Trabajo de Fin de Grado

Grado en Ingeniería Informática

Desarrollo de un sistema controlador para red domótica inalámbrica basada en protocolo ZigBee

Development of a controller system for wireless domotic network based on ZigBee protocol

Juan Ignacio Hita Manso

La Laguna, 24 de Junio de 2017

D. **Alberto Hamilton Castro**, con N.I.F. 43773884P 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:

“Desarrollo de un sistema controlador para red domótica inalámbrica basada en protocolo ZigBee”

ha sido realizada bajo su dirección por D. **Juan Ignacio Hita Manso**, con N.I.F. 45941949R

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 2 de Julio de 2017.

Agradecimientos

Agradezco a mi tutor Alberto Hamilton Castro por su ayuda, atención y consejos durante la realización de este proyecto.

A mi familia, amigos y profesores por todo el apoyo incondicional prestado durante todo este tiempo de estudio.

Licencia

Resumen

El objetivo de este trabajo ha sido desarrollar un sistema controlador para una red domótica basándose en el protocolo ZigBee y los módulos XBee.

Desde un servidor ubicado en un computador o sistema reducido como, por ejemplo, una Raspberry Pi se comunica la red de nodos por medio de una aplicación web. Podremos configurar los módulos de la red XBee, ver el estado de los diferentes actuadores y sensores, crear alertas, eventos y así actuar en los dispositivos disponibles. Por ejemplo: controlar el sistema de iluminación de una habitación, control de persianas, control de temperatura …

Entre las muchas características de este proyecto, podemos destacar las siguientes:

- *Uso de tecnologías de software actuales, buscando la eficiencia y comunicación integra del sistema.*
- *Destacar un diseño modular en el sistema, que permite reutilizar el código y facilitar el mantenimiento posterior del sistema.*
- *Está pensado para ser distribuido en un Sistema Embebido, del tipo Raspberry Pi o BeagleBoard.*
- *Pensado para ofrecer flexibilidad en la configuración de la red XBee y adaptarse a los usuarios.*

Palabras clave: ZigBee, Domótica, Internet de las cosas, Sensores, Actuadores, Python, JavaScript

Abstract

The objective of this project was to developed a controller system for a domotic network based on ZigBee protocol and the XBee modules.

From a server located on a computer or reduced computer such as a Raspberry pi communicates a network of nodes through a web application. We can configure the modules of the XBee network, view the status of the differents actuators and sensors, create alerts, events and act on the available devices. For example: control the lighting system of a room, blind control, temperatures and humidity control…

Among the many features of this project, we highlight the following:

- *Use of current software technologies, seeking the efficiency and integrated communication of the system.*
- *Modular design in the system, which allows reuse of the code and eases the later maintenance of the system.*
- *It is intended to be distributed in and embedded system, of the type Raspberry Pi or BeagleBoard.*
- *Designed to offer flexibility in configuring the XBee network and adapting to the users.*

Keywords: *ZigBee, domotic, Internet of the things, sensors, actuators, python, JavaScript*

Índice General

Índice de Figuras e Imágenes

Índice de Tablas

Capítulo 1. Introducción

1.1 Antecedentes

El avance exponencial que están viviendo las tecnologías en la era actual provee de forma incesante invenciones allí donde nazca una nueva necesidad, así como innovaciones en los sistemas domóticos existentes. Aunque ésta no se encuentre implantada por completo, es un hecho que en el futuro más próximo será una parte importante de cualquier vivienda o edificio.

Pero, ¿cómo podemos definir la domótica?, se podría definir como los diferentes sistemas integrados e interrelacionados que se instalan en una vivienda o edificio para permitir su control y automatización, además de tener objetivos específicos como el control energético, seguridad, confort y una interfaz de uso entre vivienda y usuario fácil de usar y segura.

La siguiente evolución en esta trayectoria es el concepto “IoT” (Internet of the Things), es decir, el internet de las cosas. Con el tiempo cualquier objeto cotidiano será convertido a un dispositivo electrónico, independiente y capaz de comunicarse de manera colaborativa con un servidor central u otros dispositivos de la red.
Para poder conseguir las propiedades comentadas anteriormente, será necesario que estos sistemas recojan la información de su entorno con sensores y que dispongan de la lógica necesaria para actuar en consecuencia utilizando diferentes actuadores.

Las implementaciones de los sistemas domóticos actuales tienen un costo muy alto, instalaciones complejas y en muchas ocasiones requieren realizar cambios estructurales en las viviendas. Para solventar esta problemática en este proyecto se desarrollará un sistema de control domótico basado en el protocolo de comunicación inalámbrico ZigBee y los módulos XBee con una interfaz de usuario web multiplataforma. Esta alternativa hará posible la integración de la domótica a múltiples dispositivos de forma más asequible, rápida y con ciertas ventajas que se comentarán en el desarrollo de este proyecto.

1.2 Objetivos del proyecto

El objetivo principal del proyecto es desarrollar un sistema domótico capaz de interconectar todos los dispositivos inalámbricos disponibles en los diferentes módulos de la red XBee.

A continuación, se listan diferentes objetivos específicos que se ha pretendido completar con el desarrollo del proyecto:

- Estudio y análisis del protocolo ZigBee, los diversos sensores y dispositivos que conforman la red, y la consola interactiva que hace posible la comunicación con la red.
- Implementación de protocolos socket para comunicar la API.
- Diseño y creación de la plataforma web que será la interfaz de control destinada al usuario final.
- El sistema web debe ser capaz de comunicarse y configurar otros dispositivos XBee, sensores y actuadores de funcionalidad genérica.
- El sistema debe quedar unificado en una única maquina residente, pudiendo ser un minicomputador al estilo Raspberry Pi o un servidor corriente.
- El módulo XBee que tiene la función de coordinador, debe quedar conectado electrónicamente al servidor en estado demonio.

1.3 Metodología

Para el desarrollo de este proyecto se ha intentado seguir una metodología de trabajo cercana a las metodologías ágiles de software [16], donde las dificultades evolucionan con el tiempo según la necesidad del proyecto. En primer lugar, comencé con un análisis de la tecnología Zigbee, de los objetivos del proyecto y un estudio de los códigos proporcionados por el tutor para la posterior planificación de tareas.

Los procesos ágiles de los que consta la metodología utilizan retroalimentación en lugar de planificación, como principal mecanismo de control. La retroalimentación se canaliza por medio de pruebas periódicas y frecuentes versiones del software, usando la plataforma de control de versiones “GitHub” [13] y el repositorio ofrecido por el tutor para almacenar cada nueva versión del proyecto.

Dentro de las metodologías agiles, específicamente he optado por seguir un desarrollo basado en funcionalidades [14] [15], que consta de múltiples fases en la que cada iteración del ciclo de vida incluye: planificación, análisis de requisitos, diseño, codificación, pruebas y documentación. El concepto de "Finalizado", tiene gran importancia ya que el objetivo de cada iteración no es agregar toda la funcionalidad para justificar su lanzamiento, sino incrementar el valor de cada funcionalidad.

Las múltiples fases llevadas a cabo en el transcurso del proyecto se dividen en:

1. Documentación del proyecto, análisis, pruebas y estudio del código proporcionado por el tutor.
2. Implementación del módulo Websocket y creación de una consola web simple para la manipulación de los nodos.
3. Planificación, análisis y contrastes de las múltiples maneras de afrontar el sistema domótico final, buscando posibles alternativas y herramientas para facilitar su construcción.
4. Diseño y desarrollo final de la aplicación web multiplataforma que servirá de interfaz gráfica para la manipulación del sistema domótico.
5. Construcción de la base de datos que almacenara todos los datos de los usuarios del sistema.
6. Ampliación de las funcionalidades de la aplicación y pruebas con casos de uso reales.

Capítulo 2.
Conocimientos previos

En este segundo capítulo introduciremos los distintos conceptos previos del proyecto: ¿Qué es el internet de las cosas y de qué trata la tecnología ZigBee?. Además, comentaremos su funcionamiento, características, topología y los diferentes modos de trabajo que permiten a los módulos XBee funcionar, centrándonos en su implementación y el modo API.

2.1 Internet de las cosas (IoT)

El término “Internet of Things” o Internet de las cosas (“IoT”) fue acuñado por primera vez en los años 80 en el entorno militar y posteriormente propuesto por Kevin Ashton en el Auto-ID Center del MIT en 1999, donde se investigaba en el campo de la identificación por radiofrecuencia en red (RFID) y las tecnologías de los sensores.

Podríamos definir el Internet de las cosas como un concepto de interconexión digital a través de Internet de una red con sistemas embebidos que aloja una gran multitud de objetos cotidianos o dispositivos, es decir, poder tener conectada a esta todas las cosas de este mundo como podrían ser vehículos, electrodomésticos, dispositivos mecánicos, o simplemente objetos tales como calzado, muebles, maletas, dispositivos de medición, biosensores, o cualquier objeto que nos podamos imaginar [1].

No hay un tipo específico de objetos susceptibles de conectarse al Internet de las cosas. Se trata de los chips y circuitos de los sistemas embebidos, los que cuentan con las herramientas necesarias para cumplir las funciones necesarias.
En la siguiente figura 1.1 se explica el funcionamiento del internet de las cosas, con la nube de Internet como elemento central donde se conectan el router IoT que encamina los diferentes paquetes hacia los dispositivos finales por la derecha. Y por la izquierda la plataforma de usuario y los dispositivos remotos IoT (cualquier dispositivo con acceso a internet, dispositivo móvil o computadora).

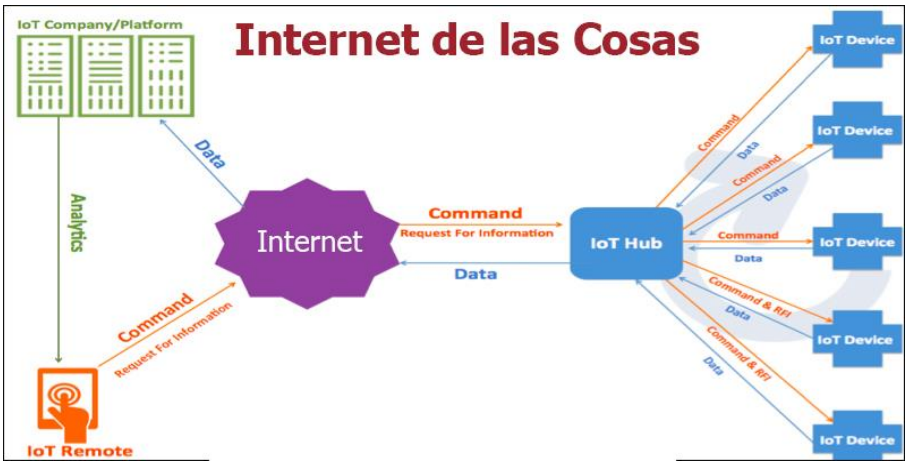

Figura 2.1: Diagrama del funcionamiento *[3]*

Sin duda ha sido en los últimos años, gracias a los grandes avances tecnológicos, cuando se ha extendido de manera espectacular su uso, aplicaciones y estudio.

Cada vez es más frecuente encontrarse con nuevos dispositivos y protocolos capaces de conectarse a Internet y permitir al usuario un control y manejo de forma remota desde cualquier parte del mundo, pero esto no ha hecho más que comenzar. El gran avance en este campo crea también nuevos retos y vulnerabilidades, cabe destacar algunos muy importantes:

- Creación de protocolos de comunicación inalámbricos de bajo consumo, aquí toma protagonismo el protocolo ZigBee como tecnología centrada en las aplicaciones domóticas e industriales con ventajas significativas respecto al consumo, robustez, seguridad, escalabilidad y control.
- La seguridad en las redes IoT, sus dispositivos e infraestructura en la nube es de mucha importancia. Existe una situación crítica ante la falta de procesos que aseguren la integridad y encriptación de los datos.
- Cambio al protocolo IPv6 debido al aumento exponencial de dispositivos, se estima que en el año 2020 habrá 26 mil millones de dispositivos conectados a este mundo [2].

2.2 Tecnologías del software

En este apartado se realiza una introducción a las tecnologías del software usadas en el desarrollo del sistema domótico planteado en este proyecto. Las tecnologías principales que he usado en el proyecto son **Python**, **JavaScript** y herramientas basadas en estos lenguajes.

En primer lugar, Python es usado para continuar el desarrollo de la consola Zigbee proporcionada por el tutor y añadir funcionalidades extras. También lo uso para implementar el Websocket que comentare más adelante.

Por otra parte, el desarrollo de la interfaz y plataforma web se realizó en JavaScript, más específicamente usando **Node.js** y siguiendo el patrón de diseño **MVC** (Model View Controller) que incorpora. [18]

El MVC que usa **Node.js** propone la construcción de tres componentes distintos diferenciados que son el modelo, la vista y el controlador. Es decir, por un lado, en el directorio "models" defino las componentes para representar la información de la red domótica en nuestro caso los paquetes, y por otro lado la interacción del usuario con el interfaz (vistas) y los distintos controladores de rutas. Este patrón de arquitectura del software se basa en las ideas de reutilización del código y la separación de conceptos, características que buscan facilitar el desarrollo de la aplicación y su posterior mantenimiento. En el apartado 3.4 "Desarrollo de la plataforma web" se explicará con más detalle la estructura del proyecto y la implementación de cada tecnología.

Por último, destacar algunas tecnologías webs presentes en el proyecto como, por ejemplo, la tecnología **Tornado** [22] disponible en Python que proporciona un servidor Websocket y servicios asíncronos de comunicación. *"*Tornado es una *tecnología web escalable que proporciona un canal de comunicación bidireccional y full-dúplex sobre un único socket TCP. [17]"* Con la tecnología de Websocket se puede tener varias conexiones en distintos puertos, y conseguir servicios de Websocket sobre un único puerto TCP y servidor HTTP. (a costa de una pequeña sobrecarga del protocolo).

Como interfaz de aplicación para la web se integró "**Materialize**", basado en el "Material Design of Google" [11] y creado por Google. Esta actualmente aplicado a sus productos tanto en web como en su sistema operativo móvil Android, específicamente la versión Lollipop. "Materialize" esta desarrollado en SASS y hace uso de buenas prácticas en HTML5, CSS3 y Javascript. Cuenta con un sistema de rejillas (grid), que dota al aplicativo web de un

sistema responsive adaptable a cualquier dispositivo tablets y móviles inclusive. También viene integrado con la fuente “Roboto” también propuesta por Google en su sistema de diseño “Material Design”. [19]

Materialize también cuenta con una serie de componentes predefinidos desde pantallas modales, botones, formularios, menú, preloaders, tablas y muchos más. Estos componentes cuentan con animaciones que siguen la línea del Material Design.

2.3 Tecnología ZigBee

2.3.1 ¿Qué es ZigBee?

ZigBee es un estándar que define un conjunto de protocolos de redes inalámbricas de área local (WPAN) de baja velocidad de datos. Fue desarrollado por la organización sin ánimo de lucro “Alianza ZigBee” [4] creada en el año 2002.

El protocolo de comunicación de esta tecnología está basado en el estándar **IEEE 802.15.4** para redes inalámbricas LR-WPAN (Low Rate-Wireless Personal Area Network)**,** adoptando sus dos primeras capas. Este estándar es ampliamente usado en redes de sensores inalámbricos (WSN), donde una tasa de transferencia baja no es inconveniente y por el contrario se consigue una comunicación fiable y bajo consumo en los nodos sensores [4].

Como dije anteriormente ZigBee es una ampliación de este estándar por lo que ambos comparten características comunes. La misión de su organización es el desarrollo y mejora del estándar teniendo como objetivos cumplir las siguientes especificaciones:

- Ultra bajo consumo de potencia por parte de los dispositivos 30mA transmitiendo y 3µA en reposo. Esto permite prolongar el uso de las baterías, pudiendo llegar a tener una duración entre 6 meses y 2 años.
- Operar en las bandas de frecuencia de radio sin licencia ISM 868Mhz (Europa), 915Mhz (EEUU) y 2.4Ghz (Mundial).
- Bajo coste de los dispositivos, instalación y mantenimiento de la red.
- Rango de alcance corto entre nodos con distancias de 30 a 50 metros en interiores y 100 metros en el exterior dependiendo del ambiente.
- Velocidad de transmisión reducida con tasas entre 20Kbp/s y máximas de 256Kbp/s.
- Optimizado para ciclo efectivo de transmisión menor a 0.1%
- Instalación fácil y sencilla

- Garantiza redes ampliables y flexibles, con un máximo de 65.535 dispositivos conectados, capaz de ajustarse completamente en topologías en malla con soporte de redundancia.

Actualmente existen muchos estándares que se pueden usar para redes de corto alcance tales como 802.11 y Bluetooth en la figura 1.2 se expone una comparativa en relación a la cobertura y velocidad de transmisión de las diferentes tecnologías. Cada una de estas está desarrollada para una aplicación determinada. ZigBee es el estándar actual más aceptado hoy en día para usar en redes de sensores y actuadores que deben operar con batería. Siendo el ámbito de la domótica, monitoreo ambiental, industrial, hospitales y hoteles donde más se ajustan sus características, ya que provee de una solución simple y de bajo costo.

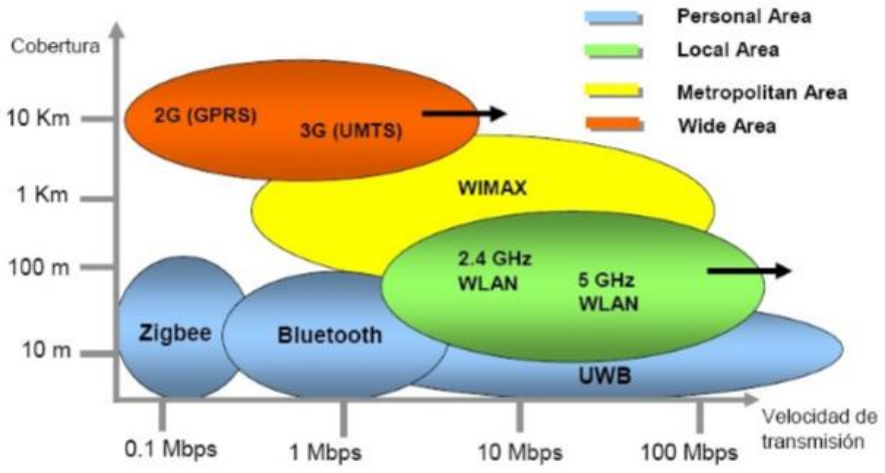

Figura 2.2: Comparativa de los estándares de comunicaciones inalámbricas

2.3.2 Protocolo ZigBee

Las capas del protocolo ZigBee que se muestran en la figura 1.3, se basan en el modelo de referencia ISO para interconexión de sistemas abiertos OSI. Este modelo cuenta con 7 capas, pero ZigBee usa solo 4 capas con el objeto de simplificar la arquitectura para el armado de una red más simple. [4].

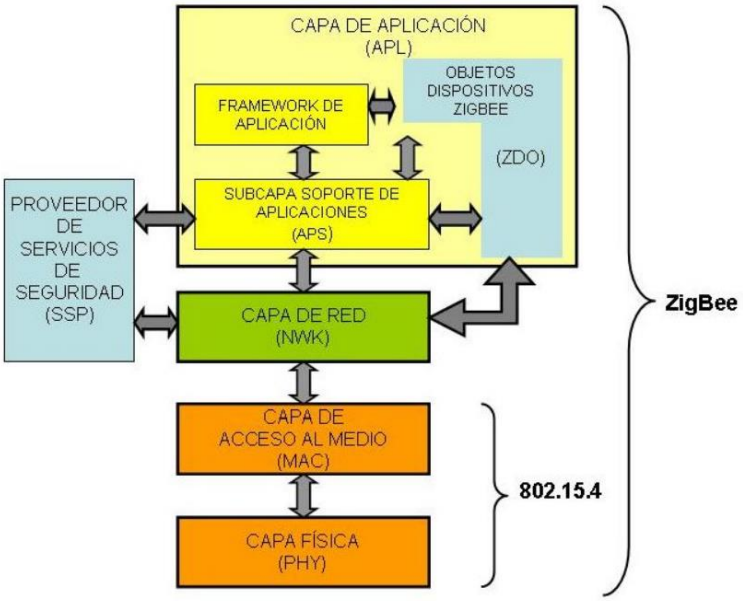

Figura 2.3: Capas de 802.15.4 y ZigBee

Las dos capas inferiores, o sea la capa física (PHY) y la capa de acceso al medio (MAC) provienen del estándar 802.15.4.

En primer lugar, la capa física define las funciones y relación con la capa MAC. Tratando aspectos como la potencia del transmisor, sensibilidad del receptor, evaluación del canal libre (CCA) e indicador de calidad del enlace (LQI).

En segundo lugar, la capa MAC provee una interfaz entre la capa física y la capa de red. Dividida en dos partes principales: el área de datos (MCPS) que comunica las capas vecinas y el área de manejo (MLME), que equivale a una unidad de control, encargada de recibir los comandos desde la capa de red y descodificarlos.

En cambio, las capas de red (NWK) y aplicación (APL) se definen en ZigBee. Estas proveen funciones para el armado y manejo de redes con una interfaz simple para relacionarla con las aplicaciones de los usuarios. La capa de red del coordinador es la encargada de asignar las direcciones de 16 bits a cada miembro de la PAN junto con la cantidad de saltos máximos que puede llegar a realizar.

Cada una de estas capas se interconectan con las capas adyacentes por medio de un SAP (Service Access Point). Un SAP es un lugar por donde una capa superior requiere de un servicio a una capa inferior

Cabe destacar también que este protocolo usa CSMA/CA (Acceso múltiple con escucha de portadora y evasión de colisiones) para solventar el problema del acceso múltiple al medio de transmisión. De esta forma, el resto de equipos de la red sabrán cuando hay colisiones y en lugar de transmitir la trama en cuanto el medio está libre, se espera un tiempo aleatorio adicional corto y

solamente si, tras ese corto intervalo el medio sigue libre, se procede a la transmisión reduciendo la probabilidad de colisiones en el canal [5].

2.3.3 Tipos de dispositivos

El estándar ZigBee define 2 tipos de nodos generales en la red según la función del nodo con el objeto de minimizar el costo en la red.

- FFD (Full Function Device): son los dispositivos capaces de funcionar en cualquier topología. Este dispositivo puede dialogar con cualquier otro.
- RFD (Reducted Function Device): son dispositivos que solamente pueden ser miembros de una red con topología en estrella. Solo pueden dialogar con el coordinador de la red, tienen una baja complejidad, menor requerimiento de procesamiento y memoria.

Con estos dos tipos de nodos se forman la red ZigBee definida por los siguientes elementos:

- Coordinador: Es el encargado de crear, inicializa y controlar la red, estableciendo aspectos como el canal de comunicación, los parámetros de red (PAN ID) y direcciones de los nodos.

  Este elemento es de tipo FFD y debe ser único en toda la red. Una vez concluidas sus funciones iniciales, su comportamiento pasa a ser el mismo que el de un router.
- Router: Son nodos de tipo FFD que se encargan de interconectar dispositivos y encaminar los mensajes hacia los nodos destino. Se emplean para extender la cobertura de la red y crear nuevas alternativas de camino que minimicen la congestión y aporten tolerancia a caídas.
- Nodos finales: Son los nodos RFD finales en la red. Su funcionalidad se restringe a poder interactuar únicamente a través de su nodo padre (controlador o router). Estos no tienen la capacidad necesaria para participar en el reparto de rutas, pero constan del modo “Sleep” para poder pasar la mayor parte del tiempo inactivos reduciendo el consumo de energía al mínimo.

2.3.4 Topologías de red

Las topologías de red que permite el protocolo ZigBee son variadas. Los nodos se pueden interconectar siguiendo las topologías principales: estrella, árbol y malla.

En la figura 1.4 se ve la representación gráfica de la topologías en estrella y entre pares de una red XBee. Ambas representaciones se componen de un nodo coordinador con una PAN ID determinada y conectado a los diferentes nodos routers o dispositivos finales a través del estandar IEEE 802.15.4.

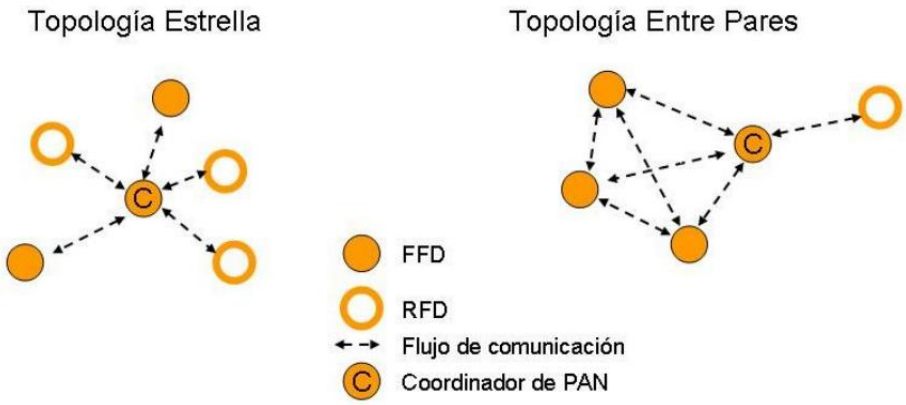

Figura 2.4: Topologías de red

Dentro de la red ZigBee los nodos son capaces de comunicarse gracias a sus direcciones de red. Cada nodo consta de dos direcciones:

- Dirección de 64 bits: esta dirección es única para cada nodo y viene fijada de fábrica al nodo que le identifica.
- Dirección de 16 bits: esta dirección se la asigna el coordinador o router a cada nodo cuando se une a la red. Es única para cada nodo, pero puede cambiar si no se puede comunicar con su padre, abandona la red o se reconecta siendo de otro tipo.

ZigBee también consta de funcionalidades para emitir mensajes en modo broadcast y multicast, de forma que cualquier dispositivo que esté conectado reciba el mismo mensaje.

2.4 Módulos XBee

Los módulos XBee pertenecen a una familia de circuitos inalámbricos de radio fabricados por la compañía “Digi International, Inc.” [6]. Esta empresa tiene varios modelos a su cargo que utilizan el protocolo ZigBee y estándar 802.15.4 para la comunicación e interconexión entre dispositivos.

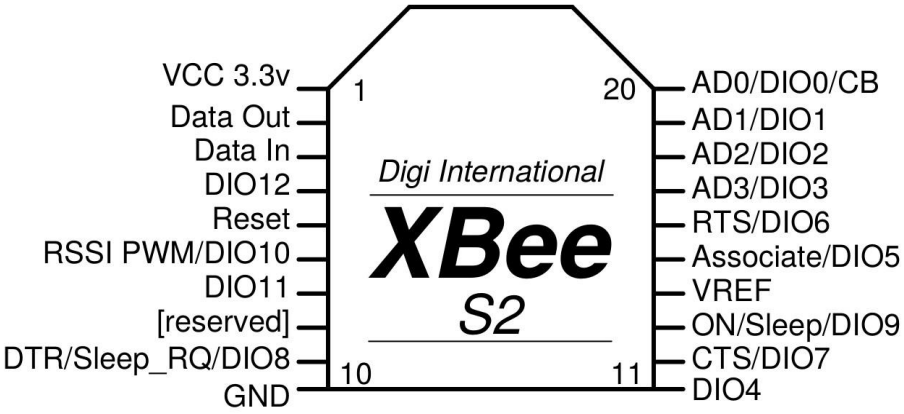

Figura 2.5: Esquema del patillaje físico de un módulo XBee

Desde la introducción inicial, hasta la actualidad Digi International, Inc. ha creado módulos inalámbricos cada vez más potentes y con hardware más evolucionado. Cabe destacar sus recientes módulos con tecnología móvil 3G que provoca un avance en el ámbito de las redes IoT [6].

2.4.1 Características y patillaje de los módulos

	XBee Serie 1	XBee Serie 2
Alcance Interior	Hasta 30 m	Hasta 40 m
Alcance exterior	Hasta 100 m	Hasta 120 m
Potencia de transmisión	1 mW (0dBm)	2 mW (+3 dBm)
RF Transmisión de datos	250 Kbps	250 Kbps
Sensibilidad	-92 dBm (1% PER)	-98 dBm (1% PER)
Voltaje	2,8 – 3,4 V	2,8 – 3,6 V
Consumo en transmisión	45 mA (@ 3,3V)	40 mA (@ 3,3V)
Consumo en reposo	10 µA	1 µA
Dimensiones	0,0960” x 1,087”	0,0960” x 1,087”
Temperatura funcionamiento	-40º a 85º C	-40º a 85º C
Topologías de red	Punto a punto, estrella, malla (con firmware DigiMesh)	Punto a punto, estrella, malla (con firmware DigiMesh)
Número de canales	16 canales de secuencia directa	16 canales de secuencia directa
Opciones de filtrado	PAN ID, canal origen / destino	PAN ID, canal origen / destino
* Estos módulos no son compatibles entre sí.		

Tabla 2.1. Comparativa de las características de los modelos XBee [7]

Hay una gran variedad de versiones, pero dos tipos más generales dentro de este ámbito son los XBee Serie 1, debido a su facilidad para trabajar y escasa configuración inicial. Y los XBee Serie 2, que mejoran el uso de la potencia y permiten hacer redes más complejas.

En la siguiente tabla 2.1 se exponen las características principales de los módulos XBee de la serie 1 y de la serie disponibles actualmente en el mercado.

A continuación, en la tabla 2.2 se detallara la función de cada pin, su tipo de transmisión y el número de pin que esta asignado en los módulo XBee [8]:

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 mínimo 200ms)
6	PWM0 / RSSI / DIO10	Entrada / Salida	PWM Salida 0 / Indicador fuera de la señal RSII / E/S digital 10
7	PWM / DIO 11	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 de asociación / 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 / Comissioning Button	Entrada / Salida	Entrada analógica 0 / E/S digital 0 / botón de comisionado

Tabla 2.2. Patillaje de los módulos XBee ZNet 2.5 [8]

Los módulos XBee para funcionar requieren de dos conexiones: VCC y GND.

2.4.2 Modos de funcionamiento y configuración

Los módulos XBee interactúan con tres modos de funcionamiento [9]:

- Modo transparente: este modo es el equivalente a un enlace con cable serie (USB) entre dos puntos. Los datos recibidos a través del pin DIN se encolan para su transmisión inalámbrica. Cuando se reciben datos a través de la red inalámbrica, se sacan a través del pin DOUT [8].
- Modo API 1 y 2: en este modo se permite enviar y recibir datos a través del envío de paquetes con comandos AT y estructuras de datos predefinidas.

- Modo de comandos: es utilizado para cambiar los parámetros de configuración del XBee.

Cabe destacar que en el modo transparente es el modo más sencillo de uso, pero al mismo tiempo hay algunas operaciones que son difíciles o imposibles de realizar, como, por ejemplo:

- Transmitir datos a todos los nodos de la red (broadcast)
- Cambiar de módulo destino con cada transmisión
- Obtener la dirección del módulo que origina el dato recibido
- Cambiar remotamente la configuración de un módulo XBee

2.4.3 Comandos AT de la API

Para lograr la comunicación efectiva del aplicativo web desarrollado en el proyecto con el módulo XBee se ha usado una Consola API. De esta forma, haciendo uso de la consola se resuelven diversos comandos AT entendibles por los módulos XBee. Los comandos se resumen en las siguientes tablas ordenadas por su función:

- Comandos AT de configuración de la red

Comando	Descripción
ND	Descubre e identifica todos los nodos hoja. Es útil para la inicialización.
NI	Almacena la cadena identificadora de cada módulo
CH	Lee el número de canal que se está usando para la transmisión entre los módulos
SH	Lee los 32 bits superiores de la dirección de 64 bits de cada módulo
SL	Lee los 32 bits inferiores de la dirección de 64 bits de cada módulo
VR	Devuelve la versión del firmware de cada módulo
AI	Devuelve la información relativa de la última petición de unión de cada módulo a la red
OP	Devuelve el identificador de la red PAN al que está asociado cada módulo

Tabla 2.3. Comandos AT para la configuración de la red

- Comandos AT de muestreo de datos

Comando	Descripción
IS	Realiza una única lectura de todas las entradas analógicas y digitales activadas
IR*[x]*	Realiza una lectura periódica de todas las entradas analógicas y digitales activadas. El período de tiempo se específica en milisegundos junto al comando.
IC*[x]*	Establece o devuelve los pines donde se están monitoreando cambios de estado digitales. El comando devuelve un mensaje cuando detecta un cambio de estado en los pines activos como monitoreados. El comando usa como *[x]* una máscara de bits hexadecimal para distinguir los diferentes canales.

Tabla 2.4. Comandos AT para muestreo de datos

- Comandos AT de configuración de los pines

PIN Asignado	Comando	Modo de configuración *[x]*
1	P0*[x]*	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
4	P2*[x]*	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
7	P1*[x]*	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
11	D4*[x]*	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
15	D5*[x]*	0 = Desactivación del pin 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
16	D6*[x]*	0 = Desactivación del pin 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
17	D3*[x]*	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
18	D2*[x]*	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
19	D1*[x]*	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)
20	D0*[x]*	0 = Desactivación del pin 2 = Pin establecido como entrada analógica 3 = Pin establecido como entrada digital 4 = Pin establecido como entrada digital, con valor en baja (0 “Desactivado”) 5 = Pin establecido como entrada digital, con valor en alta (1 “Activado”)

Tabla 2.5. Comandos AT para el muestreo de datos

- Comandos AT para configurar el modo Sleep

Comando	Descripción
SP*[x]*	Específica el tiempo que el nodo permanece dormido. Siendo *[x]* el número de centésimas de segundo. Su rango va desde 0x20 a 0xAF0. - En el nodo coordinador, representa el tiempo que guarda los paquetes en el buffer antes de descartarlos, ya que este nodo no puede dormir. - En los nodo hoja duerme por el tiempo que se establezca
ST[*x]*	Establece el tiempo antes de dormir del nodo, es decir, el tiempo que un nodo está despierto antes de dormir de nuevo. Solo es aplicable a los nodos hoja. Siendo *[x]* un número de 1 a 0xFFFE en milisegundos.
SM[*x]*	Establece el modo de sueño para los nodos hoja. 0 = Opción dormir desactivada 1 = Dormir cuando el Pin 9 (SLEEP_RQ) esté en alta (1) 4 = Dormir cíclico 5 = Dormir cíclico, pero se puede despertar al nodo con una transición de alta a baja del pin 9 (SLEEP_RQ)

Tabla 2.6. Comandos AT para configurar el modo Sleep

Capítulo 3.
Desarrollo del trabajo

En este capítulo nos centraremos en mostrar el funcionamiento del sistema y de la aplicación diseñada. Comenzaremos por explicar cómo funciona la comunicación entre el controlador de red y el Websocket que comunica con el aplicativo web, los diferentes protocolos, librerías y funcionalidades implementadas.

Para concluir, se explicará el funcionamiento lógico de la plataforma web, el interfaz diseñado, la gestión de los datos y todo lo desarrollado para la puesta en marcha.

3.1 Diagrama de bloques

En la figura 3.2 se encuentra el diagrama de despliegue que modela la arquitectura del sistema domótico en tiempo de ejecución. Se muestran la configuración de los elementos hardware (nodos) y se muestra como se conectan los diferentes elementos del software.

El diagrama está dividido en tres bloques:

- Servidor API ZigBee: desarrollado en Python, ejecuta en un hilo la consola ZigBee de forma concurrente con el servidor Websocket que proporciona la comunicación de los paquetes a la plataforma web.
- Servidor web: plataforma web desarrollada en JavaScript para ofrecer un sistema de control a la red domótico para el usuario.
- Base de datos: base de datos SQLite que almacena todos los registros de la red XBee y de los usuarios.

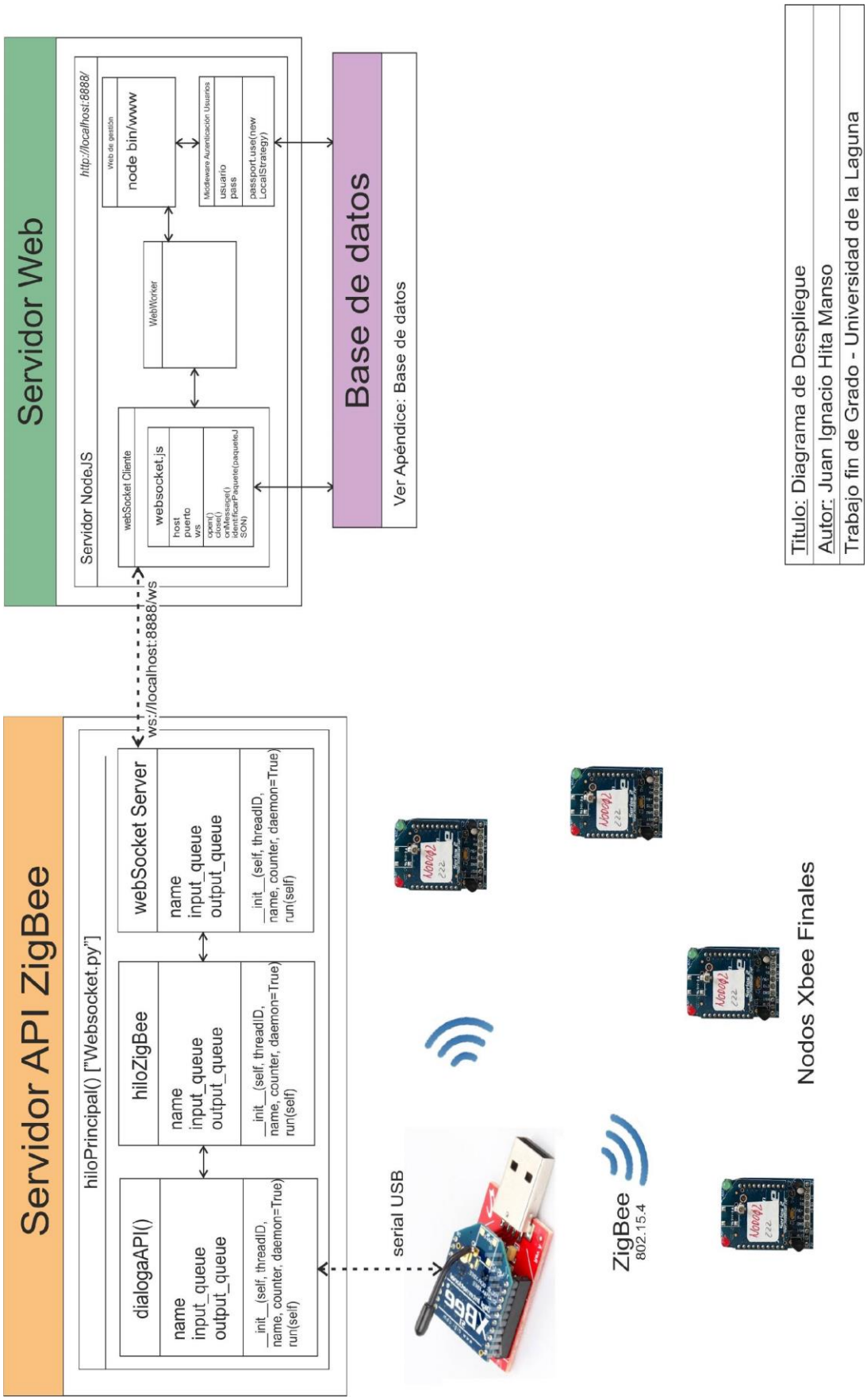

Figura 3.1. Diagrama de despliegue del proyecto

3.2 Servidores y comunicación

A lo largo de este apartado se explicará de forma detallada los diferentes hilos y conexiones concurrentes que forman parte del sistema domótico. El sistema se distribuye en dos grandes bloques: el servidor de la API ZigBee y el servidor web. Ambos servidores corriendo en la misma máquina, y pensado para estar ubicada en un computador de placa reducida al estilo Raspberry Pi o BeagleBoard.

3.2.1 Servidor API ZigBee

El servidor API ZigBee es el encargado de comunicarse con el módulo XBee Coordinador y el servicio Websocket. La máquina que ejecuta este servidor esta físicamente conectada por interfaz serie USB al nodo coordinador de la red XBee, y realiza la comunicación hasta la consola interactiva de la API.

Las clases para ejecutar la consola de control son suministradas por el tutor y desarrollada en Python, la consola está formada de las siguientes clases:

- **Conexiones.py:** clase que gestiona la conexión serial con el módulo XBee a nivel físico, necesaria para la conexión.
- **DialogaAPI.py:** clase para gestionar el dialogo mediante API, el envío de comandos y recepción de paquetes. Es necesario una conexión previa inicializada y una interfaz de comunicación.
- **ConsultaAPI.py:** clase que crea una consola de la red XBee, usando el DialogaAPI es capaz de enviar comandos y recibir retroalimentación. No se usa el fichero directamente en la ejecución del Websocket, pero es útil para realizar conexiones locales de depuración.
- **Websocket.py:** clase implementada para gestionar el servicio de la API y el servicio Websocket tornado de forma concurrente. Si los servidores están centralizados en un único dispositivo, la conexión socket cliente se realiza en la misma máquina. Si no, es necesario configurar la IP de la computadora que está conectada al nodo coordinador.

Para implementar el servidor Websocket he utilizado el módulo “threading” instanciando dos hilos concurrentes: el primero con la ejecución de la consola de control, es decir, “DialogaAPI.py” que posibilita la comunicación con el módulo XBee coordinador.

En el segundo hilo se ejecuta un servidor Websocket con soporte para conexión simultanea de varios clientes y comunicación con los eventos del hilo ZigBee.

Debido a que Python utiliza el mecanismo “GIL” (Global Interpreter Lock), que impide que múltiples “threads” modifiquen los objetos de Python a la vez en una aplicación multihilo, es necesario implementar un mecanismo de sincronización de hilos y una estructura de datos común. Para la sincronización he empleado los “Locks”, estos objetos tienen dos estados posibles: bloqueado o desbloqueado y mediante estos se coordinan la lectura y escritura de los hilos sin bloquear la comunicación.

La estructura de datos usada para comunicar la información entre los hilos es una “multiprocessing queue” heredada del módulo Queue.

Creando dos objetos de colas globales (FIFO) y con un agente (cliente web) que introduzca los mensajes en la cola, y una clase (hilo ZigBee) que produzca los mensajes de respuesta tendremos los mensajes intercomunicados entre la consola XBee y el Websocket. Los mensajes de respuesta son los paquetes que devuelven la consola de control.

3.2.2 Websocket

Como introducimos anteriormente en el apartado 2.2, para el servidor Websocket he optado por usar “Tornado” este framework web y biblioteca de redes asíncronas nos ofrece grandes ventajas para el proyecto. Con posibilidad de escalar varias conexiones abiertas sin problema, lo hace ideal para esta aplicación, que requiere varios clientes y conexiones de larga duración para cada usuario.

Imagen 3.1: Logo Tornado

Las funciones que provee el servicio de Websocket en el servidor son las siguientes:

- Mecanismo de comunicación bidireccional, rápida y segura entre el hilo demonio de la consola Zigbee y los clientes conectados al servidor.
- Conexión con varios clientes de forma concurrente
- Conexión cliente pasiva con el servidor web, aunque no hay clientes conectados a la web. La utilidad de esto se atribuye a que el sistema esta conectado a la base de datos de forma permanente para poder insertar o realizar consultas.

- Posible comunicación entre los clientes conectados a la plataforma

La implementación del servidor Tornado la he realizado mediante dos clases:

- Clase “MainHandler” heredada de “tornado.web.RequestHandler”:

  es la encargada de inicializar la sala mediante el objeto “room_handler”, añadiendo la información que identifica al cliente. Los nicks en la versión actual se establecen de forma fija, ya que no es una necesidad en el proyecto.

- Clase “WSHandler” heredada de “tornado.websocket.WebSocketHandler”: es la encargada de inicializar el servidor Websocket, añadir los nuevos clientes a la clase “RoomHandler” y se definen las funciones para el envío y recepción de paquetes al hilo Zigbee.

```
class MainHandler(tornado.web.RequestHandler):

    def initialize(self, room_handler):
        self.__rh = room_handler
        message_salida = output_queue.get()
        self.write_message(message_salida)

    def get(self):
        room = self.get_argument("room")
        nick = self.get_argument("nick")
        cid = self.__rh.add_roomnick(room, nick)

class WSHandler(tornado.websocket.WebSocketHandler):
    def initialize(self, room_handler):
        """Store a reference to the "external" RoomHandler instance"""
        self.__rh = room_handler

    def open(self, client_id):
        room = 'room'
        nick = 'user'
        client_id = self.__rh.add_roomnick(room, nick)
        self.__clientID = client_id
        print 'clientID[Open]'
        print self.__clientID
        self.__rh.add_client_wsconn(client_id, self)
        self.__rh.functionprueba(room, self)
        logging.debug('Nueva conexion con el cliente Web')
        if self not in clients:
            clients.append(self)
            logging.debug('Cliente añadido')
        self.write_message('Cliente conectado')
```

Imagen 3.1. Código de la clase Tornado Websocket

Para implementar el soporte de conexión simultanea de varios clientes en el servidor Websocket, fue necesario implementar la clase “RoomHandler” que se muestra en la imagen 3.1.

Esta clase consta de una estructura para almacenar información de cada cliente que se conecta al Websocket y gestionar las diferentes salas (en el caso de que coexistieran varias subredes ZigBee dentro del mismo sistema). La identificación de cada cliente se almacena en el vector “client_info”, junto

con la sala a la que pertenece y el “client_id”, único que se genera por primera vez cuando el cliente realiza la conexión.

La generación de este “id” se realiza mediante la generación de un objeto UUID inmutable de versión 4 en formato hexadecimal.

```
class RoomHandler(object):
    def __init__(self):
        self.client_info = {}
        self.room_info = {}
        self.roomates = {}

    def add_client_wsconn(self, client_id, conn):
        self.client_info[client_id]['wsconn'] = conn
        cid_room = self.client_info[client_id]['room']

        if cid_room in self.roomates:
            self.roomates[cid_room].add(conn)
        else:
            self.roomates[cid_room] = {conn}

        for user in self.room_info[cid_room]:
            if user['cid'] == client_id:
                user['wsconn'] = conn
                break

        self.send_join_msg(client_id)
        nick_list = self.nicks_in_room(cid_room)
        cwsconns = self.roomate_cwsconns(client_id)
        self.send_nicks_msg(cwsconns, nick_list)
```

Imagen 3.1. Código de la clase RoomHandler

Para la gestión de las diferentes salas y clientes del servidor, se usan diferentes funciones:

- add_client_wscon(self, client_id, con): realiza una conexión cliente con el servidor websocket, se le pasa por parámetro la conexión (con) y el client_id.
- remove_client(self, client_id): elimina toda la información del cliente que se le pasa por parámetro y realiza la desconexión del servidor.
- add_roomnick(self, room, nick): añade a la sala (room) el (nick) del usuario que haya establecido conexión. Esta función esta pensada para una futura implementación en la que haya comunicación con los diferentes clientes conectados a la plataforma. Creando en el aplicativo funciones como grupos de privacidad y estableciendo niveles de acceso.
- Funciones de mensajes específicos (send_join_msg, send_nicks_msg, send_leave_msg): estas funciones envían paquetes JSON cuando se producen conexiones clientes con el servidor.

3.2.3 Servidor Web

En el segundo bloque se encuentra la aplicación web “Domoz”, decidí desarrollarla con la infraestructura web “Express” en el entorno de ejecución “Node.js”. Construido con el motor de JavaScript V8 de Chrome, ofrece muchas ventajas y compatibilidades. Además, usa un modelo de operaciones E/S sin bloqueo y orientado a eventos, lo que lo hace liviano y eficiente, a la par que oportuno para la aplicación en este proyecto.

Imagen 3.4: Logo Node.js

La plataforma web consta de una conexión cliente pasiva con el Websocket para almacenar registros en la base de datos, aunque no haya clientes en la plataforma web conectados. Esto es útil para circuitos reales, donde se pueda llevar un registro de hora y fecha para cada interruptor o dispositivo remoto de la vivienda. Además de la posibilidad de registrar los valores de los sensores analógicos de temperatura y humedad, para posteriormente exportarlos y realizar un análisis.

Esta conexión se inicializa en el fichero “app.js” usando “Webworkers”. Mediante estos podemos ejecutar scripts en hilos en segundo plano al contenido web. Una vez creado, el worker puede enviar mensajes a la tarea creada mediante envió de mensajes al manejador de eventos especificado por el creador. Sin embargo, los workers trabajan dentro de un contexto global diferente al de la ventana actual. Pudiendo realizar así tareas sin interferir con la interfaz de usuario.

3.2.4 Lógica de control

La lógica de control está asociada a cada usuario y almacenada en la base de datos. Para cada usuario se almacena referencias de los nodos por defecto de la red XBee, los sensores configurados, la configuración de los nodos con las alertas y eventos. Es posible más adelante implementar relaciones entre usuarios como, por ejemplo: crear niveles de acceso a determinados actuadores o nodos de la vivienda.

Las respuestas de la red XBee llegan a la plataforma web como cadenas de caracteres que posteriormente se manipulan, serializan a JSON [21] y se reparten según el tipo de paquete que sea.

3.3 Base de datos

Para la gestión de los datos he usado una base de datos SQLite integrada en la plataforma web. SQLite a diferencia de los motores de bases de datos convencionales, no es un proceso con el que la aplicación se comunica, sino un fichero que almacena la biblioteca SQLite y se enlaza con el programa pasando a ser parte integral del mismo.

Debido a las características de SQLite ofrece ciertas ventajas al diseño embebido del sistema domótico propuesto. Como, por ejemplo:

- Base de datos completa en un único fichero y posibilidad de usarse en aplicaciones móviles u otros servicios externos.
- Es totalmente autocontenida (sin dependencias externas)
- Puede funcionar enteramente en memoria, lo que la hace muy rápida

Pasamos a describir el uso de cada tabla:

- **usuarios:** almacena la información referente a los usuarios.
- **nodos:** almacena la información referente a cada nodo XBee.
- **nodosUsuarios**: almacena la relación entre los nodos y los diferentes usuarios registrados en la web.
- **configDispositivos:** tabla que relaciona los dispositivos conectados a cada nodo y los usuarios
- **dispositivos:** tabla que define los dispositivos de cada nodo, toda la información del dispositivo y en que pin está conectado.
- **magnitudesTipo:** almacena los diferentes tipos de magnitudes de los dispositivos.
- **eventos:** guarda la información de los eventos creados por los usuarios,
- **alertas:** almacena toda la información de las alertas y la retroalimentación.
- **logSession y logComandos**
- **widget**s: se guardan los widgets de la pantalla principal que configura cada usuario en su perfil.

Diagrama Entidad / Relación

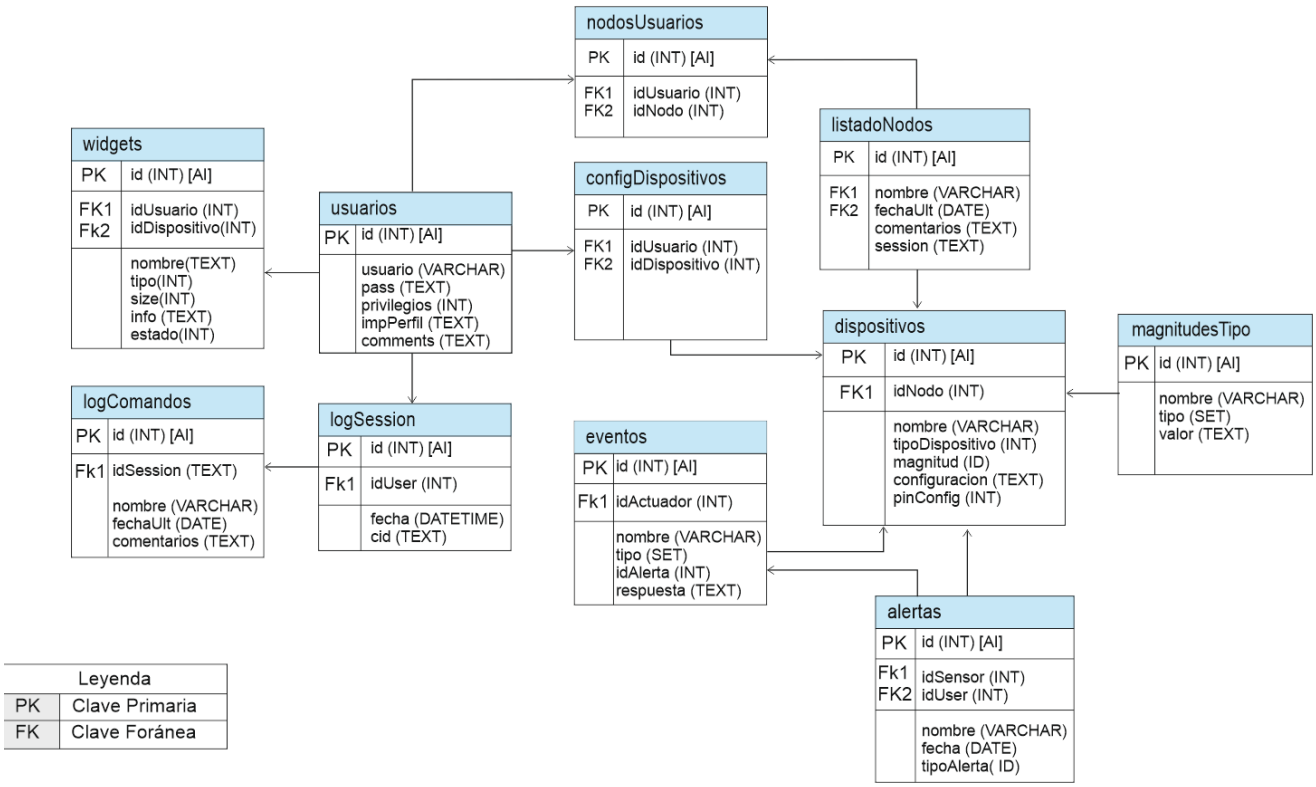

Figura 3.2. Base de datos de la aplicación.

3.4 Desarrollo de la plataforma web

3.4.1 Descripción y características

Como plataforma de control se ha desarrollado la aplicación web “Domoz” que sirve de interfaz de usuario fácil e intuitivo. Esta interfaz ofrece un modo de añadir y configurar los nodos de la red XBee, crear accesos directos a los actuadores o valores de los sensores en la página principal o crear alertas de eventos sobre los actuadores de los nodos finales.

En la figura 3.1 se muestra la estructura del proyecto con una breve descripción del contenido de cada directorio.

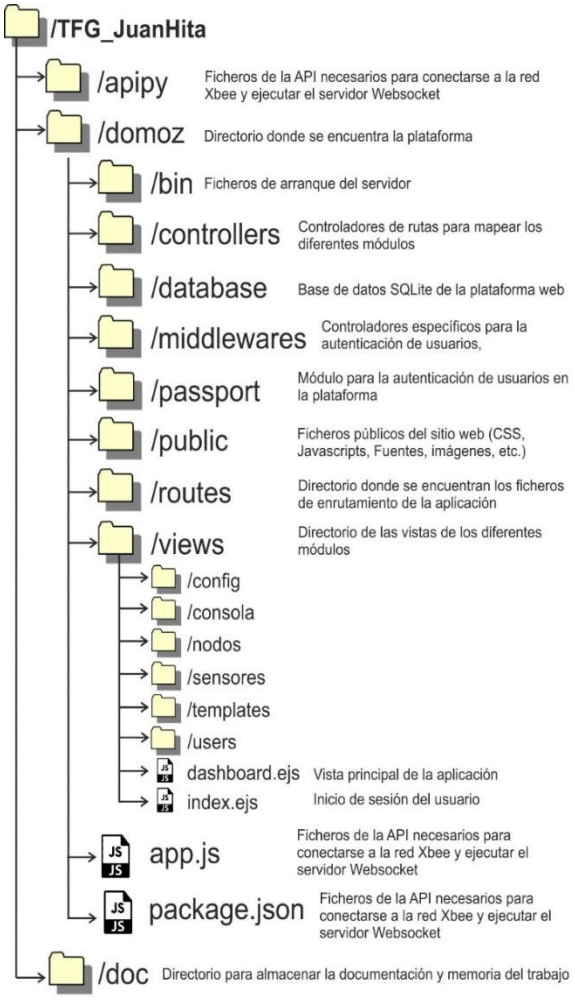

Figura 3.3. Estructura del proyecto.

Toda la información y configuraciones quedan almacenados en la base de datos y asociados a cada usuario. Las vistas iniciales de la web es el inicio de sesión de usuario y la creación de nuevas cuentas de usuario que se ven en la siguiente captura.

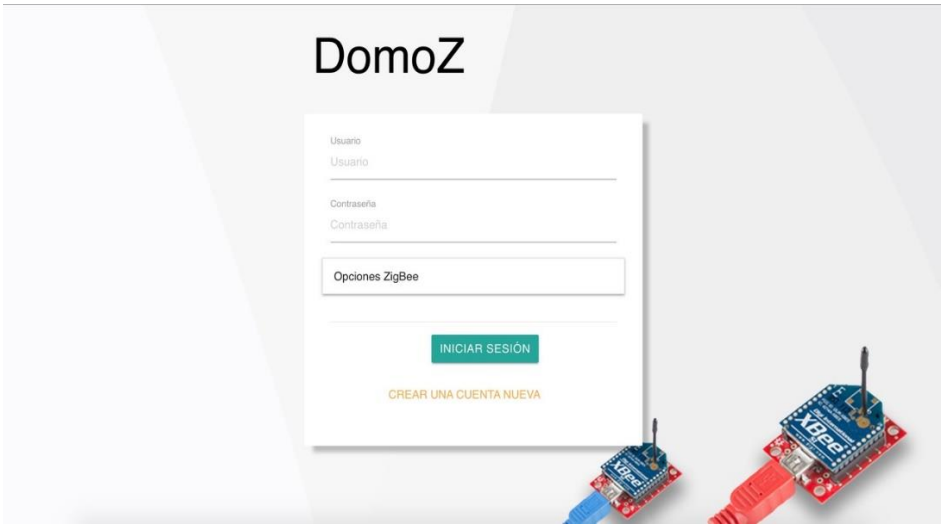

Imagen 3.4. Inicio de sesión de la web

El sistema de autentificación de usuarios de “Domoz” es local, toda la información esta almacenada en la base de datos del servidor. Para implementarlo he usado “passport-local” de la librería “Passport” [10].

“Passport ofrece un sistema de middleware de autentificación para Node.js, muy flexible, modular y con facilidad de integración con “Express”. [10] Esta librería también tiene paquetes adicionales para autentificación con servicios externos, como Google, Facebook, Twitter, Google+, como posible mejora de futuro.

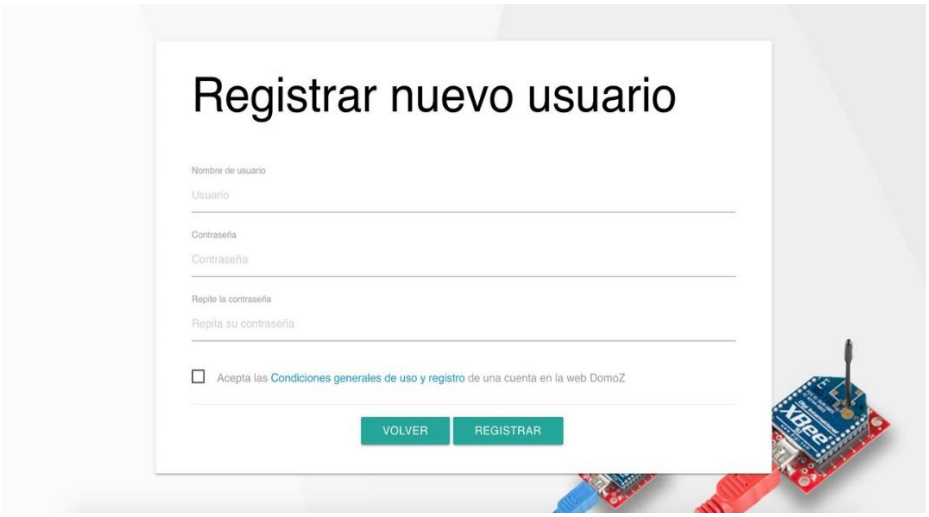

Imagen 3.5. Registro de un nuevo usuario en la web

3.5 Interfaz de la aplicación

Para la interfaz de la aplicación Domoz como se comentó en el capítulo introductorio a las tecnologías del software, he empleado mayoritariamente el framework “Materialize” [11] creando una interfaz final con simpleza y usabilidad. Además he hecho uso de otras herramientas de desarrollo web, entre ellas cabe destacar: jQuery, AJAX y ChartJS. A continuación, se explicará la integración de cada herramienta en mayor detalle.

La integración de “Materialize” en la plataforma web se realiza desde su sitio web oficial, descargando las diversas fuentes CSS, JavaScript, Fuentes y añadiendo a la carpeta pública del proyecto. He implementado en el proyecto las vistas parciales, diferenciando las distintas partes que permanecen fijas en la web para reutilizar el código estas son: barra superior (navbar), barra lateral izquierda (barleft) y el fichero templates que unifica todas las dependencias externas de la web y se añaden las fuentes de “Materialize”

Sobre las otras herramientas usadas para el desarrollo se encuentra jQuery [20], esta librería libre de código abierto escrita en Javascript permite simplificar la manera de interactuar con los documentos, la manipulación de los objetos del árbol DOM. interactuar con el Websocket por la parte cliente y realizar la lógica de control de comandos AT.

Para efectuar las diferentes consultas a la base de datos en la parte cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano he usado la tecnología AJAX (Asynchronous JavaScript And XML). De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.

La pantalla principal de la web se muestra en la imagen 3.6 consta de una barra superior fija que incluye las opciones del perfil del usuario, una barra lateral con los módulos de la web y un sistema de widgets configurables e independientes para cada usuario. Mediante el botón rojo circular superior se puede acceder a la ventana modal (Imagen 3.8), que permite añadir nuevos widgets con información sobre los actuadores, sensores , eventos, etc.

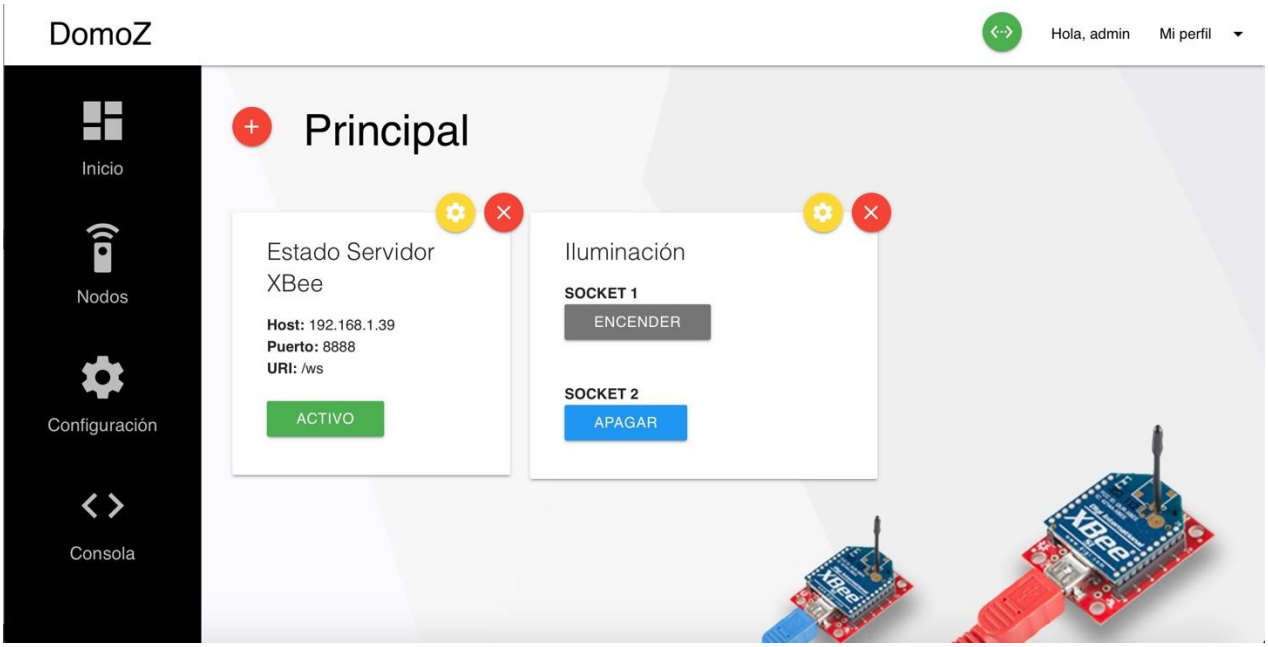

Imagen 3.6. Pantalla principal o de inicio de la web

Cada widget consta de dos botones: uno permite acceder al panel de configuración dinámica del widget (imagen 3.7) y el otro elimina el widget de la pantalla principal del usuario.

Imagen 3.7. Modal para crear nuevos widgets en la pantalla principal

Imagen 3.8. Configuración del widget de iluminación

Los diferentes tipos de widgets que se pueden configurar son los siguientes:

- Iluminación
- Lectura de sensores analógicos (Temperatura, humedad, potenciómetro…)
- Estado del servidor
- Anotaciones

La página de gestión del sistema domótico consta de varias páginas más, pasaremos a describirlas:

3.5.1 Listado de nodos

En el icono de Nodos del menú lateral se accede a la configuración de los nodos de la red. En la parte superior se encuentra los nodos en tiempo real de la red y en la parte baja los asociados al usuario. Mediante el botón de Refrescar, se fuerza a realizar una nueva comprobación de los nodos enviando el comando “ND” al nodo coordinador de la red.

Cada nodo asociado al usuario tendrá varias opciones con vistas específicas para ver una ficha del nodo hoja, editar los parámetros del nodo, la asignación de pines y su eliminación.

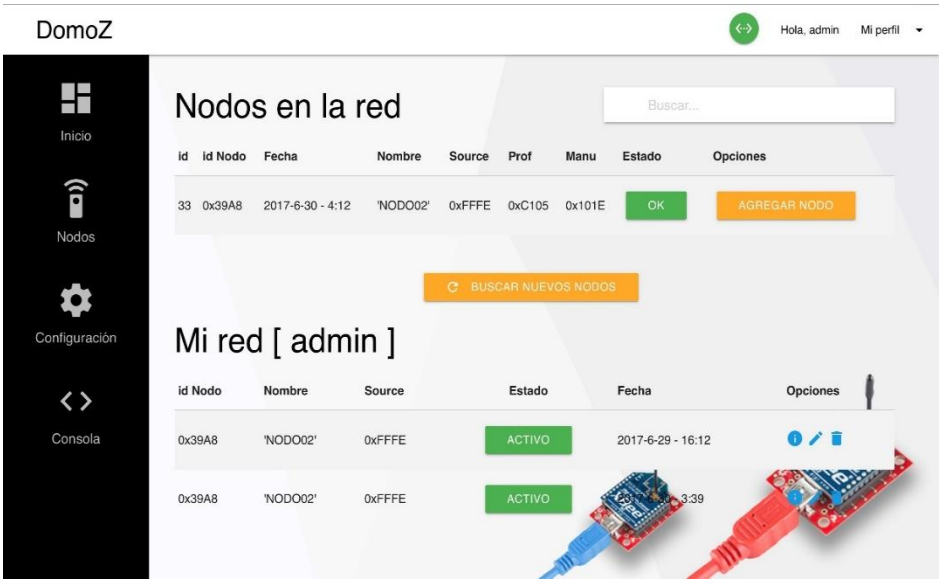

Imagen 3.9. Vista del listado de nodos y la asignación de nuevos nodos al usuario

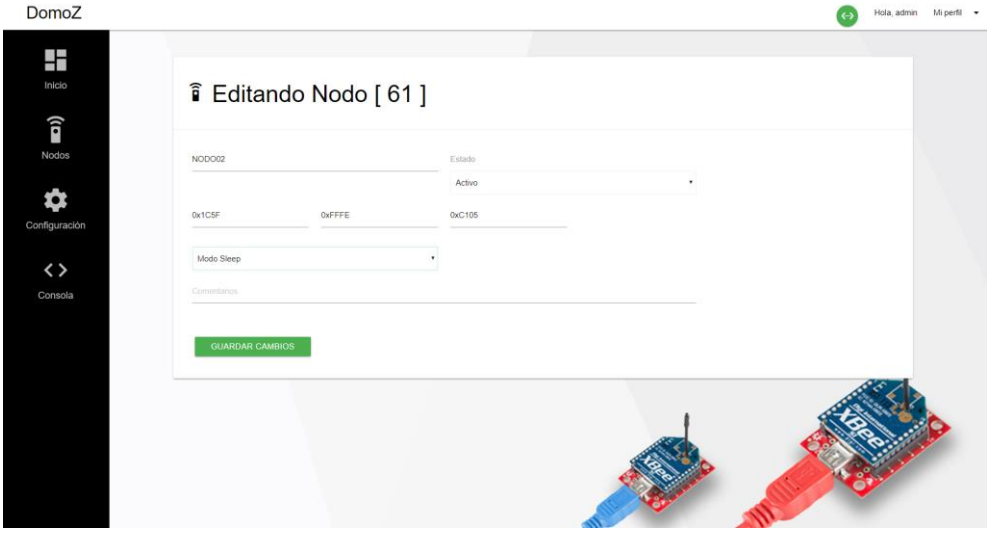

Imagen 3.10. Vista de editar un determinado nodo de la red

3.5.2 Configuración de la red

En esta sección del menú encontramos diferentes apartados divididos en pestañas para configurar los diferentes dispositivos de la red: sensores, actuadores, los diferentes eventos y las alertas asociadas.

Mediante el botón circular rojo ubicado en la parte superior de cada vista podemos añadir nuevo contenido a nuestra red y configurarla.

En cada apartado podremos ver, crear, modificar y eliminar los sensores, actuadores, alertas y eventos.

- Sensores

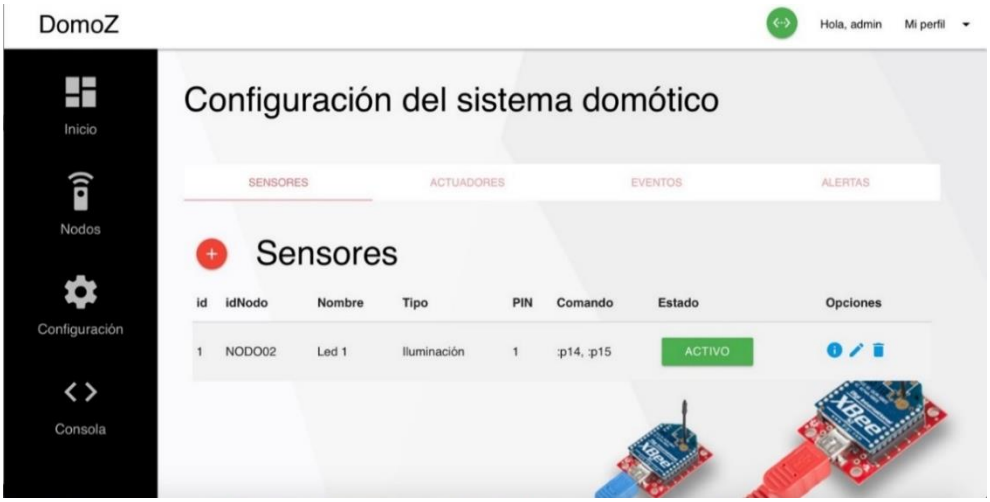

Imagen 3.11. Vista de configuración y listado de sensores

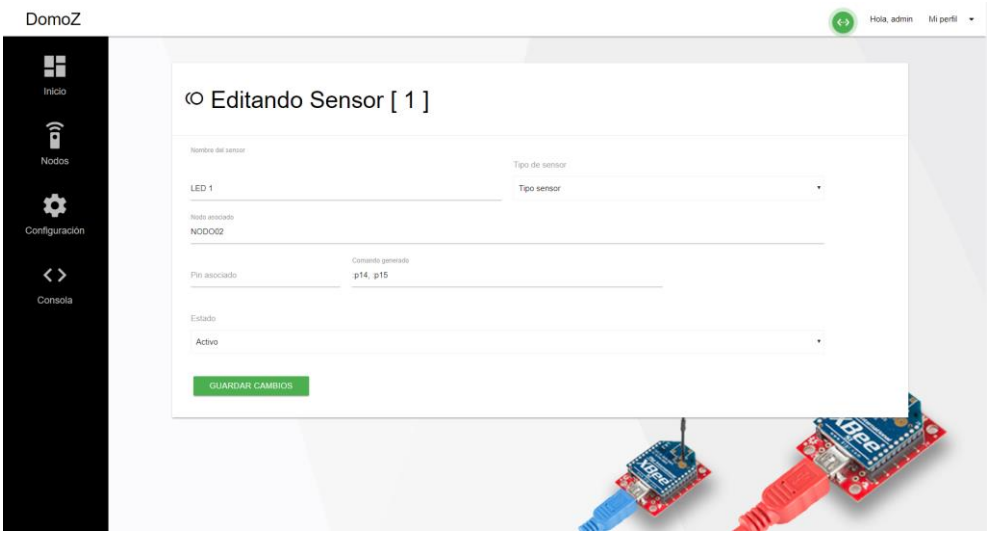

Imagen 3.12. Modificar sensores

- Actuadores

Imagen 3.13. Vista de configuración y listado de los actuadores

- Eventos

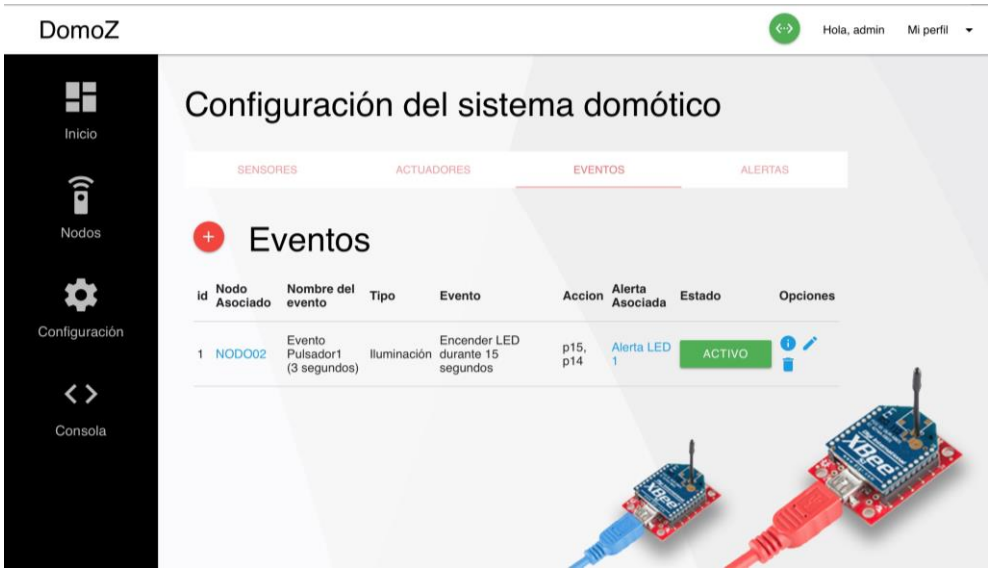

Imagen 3.14. Vista de configuración y listado de los eventos

- Alertas

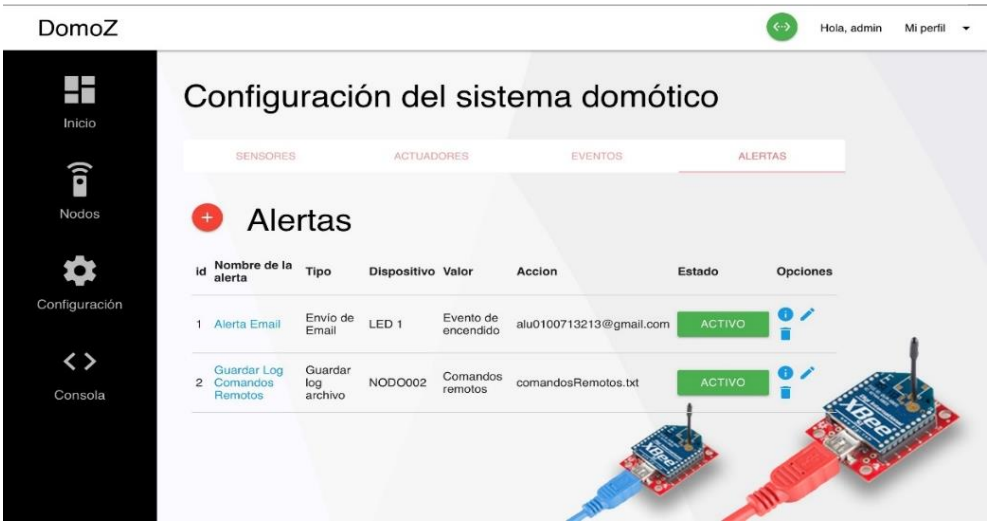

Imagen 3.15. Vista de configuración y listado de las alertas

3.5.3 Consola de XBee

Esta página nos permite mandar comandos al nodo coordinador o a un nodo “hoja” de forma directa a través del formulario.

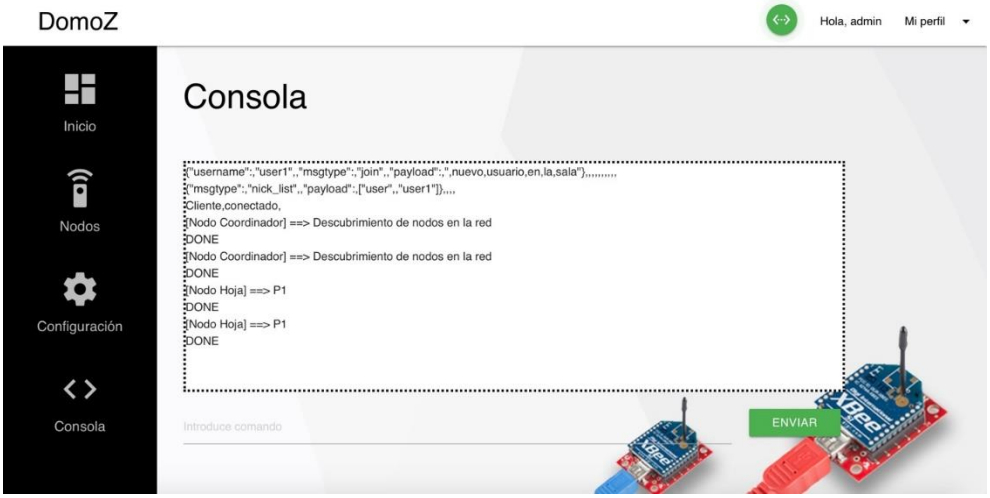

Imagen 3.16. Vista de la consola de red XBee

Capítulo 4.
Puesta en funcionamiento y resultados

La puesta en funcionamiento del sistema se ha realizado con un nodo XBee coordinador conectado a un portátil que hace de servidor y un nodo final con varios actuadores, leds y sensores alimentado por una batería externa USB.

Los dispositivos que se han empleado para la realización de este proyecto son varios módulos XBee ZNet 2.5 OEM RF Module, anteriormente conocidos como Series 2 con “Chip Antenna”.

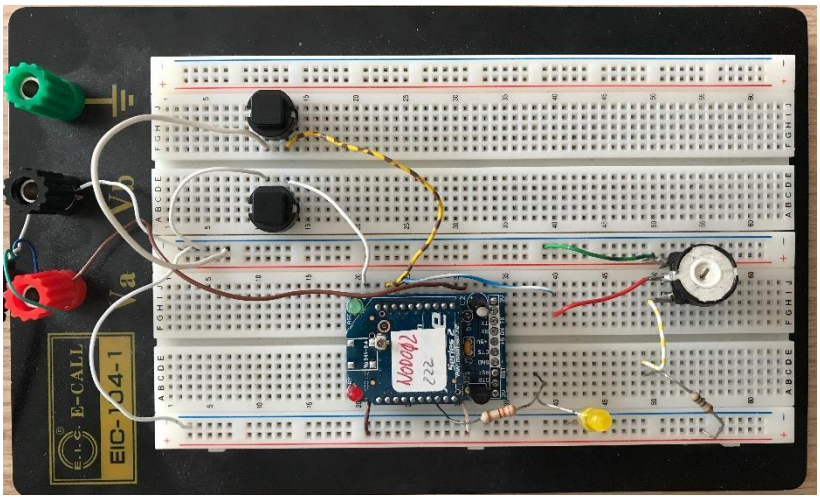

Imagen 4.1: Protoboard con el nodo final, sensores y actuadores

Imagen 4.2: Nodo coordinador utilizado

En la máquina servidor se ejecuta a la vez el servidor web y el Websocket que mantiene conexión con la red XBee. Los resultados obtenidos son favorables consiguiendo hacer conexión a través de la plataforma web, configurar el nodo final y realizar cambios en los dispositivos conectados. También se ha conseguido acceder a través desde un dispositivo móvil e interactuar con los nodos finales.

En la figura 4.1 se muestra el circuito del nodo final que se ha usado para la puesta en marcha del sistema domótico.

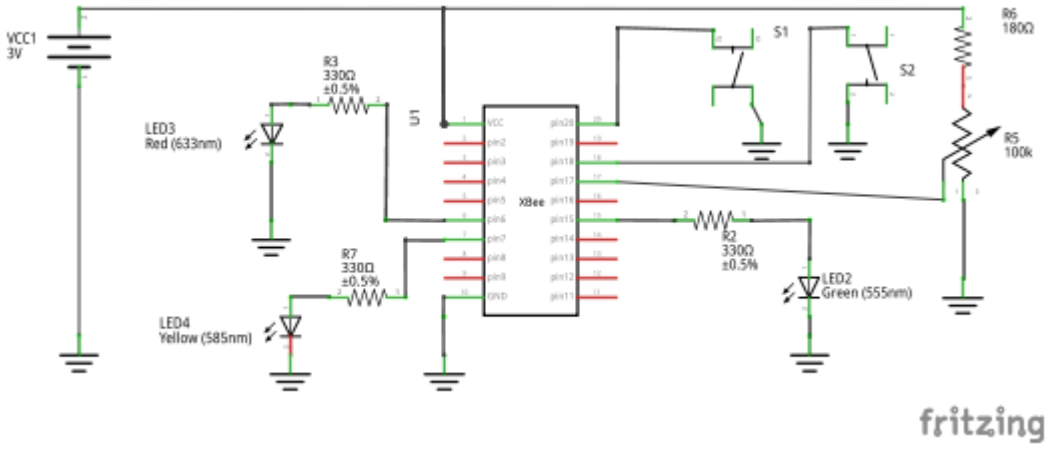

Figura 4.1 Circuito electrónico del nodo final de la red XBee

Los resultados del proyecto es la codificación de una plataforma web básica con gran posibilidad de aumentar nuevas funcionalidades. Debido al poco tiempo de realización y las dificultades que me he encontrado con la concurrencia y los procesos en paralelo no están desarrolladas todas las funcionalidades por completo, pero si las funciones básicas y la estructura general.

Un caso real de uso de la aplicación sería en primer lugar registrar una nueva cuenta de usuario, posteriormente iniciar sesión para acceder a la web. En este momento entraríamos a la plataforma vacía sin datos para ese usuario, lo primero necesario para configurar la red XBee es ir al apartado de nodos y mediante el botón “agregar nodo” se agregan los nodos a nuestra red personal, guardando registros en la base de datos.

Una vez tengamos los nodos añadidos a nuestra red personal, podremos usarlos para configurar nuevos sensores y actuadores. En el menú de Configuración mediante diferentes ventanas modales agregamos un nuevo sensor o dispositivo de iluminación LED configurando el nodo 002 y como pin asignado el 7 que ejecuta el comando “p14” y “p15” en su baja y alta. También es posible configurarlo como entrada digital monitoreada con el comando “p13” opción dentro de la vista de añadir nuevo sensor.

Para configurar los diferentes actuadores seguimos un procedimiento similar al anterior, entramos en el apartado de configuración pestaña de “Actuadores”. Y mediante el botón circular accedemos al formulario para añadir nuevos actuadores a nuestra red XBee, rellenamos el nombre, nodo asociado y el PIN asignado donde se encuentra el dispositivo.

Sobre las configuraciones de los eventos y las alertas se realizan igualmente en el apartado de configuración asociándolos a los dispositivos conectados a los nodos XBee.

Los eventos que se pueden programar pueden ser muy diferentes, por ejemplo, entre los posibles eventos en los pulsadores son:

- Evento de tiempo en la pulsación que manda una acción a un dispositivo actuador, ejemplo: encender LED durante x segundos.
- Evento de dos o tres pulsaciones seguidas sobre un pulsador que realiza una acción en un dispositivo.

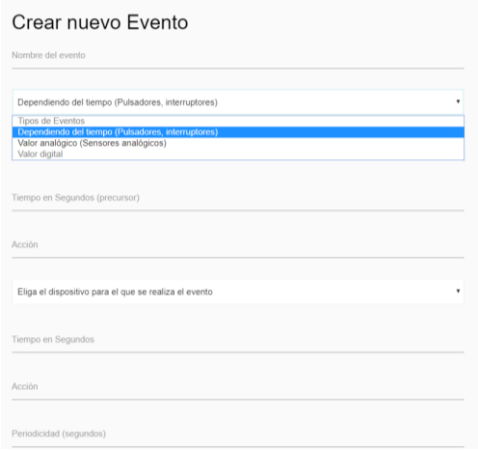

Imagen 4.1 Formulario para creación de un nuevo evento

La configuración de las alertas al igual que los eventos son muy adaptables, en la plataforma web he propuesto varios tipos diferentes que se pueden asignar a un dispositivo o un evento existente:

- Envió de email si cierto dispositivo tiene un valor menor, mayor o igual a cierto valor que se configura en la alerta.
- Envió de un mensaje de alerta a la pantalla principal de un usuario de la web.
- Guardar un log en un fichero externo de los cambios producidos en el dispositivo, función no implementada.
- Alerta sonora para cierto evento o cambio en un dispositivo, función no implementada.

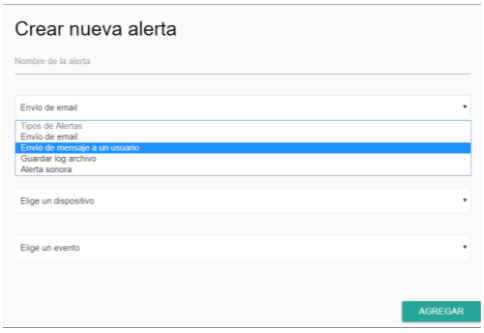

Imagen 4.1 Formulario para creación de una nueva alerta

Capítulo 5.
Conclusiones y líneas futuras

5.1 Conclusiones

- Se ha diseñado un sistema de comunicación entre los servidores usando sockets e hilos.
- Se ha desarrollado un servidor Websocket en Python para la comunicación entre la consola Zigbee y la plataforma web.
- Se ha desarrollado la aplicación web en Node.js (Javascript) orientada a usuarios para la gestión del sistema domótico y comunicación con los dispositivos.
- Se ha implementado el modelo de datos y la base de datos SQlite
- Se ha intentado desarrollar un sistema configurable, personalizable y económico para usar los nodos XBee en la domótica de una vivienda.

5.2 Líneas futuras

El sistema domótico desarrollado está pensado para poder seguir varias líneas futuras:

- Ejecutar los servidores en un computador independiente de placa reducida conectado por serial al nodo coordinador, al estilo Raspberry Pi o Beagleboard que haga de servidor central de todos los servicios domóticos. Este servidor se encargaría de mantener conectados todos los procesos necesarios para la red XBee y la plataforma web.
- Avanzar en el sistema de eventos y alertas, estos aportan mayor configuración y adaptabilidad al entorno. Con la mejora en este aspecto, se podrían realizar acciones automatizadas dependiendo de los sensores o actuadores, dependiendo del valor de algún tipo de sensor automatice el sistema. Para ello será necesario ampliar el conjunto de clases de control y control de comandos más personalizados.
- Mejorar el “tiempo real” en los paquetes IO para actualizar la web cuando se acciona un actuador en un nodo de la red del usuario.

Capítulo 6.
Summary and Conclusions

6.1 Conclusions

- It has been designed a sytem of communication between the servers using sockets and threads.
- It has been developed a server Websocket using Python for the communication between the Zigbee console and the web plataform.
- It has been developed a application web in Node.js (Javascript) user-oriented for the management of the domotic system and the communication with the devices.
- It has been developed the data models and SQLite data base.
- It has been tried to develop a configurable, customizable and economic system to use XBee nodes in the automation of a house.

Capítulo 7. Presupuesto

En este apartado se presenta un análisis del presupuesto aproximado del proyecto dividido en dos apartados: el coste humano de la mano de obra para desarrollar el sistema e implementar el servicio web y el coste cuantificable de los materiales necesarios para crear la red de sensores, los módulos XBee y los diferentes dispositivos.

7.1 Coste en recursos humanos

Concepto	Horas	Coste por hora	Subtotal
Mano de obra para el desarrollo del sistema	300	15 €	4500 €
Puesta en marcha	20	15 €	300 €
		TOTAL:	4800 €

Tabla 7.1. Tabla de costes en recursos humanos

7.2 Coste en recursos materiales

Todos los costes unitarios de cada componente se han recogido de la tienda BricoGeek [18]

Concepto	Cantidad	Coste por unidad	Subtotal
Módulos XBee	2	25 €	50 €
Pulsadores	2	0,15 €	0,30 €
LED	1	0,10 €	0,10 €
Potenciometro	1	0,20 €	0,20 €
Resistencias	2	0,15 €	0,30 €
Software	0	0 €	0 €
Raspberry Pi 3 (Opcional)	1	41,20 €	41,20 €
Fuente de alimentación LIPO externa USB	1	12,90 €	12,90 €
Protoboard Mediana	1	5,50 €	5,50 €
		TOTAL:	110,50 €

Tabla 7.2. Tabla de costes en recursos materiales

Bibliografía

[1] Internet de las Cosas.

http://www.domodesk.com/a-fondo-que-es-el-internet-de-las-cosas

[2] Internet de las Cosas - Seguridad

http://www.tyniot.com/las-ciudades-inteligentes-utilizaran-9-700-millones-de-objetos-conectados-para-2020/

http://www.gemalto.com/latam/iot/seguridad-en-iot

[3] Ilustración funcionamiento del Internet de las Cosas

http://www.businessinsider.com/what-is-the-internet-of-things-definition-2016-8

[4] Análisis del protocolo ZigBee – Jorge Pablo Dignani

http://postgrado.info.unlp.edu.ar/Carreras/Especializaciones/Redes_y_Seguridad/Trabajos_Finales/Dignanni_Jorge_Pablo.pdf

[5] Carrier sense multiple access with collision

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

[6] Digi International, Inc.

https://www.digi.com/lp/xbee

[7] XBee Series 1 vs XBee Series 2

http://icircuit.net/xbee-series-1-vs-series-2/289

[8] Product Manual XBee® ZNet 2.5/XBee-PRO® ZNet 2.5 OEM RF Modules – Digi International Inc.

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

[9] Modos de funcionamiento módulos XBee.

http://fuenteabierta.teubi.co/2014/03/arduino-y-el-xbee-series-1-modo-api.html

[10] Passport Documentation

http://passportjs.org/

[11] Materialize Documentation

http://materializecss.com/

[12] Bricogeek - Tienda de components electrónicos

http://tienda.bricogeek.com/

[13] Repositorio Github del Proyecto

https://github.com/ULL-InformaticaIndustrial-Empotrados/TFG_Juan_Hita

[14] Metodología FDD (Feature Driven Development / Desarrollo Basado en Funciones)

http://metodologiafdd.blogspot.com.es

[15] Desarrollo basado en funcionalides

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

[16] Desarrollo agil del software

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

[17] Definición de Websocket

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

[18] Modelo vista controlador (MVC)

https://fernetjs.com/2012/02/estructura-de-un-sitio-web-mvc-en-nodejs/

https://es.wikipedia.org/wiki/Modelo–vista–controlador

[19] Material Design of Google

http://www.vbote.com/vbote-solutions-academy-blog/130-materialize-el-framework-con-principios-de-material-design.html

[20] jQuery

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

https://jquery.com

[21] JSON

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

[22] Tornado Framework Websocket

http://www.tornadoweb.org/en/stable/websocket.html