Universidad de La Laguna

Grado en Ingeniería Electrónica Industrial y Automática

TRABAJO FIN DE GRADO

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

Autor: Eduardo Alfonso Rey Hamilton
Tutor: Alberto Hamilton Castro

Índice

Índice de figuras

Índice de tablas

1. Introducción

1.1. Resumen

Este proyecto trata el conexionado y configuración de una red domótica de bajo consumo basada en el protocolo inalámbrico ZigBee. Los dispositivos utilizados serán los XBee de la compañía Digi International ya que las características de dichos dispositivos nos permiten ahorrarnos el colocar microcontroladores en cada uno de los nodos lo que reduce drásticamente el coste económico de la red y el consumo eléctrico. De este modo el control queda centralizado en un nodo coordinador lo que simplifica la reconfiguración al estar todo programado en un único controlador y no en varios microcontroladores.

Se usará un ordenador, con un sistema GNU/Linux, que pondrá en funcionamiento la red y será el encargado de realizar las labores de control del sistema por medio de un programa en C++ que permita al usuario iniciar el sistema y realizar y modificar los programas o configuraciones que se desee.

En nuestro caso, la viabilidad se ha demostrado realizando una red que representa algunas habitaciones de mi propia vivienda y dónde, por medio de sensores digitales y analógicos así como actuadores digitales, se han cubierto necesidades reales existentes; diseñando el software de control y los circuitos electrónicos.

1.2. Abstract

The project involves the connection and configuration of a low_power home automation network using the ZigBee wireless protocol. Digi International's XBee devices have been used since the characteristics of these devices allow us to save the placement of microcontrollers in each of the nodes, which drastically reduces the economic cost of the network and the electric consumption. In this way the control is centralized in a coordinating node which simplifies the reconfiguration since everything is programmed in a single controller and not in several microcontrollers.

A computer will be used, with a GNU/Linux system, which will start up the network and will be in charge of carrying out the control tasks of the system through a program that allows the user to start the system and make and modify the desired programs or configurations.

In our case, viability has been demonstrated by realizing a network that represents some rooms in my own house. Designing control software and electronic circuits, existing real needs have been covered through digital and analog sensors and actuators.

1.3. Introducción

El presente documento versará sobre el desarrollo del Trabajo de Fin de Grado TFG03 *"Desarrollo de un sistema controlador para red domótica inalámbrica basada en protocolo ZigBee"* del Grado en Ingeniería Electrónica Industrial y Automática, cuyo fin es emplear el protocolo de comunicación ZigBee en una red de sensores y actuadores que nos permita simular algunas zonas de una vivienda.

Según la RAE, la domótica se define como un conjunto de sistemas que permiten automatizar las distintas instalaciones de una vivienda. El desarrollo de la domótica ha sido gracias a los grandes avances en lo que a tecnologías de la información se refiere. La invención de los microcontroladores y su producción en masa simplificó y abarató la instalación de estos sistemas lo que provocó que fueran cada vez más comunes.

Figura 1 Estructura de un sistema domótico [1]

Dichos sistemas domóticos se comunican a través de un nodo central que recibe datos del entorno mediante sensores y que coordina consecuentemente todas las operaciones que realiza la instalación. Los servicios que pueden ofrecer estos sistemas se dividen en cinco ámbitos principales:

- Ahorro energético: o lo que es lo mismo, gestión eficiente de los aparatos y sistemas del hogar. Si permitimos a un sistema domótico hacerse cargo de persianas, toldos y ventanas podríamos ahorrar mucha energía en el ajuste de temperatura, humedad o luminosidad. Además se pueden desconectar aquellos dispositivos o aparatos que no requieran una alimentación constante.
- Comodidad y confort: interfaces remotas para el control de los sistemas tales como aplicaciones webs o apps para smartphones, automatización del encendido y apagado de las luces y el almacenamiento de configuraciones, macros o programas personalizados aumentan la calidad de vida de las personas.
- Sistemas de seguridad: alarmas de intrusión, cierre de puertas y persianas, simulación de presencia, detectores de incendios o gas. Todos estos sistemas de seguridad protegen tanto a las personas como a los bienes materiales.
- Comunicaciones: los sistemas domóticos tienen la necesidad de estar interconectados todo el tiempo, es por ello que es sencillo añadir un método para interactuar remotamente. Con ello podemos revisar el estado del sistema e incluso interactuar con él sin estar físicamente en la vivienda.
- Accesibilidad: se incluyen las aplicaciones o instalaciones de control remoto del entorno que favorecen la autonomía personal de personas con limitaciones funcionales o discapacidad. La calidad de vida de estas personas es mejorada gracias a sistemas automatizados que les permitan ser autónomos.

La mayor debilidad de la domótica es el consumo energético y su consiguiente repercusión económica que no todas las familias o negocios pueden afrontar. Pensando en esto han crecido los sistemas y protocolos de comunicación que permiten realizar instalaciones de baja coste, consumo y complejidad, como ZigBee.

Este trabajo tiene como objetivo utilizar el protocolo ZigBee para simular la instalación de algunas habitaciones de mi vivienda sin usar componentes adicionales para la comunicación a los módulos XBee.

2. Conocimientos previos

Antes del desarrollo y la implementación tuve que adquirir unos conocimientos previos sobre el protocolo ZigBee y los módulos XBee Series 2 y para ello accedí, principalmente, a tres documentos. En primer lugar, *XBee ZNet 2.5/XBee PRO ZNet 2.5 OEM RF Modules* [4], el manual de la empresa Digi International para los módulos XBee. En segundo lugar, el Trabajo Fin de Grado de Javier Perez Martín: *Desarrollo de un sistema controlador para una red domótica inalámbrica* [6]. En tercer lugar, el Trabajo Fin de Grado de Gustavo Alejandro Lafranconi Peña: *Diseño de varios sensores y actuadores domóticos que utilicen comunicación inalámbrica ZigBee* [7].

Pudiese resultar que los contenidos de este (y sólo de este) apartado y de los compañeros anteriormente nombrados sean muy parecidos ya que todos provienen de la misma fuente: el manual del dispositivo facilitado por el tutor.

2.1. Protocolo ZigBee

2.1.1. Introducción

El protocolo de comunicaciones ZigBee está basado en el estándar de comunicaciones para WPAN (Wireless Personal Area Network) IEEE 802.15.4 y fue creado por ZigBee Alliance, una organización altruista. La funcionalidad de este protocolo se basa en la facilidad para comunicar inalámbricamente dispositivos electrónicos de bajo consumo, de un modo seguro y con una tasa de envío de datos baja. El bajo consumo eléctrico de estos dispositivos, la topología en malla y su sencilla integración electrónica viabilizan su uso en la domótica.

Dicho protocolo nació en 1998 respondiendo a la imposibilidad del uso de los protocolos WiFi y Bluetooth en todas las aplicaciones requeridas. Es entonces cuando un grupo de empresas del sector crearon ZigBee Alliance, organización que agrupa ya a más de 400 empresas. ZigBee Alliance impulsó la creación del protocolo diseñándolo para las redes inalámbricas de bajo coste y baja potencia.

La banda de 2.4 GHz es la elegida para realizar las comunicaciones lo que permite que las transmisiones tengan una velocidad de 256 Kb/s y una red pueda llegar a contener 65535 equipo. La antena dipolo tiene un alcance natural de 40 metros en interiores y 100 metros en el exterior, distancia más que suficiente para proyectos inalámbricos de domótica.

2.1.2. Características

Las redes ZigBee nos permite ahorrarnos el esfuerzo de cablear gracias a su comunicación serial inalámbrica. Esto también nos permite reubicar actuadores o sensores fácilmente. Estas redes cuentan con hasta tres tipos de elementos: un coordinador, uno o más routers y uno o más dispositivos finales.

- Coordinador: Es el dispositivo de mayor importancia ya que es el encargado de la creación de la red y la gestión de las direcciones. También se encarga de crear el canal de comunicaciones y fijar el identificador de la PAN *(Personal Area Network)*. Además, el coordinador puede funcionar también como router.
- Routers: Estos dispositivos se ocupan de la comunicación entre todos los dispositivos de la red. Los routers deben elegir cuál es la mejor ruta para comunicar dos puntos lo suficientemente alejados como para imposibilitar su conexión directa y gestionar dicha comunicación. Dichos dispositivos suelen estar alimentados por una toma de corriente ya que su funcionamiento es imprescindible y es necesario que estén siempre funcionando.
- Dispositivo final: Son las terminaciones de la red. Estos dispositivos sólo se encargan de recibir y enviar información referente a su estado o el de sus sensores a través de su nodo padre (un router). Ya que no son parte de la comunicación de la red, dependiendo de su uso, podría no ser necesario que estuviesen activos constantemente, permitiendo así entrar en modo ahorro o sleep.

2.1.3. Topologías de red

Los XBee tienen la capacidad de comunicarse gracias al direccionamiento de dispositivos. Este direccionamiento consta de dos direcciones: una de 64 bits y otra de 16 bits. La dirección de 64 bits es única e intransferible y asignada de fábrica. La dirección de 16 bits es asignada por el coordinador al nodo cuando este se une a la red. Estas direcciones se pueden utilizar para los algoritmos de ruteo y son únicas en toda la red lo que provoca que se abarque una menor cantidad de nodos. Haciendo uso de estas direcciones, únicas en cada dispositivo, podrán crearse diferentes rutas para interconectar los nodos.

Para diferenciar las distintas PAN que puedan coincidir en un mismo sitio, cada una de ellas tiene una dirección distinta. De esta forma no habrá conflicto ni error entre los dispositivos de dos PAN distintas.

El protocolo ZigBee trabaja con varias topologías de red para interconectar los nodos existentes: estrella, árbol y malla.

- Topología en estrella *(Star)*: En esta topología todos los dispositivos son finales excepto el coordinador que se encuentra en el centro de la red y todas las comunicaciones serán a través de él. El punto débil de esta topología es que si fallase el punto central se colapsaría toda la red.
- Topología en árbol *(Cluster tree)*: Se podría decir que la topología en árbol es una serie de redes en estrella interconectadas lo cual provoca que no exista un nodo central. Una serie de routers, con sus respectivos dispositivos finales, son conectados al coordinador que también puede tener dispositivos finales conectas a él. El inconveniente de esta red es que si alguno de los routers falla dejaría la red parcialmente inhabilitada.
- Topología en malla *(Mesh)*: Esta variante es la más fiable ya que cada nodo están conectados a varios nodos y no a uno sólo de modo que, si uno de los routers fallase, el coordinador establecerá una nueva ruta para que la comunicación no se interrumpa. Además, gracias a que es una red inalámbrica, supera la principal desventaja de esta topología: el excesivo cableado.

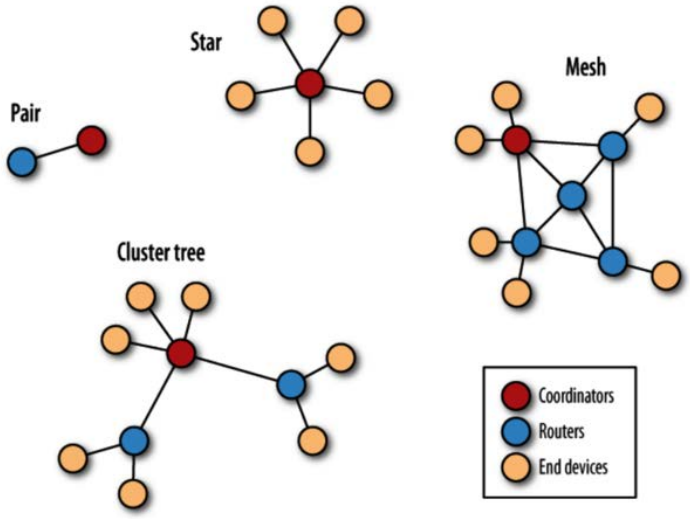


Figura 2 Tipologías de una red ZigBee [2]

2.2. XBee Series 2

2.2.1. Introducción

Los módulos XBee Series 2 son los elegidos para el desarrollo de este trabajo ya que se adaptan a las necesidades del mismo. Estos dispositivos tienen la capacidad de leer los sensores (digitales y analógicos) directamente y transmitir dicha información a su nodo padre por sí mismo sin la ayuda de un tercero. Del mismo modo, poseen salidas PWM, digitales y analógicas. Las ventajas de estos módulos son varias:

- Su simplicidad, tamaño y peso reducidos nos permiten emplearlos en una gran variedad de situaciones.
- Ahorro económico al no necesitar microcontroladores adicionales.
- Bajo consumo eléctrico.

Estos módulos también tienen sus contras. La lógica es totalmente ajena a estos módulos y las lleva el ordenador (o microcontrolador) que está conectado al coordinador. Estos módulos simplemente envían y reciben datos, y cambian remotamente el estado de sus pines. Además, los módulos no son ampliables, es decir, el número de entradas y salidas es limitado.

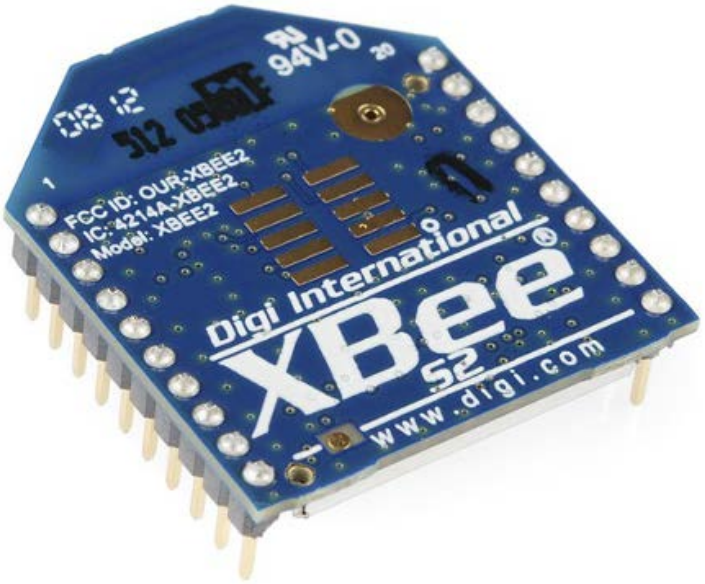


Figura 3 Módulo XBee Series 2 [3]

2.2.2. Características

Las características principales de los XBee Series 2 son:

- Características físicas: Tienen un tamaño de 2,438 cm x 2,761 cm y una temperatura de operación entre los -40 ºC y los 85 ºC.
- Desempeño: El ratio de transmisión en radiofrecuencia de 250 Kb/s y el ratio de transmisión de datos seriales de entre 1200 y 230400 bps. El alcance en interiores es de hasta 40 m, alcanzando los 100 m en exteriores y con línea de visión directa.
- Requisitos de energía: Tienen un bajo consumo eléctrico ya que solicitan una corriente máxima de 40 mA a 3.3 V. Estando encendido y sin realizar una tarea consume 15 mA, pero existe la posibilidad de configurar dicho módulo para entrar en modo hibernación donde tendrá un consumo inferior a 1 µA.
- Redes y seguridad: Crean redes de alta seguridad. La comunicación entre estos dispositivos tiene procesos de confirmación de recepción, reintento, múltiples topologías y múltiples direcciones.
- Uso sencillo. Los modos transparente y API son los modos para enviar configuraciones y órdenes AT.

Los dispositivos XBee tienen una serie de firmwares para asignar un rol a cada dispositivo y configurarlo como sea necesario. Esta configuración consiste en una

definición de parámetros tales como el alias del dispositivo, el estado de los pines, muestreos de señale, etc. Los cambios de dichos parámetros se gestionarán a través de comandos AT.

2.2.3. Dispositivo físico y patillaje

El módulo XBee que emplearemos será, más concretamente, un XBee ZNet 2.5 OEM RF Module. La disposición de los 20 pines de estos módulos se puede observar en la Figura 4 [5] y la descripción de las funciones de cada uno de ellos se describen en la Tabla 1 [7]. Las señales activas a nivel bajo están subrayadas.

Tabla 1. Asignación de pines para el XBee ZNet 2.5			
Nº pin	Nombre	Tipo	Descripción
1	VCC	-	Fuente de alimentación
2	DOUT	Salida	Salida de datos UART
3	DIN / CONFIG	Entrada	Entrada de datos UART
4	DIO12	Ambos	E/S digital 12
5	RESET	Entrada	Reseteo módulo (pulso mín. 200 ns)
6	PWM0 / RSSI / DIO10	Ambos	PWM salida 0 / Indicador fuerza señal RSSI / E/S digital 10
7	PWM / DIO11	Ambos	E/S digital 11
8	[reservado]	-	No conectar
9	DTR / SLEEP_RQ / DIO8	Ambos	Control de hibernación por pin/ E/S digital 8
10	GND	-	Tierra
11	DIO4	Ambos	E/S digital 4
12	CTS / DIO7	Ambos	Control 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	Ambos	Indicador asociado / E/S digital 5
16	RTS / DIO6	Ambos	Control flujo Request-to-send / E/S digital 6
17	AD3 / DIO3	Ambos	Entrada analógica 3 / E/S digital 3
18	AD2 / DIO2	Ambos	Entrada analógica 2 / E/S digital 2
19	AD1 / DIO1	Ambos	Entrada analógica 1 / E/S digital 1
20	AD0 / DIO0 / Botón de puesta en marcha	Ambos	Entrada analógica 0 / E/S digital 0 / Botón de comisionado

Tabla 1 Asignación de pines para el XBee ZNet 2.5 [7]

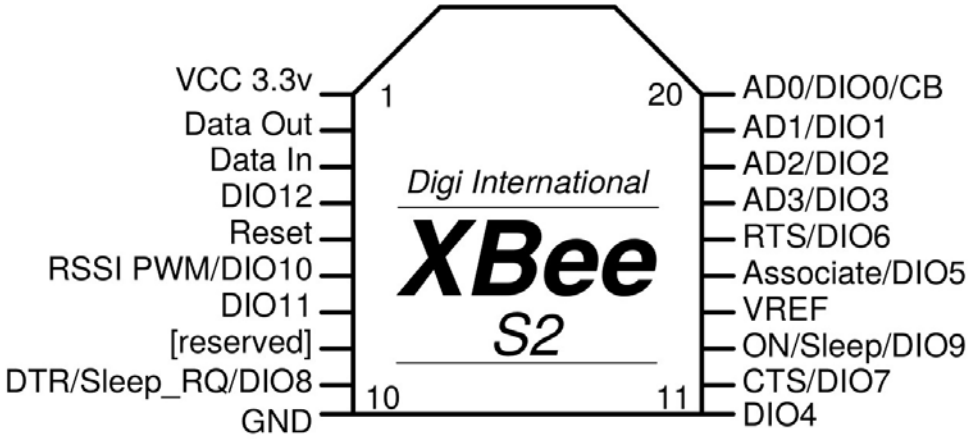


Figura 4 Patillaje de un módulo XBee (vista superior) [5]

2.2.4. Comunicaciones seriales

El XBee se comunica con su anfitrión a través de un puerto serial asíncrono. Este módulo tiene la capacidad de comunicarse con cualquier UART compatible en lógica y voltaje a través de un traductor de niveles o bien mediante su puerto serial.

Para gestionar la comunicación entre este módulo y su anfitrión a través de sus pines seriales existen dos protocolos distintos: transparente y API *(Application Programming Interface)*. Los dispositivos XBee intercomunicados en una misma red pueden utilizar protocolos distintos ya que estos modos sólo afectan a la comunicación serial del nodo y no a la comunicación con otros nodos.

En el protocolo de comunicación transparente, los módulos se comportan como si de una línea serial se tratase. Los datos recibidos por medio del pin DIN serán transmitidos vía radiofrecuencia. Por otro lado, si se recibiese por radiofrecuencia algún dato se enviará este por el pin DOUT. Para poder trabajar en modo transparente los módulos deben tener el firmware de versiones 1.0xx (coordinador) o 1.2xx (router/dispositivo final).

El modo transparente no se usa en este trabajo mientras que sí el API. El protocolo API se verá en profundidad en el apartado 2.2.6.

2.2.5. Modos de operación

2.2.5.1. Modo Idle

Mientras no esté recibiendo o transmitiendo datos, el módulo estará en modo idle. Mientras esté en dicho modo, el módulo está comprobando los datos que llegan por

radiofrecuencia en busca de datos válidos. El módulo cambia a los demás modos de operación según las condiciones que se den.

2.2.5.2. Modo de transmisión

Cuando los datos seriales sean recibidos y estén listos para el empaquetado, el módulo entrará en este modo e intentará realizar una transmisión. Antes de transmitir los datos el módulo debe asegurarse que de la dirección de 16 bits y el ruteo han sido establecidos.

Si se ha recibido correctamente el paquete el nodo emisor recibirá un mensaje de confirmación por parte del nodo receptor. El paquete sería enviado nuevamente por el emisor si dicho mensaje de confirmación no se recibiese. Dado que el XBee no tiene una función para eliminar paquetes duplicados podría incurrir en error si se diese el caso de que aun habiéndose recibido el paquete no le llegue al emisor el mensaje de confirmación.

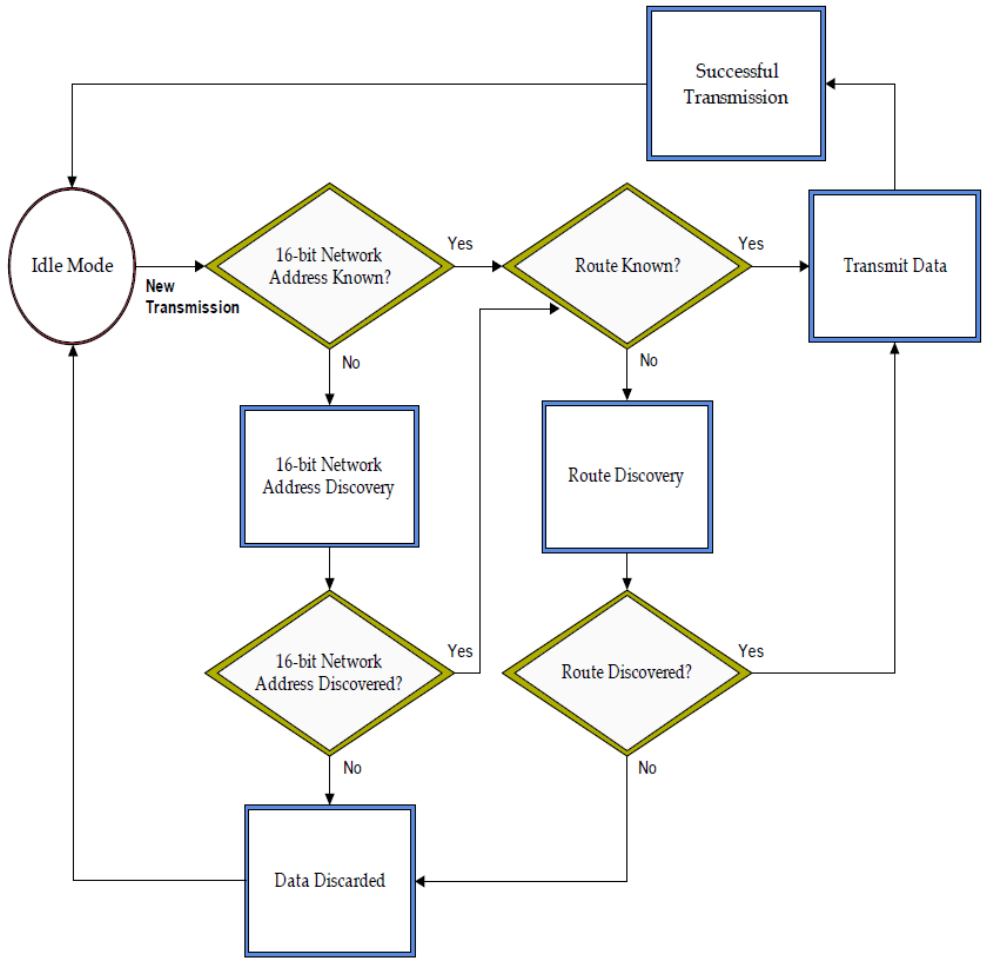


Figura 5 Secuencia del Modo de Transmisión [4]

2.2.5.3. Modo de recepción

Si se recibe un paquete válido este se transmite al búfer de transmisión serial.

2.2.5.4. Modo de comando

Para leer o modificar los parámetros del módulo este debe entrar primero en el modo comando. El modo comando es un estado en el que los caracteres seriales recibidos son interpretados como comandos. Estos comandos se pueden enviar o bien en modo transparente (del que ya hemos hablado) o bien en modo API.

2.2.5.5. Modo de hibernación (sleep)

El modo sleep permite al módulo entrar en un estado de bajo consumo cuando no está en uso. Este módulo permite tanto entrar en modo por transiciones del pin habilitado para ello como hacerlo cíclicamente.

2.2.6. Protocolo API

El uso de un API (*Application Programming Interface,* interfaz de programación de aplicaciones) basada en tramas, es una alternativa al modo transparente que con las capacidades de red del módulo es capaz de ampliar el nivel de interacción de una aplicación anfitriona. En este modo todos los datos entrantes y salientes se agrupan en tramas que se establecen como eventos u operaciones dentro del módulo. En vez de usar el modo de comandos para modificar direcciones, las tramas de datos que tienen la dirección (u otra información necesaria) pueden ser enviadas por una aplicación externa y el módulo responderá enviando otra trama con paquetes de estado entre otros datos. Para poder trabajar en modo API los módulos deben tener el firmware de versiones 1.1xx (coordinador) o 1.3xx (router/dispositivo final).

Al comparar el modo API con el modo transparente aparece múltiples ventajas:

- Modos adicionales a los comandos AT. Por ejemplo, la lectura de los estados de entradas y salidas de nodos remotos; facilitando, así, la creación de una red de sensores.
- Configuración de nodos a distancia.
- Capacidad de envío de mensajes en modo *broadcast* y un direccionamiento mucho más rápido de mensajes.
- Confirmación de la recepción de paquetes enviados mediante una respuesta y, si dicha respuesta no es recibida, reenvío del paquete.

- CRC para comprobar la integridad de los mensajes y filtrar posibles errores.

Las funciones expuestas (entre otras) son el motivo de que el modo API sea el escogido para nuestra implementación, pero la principal ventaja es el hecho de que únicamente en el modo API se pueden configurar, medir y modificar los pines de los nodos remotos, cosas que no podrían llevarse a cabo con el modo transparente.

La estructura general de una trama API es la mostrada en la Figura 6:

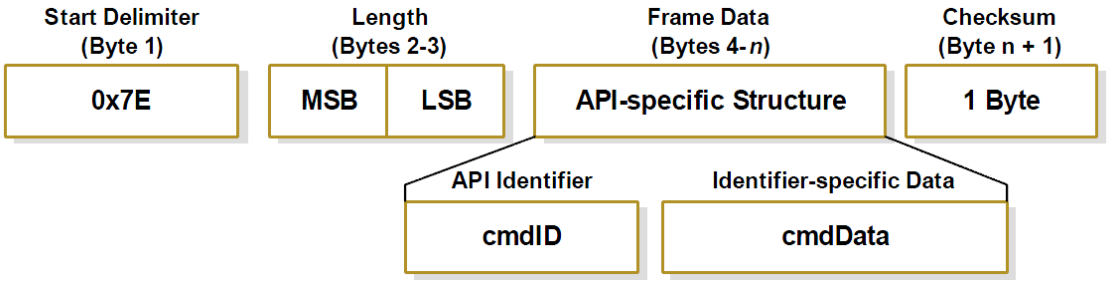


Figura 6 Estructura de un frame en modo API. [4]

En el campo de longitud se informa del tamaño (en forma de bytes) que tiene la estructura específica completa (sin contar el CRC) y tiene un tamaño de dos bytes. Dicha estructura comienza con la ID del comando a ejecutar. El *checksum* o CRC es el byte final del paquete que permite comprobar la integridad de los mensajes.

Todos los paquetes que se trabajan en el modo API se identifican con una ID. La Tabla 2 [7] recoge todos los IDs de los comandos.

Tabla 2. Nombres y valores de las tramas API [7]	
Nombre de la trama	Valor
Estado del módem	0x8A
Comando AT	0x08
Comando AT – Poner en cola valor del parámetro	0x09
Respuesta a comando AT	0x88
Solicitud de comando remoto	0x17
Respuesta a comando remoto	0x97
Petición de transmisión ZigBee	0x10
Trama de comando de direccionamiento explícito de ZigBee	0x11
Estado de transmisión de ZigBee	0x8B
Recepción de paquete ZigBee(AO=0)	0x90
Indicador de Rx explícito ZigBee (AO=1)	0x91
Indicador de muestreo de datos Rx de E/S ZigBee	0x92
Indicador de lectura de sensor XBee (AO=0)	0x94
Indicador de identificación de nodo (AO=0)	0x95

Tabla 2 Nombres y valores de las tramas API. [7]

Estas son las tramas más empleadas en este trabajo:

- 0x08 - Comando AT. La trama 0x08 nos permite enviar un comando AT al módulo conectado a través de la conexión serial, por lo que no necesitamos un campo de direccionamiento. El mensaje de confirmación por parte del módulo (si se requiriese) correspondería a la trama 0x88 – Respuesta a comando AT.
- 0x17 - Solicitud de comando remoto. Esta trama es una de las más usadas e importantes ya que nos permite enviar un comando AT pero a un módulo remoto parte de la red. En este caso sí necesitamos direccionamiento. Como sucedía en la trama anterior, se puede requerir un mensaje de confirmación de la recepción del comando 0x17 que sería el 0x97 – Respuesta a comando remoto.

- 0x92 – Indicador de muestreo de datos de E/S. Podríamos decir que esta trama es la más importante para nuestro trabajo ya que es la encargada de leer el estado de las entradas y salidas digitales de un nodo remoto. Esta trama se puede requerir mediante un comando AT o se puede configurar para que se envíe automáticamente cuando haya algún cambio en el estado de una entrada.

2.2.7. Modo de comandos

En el protocolo transparente, para poder pasar al modo de comandos debemos introducir la secuencia "+++" por terminal. En los parámetros hay especificado un tiempo que, si se cumple, provocará la salida de dicho modo. El módulo responderá con el mensaje "OK" si se completa la secuencia correctamente.

Los comandos AT empiezan con las letras "AT". Seguidamente se encuentran dos caracteres que indican el comando a emplear y, si son necesarios, otros parámetros del comando. La sintaxis general se ve en la Figura 7, donde el retorno de carro *(carriage return)* corresponde a presionar la tecla ENTER [4].

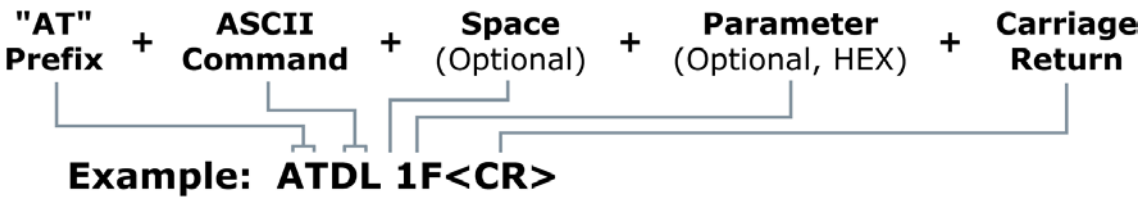


Figura 7 Sintaxis para enviar comandos AT [4]

Algunos de los comandos AT más usados en nuestro trabajo se ven en la Tabla 3 [4, pag 74].

Tabla 3. Comandos AT (funcionalidad). [4]

Comando	Nombre y descripción	Rango de parámetros	Valor predeterminado
WR	Escribir. Escribe parámetros a la memoria no volátil, conservando los cambios a pesar de reseteos posteriores.	-	-
MY	Dirección de red de 16 bits. Recibe la dirección de red de 16 bits del dispositivo.	0 – 0xFFFE (sólo lectura)	0xFFFE

NI	Identificador de nodo. Almacena un identificador en forma de cadena de caracteres. El registro sólo acepta caracteres ASCII imprimibles. Un <CR> o llegar al límite de caracteres termina el comando.	Cadena de caracteres ASCII de 20 bits	Espacio de caracteres ASCII (0x20)
ND	Descubrir nodo. Descubre e informa de todos los módulos de RF descubiertos. Se informa de la dirección de 64 bits, la de red de 16 bits, el identificador de nodo, el tipo de dispositivo, etc.	Valor de NI o MY opcional	-
IC	Detección de cambios en E/S digitales. Establece o lee los pines digitales a monitorizar para detectar cambios en el estado lógico, emitiendo un mensaje cuando suceda. IC es una máscara de bits que permite especificar los pines a monitorizar.	0 – 0xFFFF	0
IS	Fuerza la lectura de todas las entradas analógicas y digitales habilitadas.	-	-
IR	Establece y lee la frecuencia de muestreo de las entradas y salidas para permitir un muestreo periódico. Para que el muestreo periódico este activado el IR debe ajustarse a un valor distinto de cero y al menos un pin del módulo debe tener funcionalidad de entrada/salida analógica o digital habilitada.	0 – 0xFFFF (ms)	0

Tabla 3 Comandos AT (funcionalidad) [4]0

Tabla 4. Comandos AT (configuración de pines). [4]			
Comando	Nombre y descripción	Rango de parámetros	Valor predeterminado
D0	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D0.	0-5 0 = Disabled 1 = Botón de identificación del nodo activado 2 = Entrada analógica 3 = Entrada digital 4 = Salida digital, baja 5 = Salida digital, alta	1
D1	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D1.	0, 2-5 0 = Disabled 2 = Entrada analógica 3 = Entrada digital 4 = Salida digital, baja 5 = Salida digital, alta	0

D2	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D2.	0, 2-5 0 = Disabled 2 = Entrada analógica 3 = Entrada digital 4 = Salida digital, baja 5 = Salida digital, alta	0
D3	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D3.	0, 2-5 0 = Disabled 2 = Entrada analógica 3 = Entrada digital 4 = Salida digital, baja 5 = Salida digital, alta	0
D5	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D5.	0 = Disabled 1 = Associated indication LED 3 = Entrada digital 4 = Salida digital, baja 5 = Salida digital, alta	0
P0	Configuración de entrada analógica o entrada/salida digital. Configura y lee la función del pin D10.	0 = Disabled 1 = RSSI PWM 3 = Entrada digital, monitorizada 4 = Salida digital, baja 5 = Salida digital, alta	

Tabla 4 Comandos AT (configuración de pines)

2.3. Sensores

Los sensores son objetos capaces de detectar magnitudes física, tales como intensidad lumínica, temperatura, distancia, etc.; o eléctricas, tales como resistencia eléctrica, capacidad eléctrica, tensión, etc.

2.3.1. Pulsador

Los dos pulsadores utilizados para este proyecto son genéricos y normalmente abiertos. Estos pulsadores nos permiten mandar pulsos a un pin de nuestro módulo XBee. Dicho pin deberá estar configurado en como entrada digital para que pueda leer dichos pulsos.

2.3.2. LM35

Los LM35 son dispositivos de precisión con circuitos integrados para la medición de temperaturas con un voltaje de salida linealmente proporcional a la temperatura en grados centígrados. El dispositivo LM35 tiene una ventaja con respecto a los sensores lineales calibrados en Kelvin, y es que no se requiere que el usuario substraiga un alto voltaje constantemente a la salida para obtener un escalado en grados centígrados. Este dispositivo no requiere ninguna calibración o recorte para proporcionar una precisión de ± ¼ °C a temperatura ambiente y ± ¾ °C en el rango de -55 ºC a 150 ºC. La baja impedancia a la salida, la salida lineal y la calibración inherentemente precisa del dispositivo LM35 hacen que la interconexión con los circuitos de lectura o control sea especialmente sencilla. Como el dispositivo LM35 sólo extrae 60 μA tiene un calentamiento muy bajo, inferior a 0,1 ºC. [8]

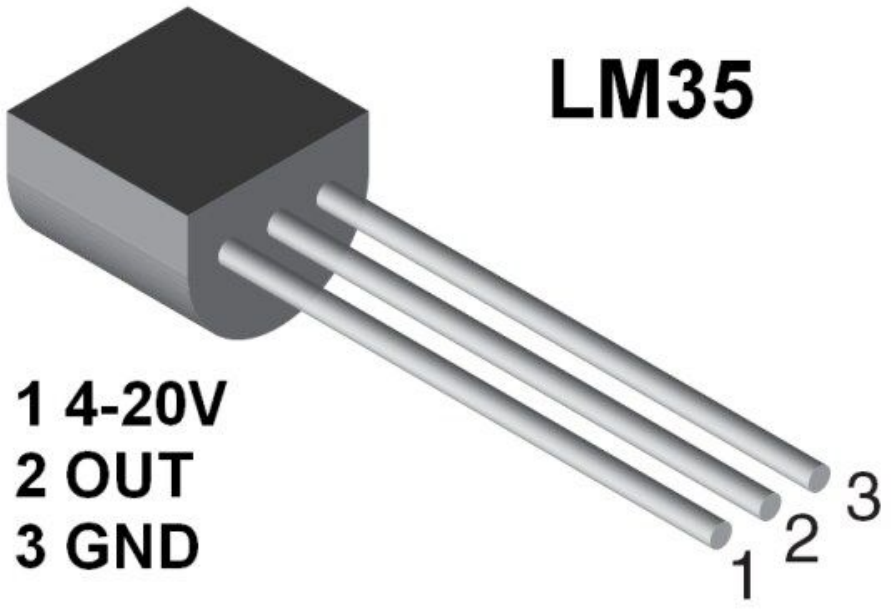


Figura 8 LM35 [9]

2.4. Actuadores

2.4.1. Leds

Los leds son componentes optoelectrónicos activos, más concretamente, diodos que emiten luz. Los cuatros leds usados en nuestro montaje son leds genéricos.

Figura 9 LEDs. [10]

2.4.2. Relé

Un relé es un dispositivo electromagnético que funciona como un interruptor, sólo que este interruptor está controlado por un circuito eléctrico por medio de una bobina y un electroimán. En nuestro caso usaremos el relé de Songle Relay SRD-05VDC-SL-C [11], cuyo datasheet se encuenta referenciado en la bibliografía.

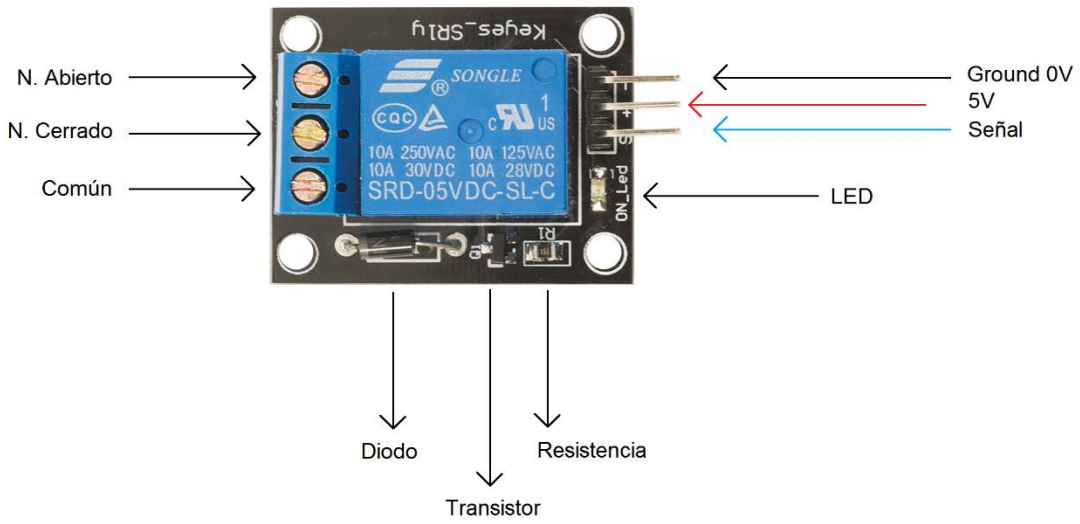


Figura 10 Relé SRD-05VDC-SL-C [12]

2.4.3. Puente H

Un puente en H es un circuito electrónico que nos permite hacer girar en ambos sentidos un motor eléctrico de corriente continua. Nuestro módulo está basado en el chip L298 [13] que nos permite controlar un motor paso a paso bipolar o dos motores de corriente continua. Este módulo tiene todos los componentes necesarios para funcionar por sí sólo.

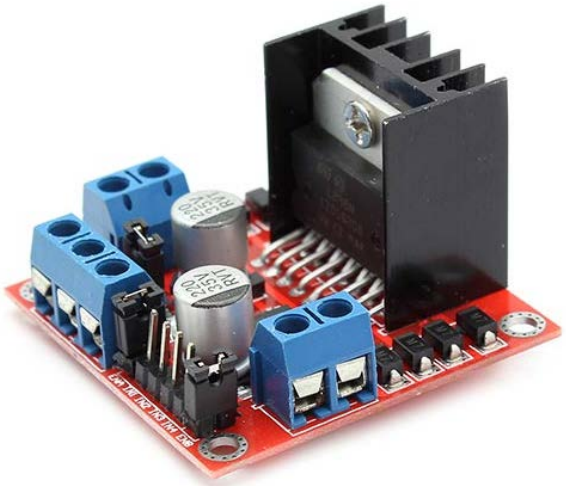


Figura 11 Doble Puente H PL298N [14]

3. Implementación

A continuación hablaremos sobre el desarrollo del TFG. Hablaremos del montaje y la programación de la lógica bajo la que debía trabajar toda la red.

3.1. Montaje

El montaje de este TFG consiste en un pequeño trabajo de domótica en el que he usado como ejemplo mi propia vivienda. He cubierto necesidades reales existentes en dos puntos importantes de la vivienda: La cocina y el dormitorio. Para ello nuestro montaje se divide en tres partes. La primera corresponde al XBee coordinador junto con mi ordenador personal. La segunda es circuito montado en una protoboard con un XBee que representa la cocina. Y la tercera es un segundo circuito montado en otra protoboard con un tercer XBee que simula el dormitorio.

Sobra decir que todo lo configurado en este montaje es totalmente reconfigurable. Número de pulsos, tiempos de espera, temperaturas límite, acciones a realizar cuando se cumplen qué condiciones, y un largo etc.

Tal y como hemos dicho durante el capítulo 2, todos los dispositivos XBee están interconectados. Esto implica que, aunque no sea así en nuestro montaje, los sensores de un XBee podrían provocar cambios en los actuadores de otro XBee remoto. Por ejemplo, podríamos tener un pulsador a la salida de la vivienda que apagase todas las luces, lo que activaría, no uno, sino varios XBee remotos.

3.1.1. Coordinador

Este XBee junto con mi ordenador personal son los encargados de coordinar y controlar el sistema. Para interconectar el XBee y el ordenador usaremos el XBee USB Explorer, que nos permite conectar un módulo XBee por USB a un ordenador para poder así utilizarlo. Esta placa se encarga de ajustar el voltaje de entrada a 3,3 V para el módulo XBee y también de hacer las veces de adaptador para poder acceder a los pines UART de nuestro módulo a través de una conexión USB.

Mi ordenador personal está en funcionamiento con un sistema operativo GNU/Linux con distribución XUbuntu. Su objetivo es ejecutar la lógica de cara a mantener el sistema funcionando correctamente. Esta lógica, escrita en C++, estará siendo ejecutada constantemente de modo que, cuando el XBee coordinador nos dé información

nueva, podamos devolverle las acciones pertinentes para que el sistema se comporte como deseamos.

El XBee coordinador está en continuamente a la espera de recibir información. En el momento que detecte un mensaje en su buffer de entrada pasará esta información al ordenador, el cuál ejecutará la lógica y le devolverá las órdenes pertinentes al XBee coordinador para que este lo envíe al XBee correspondiente (ya sea el de la cocina o el del dormitorio).

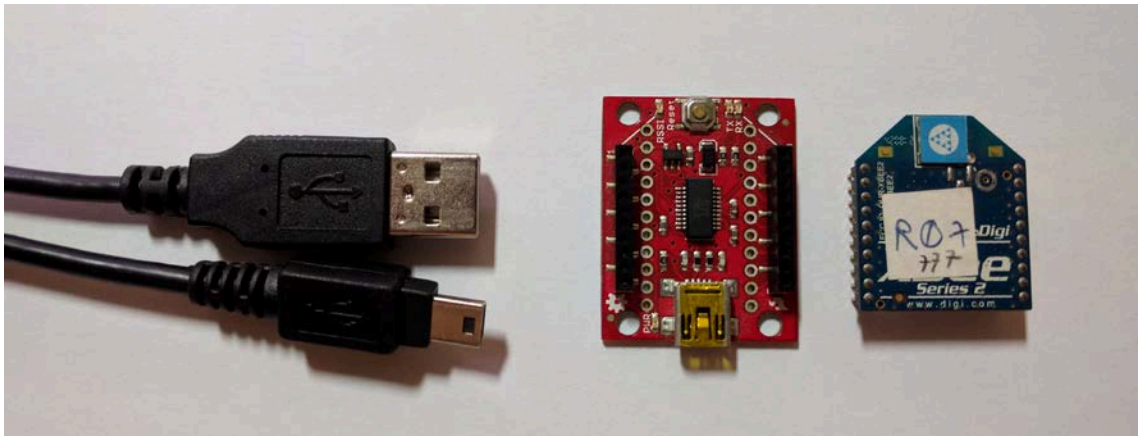


Figura 12 Circuito coordinador

3.1.2. Cocina

En la cocina se encuentran 3 actuadores, luz, extractor y alimentación de la placa vitrocerámica; y tan solo un sensor, un pulsador. La idea es simplificar la cantidad de botones e interruptores existentes y añadir un poco de seguridad. El pulsador está configurado para recibir hasta tres pulsos en un periodo de dos segundos o para ser pulsado ininterrumpidamente durante dos segundos. Las acciones que dichas interacciones causarán serán las siguientes:

- Un pulso: Se enciende o apaga la luz de la cocina.
- Dos pulsos: Se enciende o apaga el extractor situado sobre la placa vitrocerámica.
- Tres pulsos se alimenta (que no enciende) la placa vitrocerámica. De esta forma, al poder elegir si queremos alimentar o no la placa vitrocerámica podemos dotar de un nivel más de seguridad a nuestra cocina.
- Pulsador pulsado durante dos segundos: En este caso, como en todos los anteriores, pueden suceder dos cosas:

- Si tanto la luz como el extractor están apagados y la placa vitrocerámica sin alimentación. Se encenderá la luz y se alimentará la placa.
- Si la luz o el extractor están alimentados o la placa alimentada se apagará todo.

De este modo tenemos una forma de acortar tiempo a la hora de entrar a la cocina ya que no tendremos que dar un pulso, esperar a que se encienda la luz y luego otros dos pulsos; y a la hora de salir ya que no tendremos que preocuparnos por qué cosas están encendidas o no, simplemente mantenemos pulsado y todo se apagará.

En esta representación de la cocina tanto el estado de la luz como la alimentación de la placa vitrocerámica son representadas cada una por un led (amarillo para la luz y verde para la placa vitrocerámica), pero no así el extractor. Para el extractor he colocado un relé junto a un conector schucko aéreo para poder encender un ventilador de verdad y dar así una vista más próxima a lo que sería la instalación real. En la Figura 13 vemos una foto de la instalación real y en la Figura 14 podemos apreciar un esquema simbólico del circuito.

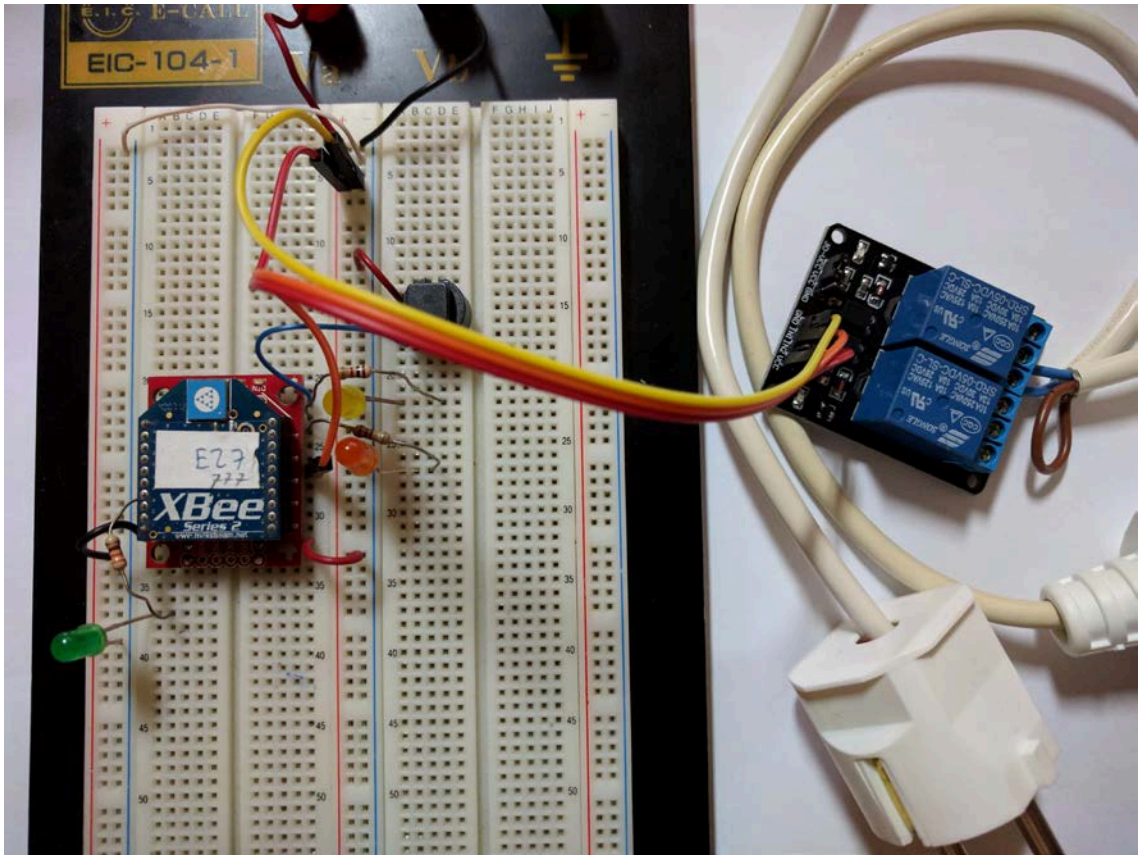


Figura 13 Circuito Cocina

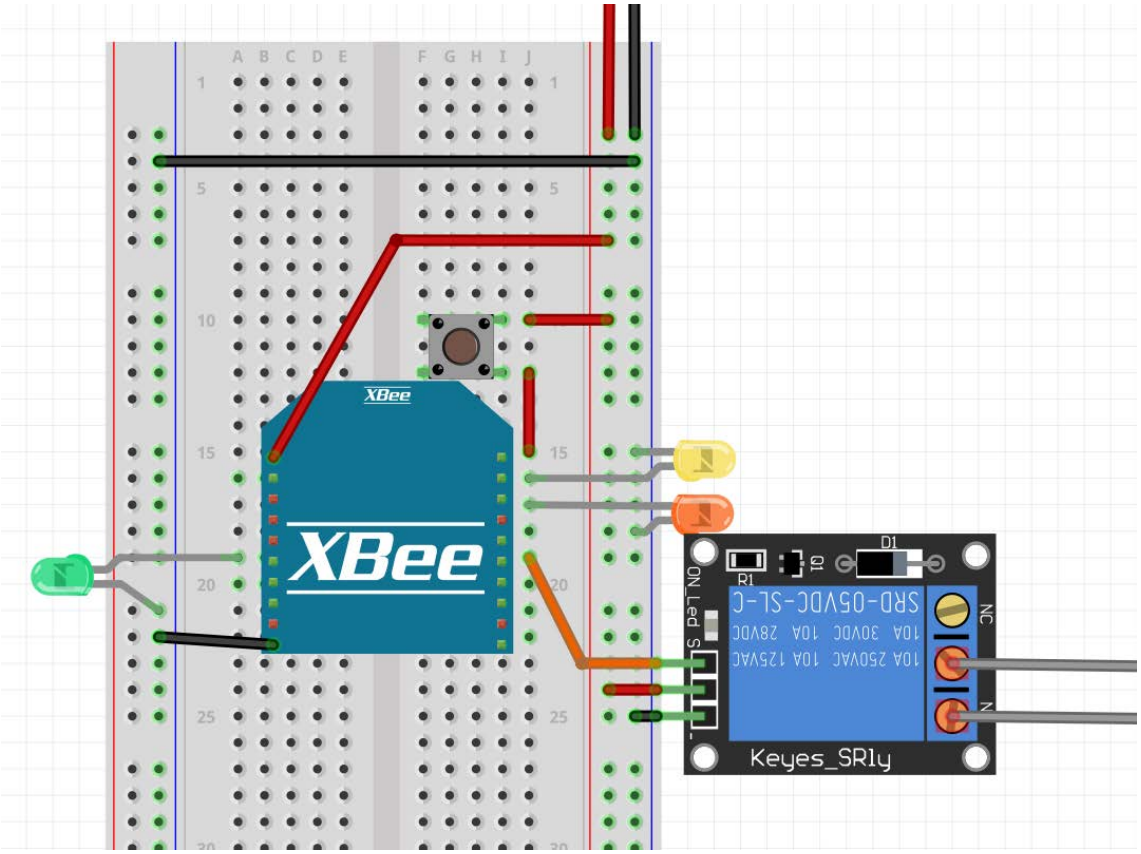


Figura 14 Circuito Cocina (Esquema)

En este montaje de la cocina también se encuentra un led (naranja) que nos indica cuando el sistema de control está funcionando ya que al iniciarlo puede tomarse unos segundos para identificar todos los módulos XBee que forman parte de la red e iniciar la ejecución de la lógica de control. Dicho led será inicializado apagado y una vez se entre en el bucle de control se encenderá para indicarnos así que el sistema está listo para ser utilizado. Este led podría colocarse en cualquier otro punto de la vivienda.

3.1.3. Dormitorio

El problema planteado en el dormitorio es que, al ser el último piso de la vivienda, pasa todo el día recibiendo la luz del Sol lo que provoca que el interior del dormitorio alcance temperaturas nada deseables. Lo que he diseñado es un pequeño montaje de control de temperatura que contiene un pulsador, dos sensores de temperatura, uno en el interior y otro en el exterior; y dos actuadores, un ventilador extractor y un aire acondicionado.

El pulsador servirá para encender o apagar este sistema de control de temperatura, mientras que los sensores externo e interno nos permitirán comparar la temperatura

interior y exterior. Conociendo estas dos temperaturas se pueden presentar 3 casos distintos:

- Que tanto la temperatura interior como la temperatura exterior excedan la temperatura máxima deseada. En tal caso se encenderá el aire acondicionado. Este se apagará cuando la temperatura llegue al punto medio entre la temperatura máxima y mínima deseadas.
- Que la temperatura interior exceda la temperatura máxima deseada pero no así la exterior. En tal caso se encenderá el ventilador extractor hasta que la temperatura en el interior sea igual o inferior a la exterior o hasta que la exterior exceda la temperatura máxima deseada, en cuyo caso se apagaría el ventilador y se reevaluaría la situación.
- Que la temperatura interior sea inferior a la temperatura máxima deseada. En tal caso no se hará nada independientemente de cuál sea el valor de la temperatura exterior.

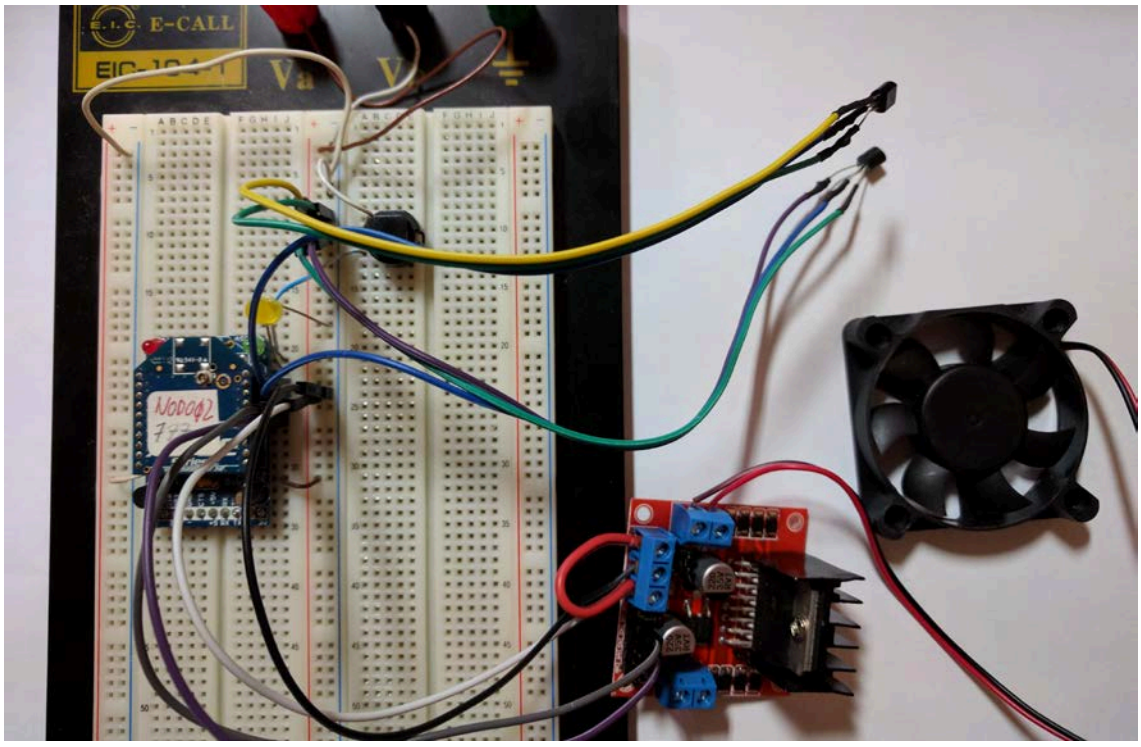


Figura 15 Circuito Dormitorio

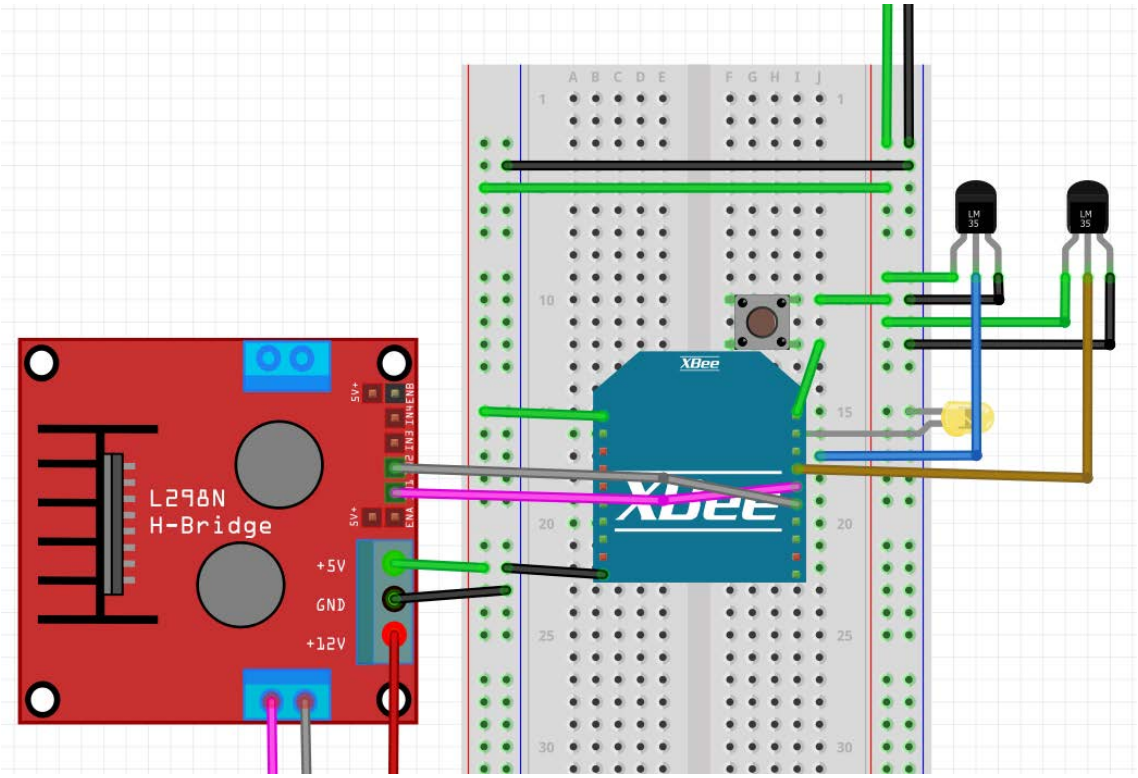
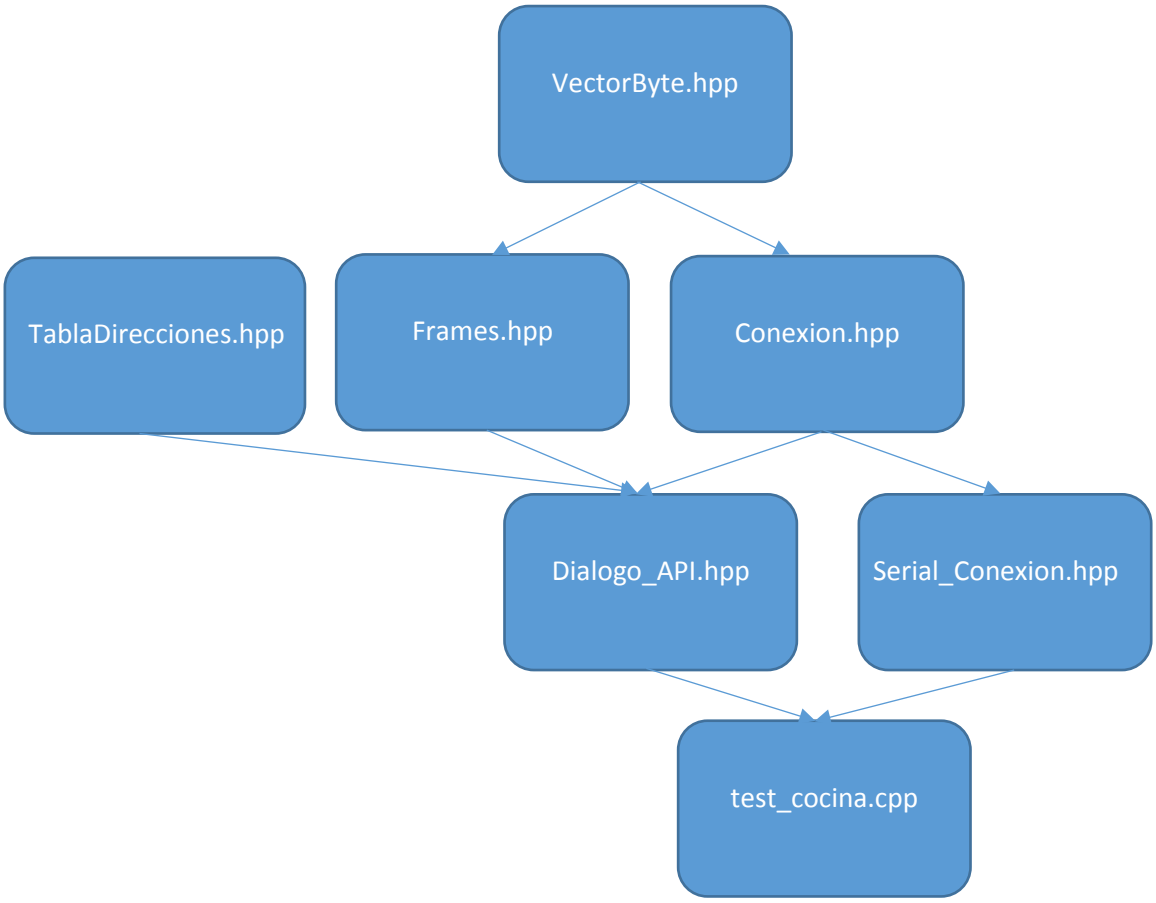


Figura 16 Circuito Dormitorio (Esquema)

En esta representación del dormitorio el led amarillo nos indica si el aire acondicionado está encendido o apagado. Para el extractor he colocado un puente H con un ventilador de 12 V de corriente continua. La instalación real de este componente sería muy parecida a la realizada para el extractor del dormitorio. Sería interesante tener en esta montaje un led que nos indicase si el control de temperatura está encendido o apagado pero no disponía de más leds así que me ha sido imposible llevarlo a cabo.

3.2. Programa

Este programa ha sido desarrollado en el lenguaje de programación C++. El tutor nos ha facilitado una librería de ficheros *.hpp*, que declaran las clases, y *.cpp*, que definen los métodos de estas, que se encargarán de la comunicación serial entre el ordenador y el XBee coordinador. La jerarquía de dichos ficheros es la siguiente:



A continuación procederemos a explicar todo el programa elaborado para este TFG. Para ello lo dividiremos en dos partes, una primera parte en la que hablaremos del uso de la librería por el tutor en nuestro programa y una segunda parte en la que hablaremos de la lógica del sistema desarrollada para que funcione todo como consideremos oportuno. Todos estos códigos se encuentran disponibles en la referencia catorce [14].

3.2.1. Ajustes y comunicación

En primer lugar tendremos que incluir en nuestro fichero *.cpp* los ficheros a utilizar.

```
29  #include "Serial_Conexion.hpp"
30  #include "Dialogo_API.hpp"
```

Figura 17 .hpp incluídos

A continuación declaramos la función “obtieneNombreNodo” el cual creará una tabla de direcciones llamada `td` con la información de los nodos de la red. Para ello le diremos al XBee coordinador, con el `comandoATlocal`, que ejecute la orden ND que, como ya dijimos en la Tabla 2, recogerá toda la información de los nodos de la red. Con

el `recibePAquete` y el `reparteMensaje` trataremos la información de los demás nodos recibida en el coordinador.

```
48  std::string obtieneNombreNodo(Dialogo_API &da){
49    TablaDirecciones td;
50    do {
51      std::cout << "\n\n";
52      da.comandoATlocal( "ND", true);
53      da.reparteMensaje( da.recibePaquete(true, 5000), true );
54
55      td = da.tablaDir();
56    } while(td.size() < numeroNODOS);
57
58    std::cout << "La tabla de direcciones es\n" << td.toString()
59        << std::endl;
60
61    return td.serial2ni(td.serialAt(0));
62  }
```

Figura 18 obtieneNombreNodo

El siguiente paso, gracias a la librería facilitada por el tutor, es la configuración de cada uno de los pines. Sólo enseñaremos 3 de ellos ya que enseñar todos convertiría esta parte en algo ilegible. Para dicha configuración necesitamos usar el comando `comandoATremoto` para mandar una orden a un nodo remoto. Este comando debe contener la orden y la dirección. En primer lugar utilizamos el comando IC, ya comentado en la Tabla 2), para dar la orden de que nos informe de cambios digitales en los pines que se requiera. Acompaña a dicho comando un uno que significa que sólo configuraremos de ese modo el D0. Posteriormente configuramos el pin D0 como una entrada digital tal y como vimos en la Tabla 4. De esta forma, con la combinación de IC1 y D03, el XBee remoto nos enviará una trama 0x92 cada vez que el pin D0 cambie su estado y así poder actuar en consecuencia. Por último configuramos el pin D1 como una salida digital a cero con el comando D14. A cada uno de estos comandos, como ya dijimos, debe acompañarlo la dirección (en este caso la serial, de 64 bits) del nodo remoto al que queramos enviar dicha orden. Bien puede ser `serialNodoE27` o `serialNodoNODO02`.

```
91      //Que nos avise de los cambios en D0
92      da.comandoATremoto("IC1", serialNodoE27, -1, debug);
93      da.reparteMensaje( da.recibePaquete(true, 5000), debug);
94
95      //D0 entrada digital (interruptor).
96      da.comandoATremoto("D03", serialNodoE27, -1, debug);
97      da.reparteMensaje( da.recibePaquete(true, 5000), debug);
98
99      //D1 salida digital (luz). Lo apagamos.
100     da.comandoATremoto("D14", serialNodoE27, -1, debug);
101     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
```

Figura 19 Configuración de los pines

A continuación, en la Tabla 5 podremos ver la configuración de todos y cada uno de los pines utilizados en este TFG.

Tabla 5. Configuración de pines.

Pin	Configuración	Conectado a…	Activa a…
Nodo E27			
D0	Entrada digital	Pulsador	Alta
D1	Salida digital	Led amarillo, luz de la cocina	Alta
D2	Salida digital	Led naranja, programa en funcionamiento	Alta
D5	Salida digital		Baja
D10	Salida digital	Led verde, placa vitrocerámica	Alta
Nodo NODO02			
D0	Entrada digital	Pulsador	Alta
D1	Salida digital	Led amarillo, aire acondicionado	Alta
D2	Entrada analógica	LM35, sensor de temperatura interior	-
D3	Entrada analógica	LM35, sensor de temperatura exterior	-
D5	Salida digital	Puente H y ventilador, extractor	Alta

Tabla 5 Configuración de pines

3.2.2. Lógica del sistema

Una vez realizada la configuración de los nodos, el programa se queda en un bucle infinito para atender los mensajes y actuar en consecuencia. Dentro de este bucle lo primero que hace el programa es comprobar la recepción de paquete. En caso de haber recibido algún paquete. Si ha recibido un paquete lo procesa para ver si es de tipo 0x92 (el que nos interesa) y en caso de que así sea mirando la dirección serial de 64 bits comprueba a cuál de los dos otros nodos proviene. Una vez sepamos de cuál de los dos nodos proviene podremos guardar dicha información en unas variables declaradas con

anterioridad. Cabe resaltar que del nodo de la cocina (E27) sólo se puede recibir información digital mientras que del nodo de la habitación (NODO02) podemos recibir tanto digital como analógica.

```
182 VectorByte paquete  = da.recibePaquete(debug, 100);
183 if (paquete.size() < 1) {
184   std::cout << "No se recibió paquete." << std::endl;
185 } else {
186   //tenemos paquete
187   if (Dialogo_API::getTipo(paquete) != 0x92) {
188     std::cout << "Paquete de tipo " << std::hex << Dialogo_API::getTipo(paquete)
189         << " no considerado por el momento." << paquete  << std::endl;
190   } else {
191     //es paquete 0x92, porcedemos a trocearlo
192     Frame92 f92 = da.trocea92(paquete);
193     if (f92.addr64 == serialNodoE27) {
194       //vemos si hay informacion de D0
195       if (f92.haveDigital(0)) {
196         valD0E27 = f92.getDigital(0);
197         std::cout << "E27. D0 está en valor " << valD0E27 << std::endl;
198       }
199     }
200 
201     if (f92.addr64 == serialNodoNODO02) {
202       //vemos si hay informacion de D0
203       if (f92.haveDigital(0) && (f92.getDigital(0) != valD0NODO02)) {
204         valD0NODO02 = f92.getDigital(0);
205         std::cout << "NODO02. D0 está en valor " << valD0NODO02 << std::endl;
206       } else if (f92.haveAnalog(2) || f92.haveAnalog(3)) {
207         tempint = f92.getAnalog(2);
208         tempint *= 100;
209         std::cout << "NODO02. La temperatura interior (D2) es de " << tempint << std::endl;
210         tempext = f92.getAnalog(3);
211         tempext *= 100;
212         std::cout << "NODO02. La temperatura exterior (D3) es de " << tempext << std::endl;
213       }
214     }
215   }
```

Figura 20 Recepción de información

Para gestionar la cuenta de pulsos usaremos un `switch` en el que, por defecto, estaremos en su etapa `a`. En dicha etapa estaremos a la espera de un primer pulso, en cuanto lo recibamos nos iremos a la etapa `b` y guardaremos la hora en la que ha tenido lugar dicho pulso ya que si pasan dos segundos tras ese primer pulso se encenderá o apagará la luz y volveremos a la etapa `a`.

```
219 switch (etapa) {
220   case 'a': //Etapa inicial donde esperamos al pulsador
221     std::cout << "E27. Etapa a. Esperamos un primer pulso." << std::endl;
222     if ((antD0E27==false) && (valD0E27==true)) {
223       std::cout << "E27. Etapa a. Primer flanco de subida en D0." << std::endl;
224       gettimeofday(&tvEnciende, NULL);
225       etapa = 'b';
226     }
227     break;
```

Figura 21 Cocina - Etapa a

La etapa b será muy parecida a la etapa a. Esperaremos la llegada del segundo pulso durante dos segundos por lo explicado en el párrafo anterior. Si recibiésemos un segundo pulso en ese tiempo nos iremos a la etapa c y guardaremos de nuevo la hora del pulso.

```
case 'b': { //Esperamos un segundo flanco o encendemos la luz tras 2 segundos
  std::cout << "E27. Etapa b. Esperamos un segundo pulso o encederemos la luz tras 2 seg." << std::endl;
  gettimeofday(&tvAhora, NULL);
  double tPasado = (tvAhora.tv_sec - tvEnciende.tv_sec) + ((tvAhora.tv_usec - tvEnciende.tv_usec)/1000000.0);
  std::cout << "E27. Etapa b. Hay tiempo y han pasado " << tPasado << "sg de los " << ESPERA << " necesarios."

  if ((tPasado > ESPERA) && (luz == 0)) {
    std::cout << "E27. Etapa b. Encendemos la luz (D1)." << std::endl;
    da.comandoATremoto("D15", serialNodoE27, -1, debug);
    da.reparteMensaje( da.recibePaquete(true, 5000), debug);
    luz = 1;
    tvEnciende.tv_sec = 0;
    tvEnciende.tv_usec = 0;
    etapa = 'a';
    break;
  }

  if ((tPasado > ESPERA) && (luz == 1)) {
    std::cout << "E27. Etapa b. Apagamos la luz (D1)." << std::endl;
    da.comandoATremoto("D14", serialNodoE27, -1, debug);
    da.reparteMensaje( da.recibePaquete(true, 5000), debug);
    luz = 0;
    tvEnciende.tv_sec = 0;
    tvEnciende.tv_usec = 0;
    etapa = 'a';
    break;
  }

  if ((antD0E27==false) && (valD0E27==true)) {
    std::cout << "E27. Etapa b. Segundo flanco de subida en D0." << std::endl;
    gettimeofday(&tvEnciende, NULL);
    etapa = 'c';
    break;
  }
}
break;
```

Figura 22 Cocina - Etapa b

La etapa c es idéntica a la etapa b pero con la diferencia de que si se sucede un tercer pulso se sucederá la acción (alimentar o dejar de alimentar la placa vitrocerámica) inmediatamente ya que no se debe esperar un cuarto pulso.

Como se ha comentado anteriormente también existe la funcionalidad encender la luz y alimentar la placa vitrocerámica o de apagarlo todo si se mantiene el pulsador durante dos segundos. Para dicha funcionalidad guardamos la hora cada vez que pulsamos el pulsador y la borramos cada vez que lo soltamos. Mientras tanto tenemos un if que comprueba constantemente si la diferencia entre la hora actual y la hora en la que se pulsó

(y no se ha soltado) el pulsador es igual o superior a dos segundos. En el momento que así sea sucederá la acción pertinente.

```
332 if ((antD0E27==true) && (valD0E27==false)){
333   gettimeofday(&tvMantener, NULL);
334 }
335
336 if ((antD0E27==false) && (valD0E27==true)){
337   tvMantener.tv_sec = 0;
338   tvMantener.tv_usec = 0;
339 }
340
341
342 // Vemos si se ha mantenido el pulsador 2 segundos
343 if(tvMantener.tv_sec  > 0) {
344   gettimeofday(&tvAhora, NULL);
345   double tPasado = (tvAhora.tv_sec - tvMantener.tv_sec) + ((tvAhora.tv_usec - tvMantener.tv_usec)/1000000.0);
346   std::cout << "E27. Manteniendo. Hay tiempo y han pasado " << tPasado << "sg de los " << ESPERA << " necesarios."
347
348   if ((tPasado > ESPERA) && (luz == 1 || estractor == 1 || pv == 1)) {
349     std::cout << "E27. Manteniendo. Apagamos todo." << std::endl;
350     da.comandoATremoto("P04", serialNodoE27, -1, debug);
351     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
352     luz = 0;
353     da.comandoATremoto("D55", serialNodoE27, -1, debug);
354     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
355     estractor = 0;
356     da.comandoATremoto("D14", serialNodoE27, -1, debug);
357     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
358     pv = 0;
359     tvEnciende.tv_sec = 0;
360     etapa = 'z';
361     tvMantener.tv_sec = 0;
362     tvMantener.tv_usec = 0;
363     tPasado = 0;
364   }
365
366   if (tPasado > ESPERA && luz == 0 && estractor == 0 && pv == 0) {
367     std::cout << "E27. Manteniendo. Encendemos luz y alimentamos placa vitrocerámica." << std::endl;
368     da.comandoATremoto("P05", serialNodoE27, -1, debug);
369     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
370     luz = 1;
371     da.comandoATremoto("D15", serialNodoE27, -1, debug);
372     da.reparteMensaje( da.recibePaquete(true, 5000), debug);
373     pv = 1;
374     tvEnciende.tv_sec = 0;
375     etapa = 'z';
376     tvMantener.tv_sec = 0;
377     tvMantener.tv_usec = 0;
378   }
379 }
```

Figura 23 Cocina - Pulso de dos segundos

La parte de programación referente al dormitorio también es basada en un switch pero mucho más sencillo ya que no hay que llevar un conteo de pulsos sino comprobar,

si está encendido el control de temperatura, las temperaturas interior y exterior y comprarlas entre ellas y con la temperatura límite y actuar en consecuencia tal y como se explicó en el punto 3.1.3. Dormitorio.

```
405 if (encendido == 1) {
406   switch (fase) {
407     case 'a': {
408       std::cout << "NODO02. Etapa a. Esperamos cambio de temperatura." << std::endl;
409       if ((tempint > TempMAX) && (tempext > TempMAX)) {
410         da.comandoATremoto("D15", serialNodoNODO02, -1, debug);
411         da.reparteMensaje( da.recibePaquete(true, 5000), debug);
412         std::cout << "NODO02. Encendemos el aire acondicionado. Tanto la temperatura interior como la exterior e
413         acondicionado = 1;
414         fase = 'b';
415       }
416
417       if ((tempint > TempMAX) && (tempext < TempMAX)) {
418         da.comandoATremoto("D55", serialNodoNODO02, -1, debug);
419         da.reparteMensaje( da.recibePaquete(true, 5000), debug);
420         std::cout << "NODO02. Encendemos el ventilador. La temperatura interior es superior a la máxima pero la t
421         ventilador = 1;
422         fase = 'c';
423       }
424       break;
425     }
426     case 'b': {
427       std::cout << "NODO02. Etapa b. Esperando que la temperatura interior alcance la teperatura media deseada."
428       if (tempint <= TempMED || (tempext <= TempMAX)) {
429         da.comandoATremoto("D14", serialNodoNODO02, -1, debug);
430         da.reparteMensaje( da.recibePaquete(true, 5000), debug);
431         std::cout << "NODO02. Apagamos el aire acondicionado. Hemos alcanzado la temperatura media deseada." << s
432         acondicionado = 0;
433         fase = 'a';
434       }
435       break;
436     }
437     case 'c': {
438       std::cout << "NODO02. Etapa c. Esperando que la temperatura interior descienda del máximo." << std::endl;
439       if ((tempint <= tempext) || (tempext < TempMAX)) {
440         da.comandoATremoto("D54", serialNodoNODO02, -1, debug);
441         da.reparteMensaje( da.recibePaquete(true, 5000), debug);
442         std::cout << "NODO02. Apagamos el ventilador. La temperatura interior es igual a la temperatura exterior.
443         ventilador = 0;
444         fase = 'a';
445       }
446       break;
447     }
448   }
449 } else {
450   std::cout << "NODO02. Control de temperatura apagado." << std::endl;
451 }
373 v = 1;
374 tvEnciende.tv_sec = 0;
375 etapa = 'z';
376 tvMantener.tv_sec = 0;
377 tvMantener.tv_usec = 0;
378
379
```

Figura 24 Dormitorio

3.2.3. Ejecución del programa

Una vez que ha sido todo el programa elaborado hay que proceder a compilar y ejecutarlo como indica la Figura 25.

```
eduardo@eduardo-SATELLITE-L855:~$ cd TFG/TFG_Eduardo_XBee/C++/
eduardo@eduardo-SATELLITE-L855:~/TFG/TFG_Eduardo_XBee/C++$ make build/test_final
eduardo@eduardo-SATELLITE-L855:~/TFG/TFG_Eduardo_XBee/C++$ ./build/test_final
```

Figura 25 Compilación y ejecución del programa

La ejecución del programa comienza reconociendo los XBee remotos y generando una tabla de direcciones visible en la Figura 26.

```
La tabla de direcciones es
[
  {0x13a20040340210, 0x15b1, 'NODO02'},
  {0x13a20040340e6a, 0x208e, 'E27'},
]
```

Figura 26 Tabla de direcciones

A continuación veremos un ejemplo de los mensajes mostrados por consola con lo referente a la ejecución en la cocina. El código ha sido limpiado para eliminar muchas líneas repetidas.

```
E27. Etapa a. Esperamos un primer pulso.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa a. Primer flanco de subida en D0.
E27. Etapa b. Esperamos un segundo pulso o encederemos la luz tras 2 seg.
E27. Etapa b. Hay tiempo y han pasado 2.00357sg de los 2 necesarios.
E27. Etapa b. Encendemos la luz (D1).

E27. Etapa a. Esperamos un primer pulso.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa a. Esperamos un primer pulso.
E27. Etapa a. Primer flanco de subida en D0.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa b. Segundo flanco de subida en D0.
E27. Etapa c. Esperamos un tercer pulso o encederemos el estractor tras 2 seg.
E27. Etapa c. Hay tiempo y han pasado 2.00349sg de los 2 necesarios.
E27. Etapa c. Encendemos el estractor (D5).

E27. Etapa a. Esperamos un primer pulso.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa a. Esperamos un primer pulso.
E27. Etapa a. Primer flanco de subida en D0.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa b. Segundo flanco de subida en D0.
E27. Etapa c. Esperamos un tercer pulso o encederemos el estractor tras 2 seg.
E27. D0 está en valor 0
E27. D0 está en valor 1
E27. Etapa c. Tercer flanco de subida en D0.
E27. Etapa c. Encendemos la placa vitrocerámica (D10).

E27. D0 está en valor 0
E27. Manteniendo. Hay tiempo y han pasado 0sg de los 2 necesarios.
E27. Manteniendo. Hay tiempo y han pasado 2.00396sg de los 2 necesarios.
E27. Manteniendo. Apagamos todo.
```

Figura 27 Mensajes por consola (Cocina)

4. Presupuesto

Artículo	Vendedor	Cantidad	Precio unitario	Total
Módulo XBee series 2	AliExpress	3	33,18 €	99,54 €
XBee Explorer USB	SparkFun	1	20,97 €	20,97 €
XBee Explorer Regulated	SparkFun	2	8,37 €	16,74 €
2mm 10pin XBee Header	SparkFun	4	0,80 €	3,2 €
2mm 10pin XBee Socket	SparkFun	4	0,84 €	3,36 €
Leds	TME.EU	4	0,12 €	0,48 €
LM35	Amazon	2	1,35 €	2,7 €
Schucko Aéreo Macho	Amazon	1	1,70 €	1,70 €
Schuko Aéreo Hembra	Amazon	1	1,25 €	1,25 €
Relé SRD-05VDC-SL-C	eBay	1	1,61 €	1,61 €
Pulsador switch	BricoGeek	2	0,40 €	0,80 €
Doble Puente H L298N	DealExtreme	1	8,13 €	8,13 €
Ventilador 12 VDC	AliExpress	1	0,98 €	0,98 €
				161,46 €

Tabla 6 Presupuesto de materiales

Hay que tener en cuenta que el presupuesto total de 161,46 € no incluye los gastos de envío ni las protoboards. No hace falta decir que si quisiéramos hacer este montaje en una vivienda posiblemente deberíamos gastarnos algo más de dinero ya no sólo en la mano de obra sino, tal vez, en componentes de mejor calidad.

Con respecto al número de horas para llevar a cabo el proyecto es obvio que variará mucho dependiendo de los conocimientos que se posean. Para una persona con conocimientos básicos de C++, habituada a usar Linux y que sepa cómo usar la librería facilitados por el tutor, el montaje y la programación no llevará más de 7 días (a 8 horas diarias). Si se empieza de cero con el uso de Linux y se desconoce lo facilitado por el tutor, el tiempo dedicado aumenta fácilmente a 20 o 25 días, como en mi caso (sin contar la redacción de la memoria).

5. Conclusión

5.1. Conclusión

En primer lugar, se estudió el protocolo de comunicación usado por los módulos XBee así como la librería para la comunicación serial facilitados por el tutor. En segundo lugar, se implementó en módulos XBee Series 2 una red inalámbrica desarrollada como red domótica de bajo consumo. A continuación, dicha red fue armada con sensores y actuadores de forma que pudiese atender y satisfacer nuestras necesidades. Y por último, se elaboró un programa de control y gestión en C++ para que ejecutase la lógica que haría funcionar todo nuestro montaje.

5.2. Conclusions

Firstly we studied the communication protocol used by the XBee devices and the serial communication library provided by the tutor. Secondly, a wireless network developed as a low-power home automation network was implemented in XBee Series 2 devices. Thirdly, this network was filled with sensors and actuators so that we could attend to our needs. And lastly, a C++ control and management program was done to execute the logic what would make all our circuit work.

5.3. Líneas abiertas

Este TFG no ha abordado la totalidad de las posibilidades presentadas por los módulos XBee y la comunicación ZigBee. Han quedado como líneas abiertas e interesantes para su desarrollo las siguientes cuestiones:

- El uso de salidas analógicas, como reguladores de luz.
- El uso del modo sleep de los módulos XBee para reducir el consumo en aquellas zonas de la red dónde no sea necesaria una revisión continua del estado de los sensores.
- La ampliación de la red a más zonas de la vivienda para poder cubrir más necesidades y así trabajar con más nodos finales y también con nodos router.
- El desarrollo de una interfaz web o de escritorio que nos permitiese modificar la lógica a seguir por cada zona de la vivienda así como los sensores o actuadores colocados.

- El desarrollar una aplicación para Smartphone que nos permitiese no sólo ver el estado de nuestro sistema sino incidir sobre él sin estar en la propia vivienda.

6. Bibliografía

[1] SLIDEPLAYER. 2014. *Introducción a la domótica.* http://slideplayer.es/slide/1041763/. Última consulta: 18 de agosto de 2017.

[2] http://jeromeabel.net/files/ressources/xbee-arduino/images/thumb_800x550/. Última consulta: 18 de agosto de 2017.

[4] DIGI INTERNATIONAL. 2015. *XBee ZNet 2.5/XBee PRO ZNet 2.5 OEM RF Modules.* http://ftp1.digi.com/support/documentation/90000866_G.pdf. Última consulta: 15 de agosto de 2017.

[5] INTJ NERD. 2016. *Petunjuk Singkat Komunikasi Wireless Arduino dengan XBee S2.* http://lang8088.blogspot.com.es/2016/06/petunjuk-singkat-komunikasi-wireless.html. Última consulta: 18 de agosto de 2017.

[6] PÉREZ MARTÍN, Javier. 2016. *Desarrollo de un sistema controlador para una red domótica inalámbrica.* http://riull.ull.es/xmlui/handle/915/1934. Última consulta: 15 de agosto de 2017.

[7] LAFRANCONI PEÑA, Gustavo Alejandro. 2014. *Diseño de varios sensores y actuadores domóticos que utilicen comunicación inalámbrica ZigBee.* http://riull.ull.es/xmlui/handle/915/249. Última consulta: 18 de agosto de 2017.

[8] TEXAS INSTRUMESNTS. 2016. *LM35 Precision Centigrade Temperature Sensors*. http://www.ti.com/lit/ds/symlink/lm35.pdf. Última consulta: 18 de agosto de 2017.

[9] ARDUINO TUTORIALES. 21 de enero de 2016. *¿Cómo usar un sensor LM35 (sensor de temperatura) con Arduino?* http://www.arduinotutoriales.com/sensor-lm35-medir-temperaturas-con-arduino/. Última consulta: 18 de agosto de 2017.

[10] GREEN PROPHET. 2017. *The history of energy saving LEDs*. https://www.greenprophet.com/2017/03/the-history-of-energy-saving-leds/. Última consulta: 19 de agosto de 2017.

[11] SONGLE RELAY. *SRD-05VDC-SL-C Datasheet*. http://files.microjpm.webnode.com/200001010-

dda41de9e1/Songle%20Relay%20Datasheet.pdf. Última consulta: 19 de agosto de 2017.

[12] FORO ARDUINO. 2015. http://forum.arduino.cc/index.php?PHPSESSID=3q7dqvnhdpjk6rve7e0bpaj540&topic=333650.msg2301360#msg2301360. Última consulta: 19 de agosto de 2017.

[13] STMICROELECTRONICS. *Dual Full-Bridge Driver*. http://pdf1.alldatasheet.com/datasheet-pdf/view/22440/STMICROELECTRONICS/L298N.html. Última consulta: 15 de agosto de 2017.

[14] GITHUB. *TFG_Eduardo_XBee*. https://github.com/ULL-InformaticaIndustrial-Empotrados/TFG_Eduardo_XBee/tree/master/C%2B%2B.

[15] MICROSPEC. 2 de octubre de 2015. *Curso de programación de controladores.* http://microspec.cl/Proyectos.php. Última consulta: 18 de agosto de 2017.

[16] STACHOWICZ, Arthur. 26 de abril de 2010. *ZigBee Wireless Networks.* http://zigbee.pbworks.com/w/page/25465049/ZigBee. Última consulta: 3 de agosto de 2017.

[17] GUALSAQUÍ VALENCIA, Edgar Aníbal. 2015. *Diseño e implementación de un prototipo (domsystem) de seguridad y control para mantener el resguardo de bienes y el confort mediante una red de sensores utilizando comunicación Wireless Bluetooth.* http://www.dspace.uce.edu.ec/bitstream/25000/4324/1/T-UCE-0011-172.pdf. Última consulta: 4 de agosto de 2017.