanza de forma inmediata un mensaje cuando detecta un cambio de estado. El comando usa una máscara de bits en formato hexadecimal con la que distinguir los canales donde detectar cambios de estado, asignando un bit a cada uno. Los canales que no se monitorean tendrán su bit correspondiente de la máscara establecido a 0.

Tabla 2.3: Comandos de muestreo

- Comandos de configuración de pines:

Estos comandos sirven para modificar el estado actual de los dispositivos conectados a los módulos, y se utilizan para activar y desactivar dispositivos de salida e inicializar correctamente cada dispositivo de entrada.

Pin que configura	Comando	Modo de configuración
Pin 1	P0	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 4	P2	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 7	P1	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 11	D4	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 15	D5	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 16	D6	0 = Desactivación del pin 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 17	D3	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 18	D2	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 19	D1	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.
Pin 20	D0	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como salida digital, con valor “desactivado”. 5 = Pin establecido como salida digital, con valor “activado”.

Tabla 2.4: Comandos de configuración de pines

2.3 Otros campos de interés

A continuación detallaremos brevemente las características de varios conceptos que ha sido necesario utilizar en el desarrollo del proyecto por diversos motivos: los lenguajes y herramientas de programación utilizados, el protocolo de comunicación UDP, y el formato de texto ligero JSON para el intercambio de datos.

2.3.1 Python y PyQt

Python [8] es un lenguaje de programación que fue diseñado para ser leído con facilidad, haciendo uso de una sintaxis que favorece crear un código legible a la hora de programar. Este lenguaje se ha utilizado para implementar la totalidad del código de la aplicación desarrollada en este proyecto. Para ello, se han utilizado ciertas características del lenguaje que facilitan en gran medida la creación de los algoritmos a desarrollar, como son:

- Estructuras de datos:

1. Listas: Se declaran usando corchetes, separando los valores que contiene mediante comas, los cuales pueden ser de diversos tipos. Es posible acceder a estos valores de forma ordenada haciendo uso de índices enteros, y además permiten modificarlos en tiempo de ejecución.
2. Diccionarios: Se declaran usando llaves, separando sus elementos mediante comas. Estos elementos están formados por un par "clave:valor", de forma que se pueda acceder a un valor determinado referenciando la clave a la que esté asociada. Mientras que las claves son inmutables y deben ser siempre únicas, los valores pueden ser de diversos tipos y pueden ser modificados en tiempo de ejecución.

- Hilos:
  Mediante el uso del módulo "threading", se crean múltiples hilos que se ejecutarán de manera concurrente al programa principal, de forma que no se interfieran entre ellos. Si fuera necesario, es posible crear métodos que comuniquen los diversos hilos y que así todos puedan trabajar de forma cooperativa.

- Otros módulos:

1. re: Permite utilizar diversas funciones que manejan expresiones regulares.
2. Queue: Permite utilizar colas de comunicación entre distintos ámbitos del programa.
3. Json: Permite leer y escribir ficheros con el formato JSON.
4. Socket: Permite crear una conexión entre aplicaciones haciendo uso de un protocolo de comunicación determinado (en nuestro caso, UDP).

5. Pickle: Permite encapsular estructuras de datos en paquetes que se puedan enviar fácilmente mediante una comunicación UDP.

Por último, para poder crear una aplicación accesible y fácil de utilizar de cara al usuario final, se ha optado por crear una interfaz gráfica para la misma.

Esta interfaz se ha realizado mediante la herramienta PyQt, una biblioteca multiplataforma ampliamente utilizada para desarrollar aplicaciones con interfaz gráfica de usuario, la cual es una versión de Qt destinada al lenguaje de programación Python.

Esta herramienta permite diseñar fácilmente interfaces completamente configurables, además de proporcionar diversas utilidades que posibilitan crear múltiples protocolos internos para la aplicación.

2.3.2 UDP

UDP (User Datagram Protocol) [9] es un protocolo del nivel de transporte basado en el intercambio de datagramas. Posee ciertas características que lo convierten en la elección oportuna para la comunicación cliente-servidor de nuestra aplicación:

- Trabaja con paquetes o datagramas enteros: La aplicación intercambia información estructurada en bloques de bytes, de forma que por cada bloque de bytes enviado de la capa de aplicación a la capa de transporte, se envía un paquete UDP.
- Provoca poca carga adicional en la red: Es sencillo y emplea cabeceras muy simples.

Por otra parte, existen dos problemas a tener en cuenta con el protocolo UDP, pero que no han supuesto un impedimento relevante en el desarrollo del proyecto:

- Trabaja sin conexión: no emplea ninguna sincronización entre el origen y el destino. Esto se ha resuelto en el código creando un protocolo que solvente en gran medida el problema.
- No es fiable: No controla el flujo ni el orden de los paquetes recibidos. Este problema no debería suponer demasiado inconveniente, si tenemos en cuenta que el contenido de los paquetes a enviar en la aplicación son de un tamaño considerablemente escaso. En cualquier caso, se ha diseñado en el código un sistema que deseche los paquetes que no coinciden con lo esperado.

En la figura 2.3 podemos ver el algoritmo que se aplica para realizar la comunicación cliente-servidor en Python:

SERVIDOR

```
# Datagram Datagram (udp) socket
try :
  self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
  print 'Socket creado'
except socket.error, msg :
  print 'Error al crear el socket. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
  sys.exit()


# Conectar el socket al host local y al puerto
try:
  self.s.bind((self.HOST, self.PORT))
except socket.error , msg:
  print 'Error al conectar. Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
  sys.exit()

print 'Conexion del socket completada'

self.s.settimeout(5.0)


def run(self):                          # Ahora permanece comunicandose con el cliente
  while self.q_salir.empty() == True:  # Señal de apagado global
    print "El servidor UDP esta corriendo"
    try:
      d = self.s.recvfrom(1024)         # Recibe data desde el cliente (data, addr)
    except socket.timeout:
      continue

    data = d[0]
    addr = d[1]


    if data == "H":                     #Caso senal H = Comprobacion de conexion con cliente
      reply = "OK"
      print "{}".format(reply)
      self.s.sendto(reply , addr)
```

CLIENTE

```
#Crear el socket dgram udp
try:
  self.s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
except socket.error:
  print 'Error al crear el socket'
  sys.exit()
else:
  print "Hilo cliente UDP creado"

self.host = 'localhost';
self.port = 8888;

self.s.settimeout(5.0)


def run(self):

  try:
    # Envia la senal
    self.s.sendto(msg, (self.host, self.port))
  except socket.error, msg:
    print 'Error Code : ' + str(msg[0]) + ' Message ' + msg[1]
  else:
    try:
      print "Recibiendo..."
      d = self.s.recvfrom(1024)
    except socket.timeout:
      print "ERROR DE CONEXION-B"
    else:
      print "Conexion establecida correctamente"
```

Figura 2.3: Algoritmo de comunicación UDP en Python

Por una parte, el servidor crea y conecta un socket para escuchar todos los mensajes que se le envíen por un puerto determinado. Al recibir un mensaje, toma por una parte el contenido del mismo, y por otro la dirección del remitente, para a continuación comprobar el contenido del mensaje. Si no se ha producido ningún error, se envía un mensaje de confirmación al emisor utilizando la dirección adquirida previamente, para pasar de nuevo a esperar por otro mensaje entrante.

Por otra parte, el cliente crea y conecta un socket que se comunica específicamente con la dirección y puerto del servidor. Al enviar el mensaje pertinente, se espera un tiempo determinado por la confirmación de llegada que proviene del servidor: si esta no llega, se comunica al usuario un mensaje de error de conexión. Una vez acabada la comunicación, el socket se cierra.

2.3.3 Json

JSON (JavaScript Object Notation) [10] es un formato para el intercambio de datos, el cual describe los mismos con una sintaxis dedicada que se usa para identificar y gestionar los datos. Esta sintaxis compone una estructura de datos que todos los lenguajes serán capaces de soportar de una forma u otra.

El formato JSON posee una ventaja fundamental: para un programador es sencillo de leer y escribir, y para una máquina es simple interpretarlo y generarlo.

En nuestro caso, ha sido necesario crear un sistema de almacenamiento de configuraciones y, dadas las virtudes de este formato, hemos optado por utilizarlo para generar de forma sencilla y visual los ficheros de configuración imprescindibles para el correcto funcionamiento de la aplicación.

Los tipos de datos disponibles en JSON son:

- Números: Pueden ser negativos o contener parte fraccional.
- Cadenas: Secuencias de cero o más caracteres, puestas entre doble comilla.
- Booleanos: Pueden tener dos valores, true y false.
- Vector: Representa una lista ordenada de cero o más valores, los cuales pueden ser de cualquier tipo, separados por comas y dentro de corchetes todo el vector.
- Objetos: Son colecciones no ordenadas de pares "Nombre":"Valor", separados por comas y puestas entre llaves. El nombre debe ser una cadena, mientras que el valor puede ser de cualquier tipo.

Mediante el lenguaje de programación Python, existen funciones que permiten leer y escribir ficheros JSON. Estas son las funciones que se han utilizado en el proyecto, haciendo uso de la librería "json" que proporciona Python:

```
#Función Guardar
with open('configuracion.json', 'w') as outfile:
  json.dump(self.new_dicpines, outfile, sort_keys = True, indent = 4, ensure_ascii=False)
```

```
#Función Cargar
with open("dispositivos.json") as f:
  self.dicdispositivos = json.load(f)
```

Figura 2.4: Funciones de lectura/escritura de ficheros JSON

La función de escritura de archivos JSON funciona sencillamente usando el nombre del fichero donde se desea escribir la información usando la función "open" (en nuestro caso, siempre han tenido formato de diccionario, con valores de diversos tipos: cadenas, listas, otros diccionarios a su vez…). Mediante el uso de la

función “json.dump” se vuelca en el fichero el diccionario que se desea guardar, mencionando a continuación el formato en el que se almacenará el mismo (indentaciones, caracteres, etc.).

La función de lectura de archivos JSON funciona sencillamente usando el nombre del fichero de donde se desea leer la información usando la función “open”. Una vez hecho esto, basta con guardar el contenido del archivo en una variable del programa utilizando la función “json.load”.

A continuación, presentaremos un ejemplo de cada fichero JSON basado en la configuración que genera una versión avanzada de nuestra aplicación.

Configuración de pines

```
"E13": {
    "1": "Indicador",
    "10": "",
    "11": "",
    "12": "",
    "13": "",
    "14": "",
    "15": "Indicador",
    "16": "",
    "17": "",
    "18": "Analogico",
    "19": "",
    "2": "",
    "20": "Pulsador",
    "3": "",
    "4": "Pulsador",
    "5": "",
    "6": "",
    "7": "Indicador",
    "8": "",
    "9": ""
},
```

Configuración de dispositivos

```
"rojo": [
    "Indicador",
    "1",
    "E13"
],
"s_luz": [
    "Analogico",
    "18",
    "E25"
],
"s_presencia": [
    "Pulsador",
    "4",
    "E25"
],
"ventilador": [
    "Actuador",
    "7",
    "E25"
],
```

Configuración de acciones de pulsadores

```
"b_plano": {
    "P_Doble": [
        "orange",
        "4",
        ""
    ],
    "P_Larga": [
        "orange",
        "2",
        "5"
    ],
    "P_Muylarga": [
        "ventilador",
        "3",
        ""
    ],
    "P_Simple": [
        "orange",
        "2",
        "30"
    ],
    "P_Tipo": "Pulsador"
},
```

Configuración de acciones de analógicos

```
"s_luz": {
    "Evento_Max": [
        "orange",
        "4",
        ""
    ],
    "Evento_Med": [
        "verde",
        "2",
        "5"
    ],
    "Evento_Min": [
        "verde",
        "1",
        ""
    ],
    "Histeresis": [
        0.015,
        0.015
    ],
    "Umbrales": [
        0.8,
        0.2
    ]
},
```

Figura 2.5: Ejemplos de fichero JSON de configuración de la aplicación

Capítulo 3.
Trabajo desarrollado

En este capítulo nos centraremos en mostrar todas las características que componen la aplicación diseñada. Comenzaremos por explicar cómo funciona la comunicación entre la aplicación y el controlador de la red, para después continuar explicando los protocolos que se han diseñado diferenciados explícitamente según el tipo de dispositivo al que va dirigido. Para concluir, explicaremos el funcionamiento de la interfaz de la aplicación mostrando numerosas figuras que reflejen lo que se vería en pantalla una vez la aplicación haya sido lanzada, diferenciando además los aspectos en los que difiera la versión de la aplicación del cliente con la versión del servidor.

En la figura 3.1 se muestra un diagrama de los elementos que intervienen en el proyecto, así como sus correspondientes relaciones:

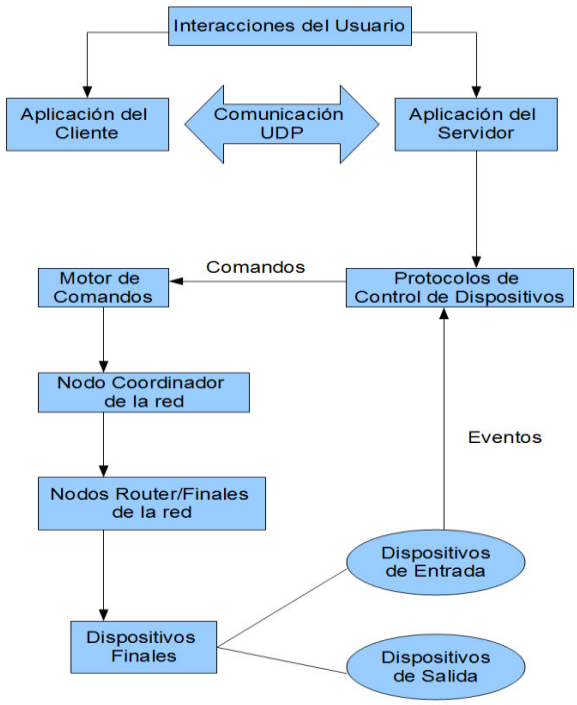


Figura 3.1: Relación de los elementos que intervienen en el proyecto

3.1 Motor de comandos

En una red de sensores, el nodo coordinador es el que se encarga de iniciar el envío de los diversos mensajes, tanto al resto de nodos que conforman la red como a los dispositivos que puedan estar conectados al mismo. Estas operaciones las realiza mediante un protocolo API que soportan los sensores XBee.

Mediante el uso de un API (Interfaz de Programación de Aplicaciones) basada en tramas, amplía el nivel de interacción de una aplicación anfitriona con las capacidades de red del módulo. En el modo API, todos los datos entrantes o salientes se agrupan en tramas que definen operaciones o eventos dentro del módulo. Una aplicación externa puede enviar tramas de datos que contienen la dirección y otra información necesaria al módulo, en lugar de utilizar el otro modo de comandos disponible para modificar direcciones. El módulo responderá enviando otra trama con paquetes de estado entre otros datos.

Las ventajas de este modo API son muy numerosas:

- Además de permitir el envío de comandos AT, permite la recepción de mensajes de estado de entradas y salidas digitales de uno o más nodos remotos, permitiendo crear una red de sensores que trasmitan información sobre variables.
- Configuración a distancia de nodos mediante comandos AT remotos.
- Rapidez de direccionamiento de mensajes y capacidad de envío de mensajes en modo broadcast.
- Respuestas a paquetes enviados para tener confirmación de la correcta recepción de los mismos, y posibilidad de reintentar el envío del paquete en caso de no recibir el mensaje de respuesta.
- CRC (Checksum) para determinar la integridad de los datos recibidos y proteger frente a fallos y perturbaciones.

En el programa proporcionado podemos observar que podemos manipular varias tramas API, las cuales pasaremos a mostrar en la tabla 3.1:

Valor de la trama	Nombre de la trama
0x88	Comando AT
0x8A	Estado del módem
0x97	Respuesta a comando remoto
0x91	Indicador de Rx explícito ZigBee -> Recepción remota
0x92	Indicador de muestreo de datos Rx de E/S ZigBee
0x95	Recepción de identificador de nodo
0x90	Recepción de paquete ZigBee

Tabla 3.1: Valores y nombres de las tramas API

Para que el programa pueda obtener estas tramas que de alguna manera debe proporcionarle el usuario mediante la interfaz, hemos creado un hilo que estará continuamente activo, preguntando si existe algún comando disponible que interpretar. Estos comandos tienen la estructura que se muestra en la figura 3.2:

```
Nombre del Módulo + : + Comando AT
```

Figura 3.2: Estructura de comandos obtenidos a interpretar

De esta forma, el programa interpreta dos partes diferenciadas del comando utilizando el símbolo ":" como separador. Por una parte, del nombre del módulo obtiene la dirección a la que irá destinado el comando, y por otra parte, utilizando el comando AT ejecutará la orden conveniente que se le haya proporcionado.

Como se puede observar, estos comandos y el hilo que los envía al intérprete son una parte fundamental de la aplicación, sin los cuales no habría canal de comunicación entre la aplicación y la red de sensores.

Por otra parte, la lógica de la aplicación siempre es implementada en la versión del servidor de la aplicación. Cuando se dan órdenes desde la versión del cliente, sencillamente se envía un mensaje al servidor mediante la comunicación UDP establecida, para que este construya y envíe el comando correspondiente a la red de sensores.

3.2 Protocolos de control de dispositivos

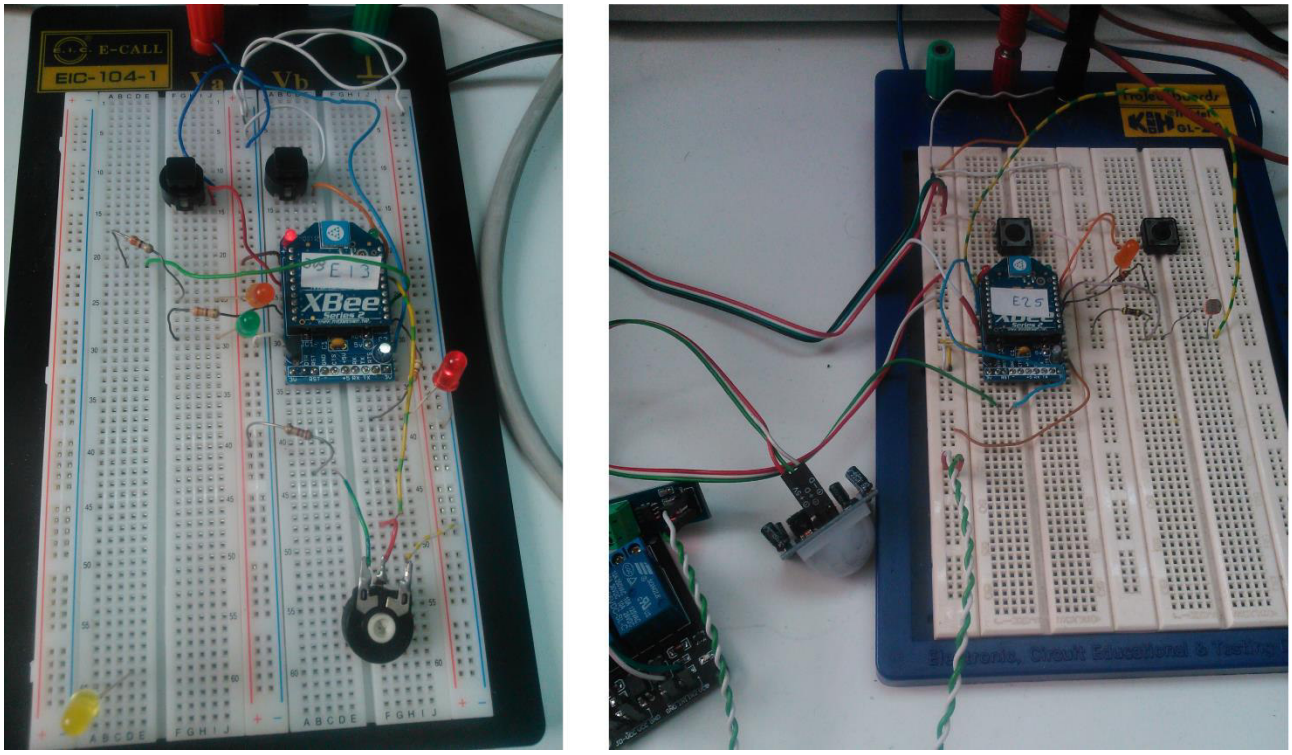


Figura 3.3: Nodos router de la red, conectados a diversos dispositivos interactuables

Antes de comenzar a analizar los distintos protocolos de acción que necesita la lógica de la aplicación para funcionar correctamente, para cada uno de los tipos de dispositivo es necesario indicar que, para que nuestra aplicación pueda tener en cuenta qué dispositivos se encuentran disponibles, es necesario realizar una pequeña inicialización de los mismos indicando el nombre y tipo de cada dispositivo, así como el pin del módulo al que se ha conectado. Esto genera un archivo de configuración en formato JSON que servirá para no perder los cambios realizados, y así ser utilizado como futura referencia interna por la aplicación para poder realizar operaciones como la construcción de comandos destinados a cada uno de los dispositivos inicializados.

Para poder interactuar con los diversos dispositivos conectados a la red de sensores, como la que se muestra en la figura 3.3, es necesario implementar una serie de protocolos específicos para cada tipo de dispositivo. También es necesario tener en cuenta que, para cada uno de los dispositivos, existe un rango determinado de acciones. Por lo tanto, es necesario crear un protocolo específico para cada tipo de interacción. Para empezar, hemos diferenciado los tipos de dispositivos que podemos conectar a la red entre dos grandes categorías, que a su vez dan lugar a un total de cuatro subcategorías. También señalaremos las acciones que se pueden realizar con cada tipo de dispositivo, y explicaremos qué protocolo se ejecuta cuando se envía la orden de inicio para cada uno de ellos:

3.2.1 Dispositivos de salida

En esta categoría se engloban aquellos dispositivos con los que el usuario no puede interactuar directamente, sólo mediante órdenes enviadas desde la aplicación, activándolos o desactivándolos según le convenga. En ella se incluyen:

a) Indicadores: Estos dispositivos se distinguen por su capacidad para proporcionar señales lumínicas cuando son activados. Sobre todo, se han utilizado para el proyecto diodos LED de diversos colores, y de esta forma poder realizar pruebas visuales a la hora de testear la aplicación.

   Las interacciones disponibles para indicadores son las siguientes:

   1. Activar indicador: Se construye y envía un comando que provoca la activación del indicador.
   2. Activar indicador durante un tiempo determinado: Mediante la creación de un hilo que se ejecuta de forma concurrente al programa principal, se envía la señal de encendido del indicador y empieza a esperar mediante un *timer* el tiempo que se haya especificado al dar la orden. Una vez alcanzado el valor proporcionado, se envía la señal de apagado del indicador. Es posible interrumpir la señal de apagado ejecutando un comando de activación/desactivación del indicador. Esta operación no puede realizarse a la vez que se esté ejecutando la operación de parpadeo del mismo dispositivo.

   3. Iniciar parpadeo con un intervalo determinado: Mediante la creación de un hilo que se ejecuta de forma concurrente al programa principal, se envía periódicamente la señal de encendido y apagado del indicador, dejando que transcurra el intervalo de tiempo entre ambas operaciones especificado al dar la orden. Para interrumpir la ejecución de la operación de parpadeo es necesario ejecutar un comando de activación/desactivación del indicador. Esta operación no puede realizarse a la vez que se esté ejecutando la operación de activación del mismo dispositivo durante un tiempo determinado.

   4. Desactivar indicador: Se construye y envía un comando que provoca el apagado del indicador.

b) Actuadores: Estos dispositivos, en cambio, se distinguen por ser capaces de realizar otro tipo de eventos cuando son activados, como por ejemplo, activar diversos tipos de dispositivos, como pudieran ser motores, luminarias, calefactores, etcétera. Además, es necesario señalar que es necesario que cada actuador esté conectado mediante lógica inversa para que las operaciones a ejecutar se realicen correctamente.

Las interacciones disponibles para actuadores son las siguientes:

1. Activar actuador: Se construye y envía un comando que provoca la activación del actuador.
2. Activar actuador durante un tiempo determinado: Mediante la creación de un hilo que se ejecuta de forma concurrente al programa principal, se envía la señal de encendido del actuador y empieza a esperar mediante un *timer* el tiempo que se haya especificado al dar la orden. Una vez alcanzado el valor proporcionado, se envía la señal de apagado del actuador. Es posible interrumpir la señal de apagado ejecutando un comando de activación/desactivación del actuador.
3. Desactivar actuador: Se construye y envía un comando que provoca el apagado del actuador.

3.2.2 Dispositivos de entrada

En esta categoría se engloban aquellos dispositivos que captan señales del entorno y con los cuales el usuario puede interactuar, produciendo un tipo de señal que es registrada e interpretada por la aplicación, produciendo a su vez un resultado que previamente pueda haber sido configurado por el usuario como, por ejemplo, la manipulación de uno de los dispositivos de salida. En esta categoría se incluyen:

a) Pulsadores: Estos dispositivos, que hemos englobado como "Pulsadores", engloban varios tipos de dispositivos de entrada digital, como pudieran ser los botones o los sensores de presencia, los cuales sólo pueden poseer estado "Activado" o "Desactivado".

Podemos asignar diversas acciones a ejecutar según el tipo de pulsación que se realice en un botón. Se debe tener en cuenta que, aunque los sensores digitales se engloben en este proyecto en la categoría de "Pulsadores", no es posible realizar distintos tipos de pulsaciones en los mismos, por lo que sólo se podrá asignar un tipo de acción al ser activados.

1. Evento de pulsación simple: Se produce una acción previamente asignada al presionar sencillamente una vez durante menos de dos segundos un pulsador determinado.
2. Evento de pulsación doble: Se produce una acción previamente asignada al presionar un pulsador determinado dos veces seguidas, con un margen de un segundo entre cada pulsación.
3. Evento de pulsación larga: Se produce una acción previamente asignada al presionar una vez durante al menos dos segundos, pero menos de cinco, un pulsador determinado.

4. Evento de pulsación muy larga: Se produce una acción previamente asignada al presionar una vez durante más de 5 segundos un pulsador determinado.
5. Evento de activación del sensor: Se produce una acción previamente asignada cuando se activa un sensor determinado. Cuando el dispositivo manipulado no es un botón sino un sensor digital, éste será el único tipo de evento a configurar. Esto se debe a que, por ejemplo, no es posible realizar "eventos de pulsación simple/doble/larga" mediante un sensor de presencia.

b) Analógicos: Estos dispositivos, en cambio, se distinguen por poseer una entrada analógica que puede variar dentro de un rango, como pudieran ser los potenciómetros o los sensores de luminosidad.

Podemos asignar diversas acciones a ejecutar cuando se sobrepasen unos determinados umbrales especificados previamente en la aplicación. Además, para prevenir continuos cambios de estado a causa de posibles variaciones no deseadas en el valor propias de cada dispositivo, se han creado también unos márgenes de histéresis que sirven como margen extra a ambos lados de cada umbral. En la figura 3.4 se pueden apreciar mejor los umbrales que dividen los eventos de cada dispositivo analógico:

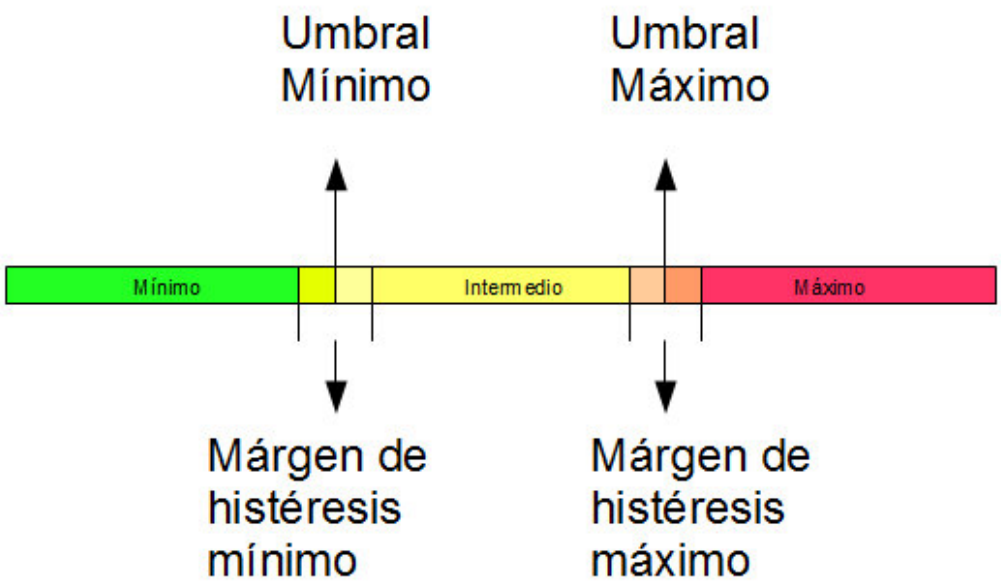


Figura 3.4: Diagrama de umbrales de dispositivos analógicos

1. Evento de umbral mínimo: Se produce una acción previamente asignada al sobrepasar el umbral mínimo hacia valores inferiores. Una vez en este estado, no cambia a otro hasta que sobrepasa el margen de histéresis mínimo hacia valores superiores.

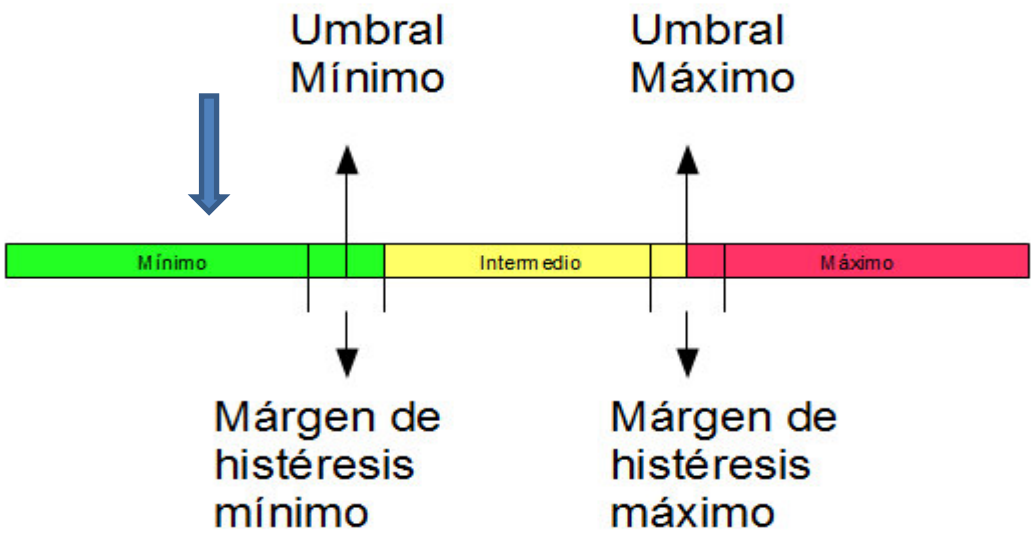


Figura 3.5: Diagrama de umbrales al alcanzar estado mínimo

2. Evento de umbral máximo: Se produce una acción previamente asignada al sobrepasar el umbral máximo hacia valores superiores. Una vez en este estado, no cambia a otro hasta que sobrepasa el margen de histéresis máximo hacia valores inferiores.

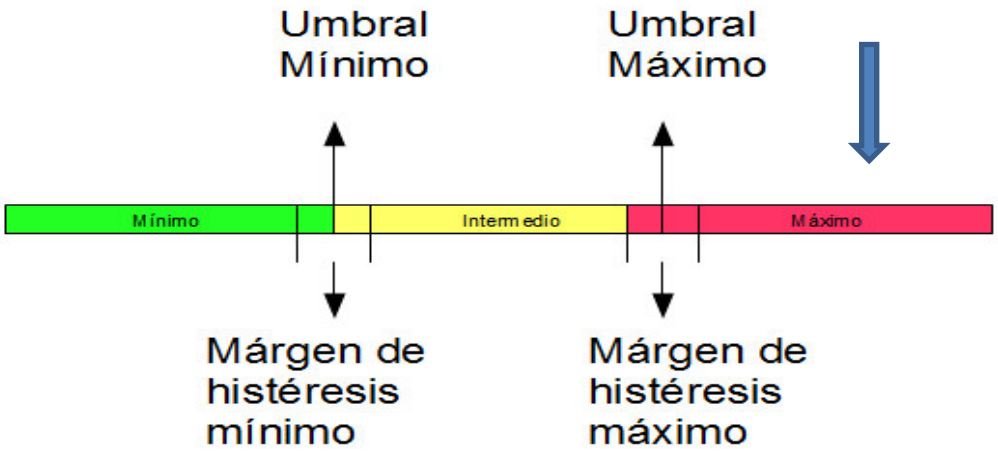


Figura 3.6: Diagrama de umbrales al alcanzar estado máximo

3. Evento de umbral intermedio: Se produce una acción previamente asignada al sobrepasar el umbral máximo hacia valores inferiores sin sobrepasar el umbral mínimo; o al sobrepasar el umbral mínimo hacia valores superiores sin sobrepasar el umbral máximo. Una vez en este estado, no cambia a otro hasta que sobrepasa el margen de histéresis máximo hacia valores superiores; o si sobrepasa el margen de histéresis mínimo hacia valores inferiores.

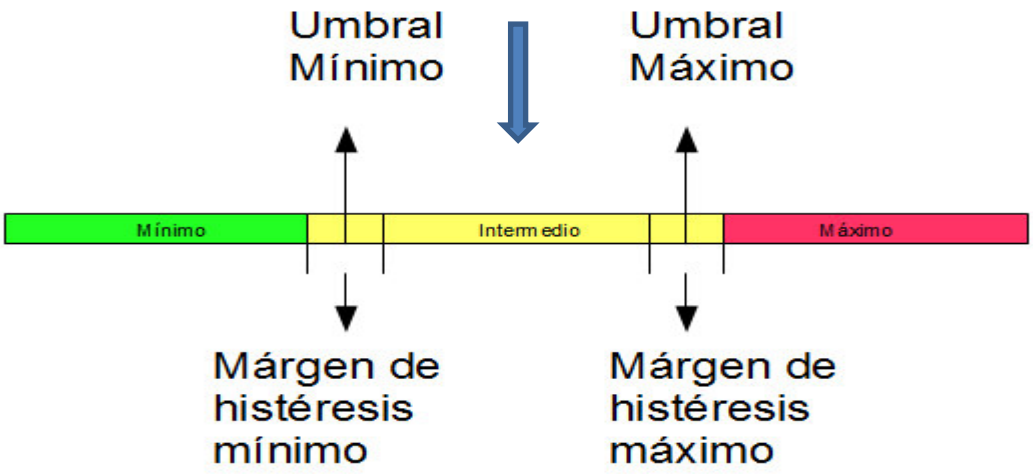


Figura 3.7: Diagrama de umbrales al alcanzar estado intermedio

Los comandos construidos poseen el formato indicado en la figura 3.2 que se mostró en el apartado anterior. Para añadir el tipo de acción que se desea ejecutar, basta con añadir el valor correspondiente de la tabla 3.2 al comando AT correspondiente:

Acción a realizar	Valor a añadir al comando	Tipo de dispositivo al que va destinado
Activación del indicador	5	Indicadores
Activación del actuador	4	Actuadores
Apagado del indicador	4	Indicadores
Activación del actuador	5	Actuadores
Tener en cuenta el dispositivo como entrada digital	3	Pulsadores
Tener en cuenta el dispositivo como entrada analógica	2	Analógicos
No tener en cuenta el dispositivo	0	Pulsadores, Analógicos

Tabla 3.2: Valores de comando a añadir según acción

Para que los dispositivos de entrada puedan ser tenidos en cuenta por la aplicación, antes de ser utilizados por primera vez es necesario que se aplique el comando correspondiente según el tipo de dispositivo.

Los módulos XBee que hemos utilizado (concretamente, los módulos XBee ZNet 2.5 OEM RF Module), permiten asignar los dispositivos de entrada y salida a una serie de pines determinados. Dado que algunos de ellos son utilizados internamente por el propio módulo para realizar operaciones reservadas, los pines configurables para poder realizar operaciones de entrada y salida son limitados.

Los pines asignables a dispositivos de entrada o salida son los que se muestran en la tabla 3.3:

Tipo de dispositivo	Lista de pines asignables
Dispositivos de Entrada: Pulsadores	4,7,11,15,17,18,19,20
Dispositivos de Entrada: Analógicos	17,18,19,20
Dispositivos de Salida: (Indicadores y Actuadores)	1,4,7,11,15,16,17,18,19,20

Tabla 3.3: Pines asignables a dispositivos de entrada/salida

Para consultar la lista completa de pines, así como toda la información relativa a los mismos, véase la tabla 2.1: Distribución de pines del XBee PRO ZNet 2.5.

3.3 Interfaz de la aplicación

Para poder manipular de forma cómoda la aplicación implementada, se han creado sendas interfaces de usuario: una versión destinada al servidor central y otra destinada a la versión que manipularán los usuarios comunes. A pesar de que es posible manipular todos los protocolos mediante una simple consola de comandos, se ha pretendido crear un programa amigable y de fácil uso para el usuario, de forma que solo tenga que preocuparse por interactuar con la aplicación mediante botones, tablas y listas, en lugar de tener que estar introduciendo de forma manual los complicados comandos que no tendría por qué conocer.

3.3.1 Casos de uso

Para poder tener una mejor visión del funcionamiento de la aplicación desarrollada, es necesario definir una por una las diferentes acciones que se pueden llevar a cabo con la misma; en primer lugar definiremos de manera general cada una de ellas, para después pasar a analizar en secciones posteriores cada uno de los elementos interactuables de forma más específica.

La aplicación está dividida en una serie de ventanas, las cuales sirven para realizar una acción específica cada una.

Partiendo siempre de un menú principal, se da la opción de, en primer lugar, inicializar la configuración de dispositivos conectados a un módulo específico de la red de sensores.

Desde la ventana de inicialización de dispositivos es posible asignar en cada uno de los pines del módulo un dispositivo determinado, diferenciándolo mediante unos parámetros concretos. Esta configuración es almacenada, si así se desea, en un fichero de configuración que perdura tras el cierre de la aplicación, por lo que no es necesario inicializar cada uno de los dispositivos cada vez que se inicia la misma.

Una vez exista una configuración mínima de dispositivos conectados a al menos un módulo de la red de sensores, desde el menú principal se permite pasar a la ventana de manipulación de dispositivos.

La ventana de manipulación de dispositivos tiene como objetivo primordial ofrecer una interfaz accesible desde la que poder manipular cada uno de los dispositivos que hayan sido inicializados previamente. Al escoger dispositivo y acción a realizar, la aplicación construye internamente un comando que será el que se le pase al nodo coordinador de la red de sensores, el cual se ocupará de que se realice la acción correspondiente en el nodo oportuno.

Desde la ventana de manipulación de dispositivos, además, es posible pasar a una nueva ventana desde la que poder configurar las acciones que se pudieran realizar al manipular de forma real los diversos dispositivos de entrada.

Una vez ubicados en la ventana de configuración de dispositivos de entrada, se permitirá alterar diversos parámetros según el tipo de dispositivo que se desee gestionar, entre ellos las posibles acciones que pudieran ocurrir al manipular de distintas maneras el dispositivo pertinente. Una vez modificados los parámetros del dispositivo, podemos guardarlos en un fichero de almacenamiento, si así se deseara, el cual perdura tras el cierre de la aplicación, por lo que no es necesario configurar cada uno de los dispositivos de entrada cada vez que se inicie la aplicación.

Es importante mencionar que es la aplicación del servidor la que se encarga de comunicarle a la red de sensores los comandos que se ejecuten por parte del usuario. De esta forma, si se ordena una acción a realizarse desde la aplicación del cliente, esta orden es transferida al servidor, el cual la procesará y se la remitirá a la red de sensores en forma de comando.

Por otra parte, tanto el servidor como el cliente deben manejar la misma configuración de la red de dispositivos. Cuando esta configuración es modificada, se transmite una copia de los cambios realizados a las otras versiones de la aplicación, para así evitar inconsistencia de los datos entre unas y otras.

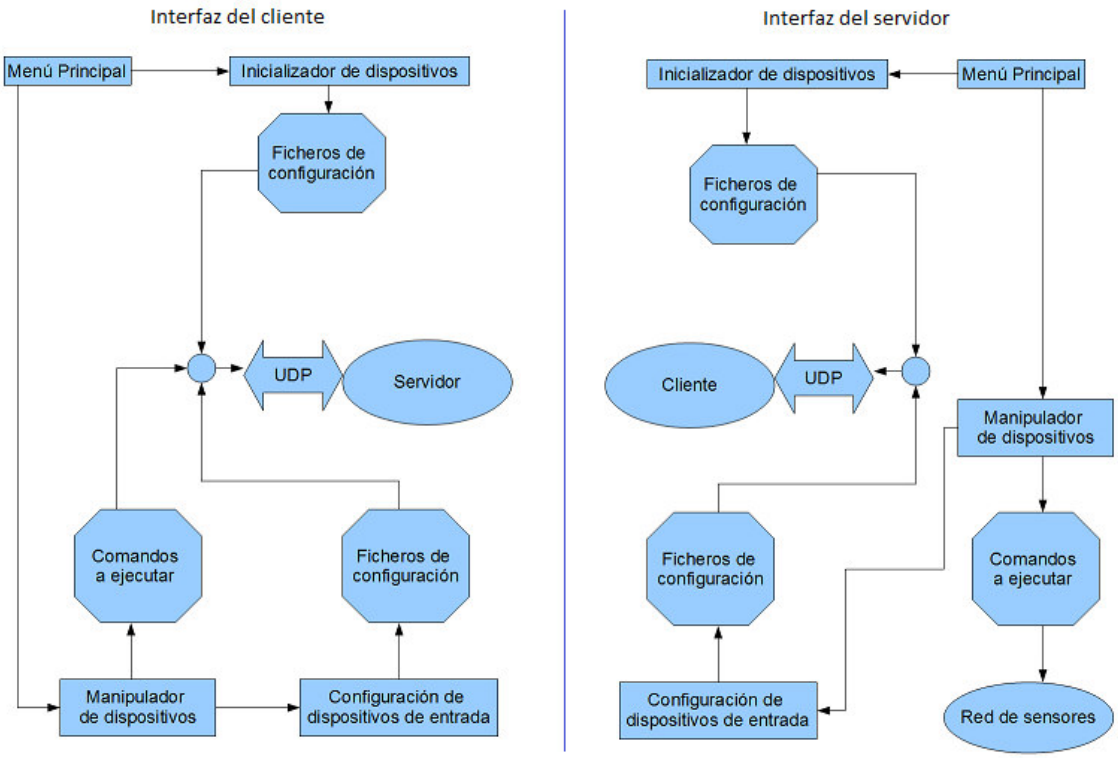


Figura 3.8: Mapa de casos de uso del cliente

3.3.2 Características de la interfaz

A continuación se mostrará el funcionamiento de los diversos componentes de cada una de las ventanas de la aplicación.

- **Menú principal**

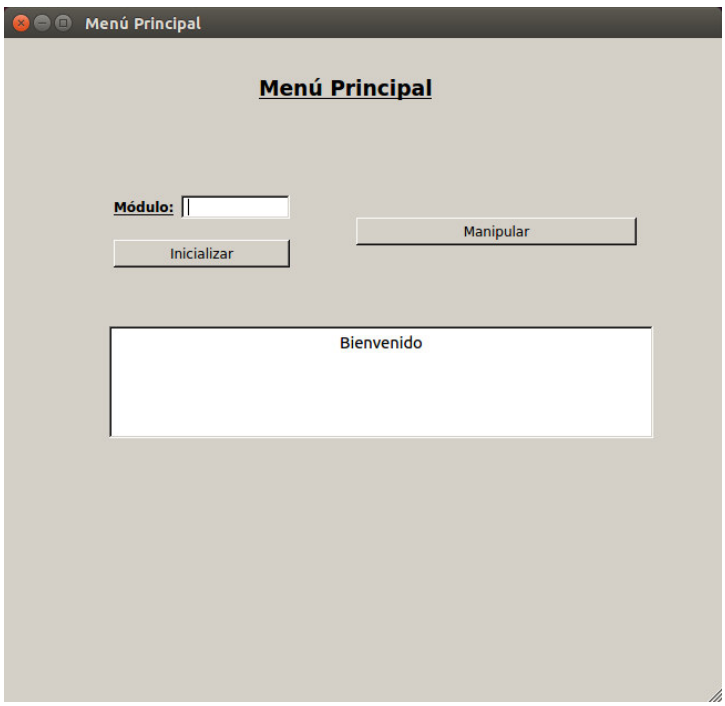


Figura 3.9: Interfaz del servidor de la ventana principal

Para empezar, al iniciar la aplicación se muestra la ventana del menú principal. Esta ventana es el portal de acceso a las ventanas de inicialización y manipulación de dispositivos, siendo imprescindible que, para poder acceder a la ventana de manipulación, exista una configuración de dispositivos de al menos uno de los módulos de la red de sensores previamente almacenada. Esta configuración se puede realizar desde la ventana de inicialización, para lo cual pide el nombre del módulo al que se conectan los dispositivos que se van a inicializar.

- **Inicializador de dispositivos**

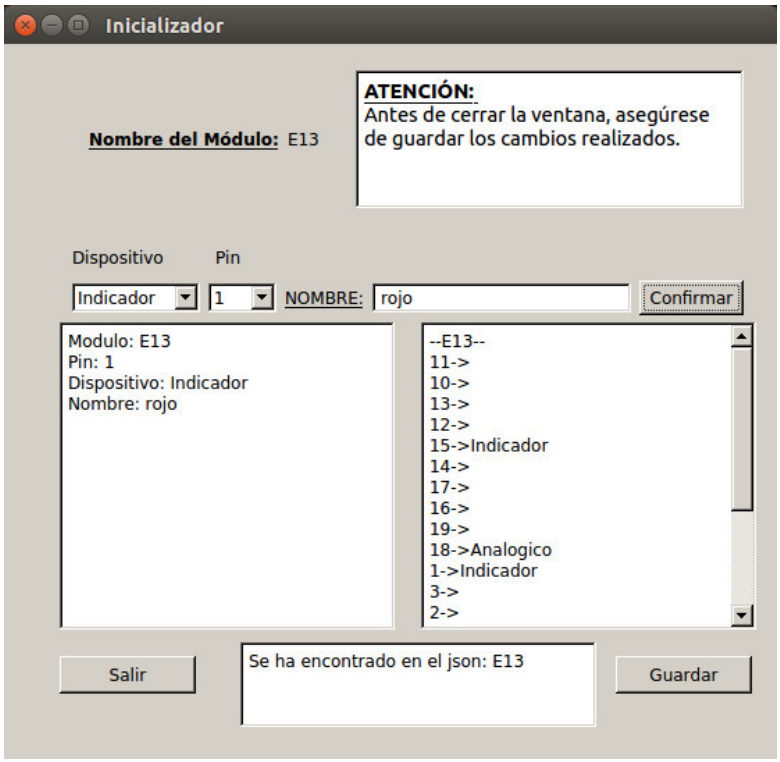


Figura 3.10: Interfaz de la ventana de inicialización

Como ya mencionamos previamente, antes de pasar a manipular los dispositivos conectados a la red de sensores es necesario inicializarlos. Para ello podemos utilizar la ventana de inicialización de dispositivos. Una vez proporcionado el nombre del módulo al que se conectan los dispositivos a inicializar, se nos muestra en pantalla la configuración actual del mismo, si la hubiera. Si se desea alterar o almacenar un dispositivo conectado al módulo indicado al principio, bastará con seguir los siguientes pasos:

1. Escoger el tipo de dispositivo que se va a almacenar de entre varias opciones reflejadas en una lista, entre las que se incluye “Indicador”, “Actuador”, “Pulsador” y “Analógico”. También existe la posibilidad de resetear el pin escogido escogiendo el espacio en blanco incluido en la lista de tipos.

2. Escoger el pin en el que está conectado el dispositivo a inicializar, pudiendo escoger de entre una serie de pines disponibles. En esta lista no se incluyen los pines reservados del módulo, así como otros que son necesarios para realizar operaciones internas por parte del módulo XBee. Para más información, consultar la tabla 2.1: Distribución de pines del XBee PRO ZNet 2.5.
3. Introducir un nombre con el que el usuario pueda distinguir al dispositivo a inicializar. No es posible nombrar a dos dispositivos con el mismo nombre.

Una vez completados todos los pasos anteriores, confirmamos los parámetros escogidos en el botón "Confirmar". Esta operación se puede realizar tantas veces como dispositivos se deseen inicializar. Si no ha habido ningún problema, aparecerá una imagen similar a la de la figura 3.10.

Hay que decir que es imprescindible guardar los cambios realizados accionando el botón "Guardar", de otra manera, todos los cambios se perderán al salir de la ventana de inicialización. De esta forma, se volcarán toda la configuración pertinente en ficheros Json que serán inmediatamente accesibles por la aplicación.

- **Manipulador de dispositivos**

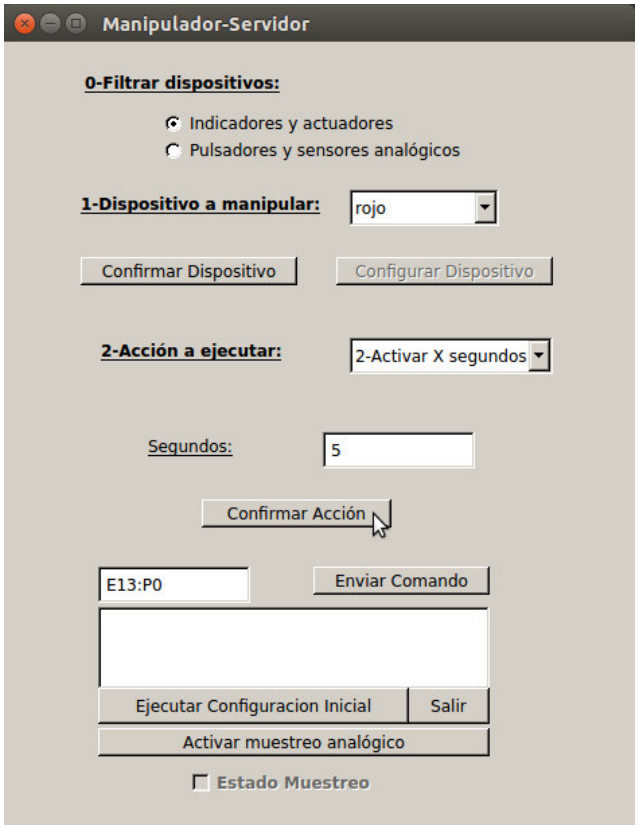


Figura 3.11: Interfaz de la ventana de manipulación

Una vez la aplicación disponga de archivos de configuración accesibles, se nos da la posibilidad de manipular todos aquellos dispositivos que hayan sido inicializados correctamente.

Desde la ventana de manipulación de dispositivos es posible realizar varias operaciones. La más importante, sin duda, es poder enviar órdenes de manera remota a los diferentes dispositivos disponibles. Para ello, es necesario seguir los siguientes pasos:

1. Para empezar, escogeremos el dispositivo que deseamos manipular. Se da la opción de filtrar los dispositivos según queramos ver en la lista dispositivos de entrada (pulsadores y analógicos) o de salida (indicadores y actuadores). Una vez escogido el dispositivo a manipular, será necesario accionar el botón "Confirmar Dispositivo" para así confirmar que es este el dispositivo que vamos a manipular.
2. A continuación se nos muestra una lista de posibles acciones a realizar por parte del dispositivo seleccionado. Una vez escogida la operación que deseamos realizar (encender, apagar, etc), será necesario accionar el botón "Confirmar Acción". Si la operación seleccionada requiere de un tiempo de operación determinado, se nos pedirá que introduzcamos en segundos el tiempo que se le pasará a la operación pertinente. Si es el caso, al pulsar "Confirmar Acción" se ejecutará la acción correspondiente.
3. Si la operación no requiere de intervalos de tiempo, aparecerá el comando preparado para ser enviado junto al botón "Enviar Comando". Al pulsar dicho botón, se enviará el comando y se producirá la acción correspondiente.

A pesar de que el último paso no debería ser necesario, se ha optado por mantenerlo en la interfaz de cara a que un usuario con conocimientos avanzados pueda escribir y enviar sus propios comandos si así lo desea.

Para poder utilizar físicamente los dispositivos de entrada, es necesario accionar el botón "Ejecutar Configuración Inicial". Esto sólo será necesario ejecutarlo una vez antes de manipular dichos dispositivos, por lo que es recomendable accionarlo en primer lugar al comenzarlo a utilizar. Al accionarse este botón se envían una serie de comandos que configuran adecuadamente cada uno de los dispositivos disponibles, proporcionando la información necesaria a la red de sensores de forma que cada uno sepa cómo se debe comportar cada uno de sus pines. Para más información, consultar la Tabla 3.2: Valores de comando a añadir según acción.

Dado que los dispositivos analógicos pueden provocar acciones de forma autónoma cuando detecten que se ha pasado de un estado determinado a otro, hemos considerado oportuno añadir un botón llamado "Activar muestreo analógico", el cual activa o desactiva según convenga al usuario la posibilidad de que estos dispositivos puedan realizar operaciones o no. Al activarse este muestreo, se crea un nuevo hilo de ejecución que trabaja de manera concurrente al programa

principal, el cual comprueba cada dos segundos en cada uno de los dispositivos analógicos si se ha producido algún cambio de estado en alguno de ellos. Si así es el caso, se encarga de construir el comando que ejecute la acción que tenga asignada el cambio de estado pertinente. Para más información, consultar la Figura 3.4: Diagrama de umbrales de dispositivos analógicos.

En la parte inferior de la ventana podemos observar un indicador que estará activado siempre que el muestreo analógico se esté realizando. Si al cerrar la ventana este indicador está activado, se ofrece la opción al usuario de desactivar el muestreo antes de salir (La desactivación del muestreo es obligatoria al cerrar la ventana de la aplicación del servidor).

Para que la aplicación sepa qué operaciones se deben realizar al detectar las señales producidas por los diversos dispositivos de entrada que haya conectados a la red de sensores, es necesario crear un nuevo fichero de configuración que sea accesible a la aplicación y que le proporcione estos datos. Para ello, será necesario escoger uno de los dispositivos de entrada disponibles y pasar a configurarlo mediante el botón “Configurar Dispositivo”. Este botón sólo es accionable si hemos escogido un “pulsador” o un “dispositivo analógico”. Al accionar este botón aparece la ventana de configuración de dispositivos de entrada, en la cual profundizaremos a continuación.

- **Configurador de dispositivos de entrada**

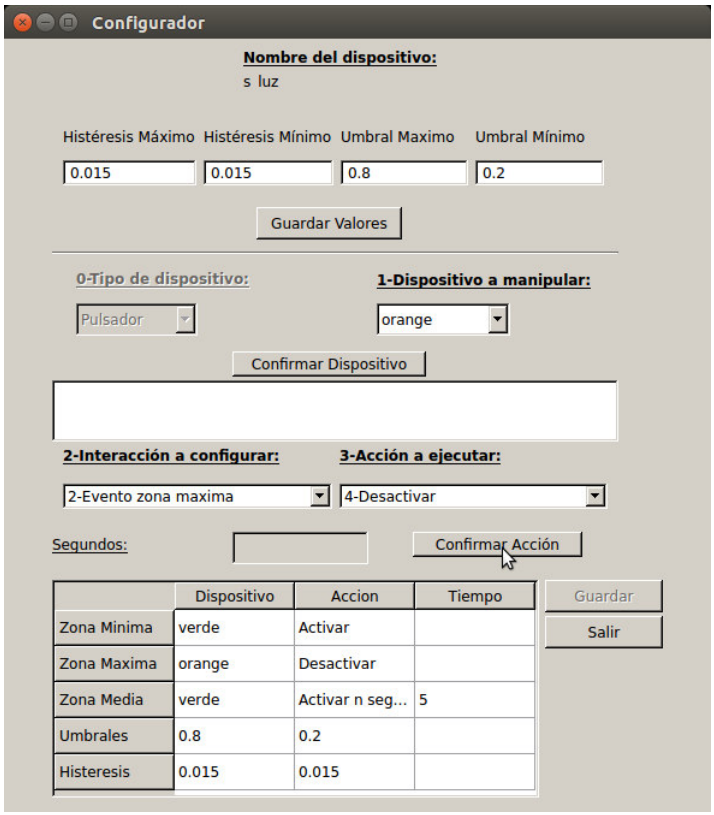


Figura 3.12: Interfaz de la ventana de configuración de dispositivos de entrada: Analógicos

La ventana de configuración de dispositivos de entrada es ligeramente distinta según el tipo de dispositivo que se vaya a configurar. Por una parte, si se trata de un dispositivo analógico, se nos da la posibilidad de configurar sus umbrales máximo y mínimo, así como los rangos de histéresis que posea en ambos umbrales. Es necesario completar cada uno de los campos antes de poder confirmar la configuración mediante el botón "Guardar Valores". Una vez confirmados, pasan a ser reflejados en una tabla en la parte inferior de la aplicación, como se puede ver en la figura 3.12.

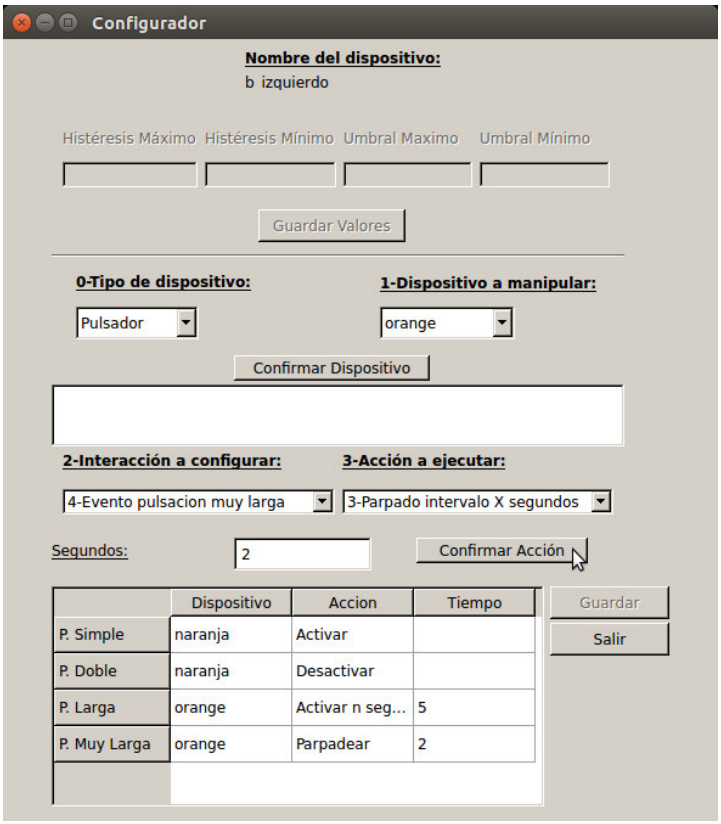


Figura 3.13: Interfaz de la ventana de configuración de dispositivos de entrada: Pulsadores

Por otra parte, si el dispositivo que se va a configurar es un pulsador, los campos de configuración de umbrales e histéresis quedan desactivados, como se puede ver en la figura 3.13.

Tanto los dispositivos analógicos como los pulsadores tienen la capacidad de producir acciones cuando son manipulados. Para saber sobre qué dispositivo se actuará al ocurrir un determinado evento en el dispositivo de entrada, habrá que crear una configuración siguiendo los siguientes pasos:

1. Para empezar, escogeremos el dispositivo sobre el que se actuará al producirse una interacción con el dispositivo que estamos configurando, de entre una lista en la que se incluyen todos los "Indicadores" y "Actuadores"

disponibles. Para ello, una vez escogido el dispositivo, accionamos el botón "Confirmar Dispositivo".

2. Aparecerán las posibles interacciones físicas disponibles que se podrán realizar por parte del usuario, las cuales difieren según el tipo específico del dispositivo de entrada (tipo de pulsación, eventos, activación del sensor…). Tras escoger una de las interacciones que ofrece la lista correspondiente, se podrá escoger el tipo de acción que se realizará por parte del dispositivo a manipular, acciones que también diferirán según el tipo de dispositivo a manipular. Si fuera necesario, se pedirá que se indique un tiempo en segundos determinado para la ejecución de la acción. Una vez escogidos ambos parámetros, será necesario accionar el botón "Confirmar Acción" para que los cambios realizados se reflejen en la tabla de la interfaz.
3. Una vez realizados todos los cambios deseados en la configuración del dispositivo de entrada, se deberá accionar el botón "Guardar" para almacenar todos los cambios en un fichero accesible a la aplicación. Si no se realizara este último paso y se saliera directamente de la misma, se avisará al usuario de que se perderán los cambios realizados, dándole la oportunidad de que los almacene si así lo deseara.

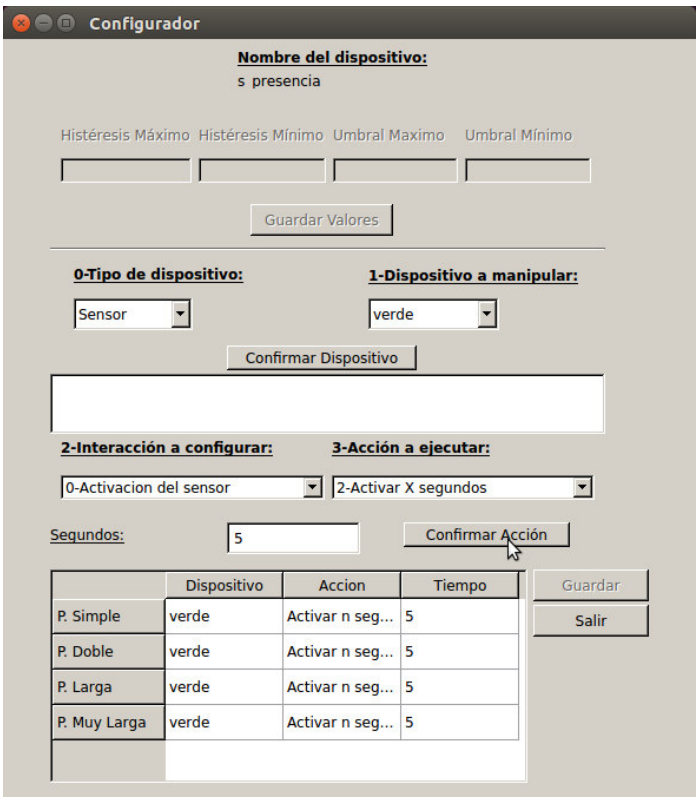


Figura 3.14: Interfaz de la ventana de configuración de dispositivos de entrada: Sensores

Hay una diferencia a destacar cuando el dispositivo de entrada a configurar sea un sensor. Aunque los sensores están englobados en este proyecto dentro de la categoría de "Pulsadores" debido a la señal digital que transmiten al activarse, no es

posible interactuar con un sensor de forma idéntica a cómo se podría interactuar con un pulsador. Es por esto que se da la posibilidad, al principio del proceso de asignación de dispositivos, de escoger el tipo de dispositivo que se va a configurar, ya sea un pulsador o un sensor (Esta posibilidad está inhabilitada cuando el dispositivo de entrada a configurar es un dispositivo analógico).

El único cambio que se producirá en la interfaz será la imposibilidad de poder escoger el tipo de pulsación que se realizará, proporcionando la única opción de "Activación del sensor".

Una vez se tenga una configuración de dispositivos de entrada adecuada, será posible realizar distintas interacciones con los dispositivos configurados de forma que se produzcan las acciones que se hayan indicado según el tipo de interacción realizada.

De esta forma, será posible, por ejemplo, activar un ventilador conectado a un relé al realizar una pulsación simple de un pulsador determinado, o apagarlo realizando una pulsación doble en el mismo; apagar una bombilla cuando un sensor de luminosidad detecte que existe un nivel elevado de luz; o activar un indicador durante un tiempo determinado cuando un sensor de presencia detecte algún movimiento cercano.

- **Menú principal del cliente**

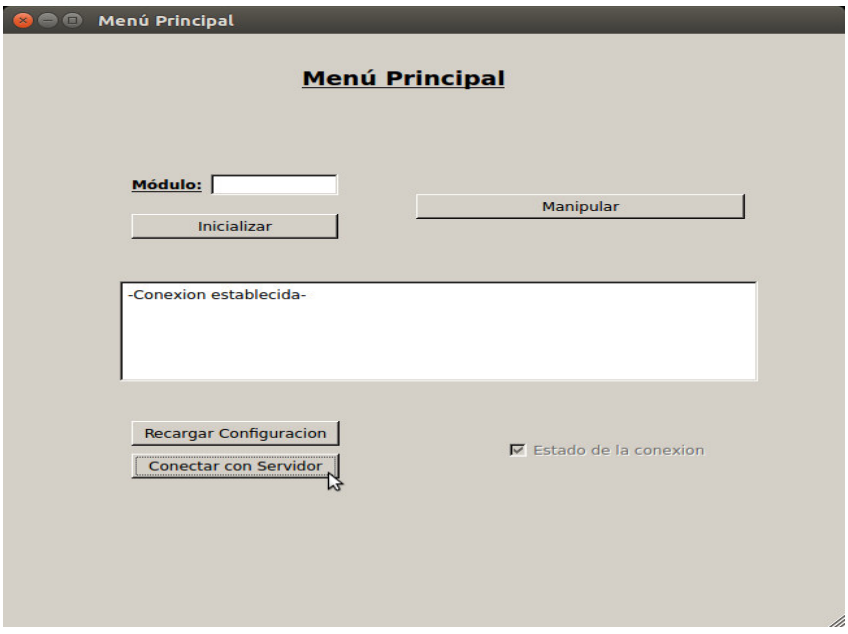


Figura 3.15: Interfaz del cliente de la ventana principal

Existen unas leves diferencias en la ventana del menú principal del cliente con respecto a la misma del servidor. Además de los accesos a las ventanas de inicialización y manipulación de dispositivos, existen dos botones más que sirven para enlazar correctamente la aplicación cliente con la del servidor. La aplicación no permite realizar ninguna operación si no se establece una comunicación funcional

con el servidor. El primer paso a realizar será comprobar que existe conexión con el servidor y, una vez hecho esto, será posible cargar en la aplicación del cliente la configuración de dispositivos alojada en el servidor. Tras realizar esta operación de comprobación de conexión, se habilitan todas las demás operaciones.

En la figura 3.16 podemos ver un esquema simplificado del protocolo apoyado en UDP que ha sido diseñado para comunicar las aplicaciones cliente-servidor:

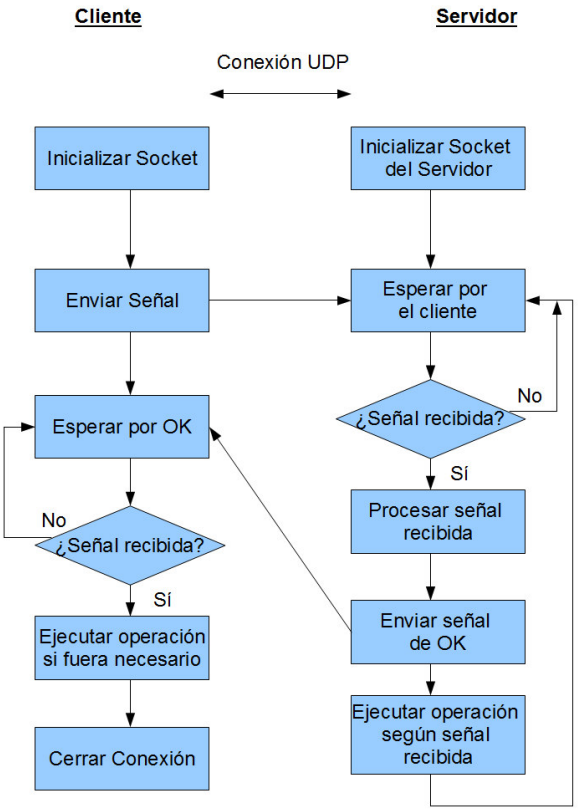


Figura 3.16: Esquema del protocolo de comunicación cliente-servidor apoyado en UDP

Como se puede observar en la figura, una vez inicializado el socket, el servidor estará continuamente esperando un mensaje a recibir desde el cliente. Cuando recibe una, envía un mensaje de confirmación al cliente, y la procesa de forma que se realice posteriormente la operación asociada a la misma, pasando nuevamente a esperar por otro posible nuevo mensaje.

El cliente, en cambio, crea un nuevo socket que se comunica con el del servidor cada vez que necesita enviarle un mensaje. Una vez enviado, espera por un mensaje de confirmación y, tras ejecutar una operación si fuera necesario, se cierra el socket del cliente.

Capítulo 4.
Resultados

Una vez creada una versión entregable de la aplicación, podemos analizar los resultados observados durante el uso de la misma.

Para empezar, hemos comprobado que, mediante la conexión UDP cliente-servidor, es posible tener una versión cliente de la aplicación que sea portátil, esto es, localizada en cualquier lugar que pueda conectarse con la versión servidor de la aplicación, la cual sí estará localizada siempre en el mismo lugar, junto al módulo coordinador de la red de sensores.

También hemos comprobado que, aparte de en un PC, es posible ejecutar la aplicación en una RaspberryPI sin ninguna incidencia reseñable, más allá de la obligación de tener que instalar los entornos gráficos y herramientas pertinentes, como son Python y PyQt.

Mediante el uso de diversos ficheros de configuración que se crean al manipular la aplicación, se logra una configuración completamente flexible y adaptable a las necesidades del usuario final.

Se ha comprobado que, siempre que la aplicación del servidor esté operativa, el nodo coordinador podrá enviar comandos y realizar operaciones pertinentes en todo momento.

Utilizando una red de un sensor coordinador, y dos sensores haciendo las veces de nodos router, los cuales se han conectado a diversos dispositivos de testeo, se ha podido comprobar que es posible mandar órdenes entre nodos, esto es, manipulando un dispositivo conectado en el nodo A se logra obtener una respuesta en otro dispositivo conectado al nodo B, siempre que se realice previamente la configuración adecuada.

Es necesario indicar que actualmente, la aplicación implementada está diseñada para funcionar con un único cliente conectado al servidor a la vez. A pesar de que es posible que la aplicación pudiera funcionar con varios clientes conectados a la vez, es posible que ocurran errores de inconsistencia entre las configuraciones de cada uno de los clientes, siendo esta una vía de mejora en el futuro.

Por último, mediante múltiples pruebas de usabilidad realizadas a posibles usuarios, se ha mejorado varias veces la visibilidad y comodidad de la interfaz de uso, añadiendo elementos más visuales y más fácilmente entendibles de cara al

usuario, como filtrado de dispositivos según tipos, o la visualización de elementos mediante tablas.

Para la realización del proyecto se utilizaron:

- 1 nodo coordinador (Figura 2.1: Nodo coordinador de la red)
- 2 nodos router (Figura 3.3: Nodos router de la red, conectados a diversos dispositivos interactuables)
- 5 diodos LED a modo de indicadores.
- 1 relé, conectado a un ventilador a modo de actuador.
- 4 pulsadores y un sensor de presencia, a modo de dispositivos de entrada digitales.
- 1 potenciómetro y un sensor de luminosidad, a modo de dispositivos de entrada analógicos.

En el siguiente enlace se encuentra el repositorio de GitHub donde se encuentra alojado el código de la aplicación final desarrollada durante el transcurso del proyecto, comentada de forma que se pueda entender fácilmente toda línea de código.

Enlace al código del proyecto:

https://github.com/ULL-InformaticaIndustrial-Empotrados/TFG_XBee_Adrian/tree/Interfaz/Python

Capítulo 5.
Conclusiones y líneas futuras

5.1 Conclusiones

Una vez llegados a la parte final de este proyecto, podemos pasar a recapitular los objetivos cumplimentados satisfactoriamente y así poder sacar las conclusiones pertinentes.

Partiendo de una red de sensores inalámbricos basada en el protocolo ZigBee y un estudio previo exhaustivo del mismo, se ha diseñado un protocolo que permite la configuración flexible del nodo coordinador de la red, y así poder comunicarse con ella de forma efectiva.

Se ha implementado una aplicación que permita configurar y manipular diversos dispositivos conectados a la red de sensores, de forma que un usuario final interactúe con ella de forma fácil y accesible, haciendo uso de una interfaz de control que ha sido mejorada en múltiples ocasiones a lo largo del transcurso del presente proyecto.

Se han implementado diversos protocolos de control de varios tipos concretos de dispositivos, que hacen que la aplicación pueda interpretar eventos ocurridos en los mismos y así actuar en consecuencia, de forma que el usuario pueda escoger en todo momento la configuración de cada dispositivo manipulable, así como los eventos que pudiera ocurrirles y las acciones que se producirán a consecuencia de los mismos.

Para lograr una disponibilidad continua de la red, y, al mismo tiempo permitir que un usuario pueda interactuar con ella de forma remota, se ha optado por desarrollar una aplicación alojada en un servidor conectado directamente a la red de sensores, que estaría diseñada para estar en continuo funcionamiento; y, además, una aplicación destinada a ser manipulada de forma puntual por los usuarios desde un puesto informático que se comunica de forma remota con el servidor, logrando así los objetivos propuestos.

Para concluir, aunque el trabajo realizado aún tiene margen para ser desarrollado aún más en diversos aspectos, actualmente presenta cualidades que lo sitúan como una línea de investigación interesante a seguir y así poder profundizar más en campos como el Internet de las Cosas, logrando que los diversos dispositivos que componen la red de sensores llegaran a actuar de manera autónoma, así como en el

campo de la domótica, el cual podría encajar perfectamente con las necesidades que podría satisfacer la aplicación desarrollada.

5.2 Líneas de trabajo futuro

Una posible línea de trabajo futuro pudiera ser implementar mejoras en la aplicación que permitan la conexión multi-cliente con el servidor, de forma que se pudieran evitar diversos problemas que surgirían a causa de esta mejora, como podría ser la posible inconsistencia entre configuraciones o la necesidad de implementar un sistema de bloqueo de acceso simultaneo a la configuración entre múltiples clientes.

Otra posible línea de trabajo sería la creación de un sistema de almacenamiento de los mensajes de información, depuración o diversas alertas que pudieran surgir durante el uso de la aplicación, para así tener un medio de documentación que poder consultar en modificaciones futuras. Actualmente estos mensajes se van mostrando según van surgiendo en una consola de comandos, sería interesante que estos mensajes pasaran a un fichero de logging mediante la correspondiente librería que proporciona Python.

También existe la posibilidad de que, por alguna razón ajena a la aplicación, se pierda la comunicación UDP entre servidor y cliente. Implementar un método de contención para estos casos sería una mejora que evitaría futuros problemas.

Además, actualmente la dirección IP del host con el que se comunica la versión del cliente de la aplicación permanece fija y no es configurable, si se pretende cambiar la dirección propia del servidor con el que se comunica, es necesario asignarla en el código de la clase de forma manual.

Por último, una línea de trabajo interesante a desarrollar más en profundidad de cara al futuro, sería una mayor integración de la aplicación en el campo del Internet de las Cosas, de forma que los aparatos sean capaces de comunicarse entre sí y puedan compartir datos, proporcionándole algún tipo de inteligencia que permita a la aplicación deducir qué es lo que puede desear el usuario en cada momento determinado, sin necesidad de que el usuario tenga que intervenir de forma directa en el acceso o control de los dispositivos. Una utilidad práctica de este sistema encajaría perfectamente en el campo de la domótica, así como en la asistencia a personas con alguna discapacidad o de avanzada edad.

Capítulo 6.
Summary and Conclusions

6.1 Conclusions

Once we have reached the final part of the project, we can summarize the accomplished tasks that were done to get easier the pertinent conclusions.

Having a wireless network of sensors based in the ZigBee protocol and a previous exhaustive research of the same, we could be able to design a protocol that let us to create a flexible configuration in the network's coordinator node, so it was able to communicate with it in an effective way.

It was implemented an application that lets the user to configure and handle many devices that would be connected to the network of sensors, so a final user would be able to interact with it in an easy and handy way, using a control interface that have been improved in so many times during the development of this project.

There were implemented many control protocols for many specific types of devices, so the application would be able to interpret many events that would happen in them so it would be able to act according to them, letting the user to pick in any moment each available device's configuration, as well as the events that could happen to them and the actions that would be produced according to them.

To make a continuous availability of the network, and at the same time letting the user to interact with it remotely, we have decided to design an application that will be located in a server directly connected to the network of sensors, which will be designed to be in continuous operation; in addition to an application whose purpose is to be manipulated in a few times by the users from a computer station that communicates remotely with the server, achieving this way the proposed objectives.

Finally, although the project still can be improved in many points, actually it presents qualities that situates it as an interesting line of investigation to follow, making easier to deepen more in fields as the Internet of Things, making the many devices that integrate the network of sensors to act independently, as well as in the field of the home automation, which could perfectly deal with the requirements that would be able to accomplish the developed application.

Capítulo 7. Presupuesto

El proyecto se ha realizado utilizando exclusivamente herramientas de software libre. Los únicos costos cuantificables vienen derivados de los diversos módulos utilizados para crear la red de sensores, los diversos dispositivos de manipulación de prueba, y por último, de la mano de obra.

Tipo	Cantidad	Coste por unidad	Coste total
Software	1	0.00€	0.00€
Módulos XBee	3	30.00€	90.00€
Pulsadores	4	0.15€	0.60€
Diodos	5	0.05€	0.25€
Relé	1	2.00€	2.00€
Potenciómetro	1	0.55€	0.55€
Sensor de luminosidad	1	0.75€	0.75€
Sensor de presencia	1	2.00€	2.00€
Mano de obra	1	4320.00€	4320.00€
TOTAL			4416.15€

Tabla 7.1. Tabla resumen de presupuesto.

7.1 Justificación del presupuesto

El proyecto se ha realizado utilizando diversos materiales con un coste propio determinado:

- Dado que se ha utilizado software libre en todo momento para desarrollar la aplicación, este apartado no conlleva un gasto real reseñable.
- El precio mínimo de los diferentes módulos XBee que hemos utilizado rondan los 30 € de media. En nuestra red hemos utilizado 3 de estos módulos, pero se podrían utilizar muchos más si se desea ampliar la red de sensores.
- Los diversos dispositivos utilizados para realizar pruebas de testeo de la aplicación también conllevan un pequeño gasto asociado. Este rango de costos puede variar bastante dependiendo de los dispositivos que se conecten a la red de sensores, pero hemos elegido estos dispositivos para abarcar todos los tipos mediante el uso de al menos uno de ellos.
- Por otra parte, el gasto derivado de la mano de obra se ha calculado en base a lo que cobraría un ingeniero informático por el desarrollo de un proyecto

utilizando su propio PC para trabajar. Suponiendo que se le pagar 18.00 € por hora, siendo este sueldo ligeramente inferior a lo que debería cobrar un ingeniero informático, y que ha trabajado 60 días en el proyecto durante 4 horas cada día, el costo asociado a la mano de obra es de aproximadamente 4320.00 €.

- Por último, el sumatorio del costo del material utilizado junto con el de la mano de obra da lugar a un total de 4416.15 € presupuestados, como se puede observar en la tabla anterior.

Bibliografía

[1] Historia sobre Internet de las Cosas

Páginas web:

http://www.quees.info/que-es-internet-de-las-cosas.html

[2] Domótica

Páginas web:

https://es.wikipedia.org/wiki/Dom%C3%B3tica

[3] Alternativas al protocolo ZigBee

Páginas web:

http://electronicdesign.com/communications/comparing-four-short-range-wireless-options-monitoring-and-control

http://www.domoprac.com/protocolos-de-comunicacion-y-sistemas-domoticos/protocolos-de-red-tipos-y-utilidades.html

https://sx-de-tx.wikispaces.com/ZIGBEE

[4] Página web oficial de la alianza ZigBee

Páginas web:

www.zigbee.org.

[5] Documentación relativa al funcionamiento de redes XBee

Páginas web:

http://www.andresduarte.com/arduino-y-xbee

[6] Guía rápida XBee S2

Páginas web:

http://www.tunnelsup.com/images/XBee-Quick-Reference-Guide.pdf

[7] Manual oficial del modelo de XBee utilizado

Páginas web:

https://www.sparkfun.com/datasheets/Wireless/Zigbee/XBee-2.5-Manual.pdf

[8] Documentación relativa a Python y PyQt

Páginas web:

https://es.wikipedia.org/wiki/Python

https://en.wikipedia.org/wiki/PyQt

[9] Documentación relativa a UDP

Páginas web:

https://es.wikipedia.org/wiki/User_Datagram_Protocol

[10] Documentación relativa a Json

Páginas web:

https://es.wikipedia.org/wiki/JSON

[11] Manuales de aprendizaje de Python 2.7

Páginas web:

https://docs.python.org/2/

[12] Manuales de aprendizaje de PyQt4

Páginas web:

http://pyqt.sourceforge.net/Docs/PyQt4/classes.html