ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

SISTEMA MULTIPROPÓSITO DE AVISO Y ANUNCIO BASADO EN ZIGBEE. PROTOTIPO FORMADO POR µC ARDUINO Y MÓDULOS XBEE

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Mención: Telemática

Autor: Juan Trujillo González

Tutores: Domingo Marrero Marrero

Fecha: Junio 2021

Resumen

La evolución constante de las Tecnologías de la Información y Comunicación (TIC) durante estos últimos años ha propiciado un desarrollo exponencial en el sector de las comunicaciones inalámbricas, obteniendo un crecimiento de los mercados e industrias en el número de proyectos y aplicaciones que se pueden realizar, elevando con ello el nivel de competencia entre ellas. Actualmente estos sistemas suponen una potente herramienta en transferir datos y comunicarse, ofreciendo una flexible y móvil solución a los problemas que pueden aparecer en la vida diaria.

En el presente trabajo Fin de Grado se realizará un sistema multipropósito de aviso y anuncio que permita generar prealertas dirigidas a posibles receptores como mecanismo de notificación y advertencia. La comunicación del sistema será de forma inalámbrica empleando para ello el protocolo basado en el estándar IEEE 802.15.4 de redes WPAN llamado ZigBee. Como implementación se realizará un prototipo constituido por µC Arduino y módulos XBee® a modo de prueba funcional del sistema.

El objetivo fundamental del sistema es diseñar un sistema de comunicación económico y simple con la tecnología ZigBee e implementar un prototipo que permita ofrecer una flexible configuración adaptable según necesidades del entorno y aplicación. Con ello se consigue el envío y recepción de la información de reducido tamaño y diversa índole a potenciales receptores que estén dentro del rango de cobertura de los módulos XBee® y cuenten con los receptores habilitados. El prototipo estará formado por dos o tres nodos, uno de ellos actuando como transmisor encargado de emitir la información pertinente mientras que en el otro extremo contaremos con el receptor encargado de procesar y visualizar los datos a través de distintos periféricos de salida.

Palabras clave: TIC, IEEE 802.15.4, WPAN, ZigBee, XBee, Arduino.

Abstract

The constant evolution of Information and Communication Technologies (ICT) in recent years has led to an exponential development in the wireless communications sector, resulting in a growth of markets and industries in the number of projects and applications that can be carried out, thereby raising the level of competition between them. Currently these systems are a powerful tool in data transfer and communication, offering a flexible and mobile solution to the problems that may arise in everyday life.

In this thesis, a multipurpose warning and announcement system will be developed to generate pre-alerts directed to possible receivers as a notification and warning mechanism. The communication of the system will be wirelessly using the protocol based on the IEEE 802.15.4 standard for WPAN networks called ZigBee. A prototype consisting of µC Arduino and XBee® modules as a functional test.

The system's main objective is to design an economical and simple communication system with ZigBee technology and implement a prototype that allows to offer a flexible and adaptable configuration according to the needs of the environment and application. This allows sending and receiving small and diverse information to potential receivers that are within the coverage range of the XBee® modules and have enabled receivers. The prototype will consist of two or three nodes, one of them acting as a transmitter in charge of emitting the relevant information while at the other end we will have the receiver in charge of processing and visualizing the data through different output peripherals.

Keywords: ICT, IEEE 802.15.4, WPAN, ZigBee, XBee, Arduino.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

ACK – Acknowledgement

AIB – APS Information Base

AP – Application Profile

APDU – Application Protocol Data Unit

API – Application Programming Interface

APL – Application Layer

APS – Application Support Layer

BPSK – Binary Phase Shift Keying

BSN – Beacon Sequence Number

CA – Collision Avoidance

CCA – Clear Channel Assessment

COITT – Colegio Oficial de Ingenieros Técnicos de Telecomunicación

CSMA – Carrier Sense Multiple Access

DIN – Data In

DIP – Dual In–Line Packet

DOUT – Data Out

DSSS – Direct Sequence Spread Spectrum

ED – Detection Energy

FFD – Full Function Device

GTS – Guaranteed Time Slots

I2C – Inter-Integrated Circuit

IDE – Integrated Development Environmet

IEEE – Institute of Electrical and Electronics Engineers

IoT – Internet of Things

LCD – Liquid Crystal Display

LIPo – Lithium Polymer

LLC – Logical Link Control

LR-WPAN – Low Rate-Wireless Personal Area Network

LSB – Least Significant Byte

MAC – Medium Access Control

MCPS – MAC Common Part Sublayer

MFR – MAC Footer

MHR – MAC Header

MLME – MAC Sublayer Management Entity

MSB – Most Significant Byte

MSDU – MAC Service Data Unit

NIB – Network Layer Information Base

NLDE – Network Layer Data Entity

NLME – Network Layer Management Entity

NWK – Network Layer

OEMs – Original Equipment Manufacture

OQPSK – Offset Quadrature Phase Shift Keying

PAN – Personal Area Network

PHR – PHY Header

PHY – Physical Layer

PLME – Physical Layer Management Entity

PPDU – PHY Protocol Data Unit

PSDU – Physical Service Data Unit

PWN – Pulse Width Modulation

RF – Radio Frequency

RFD – Reduced Function Device

RSSI – Received Signal Strength Indicator

SAP – Service Access Point

SAP – Service Access Point

SFD – Start of Frame Delimiter

SHR – Synchronization Header

SPDT – Single Pole Double

SSCS – Service Specific Convergence Sublayer

SSID – Service Set Identifier

UART – Universal Asynchronous Transmitter

WPAN – Wireless Personal Area Network

XCTU – XBee Configuration and Test Utility

ZC – ZigBee Device

ZDO – ZigBee Device Objects

ZED – ZigBee End Device

ZR – ZigBee Router

MEMORIA

Capítulo 1. INTRODUCCIÓN

En este capítulo se hará una breve reseña exponiendo los antecedentes y las ideas que dan origen a la creación de este Trabajo Fin de Grado. Posteriormente, en el apartado 1.2 se contemplan los objetivos a los que se desea llegar durante su desarrollo y por último se describe la estructura que sigue esta memoria.

1.1 Antecedentes

El desarrollo y avance tecnológico que se ha producido en las últimas décadas en el área de las comunicaciones inalámbricas ha permitido una notable mejoría en el tratamiento de los datos de una forma rápida, flexible y autónoma, teniendo como consecuencia un gran impacto en la sociedad actual. Han aparecido nuevos métodos y sistemas que mejoran la calidad de vida de las personas, facilitando la integración de dispositivos inalámbricos en la vida cotidiana de la gente. Estos avances tecnológicos han permitido desarrollar nuevos circuitos integrados de muy pequeño tamaño, teniendo una alta eficiencia energética y suponiendo un coste bastante bajo. Algunos de estos avances lo vemos en las tecnologías de comunicación de Internet de las Cosas (IoT, del inglés *Internet of Things*), permitiendo habilitar una interconexión digital con dispositivos, máquinas, sensores o “cosas” cotidianas que generan y procesan datos desde cualquier región geográfica del planeta comunicándose con internet.

En este tipo de redes de corto alcance y destinadas a abarcar áreas de algunas decenas de metros, se encuentran las Redes de Área Personal (de ahora en adelante WPAN, del inglés *Wireless Personal Area Network*) [1]. Una particularidad relevante de estos sistemas es que están enfocados en que sean de bajo consumo energético, como es el caso de las redes 6LoWPAN (del inglés, *IPv6 over Low power Wireless Personal Area Networks*) [2]. Es aquí donde entra la tecnología ZigBee, una tecnología relativamente nueva y con gran potencialidad debido a su creciente, amplio y extremadamente flexible uso, permitiendo que dispositivos electrónicos de bajo consumo puedan realizar comunicaciones inalámbricas, especialmente en entornos industriales, médicos y domóticos entre otros [3]. La tecnología ZigBee está desarrollada para apoyarse en la estandarización del protocolo IEEE 802.15.4 (del inglés *Institute of Electrical and Electronics Engineers*) para comunicaciones de baja transferencia de datos, en el que uno de los enfoques principales es lograr que los dispositivos alimentados con baterías que utilicen esta tecnología no tengan un gasto energético elevado para garantizar un alto tiempo de vida [3].

Este tipo de tecnología ha hecho realidad el desarrollo de unos mecanismos distribuidos, diminutos, baratos y de consumo reducido que son capaces tanto de procesar información localmente como de comunicarla de forma inalámbrica. La posibilidad de implementar este tipo de dispositivos con las características energéticas que permiten su duración y su precio reducido sin mantenimiento capaces de obtener información del entorno y transmitirla, ofrece posibilidades inimaginables en multitud de aplicaciones. Es de aquí donde nace la idea de profundizar en el conocimiento de esta tecnología y buscar un uso o aplicación de esta, aprovechando todas estas características mencionadas.

1.2 Objetivos

El objetivo principal de este trabajo fin de grado es definir y desarrollar un sistema multipropósito de aviso y anuncio como elemento de seguridad proactivo que permita generar prealertas y anuncios utilizando la tecnología ZigBee. La idea básica es disponer de transmisores ZigBee configurados convenientemente según su necesidad para emitir información de forma constante y dirigida a potenciales receptores que puedan hacer uso de dicha información.

Más concretamente, se diseñará un sistema de comunicaciones basado en celdas aisladas y en pequeñas áreas de cobertura. Estas celdas constituirán diferentes subredes y estarán basadas en la tecnología ZigBee y el estándar IEEE 802.15.4. De forma más precisa, se permitirá que diferentes emisores ZigBee configurados de varias posibles formas (autónomos o gobernados por microcomputador) emitan información a modo de aviso o alerta, anuncio, difusión, etc. con un amplio ámbito de utilización. Cualquier receptor configurado para permitir su vinculación a estos emisores podrá procesar los mensajes convenientemente. Este sistema integrado se propone que pueda ser configurado de diversas opciones, desde un modo simple y más económico hasta un modo más completo basado en microcomputador, todos ellos permitiendo dotar a cualquier posible usuario o beneficiario de esta red de datos del hardware necesario para la comunicación entre ellos. Estas diferentes opciones permiten que, por ejemplo, en el caso más simple este pudiera llevarse en un bolsillo

o mediante sujeción en una bicicleta, guantera de un coche y cualquier otra ubicación para recibir la información de que se trate e interactuar con el usuario mediante alguna indicación visual o sonora hasta el más avanzado o completo usando el microcomputador Arduino. Se desarrollará el software configurable para diferentes situaciones y se implementará en un prototipo básico a modo de test de funcionalidad y posibilidades. Como dispositivos hardware para la experimentación, se usarán microcontroladores Atmel sobre plataformas Arduino y módulos XBee®.

1.3 Estructura de la memoria

La presente memoria se compone de cinco capítulos. En el primer capítulo se inicia una introducción al proyecto, así como los principales objetivos que se desean cumplir. En el segundo capítulo se describen los aspectos teóricos que lo componen, concretamente la tecnología ZigBee, sobre el que operan los módulos XBee® describiendo estos dispositivos, seguido de los sistemas microcomputadores Arduino. Se hará un estudio teórico de la tecnología y estos sistemas básicos, exponiendo sus características y permitiendo analizar de que manera se va a abarcar los pasos que den lugar a una solución técnica.

Posteriormente, en el tercer capítulo se presenta la descripción del diseño que se llevará a cabo con las diferentes opciones o configuraciones propuestas. Se describen varias propuestas de desarrollo y se especifican los componentes a utilizar en las fases de la implementación.

A continuación, en el cuarto capítulo se procederá a implementar el sistema diseñado. Se realizará la configuración de los dispositivos, así como de elaborar el montaje necesario para lograr su funcionamiento. Posteriormente se implementará el programa desarrollado y se llevará a cabo las pruebas necesarias para realizar casos experimentales y mostrar sus resultados.

Finalmente, en el quinto capítulo, se presentan las conclusiones obtenidas de este trabajo y se esbozan algunas líneas futuras que se pueden desarrollar a partir de este

proyecto como Trabajo Fin de Grado. En las tres siguientes secciones, fuera de la memoria, se muestra el anexo referente a la información empleada para el desarrollo de este proyecto junto con los diseños realizados, se describe el material utilizado y los costes que vienen acompañados para este proyecto.

Capítulo 2. FUNDAMENTOS TEÓRICOS

En este apartado, se pretende aportar información básica relativa a los principales aspectos de las tecnologías usadas. Se describe como es el funcionamiento y estructura básica de la tecnología ZigBee. Posteriormente, se analizará en detalle como funcionan los módulos hardware que utilizaremos para este proyecto.

2.1 Estándares ZigBee e IEEE 802.15.4

2.1.1 Introducción

ZigBee es un estándar especificado por dos organizaciones, el grupo de trabajo IEEE 802.15.4 y la asociación ZigBee Alliance [4]. Se basa en el estándar IEEE 802.15.4 de redes WPAN que se estableció en mayo de 2003 [5] y define el nivel físico y el control del acceso al medio de redes inalámbricas de área personal con tasas bajas de transmisión de datos (LR-WPAN, del inglés *Low Rate-Wireless Personal Area Network*).

Este estándar abarca las definiciones de las dos capas inferiores de la arquitectura de pila ZigBee, la capa física y la subcapa del control de acceso al medio, mientras que ZigBee Alliance definió la subcapa de soporte de aplicaciones, el objeto de dispositivo ZigBee, el perfil de dispositivo ZigBee, el marco de aplicación, la capa de red y los servicios de seguridad de ZigBee. En la Figura 1 se muestra el logo ZigBee Alliance.

Figura 1. Logo ZigBee Alliance.

En la actualidad, ZigBee Alliance está constituido por numerosas compañías, universidades y agencias gubernamentales, desde constructores de semiconductores y desarrolladores de software a fabricantes de equipos originales (OEMs, del inglés *Original Equipment Manufacturer*) e instaladores, convirtiéndose en el único estándar inalámbrico global de redes cualificado en las aplicaciones de monitorización, control y de sensores.

El objetivo de esta tecnología no es la de obtener velocidades muy altas, si no la de gestionar información de sensores cuyos transceptores tengan un bajo consumo de energía. De hecho, algunos dispositivos alimentados mediante pilas puedan aguantar

largos períodos de tiempo sin la necesidad de sustituirlas por otras nuevas. Debido a esto, dichos dispositivos pasan la mayor parte del tiempo en un estado latente, es decir, durmiendo para consumir mucho menos. Por lo tanto, ZigBee está dirigido a aplicaciones de radiofrecuencia que requieren una baja transferencia de datos, una batería de larga duración y una red segura. El bajo coste permite que la tecnología se implemente ampliamente en aplicaciones de control y monitoreo inalámbrico, su consumo de energía permite una vida útil más larga con baterías más pequeñas, y su red característica de malla proporciona alta confiabilidad y mayor alcance. Pretende ser más simple y económico que otras tecnologías WPAN, como por ejemplo Bluetooth [6].

Esta tecnología, generalmente se usa para control industrial, detección integrada, recolección de datos médicos, advertencia de humo e intrusos, automatización de edificios, domótica, etc.

2.1.2 Bandas de frecuencia de operación y rango velocidades

IEEE 802.15.4 tiene dos capas físicas que operan en dos rangos de frecuencia separados, 868/915 MHz y 2.4 GHz. La capa física de baja frecuencia cubre tanto la banda de 868 MHz como la banda de 915 MHz. La capa física de mayor frecuencia, 2.4 GHz, se usa prácticamente en todo el mundo. El estándar ZigBee se ha implementado y especifica el funcionamiento en las bandas ISM sin licencia de 2,4 GHz (en todo el mundo), 915 MHz (América y Australia) y 868 MHz (Europa). Dieciséis canales están asignados en la banda de 2.4 GHz, y cada canal requiere 5 MHz de ancho de banda. Las radios utilizan codificación de espectro ensanchado por secuencia directa (DSSS, del inglés *Direct Sequence Spread Spectrum),* que es administrada por la corriente digital en el modulador. La modulación por desplazamiento de fase binaria (BPSK, del inglés *Binary Phase Shift Keying*) se utiliza en las bandas de 868 y 915 MHz, y la modulación por desplazamiento de fase en cuadratura ortogonal (OQPSK, de inglés *Offset Quadrature Phase Shift Keying*) que transmite dos bits por símbolo se utiliza en la banda de 2,4 GHz.

Las frecuencias de 868/915 MHz y 2.4 GHz poseen una buena incidencia de penetración a la hora de atravesar techos o paredes, aunque tienen límite de rango. Este límite de rango es deseable para disminuir las interferencias. Regresando a los aspectos ideales de sistemas basados en la tecnología *ZigBee*, deben tener una instalación automática o que se acerque, buscando la finalidad de que los usuarios puedan instalar estas redes fácilmente. En la Tabla 1 se muestran diferentes valores vinculados a cada banda de frecuencia.

Tabla 1. Bandas de frecuencia en ZigBee.

	Frecuencia (MHz)	Número de canales	Modulación	Tasa de chip (Kchip/s)	Tasa de Bit (Kb/s)	Tasa de símbolos	Spreading Method
	868-868.6	1	BPSK	300	20	20	DSSS Binario
	902-928	10	BPSK	600	40	40	DSSS Binario
Opcional	868-868.6	1	ASK	400	250	12.5	20-bit PSSS
	902-928	10	ASK	1600	250	50	5-bit PSSS
Opcional	868-868.6	1	O-QPSK	400	100	25	16- array ortogonal
	902-928	10	O-QPSK	1000	250	62.5	16-array ortogonal
	2400	16	O-QPSK	2000	250	62.5	16-array ortogonal

Generalmente, las aplicaciones en 2.4 GHz permiten un ancho de banda más grande, así como más canales. No obstante, hay que tener en cuenta otros sistemas como WLAN y Bluetooth, que funcionan con 2.4 GHz, a la hora de coexistir. En la Figura 2 se observa una gráfica relacionando el alcance con la tasa de bits de diferentes estándares desarrollados por el IEEE 802.

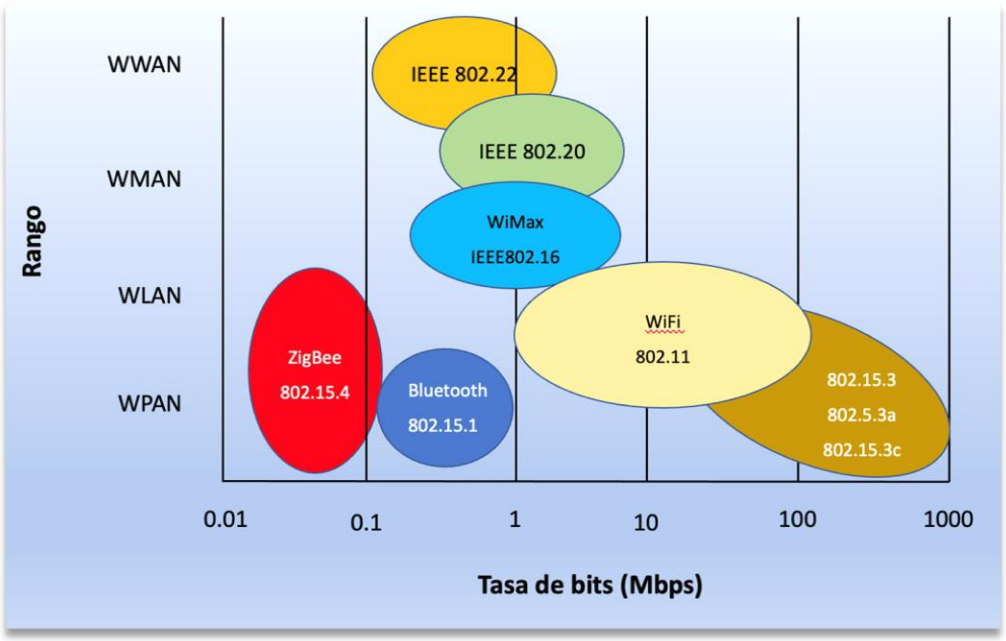


Figura 2. Tecnologías en banda de 2.4 GHz.

El estándar ZigBee permite tener una velocidad máxima de transferencia de datos de hasta 250 Kbps. Concretamente, la velocidad de datos sin procesar en el aire es de 250 kbps por canal en la banda de 2,4 GHz, 40 kbps por canal en la banda de 915 MHz y 20 kbps en la banda de 868 MHz. El rendimiento de datos real será menor que la velocidad de bits máxima especificada debido a la sobrecarga del paquete y los retrasos en el procesamiento.

Para aplicaciones en interiores, según se especifica [7], la distancia de transmisión dentro de la banda de 2.4 GHz, pueden variar desde los 10 a 100 metros de la línea de visión, dependiendo de la potencia de salida y las características ambientales tales como los materiales de construcción, número de paredes a penetrar y la potencia de salida permitida en esa ubicación geográfica. En exteriores con línea de visión, el alcance puede ser de hasta 1500m dependiendo de la potencia de salida y las características ambientales. La potencia de salida de las radios está generalmente entre 0-20 dBm (1-100 mW).

2.1.3 Tipos de dispositivos

IEEE 802.15.4 y ZigBee definen diferentes tipos de dispositivos. Por un lado, IEEE 802.15.4 especifica componentes basados en su mecanismo de transmisión, denominados “dispositivos” y son los más básicos. Las redes LR-WPAN consisten en estos dispositivos, que basados en su funcionalidad a su vez se diferencian en dos tipos:

- **Dispositivos de funcionalidad completa:** FFD, del inglés *Full Function Device*.
- **Dispositivos de funcionalidad reducida:** RFD, del inglés *Reduced Function Device*.

Como indican sus nombres, un FFD tiene más funcionalidades que un RFD. En general, FFD tiene tres roles diferentes en la red, es decir, puede operar en tres modos que sirven como:

- **Coordinador de red de área personal:** Coordinador PAN (del inglés *Network Personal Area Network*). Identifica la red y sus configuraciones.
- **Coordinador:** que proporciona servicios de sincronización a través de la transmisión de balizas.
- **Dispositivo:** que no implementan las funcionalidades anteriores y deben asociarse con un coordinador antes de interactuar con la red.

Los FFD se encargan de la transmisión de la fuente de datos principal en una red, pueden comunicarse con cualquier otro dispositivo. Los RFD actúan como un dispositivo final que solo puede asociarse con un FFD a la vez. En una red IEEE 802.15.4 se necesita al menos un FFD para actuar como coordinador. En cuanto al direccionamiento, todos los dispositivos deben tener una dirección extendida de 64 bits o utilizar una dirección corta de 16 bits asignada por el coordinador. Comúnmente, los RFD usan baterías mientras que los FFD usan la línea eléctrica.

ZigBee implementa su base en esto y según su papel en la red, clasifica los dispositivos en tres tipos que son, coordinador ZigBee (ZC, del inglés *ZigBee Coordinator*), router ZigBee (ZR, del inglés *ZigBee Router*) y dispositivo final ZigBee (ZED, del inglés *ZigBee End Device*). A continuación, se describen las principales características de estos dispositivos.

2.1.3.1 Coordinador ZigBee

Es el dispositivo más completo, debe existir solo uno por red y su presencia es obligatoria. Las funciones y responsabilidades del coordinador son:

- Arranque de la red.
- Configuración de los parámetros de la red.
- Admisión de nodos a la red.
- Asignación de direcciones de red.

Para estos elementos o nodos de una red ZigBee, es importante que la fuente de alimentación sea permanente y segura, ya que este dispositivo nunca entrará en el llamado *modo Sleep*, pues debe controlar la red constantemente.

2.1.3.2 Router ZigBee

Es un nodo de tipo FFD pero que no es el coordinador ZigBee, sin embargo, puede actuar como un coordinador según IEEE 802.15.4 en su propio canal operativo. La utilidad de éstos es para extender la cobertura de la red y aumentar la confiabilidad con la creación de rutas adicionales de datos. Por lo tanto, cumple con la finalidad de router permitiendo localizar al coordinador mediante el intercambio con otros router. Además de ello, es posible su configuración permitiendo la ejecución de código por parte de un anfitrión. La obtención de los datos que dispone el coordinador o

dispositivo final los reenvía a través de la misma antena por la ruta que esté más cercana, llevándolos al elemento de la red al que está destinado la transmisión.

Se suelen utilizar cuando tenemos dos elementos ZigBee y la señal de transmisión que hay entre estos dos es débil, ya sea porque tenemos algún tipo de obstáculo en el camino viéndose atenuada la comunicación o cuando tenemos una distancia superior al límite establecido en las especificaciones. También se utilizan para localizar al coordinador.

2.1.3.3 Dispositivo Final ZigBee

Contiene la funcionalidad suficiente para comunicarse con el nodo principal (ya sea el coordinador o un router). Estos nodos tienen menos potencia de cómputo y no pueden transmitir datos desde otros dispositivos (solo emisor o receptor de información, no puede encaminar dichos datos). Esta relación permite que el nodo pueda estar en un modo de bajo consumo, *modo Sleep*, una cantidad significativa de tiempo, por lo que generalmente son alimentados con baterías. Son dispositivos FFD o RFD según el estándar IEEE 802.15.4 y requiere la menor cantidad de memoria, por lo tanto, puede ser menos costoso de fabricar que un dispositivo router o coordinador.

En la Figura 3 se puede apreciar las clasificaciones ZigBee e IEEE 802.15.4 y sus analogías.

Figura 3. Tipos dispositivos ZigBee / IEEE 802.15.4

Los FFDs operan en cualquiera de las topologías de red que ofrece el estándar y puede desempeñar el papel de coordinador PAN, coordinador ZigBee, router ZigBee o dispositivo final ZigBee. En el siguiente apartado se detallan las distintas topologías de redes existentes en el estándar ZigBee.

2.1.4 Topología de red

Dependiendo de los requisitos de la aplicación, una red LR-WPAN puede funcionar en cualquiera de las siguientes dos topologías: topología en estrella y topología punto a punto.

- **Topología en estrella.**
  En la topología en estrella, la comunicación se establece entre dispositivos y un único controlador central, que es el coordinador PAN. Un dispositivo generalmente tiene alguna aplicación asociada y es el punto de inicio o finalización para las comunicaciones de la red. Un coordinador PAN también puede tener una aplicación específica, pero se puede usar para iniciar, finalizar o encaminar la comunicación por la red. La topología punto a punto también tiene un coordinador PAN, pero difiere de la topología en estrella en que cualquier dispositivo puede comunicarse con cualquier otro dispositivo siempre y cuando estén dentro del alcance del otro. La estructura básica de la topología en estrella se puede ver en la Figura 4.

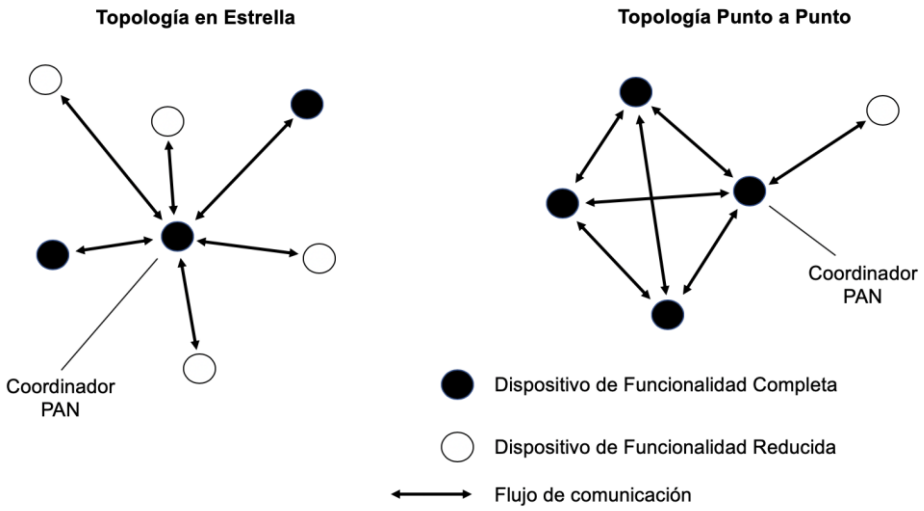


Figura 4. Topología de red de IEEE 802.15.4.

Después de activar un FFD por primera vez, este puede establecer su propia red y convertirse en el coordinador de la red. Todas las redes dentro de esta topología operan independientemente de todas las demás redes estrella que estén funcionando. Esto se logra eligiendo un **identificador PAN**, que no es utilizado por ninguna otra red dentro de su radio de influencia. Este ID es importante para diferenciar una red de otra. Una vez que se elige el identificador PAN, el coordinador PAN puede permitir que otros dispositivos se unan a su red, tanto los FFD como los RFD. Las aplicaciones que se benefician de una topología en estrella incluyen domótica, periféricos de ordenadores personales, juguetes y atención médica personal entre otros.

- **Topología punto a punto**

  En una topología punto a punto, cada dispositivo es capaz de comunicarse con cualquier otro dispositivo dentro de su radio de influencia. Un dispositivo será designado como coordinador PAN, por ejemplo, para ser el primer dispositivo en comunicarse en el canal. Se pueden construir estructuras de red adicionales a partir de la topología punto a punto y pueden imponer restricciones topológicas a la formación de la red. Un ejemplo del uso de la topología de comunicaciones punto a punto es la topología en redes de árboles (del inglés *Topology Cluster Tree*). Esta topología es un caso especial de una red punto a punto en la que la mayoría de los dispositivos son FFD.

Ahora bien, las especificaciones del estándar ZigBee y su capa de red apoyándose a partir de las topologías citadas anteriormente, soportan tres tipos de topologías de red mejorando las anteriores: topología en estrella, topología en red de malla y topología en árbol. En la Figura 5 se muestran dichas topologías.

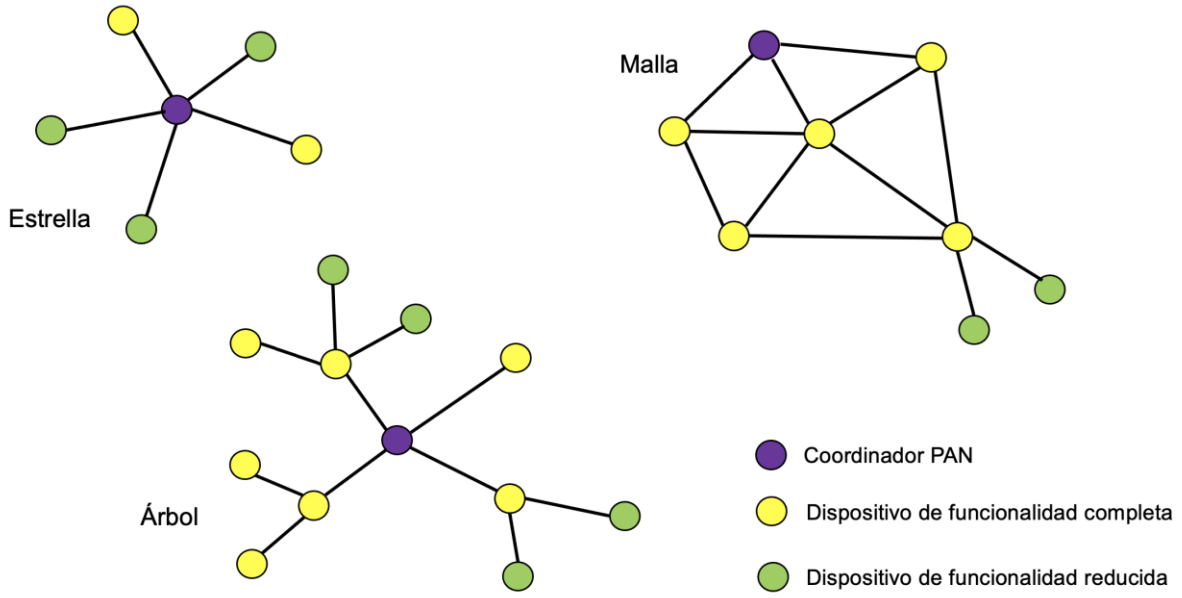


Figura 5. Topologías de red especificadas en ZigBee.

2.1.4.1 Topología en estrella

En una topología en estrella la red está controlada por un solo dispositivo. En esta topología el coordinador es el centro de la red y es el que se conecta con los demás dispositivos (finales). Es responsable de iniciar y mantener los dispositivos en la red. Por lo tanto, todos los mensajes deben pasar por el coordinador. Dos dispositivos finales no pueden comunicarse entre si directamente, sino que tiene que comunicarse directamente con el coordinador ZigBee. En la Figura 6 se muestra dicha topología.

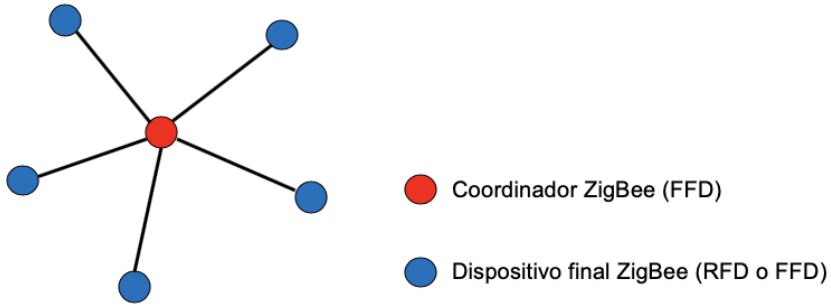


Figura 6. Topología en estrella.

2.1.4.2 Topología en red de malla

El coordinador ZigBee es responsable de iniciar y de elegir ciertos parámetros clave de la red, pero la red puede extenderse mediante el uso de enrutadores ZigBee. En esta topología se tiene conectividad total de todos los FFD que forman la red con el FFD que actúa como coordinador PAN según se muestra la Figura 7. Si en algún instante un nodo de la red o camino fallan en la comunicación, se puede volver a trazar la ruta rehaciendo los caminos, el cual es responsable el coordinador. Este tipo de topología pertenece a la topología punto a punto.

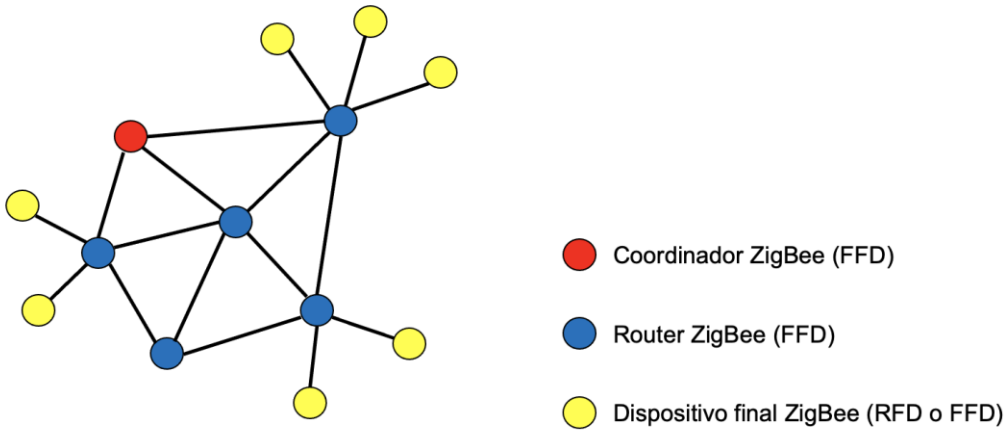


Figura 7. Topología en Malla.

2.1.4.3 Topología en árbol

Un caso especial de una red punto a punto es la topología en árbol de clúster en el que la mayoría de los dispositivos son FFD, tal y como aparece en la Figura 8. Un RFD se conecta a una red actuando como una hoja al final de una rama, ya que los RFD no permiten la asociación de otros dispositivos. Cualquiera de los FFD puede actuar como coordinador, proporcionando servicios de sincronización a otros dispositivos u otros coordinadores. Solo uno de estos coordinadores puede ser el coordinador general del PAN, que puede tener mayores recursos computacionales que cualquier otro dispositivo en la red. Esto requiere un mecanismo para decidir el coordinador del PAN y un mecanismo de resolución de conflictos si dos o más FFD intentan establecerse simultáneamente como un Coordinador PAN.

En las redes de árbol, los enrutadores mueven datos y controlan mensajes a través de la red utilizando una estrategia de enrutamiento jerárquico. Las redes de árbol pueden emplear comunicación orientada a balizas.

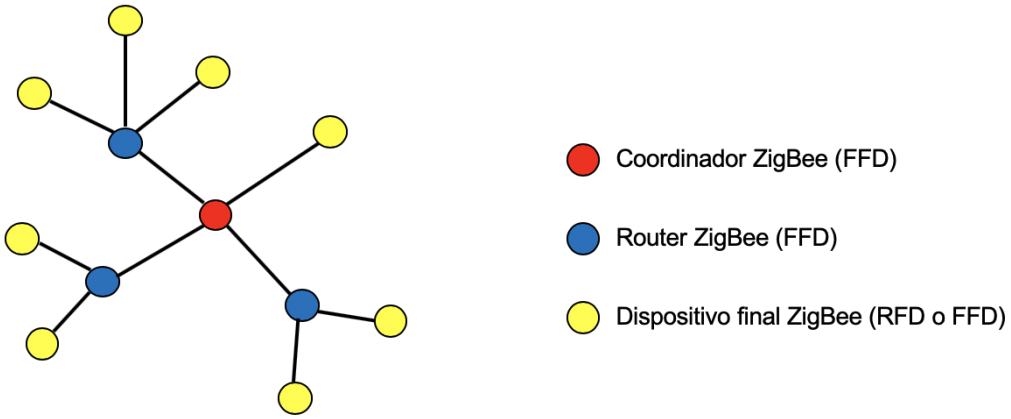


Figura 8. Topología en árbol.

Dentro de todas las topologías analizadas en la red ZigBee, la más destacada es la topología en malla, ya que permite un encaminamiento dinámico, por lo tanto, otorga a la red ganar sobretodo en fiabilidad en la comunicación.

2.1.5 Conceptos básicos de la arquitectura de red

2.1.5.1 Visión general

La arquitectura LR-WPAN (IEEE 802.15.4) se define en términos de varios bloques para simplificar el estándar. Estos bloques se llaman capas o niveles. Cada capa es responsable de una parte del estándar y ofrece servicios a las capas superiores. Las interfaces entre las capas sirven para definir los enlaces lógicos que se describen en este estándar. Un dispositivo LR-WPAN comprende la capa física (de ahora en adelante PHY, del inglés *Physical Layer*), que contiene el transceptor de radiofrecuencia (RF, del inglés *Radio Frequency*) junto con su mecanismo de control de bajo nivel y una subcapa de control de acceso al medio (de ahora en adelante MAC, del inglés *Medium Access Control*), que proporciona acceso al canal físico para todos los tipos de transferencia. La Figura 9 muestra estos bloques en una representación gráfica.

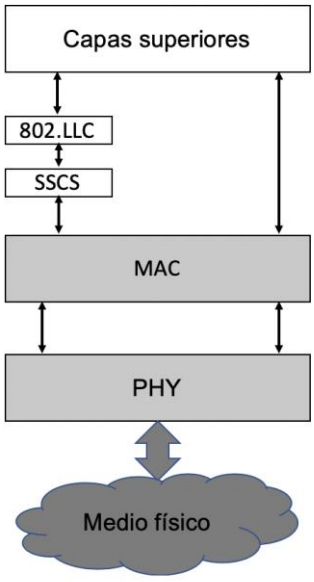


Figura 9. Arquitectura WPAN.

Las capas superiores consisten en una capa de red que proporciona configuración de la red, manipulación, enrutamiento de los mensajes, y una capa de aplicación, que proporciona la función prevista del dispositivo. La definición de estas capas superiores

está fuera del alcance del estándar IEEE 802.15.4. Un control de enlace lógico (LLC, del inglés *Logical Link Control*) puede acceder a la subcapa MAC a través de la subcapa de convergencia específica del servicio (SSCS, del inglés *Service Specific Convergence Sublayer*). La arquitectura LR-WPAN se puede implementar como dispositivos integrados o como dispositivos que requieren el soporte de un dispositivo externo como un PC.

La arquitectura de pila ZigBee, como se ilustra en la Figura 10, está separada también por capas. Como se ha mencionado, cada capa realiza un conjunto de servicios específicos para la capa superior. Una entidad de datos proporciona un servicio de transmisión de datos y una entidad de gestión proporciona todos los demás servicios. Cada entidad de servicio expone una interfaz a la capa superior a través de un punto de acceso al servicio (SAP, del inglés *Service Access Point*), y cada SAP soporta una serie de primitivas de servicio para lograr la funcionalidad requerida.

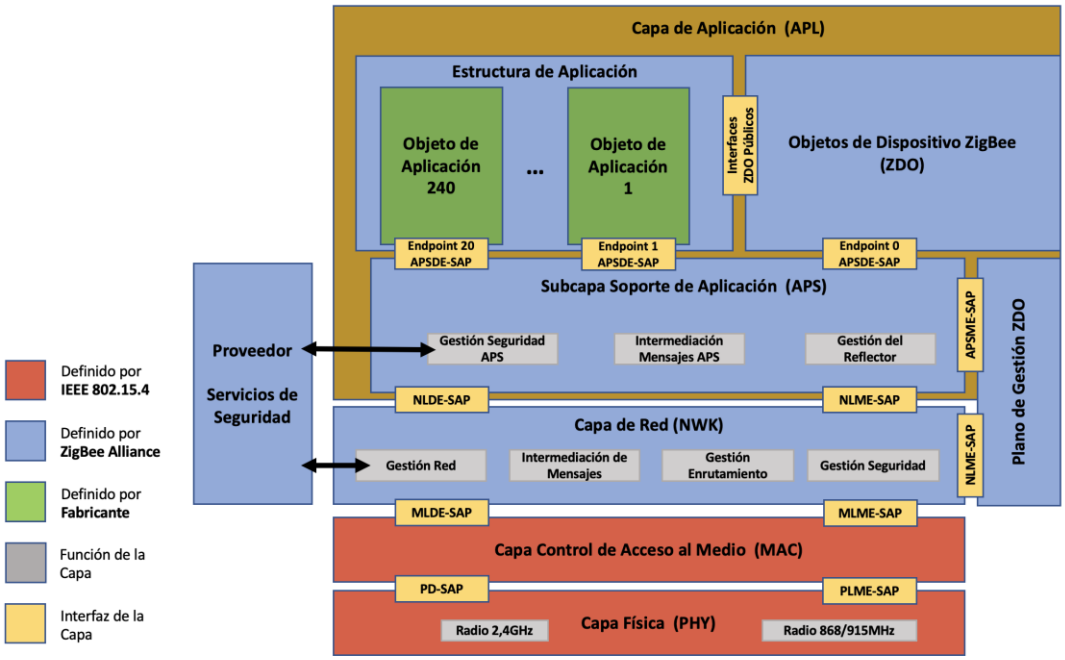


Figura 10. Pila de protocolo ZigBee y IEEE 802.15.4.

El estándar IEEE 802.15.4 define las dos capas inferiores como se ha dicho anteriormente. Primero la capa PHY, seguida de la subcapa MAC. ZigBee Alliance se basa en este estándar para sus dos capas inferiores, y proporciona una capa de red (NWK, del inglés *Network Layer)* y el marco para los estándares de la capa de

aplicación (de ahora en adelante APL, del inglés *Application Layer*), que incluye la subcapa de soporte de aplicaciones (APS, del inglés *Application Support Sublayer*), los objetos del dispositivo ZigBee (ZDO, del inglés *ZigBee Device Objects*) y los objetos de aplicación definidos por el fabricante, que usan este marco y comparten APS y seguridad con el ZDO.

A continuación, de una breve forma se exponen los niveles que conforman las capas citadas, describiendo los principales detalles de cada una de ellas.

2.1.5.2 Capa Física

Esta capa corresponde al estándar IEEE 802.15.4 siendo la más cercana al hardware, controlando y comunicando el transceptor. Además de definir las funciones y la relación con la capa MAC, define aspectos como la potencia del transmisor y la sensibilidad del receptor.

Proporciona dos servicios: el servicio de datos PHY y el servicio de gestión PHY que interactúan con la entidad de gestión de la capa física (PLME, del inglés *Physical Layer Management Entity*). El servicio de datos PHY permite la transmisión y recepción de unidades de datos del protocolo PHY (PPDU, del inglés *PHY Protocol Data Unit*) a través del canal de radio físico.

La capa PHY es responsable de:

- Activación y desactivación del transceptor de radio.
- Detección de energía dentro del canal actual (ED, del inglés *Detection Energy*).
- Indicación de calidad de enlace para paquetes recibidos.
- CCA (del inglés *Clear Channel Assessment*) para CSMA / CA (del inglés *Carrier Sense Multiple Access / Collision Avoidance*).
- Selección de frecuencia de canal.
- Transmisión y recepción de datos.

La capa PHY es responsable de la transmisión y recepción de datos utilizando un determinado canal de radio y de acuerdo con una técnica de modulación y difusión específica. Las frecuencias más bajas son más adecuadas para rangos de transmisión más largos debido a menores pérdidas de propagación. Las transmisiones de baja velocidad proporcionan una mejor sensibilidad y un área de cobertura mayor. Una tasa más alta significa un mayor rendimiento, menor latencia o ciclos de trabajo más bajos. Todas estas bandas de frecuencia se basan en la técnica de propagación del espectro ensanchado por secuencia directa (DSSS).

2.1.5.3 Capa de enlace

La subcapa MAC proporciona dos servicios: el servicio de datos MAC (del inglés *MAC Data Service*) y el servicio de gestión MAC (del inglés *MAC Management Service*). Estos dos servicios también son llamados MLME (del inglés *MAC Sublayer Management Entity*) y son accedidos por dos tipos de SAP:

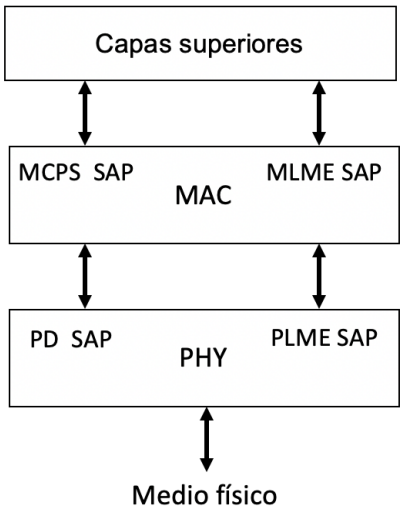


Figura 11. Servicios de enlace de la subcapa MAC 802.15.4.

- **MCPS-SAP:** MAC Common Part Sublayer Service Access Point. Sirve para el enlace de la subcapa MAC.
- **MLME-SAP:** MAC Management Access Point. Útil para servicios de administración.

LLC accede a la MAC mediante estos dos tipos de SAP. El servicio de gestión MAC interactúa con MLME-SAP. El servicio de datos MAC permite la transmisión y recepción de unidades de datos del protocolo MAC a través del servicio de datos PHY.

La subcapa MAC maneja todo el acceso al canal de radio físico y es responsable de:

- Activación y desactivación del transceptor de radio.
- Detección de energía dentro del canal actual (ED, del inglés *Detection Energy*).
- Indicación de calidad de enlace para paquetes recibidos.
- CCA (del inglés *Clear Channel Assessment*) para CSMA / CA (del inglés *Carrier Sense Multiple Access / Collision Avoidance*).
- Selección de frecuencia de canal.
- Transmisión y recepción de datos.

Entre estas características, cabe destacar la gestión de balizas. Dentro de los protocolos de comunicación inalámbrica, una baliza o también llamada beacon es una trama que es enviada periódicamente por el coordinador y que va dirigida al resto de los módulos de la red ZigBee notificando que este se encuentra activo y preparado para utilizarse en la transferencia de los datos. Mientras se están enviando los beacons, los dispositivos finales se encuentran activos, pero una vez que el coordinador deja de enviarlos, estos dispositivos de la red pasan a estar en un modo de bajo consumo llamado *modo Sleep*. Se pueden configurar estos dispositivos para que se salga de este modo automáticamente en un tiempo dado. A la hora de configurar el dispositivo, es fundamental establecer el tiempo en que los dispositivos ZigBee permanecerán inactivos y también dicho período de tiempo.

En cuanto al formato de las tramas MAC se dividen en 4 tipos:

- Beacons.
- Datos.
- Confirmación (ACK).
- Comandos.

Todas estas se componen de tres partes las cuales incluyen un encabezado MAC (MHR, del inglés *MAC Header*), un campo de carga útil o de datos y un pie de página (MFR, del inglés MAC Footer). La parte del encabezado contiene los campos de control de la trama MAC, números de secuencia de beacons (BSN, del inglés *Beacon Sequence Number*) y campos de información de direccionamiento. Mientras que el MFR contiene una secuencia de verificación de tramas (FCS, del inglés *Frame Check Sequence*).

El MHR, MSDU y MFR forman la trama beacon MAC. En esta trama, una unidad de datos del protocolo MAC (MPDU, del inglés MAC Protocol Data Unit) se pasa entonces a la PHY como la carga útil del paquete beacon. La unidad de datos del servicio físico (PSDU, del inglés *Physical Service Data Unit*) tiene un encabezado de sincronización (SHR, del inglés *Synchronization Header*), que contiene la secuencia de preámbulo y los campos de delimitador de inicio de trama (SFD, del inglés *Start of Frame Delimiter*), y un encabezado PHY (PHR, del inglés PHY Header) que contiene el campo de longitud de la trama. En la Figura 12 se observa la estructura de la trama beacon. La secuencia de preámbulo permite al receptor lograr la sincronización de símbolos. SHR, PHR y PSDU juntos forman el paquete de baliza PHY.

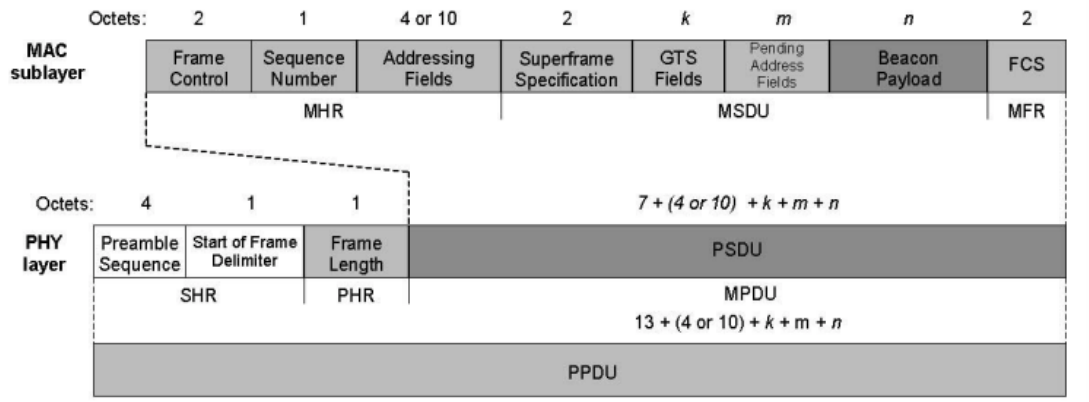


Figura 12. Estructura de una trama beacon.

En una trama de datos, véase Figura 13, la carga útil de datos se pasa a la subcapa MAC y se denomina unidad de datos del servicio MAC (MSDU, del inglés *MAC Service Data Unit*). La MSDU tiene un prefijo MHR y un MFR adjunto. El MHR contiene los campos de información de direccionamiento, número de secuencia y control de trama. El MFR está compuesto por un FCS de 16 bits. El MHR, MSDU y MFR juntos forman la trama de datos MAC como se aprecia en la Figura *13*. La MPDU se pasa a la PHY como la carga útil de la trama de datos de la PHY. La PSDU tiene el prefijo SHR, que contiene la secuencia de preámbulo y los campos SFD, y una PHR que contiene la longitud de la PSDU en octetos. La secuencia de preámbulo y los datos SFD permiten que el receptor logre la sincronización de símbolos. SHR, PHR y PSDU juntos forman el paquete de datos PHY.

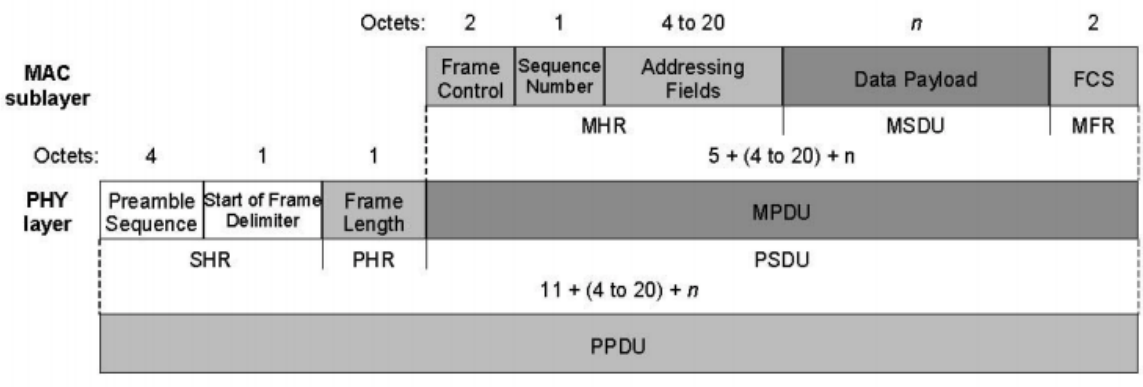


Figura 13. Estructura de una trama de datos.

2.1.5.4 Capa de Red

Se requiere que la capa de red proporcione funcionalidad para garantizar el correcto funcionamiento de la subcapa MAC y para proporcionar una interfaz de servicio adecuada para la capa de aplicación. Para interactuar con la capa de aplicación, la capa de red incluye conceptualmente dos entidades de servicio que proporcionan la funcionalidad necesaria:

- **Servicio de datos de red:** NLDE, del inglés *Network Layer Data Entity*. Permite que una aplicación transporte unidades de datos de protocolo de aplicación (APDU, del inglés *Application Protocol Data Unit*) entre dos o más dispositivos. Proporciona servicios tales como generación de PDU a nivel de red, enrutamiento especifico de la topología y seguridad garantizando tanto la autenticidad de la red como la confidencialidad de una transmisión.
- **Servicio de gestión de red:** NLME, del inglés *Network Layer Management Entity*. Permite que una aplicación interactúe con la pila. Proporciona servicios tales como configuración de un nuevo dispositivo, inicialización de una nueva red, unirse y salir de una red, direccionamiento, descubrimiento de vecinos, descubrimiento de rutas, control de recepción y mecanismo de enrutamiento como difusión o multidifusión.

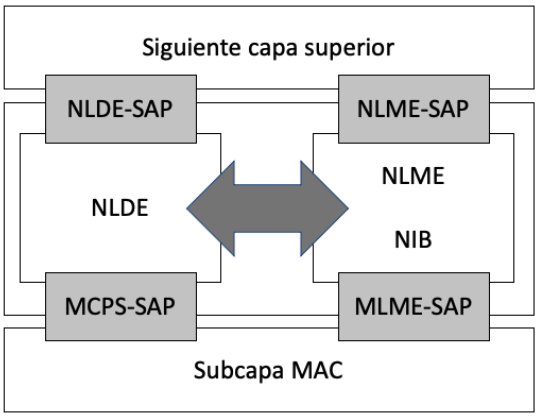


Figura 14. Capa de red especificada en ZigBee.

La NLDE y NLME proporcionan respectivamente el servicio de transmisión de datos y gestión de la red a través de su SAP asociado, el NLDE-SAP, y la NLME-SAP. El

NLME utiliza el NLDE para lograr algunas de sus tareas de administración y también mantiene una base de datos de objetos administrados (NIB, del inglés *Network Layer. Information Base*) conocida como NIB.

2.1.5.5 Capa de Aplicación

La capa de aplicación consta de la subcapa de soporte de aplicaciones (APS, del inglés *Application Support Subl*ayer), los objetos de dispositivo ZigBee (ZDO, del inglés *ZigBee Device Objects*) y el marco de aplicación (AF, del inglés *Application Framework*) definidos por el fabricante. Los fabricantes desarrollaron esta capa para poder personalizar un mismo dispositivo para varias aplicaciones, por lo que se puede decir que esta capa es la que hace a los dispositivos versátiles. Antes de describir las funciones de las subcapas de este nivel, es preciso detallar algunos conceptos que se utilizan en esta capa de aplicación:

- **Perfil de aplicación:** AP, de inglés *Application Profile*. Se encarga de enviar y recibir todas las peticiones de datos en ZigBee. Un perfil especifica un dominio contenido por un tipo de aplicación y dispositivo. Estos perfiles pueden ser públicos (especificados por la ZigBee Alliance), con el fin de que un producto de un fabricante funcione con el de otro distinto, y privados.
- **Cluster:** Utilizan un identificador, cluster ID. Definen el comportamiento de una aplicación a través de comandos y atributos. Un ejemplo sería el encendido de una luz, comando, o ver si está encendida que en este caso sería atributo.
- **Puntos de acceso**: *Endpoints*. Se identifican mediante un número comprendido entre 1 y 240. Estos definen cada aplicación que funciona dentro de un nodo ZigBee.

Marco de aplicación (AF)

El marco de aplicación de ZigBee es el entorno en el que se alojan los objetos de la aplicación en los dispositivos ZigBee. Se pueden definir hasta 240 objetos de aplicaciones distintas, cada uno de ellos identificado por un *endpoint* de 1 a 240. Se

definen dos *endpoint* adicionales para el uso de APSDE-SAP. La dirección de *endpoint* 0 está reservado para la interfaz de datos a la ZDO, y el 255 está reservado para la función de interfaz de datos para transmitir datos a todos los objetos de aplicación.

Objetos de dispositivo ZigBee (ZDO)

Los objetos del dispositivo ZigBee representan una clase de funcionalidad básica que proporciona una interfaz entre los objetos de la aplicación, el perfil del dispositivo y el APS. El ZDO se encuentra entre el marco de la aplicación y la subcapa de soporte de la aplicación. Satisface los requisitos comunes de todas las aplicaciones que operan en una pila de protocolos ZigBee. La ZDO es responsable de lo siguiente:

- Iniciar la subcapa de soporte de aplicaciones, la capa de red y el Proveedor de Servicios de Seguridad.
- Ensamblar la información de configuración de las aplicaciones finales para determinar e implementar el descubrimiento, la gestión de seguridad, la gestión de red y la gestión de vinculación.

Subcapa de soporte de aplicaciones (APS)

La subcapa de soporte de aplicaciones proporciona la interfaz entre la capa de red y la capa de aplicación a través de un conjunto general de servicios para su uso tanto por el objeto de dispositivo ZigBee como por los objetos de aplicación definidos por el fabricante. Estos servicios se ofrecen a través de dos entidades.

- **Entidad de datos APS**: APSDE, del inglés *Application Support Data Entity*. Proporciona el servicio de transmisión de datos a través de su SAP asociado, el APSDE-SAP.
- **Entidad de gestión APS**: APSME, del inglés *Application Support Management Entity*. Proporciona el servicio de gestión a través de su SAP asociado, el APSME-SAP, y mantiene una base de datos de objetos gestionados conocida como base de información APS (AIB, del inglés *APS Information Base*).

Para una información más detallada sobre las capas superiores ZigBee incluyendo su seguridad en las comunicaciones se dispone de la especificación ZigBee [7] para profundizar en los mismos.

2.2 Módulo XBee®. Solución ZigBee de Digi

2.2.1 Introducción

Una solución basada en ZigBee es la desarrollada por la empresa Digi [8]. Estos módulos hardware hacen posible las comunicaciones inalámbricas bajo este estándar con las especificaciones del mismo, pero siguiendo una implementación propietaria; estos son los muy conocidos módulos XBee® [9]. De acuerdo con Digi, fabricante de dichos módulos, los módulos XBee® son soluciones integradas que brindan un medio inalámbrico para la interconexión y comunicación entre dispositivos. Estos módulos utilizan el protocolo de red especificado en el estándar IEEE 802.15.4 para crear redes punto a punto o redes punto a multipunto.

Los módulos XBee® tienen una dirección única denominada dirección MAC. Por lo que cada módulo, tiene una dirección única de 64 bits que viene incorporada de fábrica. En una red ZigBee, se utilizan direcciones de 16 bits en los algoritmos para elegir la ruta. Cada vez que un dispositivo se une a la red, y por lo tanto se asocie al coordinador, este le asigna una dirección única de 16 bits. Por esta razón, el número máximo teórico de elementos que puede haber en una red ZigBee es de $2^{16} = 65535$, que es el número máximo de direcciones de red que se pueden asignar.

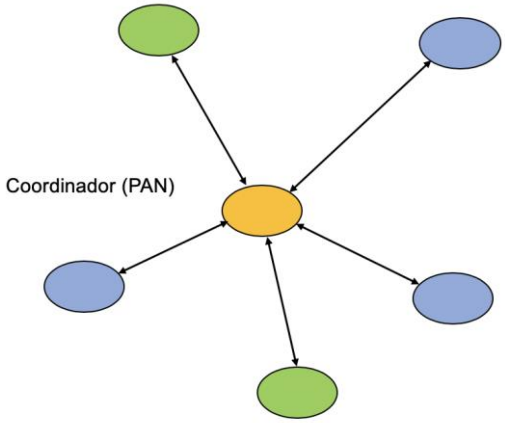


Figura 15. Coordinador PAN con múltiples nodos.

2.2.2 XBee® Serie 2

Para el desarrollo de este trabajo se han utilizado los módulos *Digi XBee® S2C ZigBee* [10] ya que son los disponibles y se adaptan perfectamente a las necesidades y requisitos de este trabajo.

Uno de los principales detalles de estos dispositivos es que pueden leer los sensores, tanto digitales como analógicos directamente y transmitir a su nodo de origen o destino la información por sí mismo. También incorporan salidas PWN (del inglés *Pulse Width Modulation*), digitales y analógicas. Destacan principalmente por su tamaño y gracias a su peso reducido podemos emplearlos en amplia gama de situaciones, teniendo un bajo consumo eléctrico. En la Figura 16 podemos observar el módulo XBee® S2C.

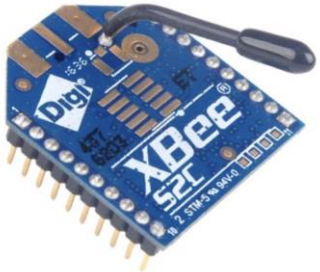


Figura 16. Módulo XBee® S2C.

2.2.2.1 Especificaciones y patillaje

Las especificaciones de los XBee® S2C se resumen en la Tabla *2*:

Tabla 2. Principales especificaciones XBee® S2C.

Fabricante	Digi International
Serie	XBee® ZigBee®
RF Familia / Estándar	802.15.4
Frecuencia	2.4 GHz
Potencia de salida máxima	+10dBm
Sensibilidad del receptor	-100dBm / -102dBm
Interfaces seriales	UART, SPI
Tasa de bits	RF 250 Kbps, serial hasta 1Mbps
Rango en interior/urbano	Hasta 60 m
Rango en exterior	Hasta 1200 m
Tensión de alimentación	2.1 -> 3.6V dc
Dimensiones	24 mm de ancho x 28 mm de profundidad x 8 mm de altura
E/S digitales	15
Tipo de montaje	Through Hole
Temperatura de Funcionamiento	- 40°C a + 85°C
Canales	16
Peso	3.25 g

La Tabla 3 muestra las asignaciones de los 20 pines disponibles para el módulo XBee® (las señales de nivel bajo activas se distinguen subrayadas) mientras que su disposición se puede observar en la Figura 17.

Tabla 3. Asignación de pines para módulo XBee® S2C.

Pin	Nombre	Tipo	Descripción
1	VCC	-	Fuente de alimentación
2	DOUT / DIO13	Salida	Salida de datos UART
3	DIN / CONFIG/ DIO14	Entrada	Entrada de datos UART
4	DIO12	Ambos	E/S digital 12

5	RESET	Entrada	Reseteo módulo (pulso mín. 200 ns)
6	PWM0 / RSSI / DIO 10	Ambos	PWN salida 0 / Indicador fuerza señal RSSi / E/S digital 10
7	PWN / 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 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	Ambos	Indicador asociado / E/S digital 5
16	RTS / DIO6	Ambos	Control de 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 / DI0 Botón de puesta en marcha	Ambos	Entrada analógica 0 / E/S digital 0 / Botón de comisionado

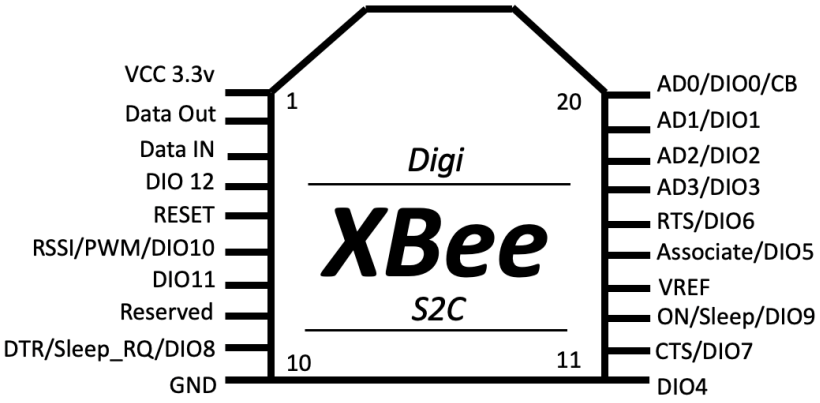


Figura 17. Patillaje del módulo XBee®.

2.2.2.2 Comunicación Serie

La forma que tiene el módulo XBee® de comunicarse con su anfitrión es mediante un puerto serie estándar. Está dotado para la comunicación con cualquier transmisor-

receptor asíncrono universal (de ahora en adelante UART, del inglés *Universal Asynchronous Receiver-Transmitter*).

A la hora de encargarse de la comunicación con su anfitrión, el módulo XBee® mediante sus pines seriales dispone de dos protocolos: transparente e interfaz de programación de aplicaciones (de ahora en adelante API, del inglés *Application Programming Interface*). Estos protocolos se detallarán en los siguientes apartados. Cuando los dispositivos XBee® se encuentran en una misma red intercomunicados, estos pueden emplear protocolos diferentes no afectando a la comunicación con otros nodos, solo a la comunicación serial del propio nodo.

Cuando los módulos emplean el modo transparente, actúan como si fuesen una línea serie básica. Los datos que reciben a través del pin DIN (del inglés *Data In*) se transmitirán directamente sin modificación por radiofrecuencia. En el caso contrario, al recibir por radiofrecuencia los datos, se enviarán por el pin DOUT (del inglés *Data Out)* tal como se reciben.

En el desarrollo de este trabajo el modo transparente no se utilizará por su sencillez, si en cambio el protocolo API será el elegido, se estudiará en profundidad y se utilizará para implementar el formato de los diferentes mensajes o PDUs de la comunicación.

2.2.3 Funcionamiento básico

La Figura 18 muestra el conexionado mínimo que necesita un módulo XBee® para que un dispositivo generador de datos (un microcomputador, un ordenador personal, u otro sistema) mediante el puerto serie envíe sus mensajes o reciba los mismos. Por defecto, los módulos están configurados para leer mensajes recibidos por la línea RXD (del inglés *Reception Data*). Estos pueden ser de diferentes tipos: mensajes especiales con cabeceros que especifican que deben enviarse por RF y otros, que representan comandos de programación o configuración del módulo.

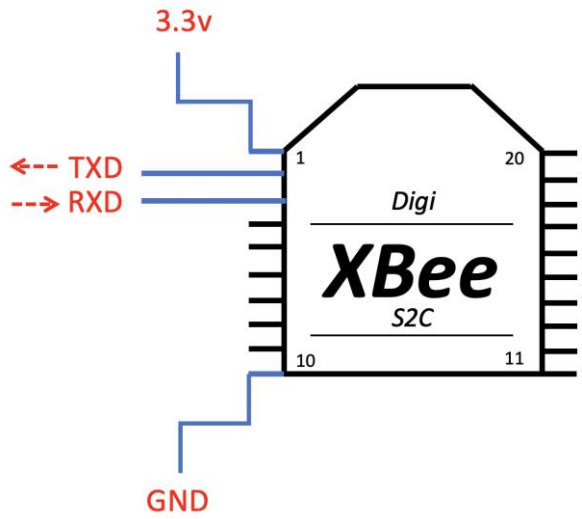


Figura 18. Conexiones mínimas requeridas para el módulo XBee®.

Según las especificaciones del fabricante [10], solamente se requiere una alimentación desde 2.1 a 3.6 Voltios, la conexión a tierra y para poder comunicarse con un microcontrolador por medio de la UART (TXD y RXD) como líneas de transmisión de datos con niveles 0 y 3,3 aproximadamente. Si este último utiliza otros niveles de voltaje, deberán convertirse convenientemente.

Si bien parece un dispositivo sencillo, desde el punto de vista de configuración y programación es bastante potente incorporando un sistema software microcontrolador elaborado y muy amplio para configurar muchas posibilidades. Estas van desde el modo de trabajo de los puertos de entrada/salida, duración del modo Sleep, activación o modos de despertar, configuración del puerto de serie, etc. Para su programación se utiliza un software propietario y el mismo puerto serie que para las comunicaciones, no pudiendo coexistir ambos modos de operación (programación o configuración y comunicación) salvo mediante el uso de telecomando (comandos específicos de configuración), que no forman parte de este trabajo.

Para la programación de los módulos XBee® el fabricante ofrece el software denominado XCTU (del inglés *XBee Configuration and Test Utility*) para su configuración [11]. Este software permite a través de una interfaz gráfica interactuar con los módulos vía serie y nos ofrece herramientas para la configuración, inicialización, actualización y testeo de los XBee®. En la Figura 19 se puede ver un

ejemplo de la interfaz de este software. Para obtener instrucciones sobre cómo descargar y usar XCTU, se puede consultar la Guía del usuario de XCTU [12].

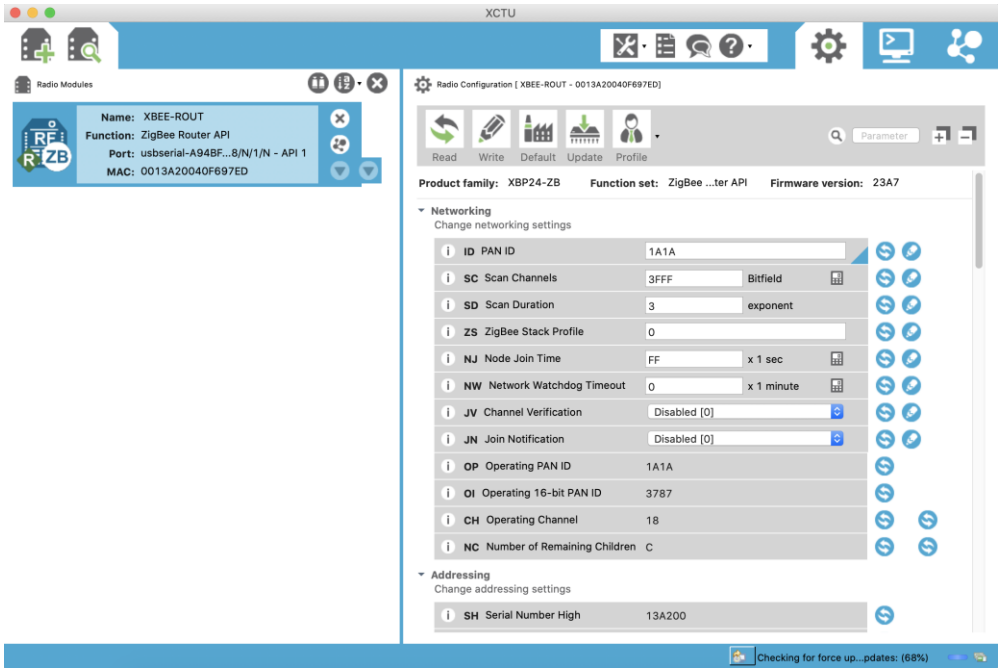


Figura 19. XCTU: Interfaz gráfica del software.

2.2.4 Modos de operación

Los módulos XBee® pueden operar en los siguientes 5 modos:

- Modo IDLE.
- Modo Recibir.
- Modo Transmitir.
- Modo Comando.
- Modo de Bajo Consumo (Sleep Mode).

A continuación, se describirán las principales características de cada uno de ellos.

2.2.4.1 Modo IDLE

Cuando no se reciben ni se transmiten datos, el dispositivo se encuentra en modo Idle. Es un estado de transición entre los otros modos disponibles. Durante este modo, el dispositivo escucha datos válidos en el puerto serie y por el enlace RF. El dispositivo cambia a otro modo de operación bajo las siguientes condiciones:

- Modo transmitir. Los datos serie en el buffer de recepción serial están preparados para ser empaquetados.
- Modo recibir. Los datos RF válidos son recibidos a través de la antena.
- Modo comando. Se emite una secuencia de modo de comando.

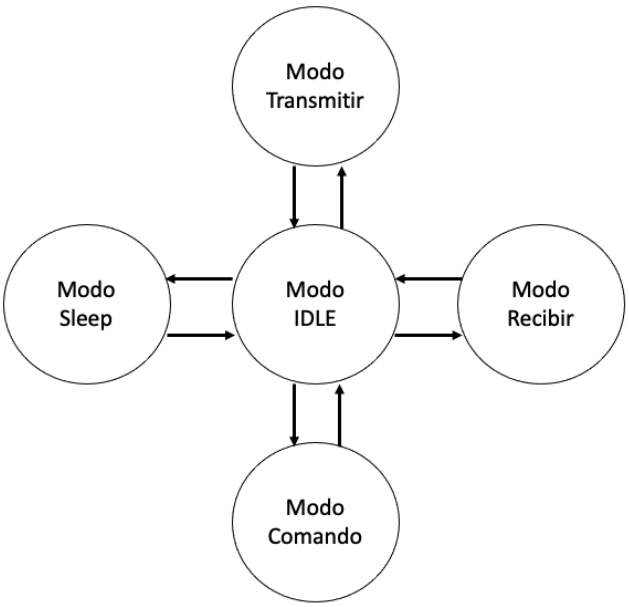


Figura 20. Modos de operación XBee®.

2.2.4.2 Modo Recibir y Transmitir

El dispositivo se encuentra el modo recibir cuando no está transmitiendo datos y se recibe por la antena que dispone uno o varios paquetes RF. Por otra parte, el dispositivo cambiará a modo transmitir cuando la información es enviada vía serial

dirigida al buffer, es decir, cuando tenga datos en el buffer. Posteriormente se transmite por RF.

A su vez, la información que se transmite puede ser en:

- Modo directo: la información es enviada seguidamente a la dirección que tiene de destino.
- Modo Indirecto: el módulo detiene la información un tiempo determinado y la envía únicamente cuando es solicitada por la dirección de destino.

Asimismo, desde un punto de vista de los destinatarios, hay dos formas diferentes de poder enviar la información:

- Unicast: comunicación desde un módulo origen a otro módulo XBee®. En este modo se admite respuesta de quien recibe el paquete RF. Una vez recibido debe enviar un ACK (del inglés, *Acknowledgement*), que básicamente es un paquete que notifica que se recibió el paquete, confirmando su recepción a la dirección de origen. El módulo que envió el paquete espera hasta haber recibido el ACK y si no llega dicho paquete, se reenvía hasta un límite de tres veces o hasta que haya recibido el ACK.
- Broadcast: comunicación desde un nodo a todos los que se encuentran en la red. En este modo no se dispone de confirmación mediante ACK.

Estas direcciones serán las especificadas y únicas de 64 bits por cada módulo o especiales para soportar el modo broadcast y la dirección reservada y única para el coordinador de la red.

2.2.4.3 Modo comando

En este modo, el firmware interpreta los caracteres entrantes como comandos. Permite modificar la configuración del dispositivo usando parámetros que se pueden establecer a través de los comandos AT [13]. Cuando se desee leer o configurar

cualquier parámetro del módulo usando este modo, se debe enviar un comando AT. Cada comando AT comienza con las letras AT seguidas por los dos caracteres que identifican el comando y luego por algunos valores de configuración opcionales.

Permite gobernar el módulo por la misma línea de serie; ya que con este tipo de comandos se pueden leer, ajustar o modificar la configuración de los nodos vía RF. Estos comandos AT en su formato válido pueden ser entregados por un microcomputador o hacer uso de un ordenador personal con un software de comunicaciones serie, por ejemplo, el uso del programa Hyperterminal de Windows o el propio programa de configuración XBee®.

En la Figura 21 se observa un ejemplo de la sintaxis de un comando AT. Luego de ingresar a este modo, se debe ingresar el comando deseado para ajustar los parámetros del módulo XBee®. Este ejemplo real cambia la dirección de destino del dispositivo (*Destination Address Low*) a 0x1F.

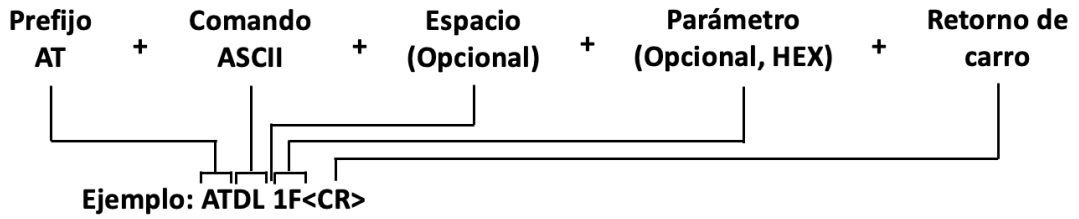


Figura 21. Ejemplo Comando AT.

Los modos de funcionamiento del módulo XBee® están controlados por la configuración del parámetro AP (API Enable) pero el modo de comando siempre está disponible como un modo en el que el dispositivo puede ingresar mientras está configurado para cualquiera de los modos de funcionamiento.

2.2.4.4 Modo Sleep

En este modo el módulo XBee® entra en un estado de inactividad (como si estuviera dormido). Se caracteriza por permitir que el dispositivo tenga un bajo consumo de energía dependiendo de la configuración en la que se encuentra mientras no se utiliza. El dispositivo está casi completamente apagado durante este modo y es incapaz de enviar o recibir datos hasta que se activa. Los módulos admiten la suspensión de los pines, en la que los dispositivos entran en el modo Sleep tras la transición de estos pines donde el dispositivo se queda en un estado mínimo de energía por un tiempo fijado.

2.2.5 Protocolos de interfaz serial

El módulo RF XBee® soporta interfaz serial Transparente y modo API como ya se comentó. A continuación, se describe su funcionamiento más detalladamente.

2.2.5.1 Modo de operación transparente

Los dispositivos funcionan en este modo de manera predeterminada. Actúan como un reemplazo de la línea serie, guardando en el buffer de entrada todos los datos UART que recibe a través del pin 3 (RXD) para la transmisión de RF. Cuando un dispositivo recibe datos de RF, se guardan en el buffer de salida hacia el puerto serie y luego los envía a través del pin 2 (TXD). Se pueden establecer los parámetros de configuración utilizando la interfaz de comando AT.

El uso de este modo es sobre todo para la comunicación punto a punto, en la que no se necesita ningún tipo de control. Igualmente se utiliza para sustituir por cable una conexión serial, debido a que es la configuración más simple y no exige una configuración más compleja. Estos buffers son mostrados en la Figura 22.

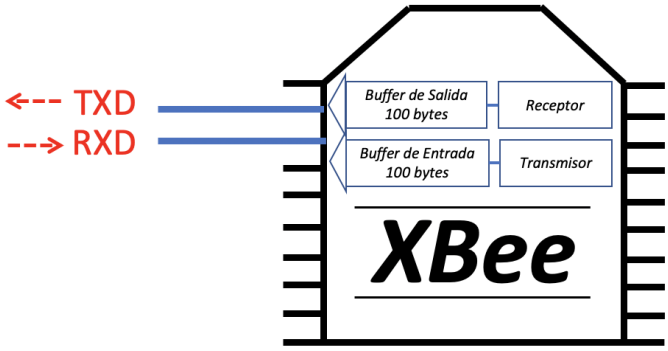


Figura 22. Correspondencia de Buffer con entrada y salida.

2.2.5.2 Modo de operación API

El modo de operación API es una alternativa al modo transparente. Es un protocolo basado en tramas (*frames*) o unidades de datos estructurados con cabeceras especiales que permiten garantizar la transmisión de los datos y una amplia gama de posibilidades para el manejo de esos datos. El dispositivo comunica datos UART en forma de paquetes (encapsulando los bytes), también conocidos como *API frames*. Dispone de varias maneras para configurar los módulos, como también diferentes alternativas del ruteo de los datos. Los datos pueden ser enviados a distintos módulos sin la necesidad de que la dirección de destino tenga que ser cambiada mediante la configuración del módulo, esto es debido a que cada trama que se envía contiene un campo con la información de la dirección de destino. En este modo la confirmación cuando una transmisión es realizada está permitida, de tal forma que se puede saber cuando un paquete ha llegado correctamente cuándo un paquete es recibido también se permite identificar el módulo que lo envió, ya que en la propia trama incluye un campo con la dirección del nodo que lo ha transmitido.

Todos los datos entrantes y salientes en este modo deben estar ordenados de forma estructurada a través de las tramas. Utilizando las tramas de datos de la UART la API detalla cómo los comandos y las tramas de los estados o eventos del modulo se envían o reciben. Por lo tanto, se encapsula dos veces ya que hay una estructura

simple para los comandos que se envían por la UART y una sintaxis propia de la API, que está reservada en un campo de datos.

Solo en este modo se permite la lectura y escritura de las entradas y salidas del módulo. Un ejemplo sería que el módulo enviase periódicamente lecturas de los sensores que tenga conectados. También se puede utilizar comandos AT manualmente cuando el módulo está conectado al puerto serie.

Por consiguiente, en el modo API los mensajes, respuestas y los comandos son encapsulados en una trama, la cual permite enviar y recibir mensajes a módulos remotos y recibirlos de estos a través de una única interfaz serie y de una manera sencilla.

El firmware admite la configuración del parámetro AP, siendo el modo operativo de trabajo. Dependiendo del comando API se habilita cualquiera de los modos:

Tabla 4. Modos del parámetro AP.

Comando AP de configuración	Descripción
AP = 0	Modo de operación transparente, línea serial de la UART reemplazada con modo API desactivado. Esta es la opción por defecto.
AP = 1	API operativo (Sin caracteres de escape).
AP = 2	API operativo con caracteres de escape (solo posible en la UART).

Al configurar el modo API con caracteres de escape se consigue evitar varios bytes que podrían entrar en conflicto con la UART. En concreto los bytes 0x7E (*Delimitador de inicio*), 0x11 (Escape) y 0x13 (XOFF). Se soluciona introduciendo donde está alguno de estos bytes de la trama dos nuevos bytes en la misma posición.

La Tabla 5 muestra la estructura general de una trama de datos en modo API detallando donde están contenidos los comandos o datos que se quieran transmitir, así como el número de byte con el que corresponden.

Tabla 5. Estructura de una trama de datos.

Delimitador de inicio	**Longitud**		**Dato de la trama**							**Checksum**
			Tipo de trama	Datos						
1	2	3	4	5	6	7	8	9	…	n	n+1
0x7E	MSB	LSB	Tipo de trama API	Datos						Un byte

- **Delimitador de inicio:** Este campo indica el comienzo de una trama. Siempre es 0x7E. Esto permite que el dispositivo detecte fácilmente una nueva trama entrante.
- **Longitud:** El campo de longitud especifica el número total de bytes incluidos en el campo de datos de la trama. Su valor es de dos bytes, MSB (del inglés, *Most Significant Byte*) y LSB (del inglés, *Least Significant Byte*). Se excluye el delimitador de inicio, la longitud y la suma de comprobación (en inglés, *Checksum*).
- **Datos:** contienen los datos en sí. Esta información y su orden dependen del tipo de trama que defina el campo tipo de trama. Generalmente incluyen valores numéricos de muestras obtenidas desde el exterior o entregadas desde una fuente externa o representando caracteres alfanuméricos en ASCII según se programen. Esta información es la que un dispositivo recibe desde una fuente serie y transmitirá por RF. La estructura de los datos de la trama depende del propósito de la trama API.
- **Checksum:** Suma de comprobación o verificación. Es el último byte de la trama de datos y ayuda a probar la integridad de los datos. Se calcula tomando la suma hash de todos los bytes de trama API anteriores, excepto los primeros tres bytes (delimitador de inicio y longitud). Si la suma de verificación es incorrecta el dispositivo no procesa las tramas enviadas a través de la interfaz serial e ignora sus datos, así como descarta la trama recibida en el receptor.

En el modo API cualquier trama con la que se trabaje viene identificada mediante un ID. La siguiente tabla muestra un resumen de los tipos de tramas con sus identificadores:

Tabla 6. Identificadores y tipos de tramas API.

Identificador API	Tipo de trama API
0x8A	Estado del modem
0x08	Comando AT
0x09	Comando AT – Poner en colar valor del parámetro
0x88	Respuesta de comando AT
0x17	Comando AT Remoto
0x97	Respuesta de comando Remoto
0x10	Petición de transmisión ZigBee
0x11	Trama de comando de direccionamiento explícito de ZigBee
0x8B	Estado de transmisión ZigBee
0x90	Recepción de paquete ZigBee (AO=0)
0x91	Indicador de RX explícito ZigBee (AO=1)
0x92	Indicador de muestreo de datos RX de E/S ZigBee
0x94	Indicador de lectura de sensor XBee (AO=0)
0x95	Indicador de identificación de nodo (AO=0)

2.3 Conceptos básicos µC ARDUINO

2.3.1 Visión general

Arduino [14] es una plataforma de hardware y software libre, además abierto para la utilización y contribución de toda la sociedad. Fue ideado en el año 2005, con el objetivo de obtener una herramienta de hardware que supusiese una fácil compresión, programación y aplicación para cualquier usuario no experimentado en ordenadores y que no fuese muy cara, estando orientado para el uso entre diversas plataformas, pudiendo configurarlo en entornos como Windows, GNU/Linux y Mac OS.

Está compuesto por circuitos electrónicos basados en microcontroladores por parte del fabricante Atmel, actualmente adquirida por Microchip Technology [15], caracterizados por ser chips que no son tan complejos y no suponen un coste muy

elevado. Su circuitería permite un uso rápido y simple del microcontrolador. Por otra parte, existen numerosas placas o módulos (llamados shields) adicionales de hardware que permiten una conexión sencilla a la placa base del Arduino, que a su vez estás proveen conectores estandarizados, así como un conjunto extenso de librerías de software libre [16]. Asimismo, se dispone de conectividad USB en las tarjetas con lo que es posible programar de una manera sencilla el microcontrolador que incorporan. Todas estas características hacen accesible a cualquier usuario el poder hacer uso de la programación de objetos o de dispositivos interactivos, los cuales pueden ser indicadores luminosos, interruptores, etc.

Arduino se presenta con una variedad de circuitos impresos, mostrados en la Figura 23, pudiendo encontrar multitud de modelos, en la que todos tienen un mismo fin, pudiendo ser compatibles con los shields oficiales y el entorno que ofrece Arduino para programar (Arduino IDE). Entre algunas de las placas Arduino existentes, los modelos más utilizados son UNO, Mega, Pro, Leonardo, Mini, Nano, Fio, etc. [17].

Figura 23. Ejemplo de modelos de placas Arduino disponibles.

En el mercado hay una gran variedad de microcontroladores y plataformas disponibles similares a Arduino, algunos ejemplos son Raspberry Pi, Waspmote, Nanode y muchas otras que tienen una funcionalidad similar; quizás, en el primer caso supera enormemente las posibilidades, prestaciones y el consumo, constituyendo más un microcomputador que un microcontrolador. Muchos de estos microcontroladores disponibles en el mercado recogen las características de una programación sencilla y

las reúne en un paquete fácil de usar. Arduino facilita el proceso de trabajo con microcontroladores, pero se caracteriza por ofrecer algunas ventajas dirigidas a profesores, estudiantes y aficionados que tengan interés en estos sistemas entre otros:

- **Bajo coste:** el precio de las placas Arduino suelen ser relativamente barato comparado con otros sistemas de plataformas microcontroladores que hay en el mercado.
- **Multiplataforma:** Arduino dispone del software de edición, simulación de código y programación de la memoria interna del microcontrolador para poder ejecutarlo en sistemas operativos Windows, Mac OS y GNU/Linux.
- **Entorno de programación simple y claro:** Sencillo de utilizar para usuarios que sean principiantes, pero a su vez flexible para que programadores avanzados puede aprovecharlo también. Es de código abierto y su software es bastante extensible. El lenguaje puede ser ampliado mediante librerías C++.
- **Hardware extensible**: La licencia Creative Commons [18] permite duplicarlas y los diseñadores pueden crearse su propia versión del módulo, extendiendo los componentes y mejorando la placa.
- **Flexible:** Una de las ventajas es que tenemos la posibilidad de agregar módulos según la utilidad que queramos darle, ya sea controlar un motor, conectividad a internet, etc. Se dispone de una amplia variedad de módulos a la hora de comprarlos.

2.3.2 Características del modelo UNO

Nos centraremos en describir de forma general el modelo **Arduino UNO** (véase Figura 24), dado que es el utilizado para la realización de este proyecto. En concreto para el desarrollo de este se ha proporcionado el modelo UNO R3, caracterizado por su bajo peso y dimensiones. Posee un microcontrolador ATmega328P [19] y su alimentación puede darse a través de USB (5V) o mediante alimentación externa. Sus pines son analógicos y digitales.

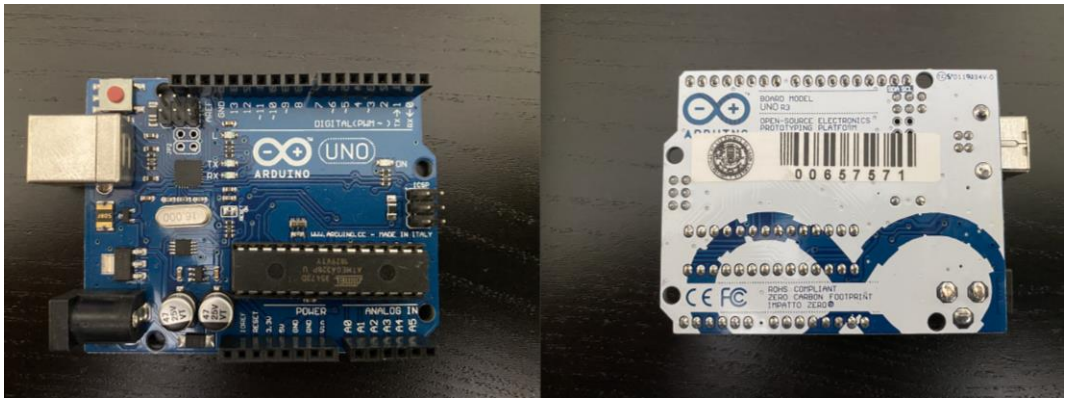


Figura 24. Parte frontal y reverso de la placa Arduino Uno.

En la Tabla 7 se resume las principales características técnicas de la placa Arduino UNO. La descripción completa se puede encontrar en su página web oficial [20].

Tabla 7. Principales especificaciones técnicas de Arduino UNO.

Microcontrolador	ATmega328P
Voltaje de funcionamiento	5 V
Tensión de entrada (recomendado)	7-12 V
Tensión de entrada (límite)	6-20 V
Pines digitales de E/S	14 (6 son salidas PWN)
Pines de entrada analógicos	6
Intensidad por pin E/S	20 mA
Intensidad para Pin 3.3 V	50 mA
Memoria Flash	32 KB
SRAM	2 KB
Velocidad de reloj	16 MHz
Pin de led	13
Longitud	68.6 mm
Ancho	53.4 mm
Peso	25 g

2.3.2.1 Entradas y salidas

En cuanto a los pines de entrada y salida del circuito integrado Arduino UNO, estos se usan para las entradas de los datos procedentes, generalmente de un sensor o para activar un actuador, o sea con la finalidad de leer la información del entorno o

dar la ordenes pertinentes. Dispone de 14 pines digitales de entrada o salida según su uso programación, estos están numerados desde el pin 0 al pin 13. Operan a 5 voltios y cada pin proporciona o recibe hasta 40mA. Disponen de una resistencia pull-up [21] interna de entre 20-50 KΩ, la cual está desconectada por defecto.

Aunque cada pin por separado pueda proporcionar hasta 40 mA, algo importante es que el circuito integrado en su interior junta los pines digitales de manera que solo son capaces de dar 100 mA a la vez, los cuales son los pines 0, 1, 2, 3 y 4, y 100 mA a los pines restantes, los pines 5, 6, 7, 8, 9, 10, 11, 12 y 13. Con lo que como máximo se pueden tener hasta 10 pines dando 20 mA a la vez. Algunos pines están reservados para diversos usos:

- **6 entradas analógicas:** pines etiquetados como A0, A1, A2, A3, A4 y A5. Admiten voltajes en unos rangos continuos entre 0 y 5 V. Sin embargo, el C.I del Arduino únicamente trabaja con valores digitales, así pues, se necesita antes convertir el valor analógico recibido en un valor digital lo más cercano posible. Esto es posible ya que la placa posee un circuito que convierte los valores analógicos/digitales.
- **6 salidas analógicas PWN:** son los pines números 3, 5, 6, 9, 10 y 11. Proveen una salida de 8 bits en modo PWN.
- **Serie:** pin 0 (RX) pin 1 (TX). Utilizados para recibir (RX) y transmitir (TX) directamente datos en serie sin la conversión USB a Serie que se realiza en el chip ATmega328P. En la placa vienen incrustados dos leds nombrados TX y RX, sin embargo, a la hora de transmitir o recibir datos del pin 0 y 1, estos pines no se activan. Su activación solo ocurre cuando estos datos vienen de una conexión USB a través del chip incorporado.
- **Pin 13:** este pin se conecta a un led que está incrustado en la placa, y tiene como nombre la etiqueta L. Este led se enciende cuando el pin recibe un voltaje y su valor es High o Alto, en caso contrario, al recibir el valor como Low o bajo, el led se apaga.

Se disponen también de dos pines hembra con su entrada analógica los cuales cumplen con una función determinada aparte de la habitual:

- **I2C:** pin A4 (SDA) y A5 (SCL). Estos pines se usan con el objetivo de conectar aquellos dispositivos con los que se quieren llevar a cabo comunicaciones a través del protocolo I2C (del inglés *Inter-Integrated Circuit*), que significa inter circuitos integrado, utilizando la librería Wire [16].

A continuación, en la Figura 25, podemos observar los principales elementos descritos anteriormente junto con otros que componen la placa Arduino UNO.

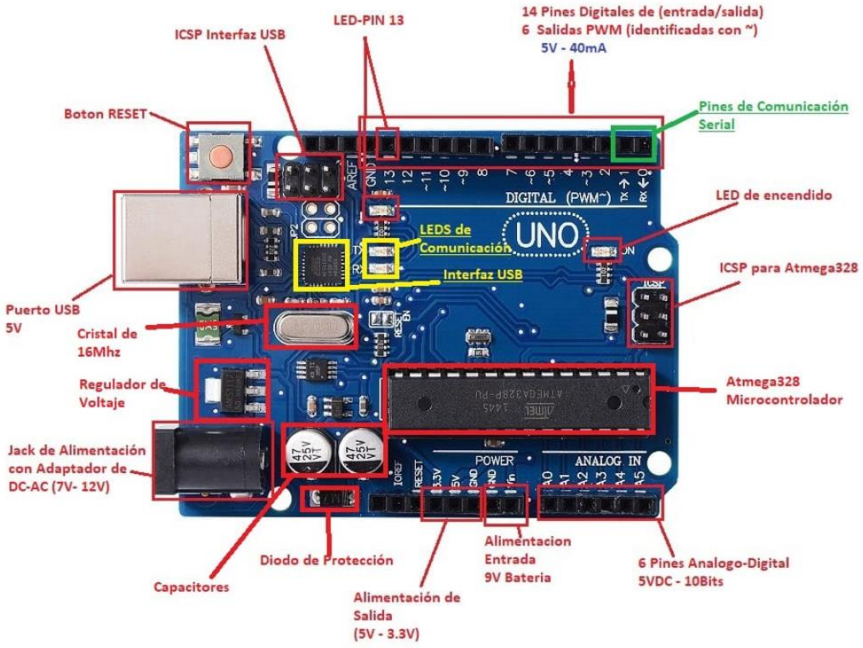


Figura 25. Principales elementos de la placa Arduino Uno. Fuente: https://tecmikro.com/content/17-arduino-uno-r3-caracteristicas.

2.3.3 Entorno de desarrollo

El entorno de desarrollo (IDE, de inglés *Integrated Development Environment*) de Arduino, como cualquier otro IDE, es básicamente una manera de invocar al conjunto de herramientas software que concede a los programadores el poder desarrollar (escribir y probar) sus propios programas. En nuestro caso, Arduino, necesitamos un IDE con el que podamos escribir y editar nuestro programa, a esto en el mundo de Arduino se le suele llamar “sketch”. Con ello podemos verificar que no tenemos errores

y que cuando tengamos la seguridad de que el sketch está correctamente programado, grabarlo en la memoria del microcontrolador y por tanto que el programa se ejecute autónomamente.

La plataforma Arduino utiliza un lenguaje basado en C/C++, soportando las funciones del estándar C y algunas de C++. También permite usar otros lenguajes de programación y aplicaciones como Java, Processing, Python, Matlab, Visual Basic, etc. ya que Arduino utiliza la comunicación mediante la transmisión de datos en serie, y los lenguajes citados anteriormente lo soportan. Es posible utilizar software intermediario permitiendo una comunicación fluida para los casos en que no sea posible utilizar el formato serie que viene nativo. Con esta variedad de lenguajes y sistemas podemos interactuar con el Arduino teniendo una amplia gama de posibilidades, ya que según nuestra necesidad y el problema que vayamos a solucionar podemos usar la gran compatibilidad que nos ofrece a la hora de comunicarnos.

El entorno de desarrollo Arduino es bastante sencillo además de intuitivo. Su descarga es proporcionada gratuitamente a través de su página oficial [14] y está disponible para distintos sistemas operativos como hemos comentado. Ha sido implementado mediante el lenguaje Processing. Su última versión es la 1.8.2, que es la que se ha usado en este proyecto. Dispone de una serie de menús, barra de herramientas para las funciones comunes, el editor de texto, un área de mensajes y una consola de texto. En la Figura 26 se puede apreciar el software de Arduino. Una vez instalado el IDE que nos proporciona los fabricantes del proyecto Arduino ya podemos empezar a desarrollar nuestros propios sketches. Este IDE también nos facilita un sistema de gestión de librerías y placas.

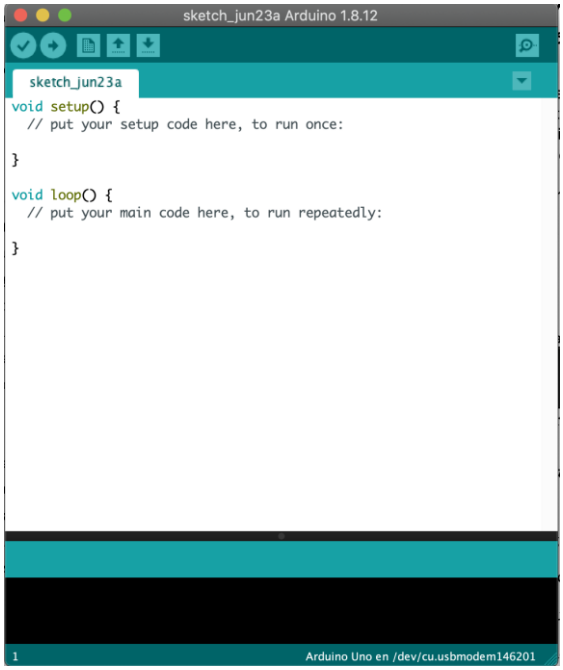


Figura 26. IDE Arduino.

Para que la placa Arduino sea reconocida por el IDE y podamos cargar su código en ella es necesario conectar el puerto USB (a través de un cable tipo A o B) que viene incorporado en la placa al puerto USB que dispone la computadora utilizada. Una vez realizada la conexión, para asegurarnos de que se ha reconocido la placa, nos dirigimos al menú superior y seleccionamos la opción *Herramientas* seguido de *Puerto* y verificamos que hay un dispositivo conectado en *COM**, comprobando también que la tarjeta Arduino seleccionada es la que queremos.

La estructura de un sketch es la siguiente:

- Introducir bibliotecas o librerías que se quieran utilizar.
- Inicialización de las variables del programa
- **void setup ():** esta función solo se ejecuta una vez al inicio del programa. Con ella se asignan como entrada o salida cada pin de la placa que se va a utilizar entre otras cosas.
- **void loop ():** en esta función es donde pondremos el código principal del programa. Se caracteriza por ejecutarse continuamente en bucle.

Capítulo 3. DESCRIPCIÓN DEL SISTEMA

En este capítulo se describe el sistema propuesto, se presentan las distintas formas y opciones de configuración del diseño y su modo de operación previo a la implementación del prototipo. Posteriormente se detallan los componentes hardware a utilizar para llevar a cabo su implementación.

3.1 Introducción

Una vez estudiado los fundamentos teóricos vistos en el segundo capítulo de este documento y ya conocida la configuración y modos de operación de las redes ZigBee, así como la de los módulos y el microcomputador Arduino UNO que vamos a utilizar en nuestro sistema de pruebas XBee®, pasamos a describir las bases del sistema propuesto.

Para ello, previamente debe realizarse un análisis previo detallado antes de realizar el diseño y configuración, así podremos ver de qué manera se puede adaptar y configurar el sistema para alcanzar el objetivo final propuesto. Como se comentó en la introducción, la idea básica consiste en posibilitar que diferentes dispositivos programados convenientemente, emitan información de aviso, alarma o anuncio para que cualquier potencial receptor capte dichos mensajes y actúe en consecuencia, según su uso.

Como las redes ZigBee constituyen un sistema de comunicación de corto alcance que permiten el intercambio de mensajes sin requisitos de alta velocidad, se considera que nuestro sistema multipropósito de aviso y anuncio debe plantear una solución “uno a muchos” y puede ser implementado con esta tecnología y con los módulos XBee®. Como no pretendemos alterar el estándar, nuestra propuesta consiste en que sea un coordinador el que emita mensajes en modo difusión y los potenciales receptores sean dispositivos aislados y autónomos, que al pasar por la zona de cobertura capten los mensajes tras la correspondiente asociación y vinculación de los mismos. En la Figura 27 se puede observar la idea del sistema propuesto. Se considera importante resaltar que el modelo de uso del dispositivo ZigBee suelen operar de manera inversa, los emisores son los dispositivos finales que captan generalmente información del entorno y el coordinador los recibe y procesa o retransmite a Internet (generalmente un servidor IoT en la nube).

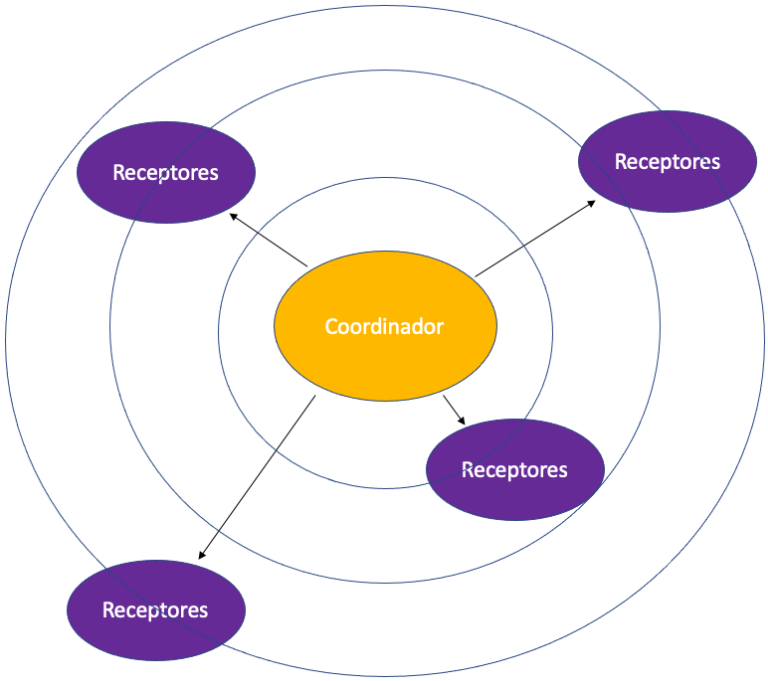


Figura 27. Modelo conexión: coordinador - receptores.

Para el estudio y diseño de la red de nuestro sistema, el nodo que transmite la información será nuestro dispositivo coordinador de la red y el nodo o nodos receptores de estos datos actuarán como dispositivos routers. Como hemos visto en el capítulo anterior, el coordinador se encargará de iniciar y mantener la red ZigBee, debe de haber como mínimo uno y su función es la de administrar las acciones de control requeridas por el usuario y transmitir las ordenes de control a los demás nodos de la red. Los nodos receptores de esta red se encargarán de procesar los mensajes enviados por dicho coordinador.

El aspecto del consumo es de suma importancia, ya que en general los receptores suelen estar alimentados por una simple batería (gran durabilidad) y ocupar el mínimo espacio posible. Además, si se pretende hacer uso del sistema en cualquier lugar, pudiendo ser zonas aisladas, se justifica buscar soluciones de bajo consumo, tanto para dispositivos coordinadores como dispositivos routers. Evidentemente, si se incorpora un µC Arduino ese problema se acrecienta (aumento del consumo). Por tanto, habrá que seleccionar la mejor solución que se adapte a cada caso de entre las diferentes que proponemos. Si bien los módulos XBee® como se vio, pueden ser alimentados con baterías simples, el Arduino requiere recursos de mayores exigencias

si se desea un tiempo razonable de uso sin requerir recargar o cambios de batería. Evidentemente, si la ubicación de una genérica implementación o despliegue permite acceder a la red eléctrica estos problemas se ven reducidos e incluso se puede dotar de manera ideal con sistemas de alimentación combinados. El estudio o diseño del mejor sistema de alimentación no se ha planteado en este proyecto, pues puede haber múltiples alternativas; si hemos analizado que las diferentes configuraciones tendrán una dependencia de características de consumo y requisitos de alimentación eléctrica para cada implementación. Esto se ve incrementado, según se hagan uso de sensores y otros dispositivos externos en el coordinador que habría que ver qué incremento producen en el consumo. Como todos los casos no se pueden considerar, si se ha realizado algunas consideraciones sobre este tema a la hora de especificar varias alternativas. Para nuestra implementación a modo de prototipo del sistema, se ha utilizado conexiones USB (+5 voltios de tensión) directas a través de una computadora portátil para alimentar los nodos utilizados.

3.2 Configuración

Como se ha indicado en el apartado anterior, partimos de la idea de que el coordinador va a transmitir de forma regular alguna información simple para que el o los receptores (routers) XBee® capten la información y sean notificados. Como se expuso en la descripción de la tecnología ZigBee, su uso está orientado a mensajes cortos de sensores u otros dispositivos, teniendo limitaciones intrínsecas para gestionar mensajes pesados en cuando a volumen de información y elevados requisitos de velocidad, por lo que no se les puede dotar de recursos hardware de entrada y salida complejos o de elevado consumo como por ejemplo puerto USB, HDMI para vídeo, entrada y salida de audio u otros. La información a transmitir en cualquier aplicación de nuestro sistema sería simple, precisa y de tamaño muy reducido (campo de datos de tramas API muy pequeñas); incluso teniendo que recurrir a notificaciones visuales (basadas en simples leds o displays de 7 segmentos), sonoras (con simples zumbadores, o similares) u otras. Si se requieren mensajes más complejos y de mayor cobertura, esta tecnología no sería la requerida, sino Bluetooth, WiFi, LTE/5G, etc.

Para cubrir el aspecto multipropósito o un mayor número de potenciales usos de este sistema, se ha procedido a configurarlo de tres formas diferentes según sus necesidades. Estas tres formas las vamos a denominar *configuración básica*, *configuración ampliada en el coordinador* y *configuración completa*. A continuación, se detallan el diseño de cada uno de ellos, así como su funcionamiento. Las dos últimas configuraciones requieren el uso de microcomputador u ordenador personal con los consiguientes cambios en el código y conexionado.

3.2.1 Configuración básica

Este tipo de configuración está formado únicamente por módulos XBee®. Tanto el nodo transmisor como el nodo receptor estarán equipados por estos módulos. En la Figura 28 vemos una ilustración de la configuración.

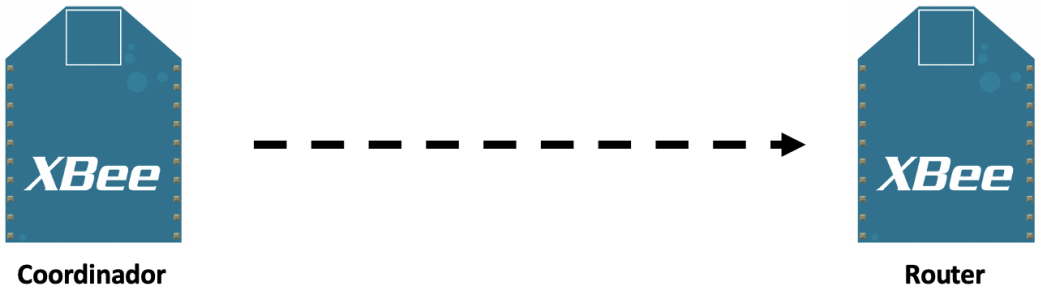


Figura 28. Configuración básica.

Usando esta configuración en la que la comunicación es exclusivamente entre módulos XBee®, la funcionalidad que otorga es la que ya viene limitada por el propio software (firmware) y hardware del mismo. En esta configuración no se requiere microcontrolador y el módulo transmitirá con sus propios medios, la información que se le suministre desde el exterior o pueda recibir del exterior por interacción del administrador de la red ZigBee. Un ejemplo puede ser la lectura periódica de sensores conectados a sus líneas de entrada analógica (por ejemplo, un sensor de temperatura), digital o mediante activación por interrupciones (detección de nivel o

cambio de valor de señal). Esta configuración está muy limitada en cuanto a sus posibilidades de multiuso (aspecto multipropósito) y variedad de señales de entrada (reducido número de ellas y características, véase Tabla 3), requiriendo de gestionar directamente las señales de entrada (sensores externos) y preprogramar los mensajes asociados usando el software propietario o vía telecomandos. Por el contrario, tiene la ventaja de ser la que menos consumo genera.

3.2.2 Configuración ampliada en el coordinador

En el nodo transmisor se incluye una placa Arduino UNO, que se interconecta con el módulo XBee® y en el nodo receptor únicamente el módulo XBee®.

Figura 29. Configuración ampliada en el coordinador.

En este caso el coordinador al contar con el µC, la posible ejecución de código en el mismo y el gobierno que pueda hacerse del nodo transmisor lo dota de un potencial mucho mayor, tanto en recursos de entrada y salida como de almacenamiento de datos y transferencia a otros sistemas remotos (ordenador personal, uso de Internet, etc.). En cuanto al receptor, sigue manteniendo las mismas posibilidades que la configuración anterior, en es especial la ventaja de bajo consumo y ubicuidad. Evidentemente, al ser los nodos receptores dispositivos router simples, estos presentan las mismas limitaciones de no contar con dispositivos de salida visuales complejos. Esta configuración puede resultar de interés para receptores móviles (por ejemplo, personas caminando) y con limitadas capacidades de alimentación por peso, espacio o recursos.

3.2.3 Configuración completa

Tanto el nodo transmisor como el receptor están formados por una placa Arduino UNO junto con el módulo XBee® incorporado.

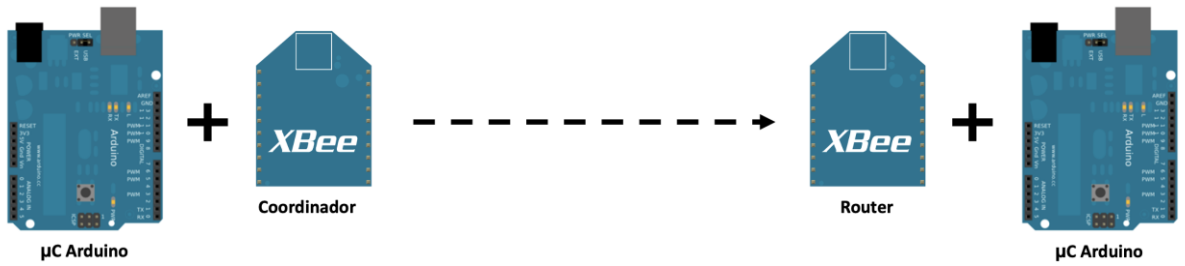


Figura 30. Configuración completa.

Esta es la configuración más completa y con mayor número de posibilidades para el sistema propuesto. Esta configuración introduce frente a la anterior tanto ventajas como inconvenientes a los nodos receptores. Las ventajas evidentes son la capacidad de procesado, almacenamiento más complejo y ampliado, así como mejores posibilidades de salida de la información recibida. Sin embargo, el consumo que genera y el espacio que ocupa pueden llegar a ser inconvenientes a tener en cuenta. Enfocándonos en un caso práctico como por ejemplo, personas que caminan, puede ser complicado que incorporen el modulo receptor junto con la placa Arduino y la batería que necesitan. Por tanto, las configuraciones básica y ampliada estarían mejor adaptadas a ser de aplicación para este tipo de usuarios finales. El caso que si puede considerarse para esta configuración sería por ejemplo para un ciclista (puede incorporar baterías más potentes), incluido en un vehículo usando la energía propia de sus baterías (con las correspondientes adaptaciones) o en sistemas destino fijos y alimentados con energía de red eléctrica. Nos centraremos en detallar la configuración del sistema en modo completo al considerar que es la que más dificultad presenta y capacidad multipropósito.

3.2.3.1 Descripción configuración completa

Dado que la configuración completa incluye el µC Arduino, se debe tener presente que este requiere de un programa, elaborado adhoc que gobierne al módulo XBee® y este último se encargue de la emisión por RF de los mensajes que debe recibir el receptor. Por tanto, se tendrá que especificar qué información (puede ser fija o pregrabada, dinámica según las condiciones que presente interna o externamente; y/o el estado de una o varias variables tomadas del exterior) debe procesar el µC para su captura (si fuera necesario) y su envío por las líneas serie junto con las tramas API correspondientes al XBee® Transmisor (dispositivo coordinador). Como punto de partida de nuestro sistema y ejemplo de aplicación, se considera que el módulo transmisor no debe requerir la interacción con operario alguno. O sea, que el transmisor esté configurado o programado para emitir constantemente un mensaje o unos datos determinados, se ubique el lugar deseado, haga su función hasta que se reprograme para otro uso o ubicación, permitiendo cierta configuración y diferentes usos. Para el caso del prototipo a implementar, se materializará y simulará mediante el uso de jumpers o un DIP Switch de configuración; según sus valores activará uno u otro mensaje que especificará como operar tanto al µC Arduino como coordinador y que información a enviar a los receptores vía RF. En el caso de los receptores, como solo será destinos finales y no emisores de información, tampoco será necesario ninguna acción, que aunque sean personas a las que van dirigidos los mensajes, estos sean solo indicaciones visuales y/o sonoras; y cuando se cuente con recursos adecuados más sofisticados, como display LCD y otras similitudes para poder mostrarlos convenientemente. En cada caso, dependiendo del tipo y recursos disponibles del receptor, habrá que adaptar tanto los mensajes como el código del receptor.

En este caso, el nodo transmisor lo vamos a dotar de tres partes:

- Microcontrolador Arduino.
- Interruptores o jumpers de configuración (modo práctico de posibilitar varios usos).
- Dispositivo XBee®.

El Arduino será programado para captar los datos externos y/o preprogramados y enviarlos (como mensajes) al dispositivo XBee® por el puerto serie, para posteriormente transmitir dichos mensajes por RF a cualquier nodo que pueda recibirlo (en zona de cobertura). El dispositivo XBee® habrá sido programado necesariamente como coordinador pues es el elemento central que difunde los avisos, anuncios, notificaciones, etc.

El nodo receptor tiene tres módulos diferentes:

- Microcontrolador Arduino.
- Dispositivo XBee®.
- Componentes de salida (leds, zumbadores, etc.).

El Arduino está programado para recibir las tramas API que les entrega por el puerto serie el dispositivo XBee® receptor, que a su vez este habrá recibido vía RF y convertida en datos de salida. El tratamiento y análisis detallado de los datos recibidos, como parte de los datos de los mensajes (tramas API) por el dispositivo XBee®, se realizará en el µC para su comprobación de error, mensaje valido soportado y su representación, según cada caso, que dependerá de los dispositivos de salida disponible. Para nuestro prototipo y a modo de prueba se contará con indicación visual (mediante leds), sonora (zumbador) o mediante display LCD. El dispositivo XBee® habrá sido programado como router y de manera muy especial, configurado para unirse a la PAN que el coordinador haya prefijado por el administrador de la subred o celda. Este aspecto debe ser conocido por todos los receptores que quieran participar de este sistema de anuncio dado que este valor en el estándar debe ser definido por el administrador de la red y no se difunde (debe ser reprogramado), a diferencia de la tecnología WiFi, que su denominación sería el equivalente al identificador SSID (del inglés *Service Set Identifier*), donde suele divulgarse.

El sistema propuesto, en cualquiera de sus tres configuraciones puede ser replicado de forma aislada sin interacción entre ellos, bajo un mismo uso o diferentes en una misma área geográfica. Es decir, allí donde se requiera notificar algún aviso o alerta

se configurará por un administrador en cada coordinador y con ello se habilita a los potenciales receptores el recibo de notificaciones, y consecuentemente serán procesados con los medios que cada receptor disponga, ya sea de forma luminosa, sonora, display de caracteres (LCD); y excepcionalmente un monitor u otras presentaciones. En la Figura 31 se muestra el esquema del sistema replicado con las tres configuraciones, representando celdas independientes con diferentes formas de alimentar el sistema y potencial formas de salida tales como leds, zumbadores, etc.

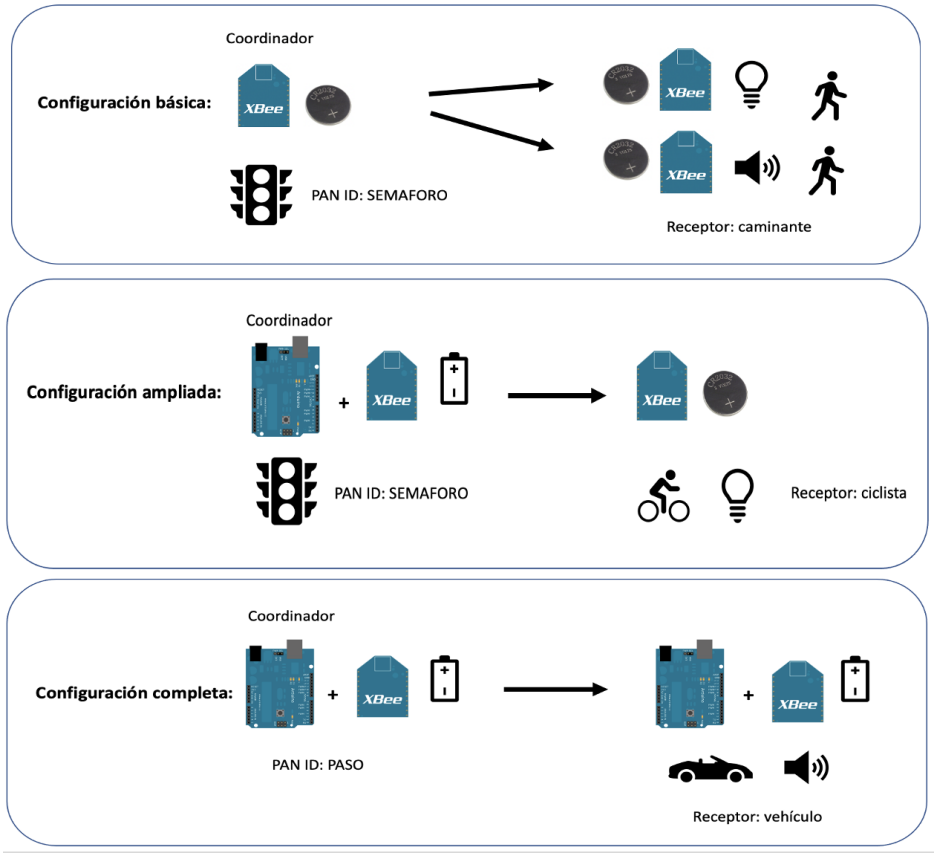


Figura 31. Sistema replicado con las tres configuraciones en casos prácticos.

Si bien los módulos XBee® pueden ser una fuente de datos captadas desde sensores externos y emisores de salidas locales, no pueden programarse para realizar un procesamiento de información lógica dada sus limitaciones tecnológicas y firmware, así mismo bien pueden ser programados para enviar datos de forma repetitiva. Si la información de un sensor requiere ser procesada o sea tomada alguna decisión local, será necesario agregar un microcontrolador para manejar esos procesos, es decir, configuración ampliada o completa. El objetivo y ventaja de implementar el sistema

con un µC, y en especial un Arduino, es desarrollar una técnica que permita su configuración y emisión de mensajes diferenciados de manera remota para cualquier sistema de aviso y anuncio. Evidentemente, para cada necesidad o despliegue concreto, sería necesario revisar diferentes aspectos, el primero corresponde con la captura de datos del exterior si es el caso y lo que motiva o define la notificación, segundo la comunicación entre el coordinador con la placa Arduino y tercero corresponde a adaptar su programación y montaje según su aplicación.

Para cada aplicación y uso del sistema se tendría un modelo de funcionamiento y operación concreto multipropósito y replicable en diferentes celdas, ofreciendo la posibilidad de enviar diferentes datos provenientes de varios sensores o dispositivos de entrada en origen y de activar varios dispositivos de salidas en destino. Con cualquier µC, y en particular con Arduino se puede implementar una variedad de sistemas en comparación con el uso solamente de los módulos XBee® con las restricciones indicadas.

3.3 Componentes hardware. Descripción de recursos

Previamente a describir e implementar en un prototipo el sistema propuesto, se expondrán aquellos componentes que se utilizarán en el mismo, y así poder llevar cabo el montaje y la fase de pruebas posteriormente. Los módulos XBee® y las placas Arduino utilizadas se han descrito previamente en la sección 2.2.2 y 2.3.2 del segundo capítulo de este documento. A continuación, se detallan los demás componentes pasivos/activos básicos restantes. En cuanto a los periféricos se describirán de forma básica los que hemos considerado útiles (la cantidad de ellos y sus posibilidades son muy amplias) y serán los utilizados en el prototipo.

- **Arduino Starter Kit.** Para la elaboración de este proyecto se han utilizado 2 kits de desarrollo *Arduino Starter Kit* proporcionados por parte de la universidad (Telemática). Estos kits destacan por su simplicidad, ya que incluyen el microcontrolador Arduino y diferentes componentes electrónicos tales como resistencias, jumpers, sensores, entrada-salidas, puertos series y demás

periféricos. De entre todos los disponibles se describirán solo los componentes que se han escogido para realizar la implementación del sistema.

Figura 32. Kit de desarrollo Arduino Starter Kit.

- **Adaptador XBee Shield.** Se trata de una placa que sirve como puente entre el µC Arduino y el módulo XBee®. Con este módulo es posible interconectar las dos placas proporcionándonos una conexión segura además de trabajar de una manera adecuada. Incorpora un switch SPDT (del inglés *Single Pole Double. Throw*) que nos permite ir seleccionando si queremos conectarlo a la UART (D0, D1) o a los pines digitales 2 y 3 del Arduino. En cuanto a la alimentación, la recibe del Arduino por lo que son 5V y también dispone de un regulador o estabilizador del voltaje de 3.3V pudiendo alimentar al módulo XBee®.

Figura 33. Módulo XBee Shield V3.

- **XBee USB Adapter.** Permite conectar y utilizar un módulo XBee® directamente mediante un puerto USB. Dispone de un conversor USB a Serial, que hace posible convertir o traducir los datos de un formato a otro, entre nuestro PC y el módulo XBee®. Conectando el módulo XBee® a esta placa, será la manera

de poder interactuar con el y configurar los módulos a través del software XCTU. Con ello podremos modificar y establecer los parámetros necesarios con el fin de aplicarlos según cada caso.

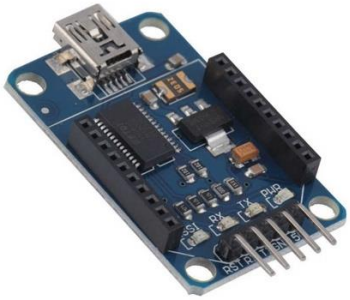


Figura 34. XBee USB Adapter.

Los elementos pasivos más básicos y de menos consumo considerados son:

- **Leds.** Mediante uno o varios diodos led podemos facilitar información visual a los nodos destinatario.

Figura 35. Diodo led común.

- **Display LCD.** Del inglés *Liquid Crystal Display*. Muestra información más detallada, ya sea textual o gráfica con restricciones, en particular, mensajes de texto o gráficos básicos en 7 segmentos o multisegmentos. El display disponible y utilizado para nuestro prototipo es el formato 16x2, es decir, 16 caracteres x 2 líneas como el mostrado en la Figura 36.

Figura 36. Display LCD 16x2.

- **Zumbador.** También llamado buzzer, son dispositivos que emiten una onda de sonido tras la conversión de una señal eléctrica. Para nuestro prototipo utilizaremos un buzzer activo, que contiene un oscilador interno y únicamente basta con alimentarlo para reproducir el sonido.

Figura 37. Zumbador o buzzer activo.

- **DIP switch.** Del inglés *Dual In – Line Package*. Son conmutadores que si conectan por un extremo a un nivel lógico (0, 3v, o 5v) y por el otro a las entradas de un circuito digital puede tomarlos como 0 lógico o 1 lógico, tomando decisiones según sean los mismos. Estos dispositivos suele ser una alternativa a los jumpers. En nuestro prototipo hemos incorporado un DIP Switch 4 que consta de cuatro switchs, con ello consideramos que tenemos suficientes opciones (2^4 valores posibles); con estas 16 combinaciones podríamos preprogramar hasta 16 acciones o aplicaciones y mensajes diferentes asociados.

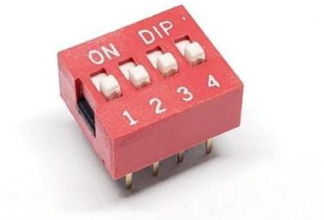


Figura 38. DIP Switch 4.

- **Breadboard.** Es una placa de pruebas, que consiste en un tablero lleno de orificios en los que a modo de hileras se encuentran conectados eléctricamente entre sí.

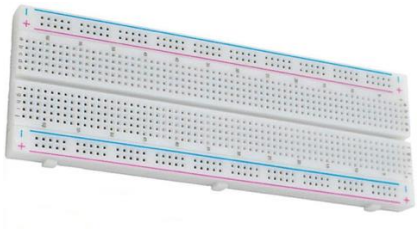


Figura 39. Breadboard MB-102.

3.3.1 Alimentación del sistema.

Para implementar el sistema, tanto a los Arduino como a los XBee® para que puedan funcionar y comunicarse, se requiere proporcionarles la alimentación necesaria. Según las especificaciones del fabricante [22], el XBee® S2C requiere una alimentación desde los 2.1 voltios hasta los 3.6 voltios de tensión continua. Mientras que el µC Arduino UNO, requiere un voltaje de funcionamiento de 5 voltios, consumiendo 46 mA en reposo y por sí mismo.

A continuación, se recogen las posibles alternativas de las que se disponen en el mercado para alimentar el sistema en cualquiera de sus tres configuraciones propuestas.

- **Conexión USB.** Arduino UNO dispone de un puerto USB (Tipo B de 4 pines) para comunicar el software de programación, monitor serie y línea DC (5 voltios), por el cual se podrá alimentar la placa. A través de este puerto de entrada, se admiten 5 voltios únicamente. Conectada esta línea USB con un ordenador personal o un cargador de 5 voltios (con salida conector USB tipo B), sería factible. Dado que tanto el transmisor como el receptor XBee® toman la tensión de la placa Arduino, con esta conexión se podrá alimentar ambos

circuitos sin temer por un exceso de consumo que estropee la línea USB. Este es el modo utilizado para realizar las pruebas en el prototipo.

En esta opción, para el caso de la configuración básica, habría que disponer del cable adecuado para extraer las líneas 5v/Gnd y convertirlas a 3v3/Gnd.

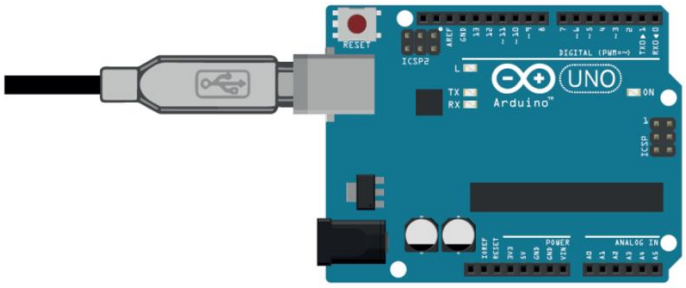


Figura 40. Arduino UNO alimentado por conexión USB.

- **Alimentación mediante baterías.** Mediante este tipo de alimentación no se depende de tomas de corriente; asimismo posibilita la movilidad para montarlo en cualquier parte. Las baterías suelen encontrarse de dos tipos principales: baterías de un solo uso, es decir, que se agotan y una vez usadas y gastadas hay que retirarlas no pudiendo recuperar su carga (desechables). La segunda opción son las recargables, estas nos permiten recuperar su carga y volver a almacenar la energía en ellas (se tendrá una limitación en la cantidad de veces que se puede recargar).

  Lo que nos interesa a la hora de la elección de nuestra batería, es la tensión de salida y los mA/h que suministran, que determinan en gran medida su duración. Por ejemplo, las pilas de 9 voltios suelen ser una de las opciones más extendidas; su desventaja es que tienen una baja capacidad energética, siendo de 500-600 mA y su tensión puede no ser adecuada para alimentar los sensores o actuadores a nuestro Arduino. Aunque su precio es reducido, a largo plazo puede que no resulten económicas. Además, como hay que alimentar el módulo XBee® se requiere estudiar la durabilidad de la misma al añadirse al consumo del µC Arduino el consumo del módulo XBee®; así mismo los sensores que pudieran usarse pueden consumir bastante y requerir convertidores (habría que estudiar cada caso).

Figura 41. Pila alcalina no recargable (9V).

Utilizar pilas alcalinas AA o AAA en serie, más específicamente, empleando cuatro pilas de 1.5 V en serie. El voltaje de 6 voltios es perfecto para alimentar algunos sensores/actuadores pero puede ser excesivamente bajo para atacar la placa Arduino. Cuatro pilas AA alcalinas convencionales pueden proporcionar una capacidad de 1700-2800 mAh, aunque hay muchas variantes. Otra opción es la de usar baterías de polímero de litio (LIPo, del inglés *Lithium Polymer*), es una opción más cara, pero tienen mayor duración, ya que la cantidad de mAh es mayor, llegando a los 4500 mAh y corrientes mayores a los 5 A. Existen de diferentes voltajes y capacidades, para nuestro prototipo, una opción podría ser la de 7.4 V y 2000 mAh, que será suficiente para alimentar tanto al µC Arduino como al módulo XBee® con sus correspondientes reductores de tensión. Al ser recargables, se requerirá de cargadores.

Figura 42. Batería LiPo 7.4V 2000mAh.

- **Adaptadores de corriente.** El µC Arduino UNO incorpora una entrada de alimentación mediante el jack estándar. Se utiliza para ello un adaptador de corriente (AC/DC). El voltaje utilizado suele ser entre 7 V y 12 V.

Figura 43. Cargador/fuente de alimentación.

- Otras opciones podrían ser combinaciones de paneles solares y otras energías que en combinación con baterías pueden hacer todavía más autónomos y duraderos tanto el transmisor como los receptores.

Capítulo 4. IMPLEMENTACIÓN DEL PROTOTIPO

En el capítulo anterior se expusieron las diferentes configuraciones del sistema propuesto y se detalló las características de cada una. En este capítulo se especifica como implementar el sistema en un prototipo, tanto del hardware como del software que alcance los objetivos propuestos inicialmente. Una vez implementado el prototipo, se procederá a realizar una fase de pruebas de su funcionamiento y finalmente mostrar los resultados obtenidos.

4.1 Montaje

La implementación de un prototipo para este sistema multipropósito de aviso y anuncio se hará en el modo de configuración completa visto en el capítulo anterior, que se divide en dos partes. La primera parte tendrá un módulo XBee® configurado como coordinador, un módulo Arduino Uno y un interruptor DIP Switch 4. En la segunda parte del montaje se tendrá otro módulo XBee® configurado como router, conectado a otro módulo Arduino UNO, un led estándar, un display LCD 16x2 para visualizar los mensajes y un buzzer activo para emitir la alarma sonora. En cuanto a la utilidad, se habilitará que desde el emisor se pre-configure con switchs que mensaje se desea emitir y el receptor, tras su procesado y dependiendo de los dispositivos de salida que disponga, se visualice en led, se muestre en modo caracteres en un display o en un aviso sonoro. El prototipo usará, tanto en el emisor (µC Arduino + Coordinador) como en Receptor (µC Arduino + Router) como alimentación la obtenida de la salida USB como se ha mencionado anteriormente, que estará conectada a una computadora portátil.

4.1.1 Circuito Emisor

Esta parte del sistema y prototipo será el “cerebro del sistema”, y es la parte encargada de controlar y coordinar todo el sistema. Se encarga de procesar y adaptar o crear los diferentes mensajes a enviar, según la configuración por switchs. Estos mensajes deberán ser enviados a los nodos remotos para que se ejecute la acción deseada en el receptor. El circuito estará compuesto por el módulo XBee® S2C conectado a su vez al µC Arduino. Para conectar ambas partes se utilizará la placa XBee Shield, la cual permite acoplar cualquier módulo XBee® sobre el Arduino y conectar las líneas serie necesarias para la comunicación entre ambos (TxD y RxD). En la Figura 44 se puede ver el diseño de este circuito emisor-coordinador.

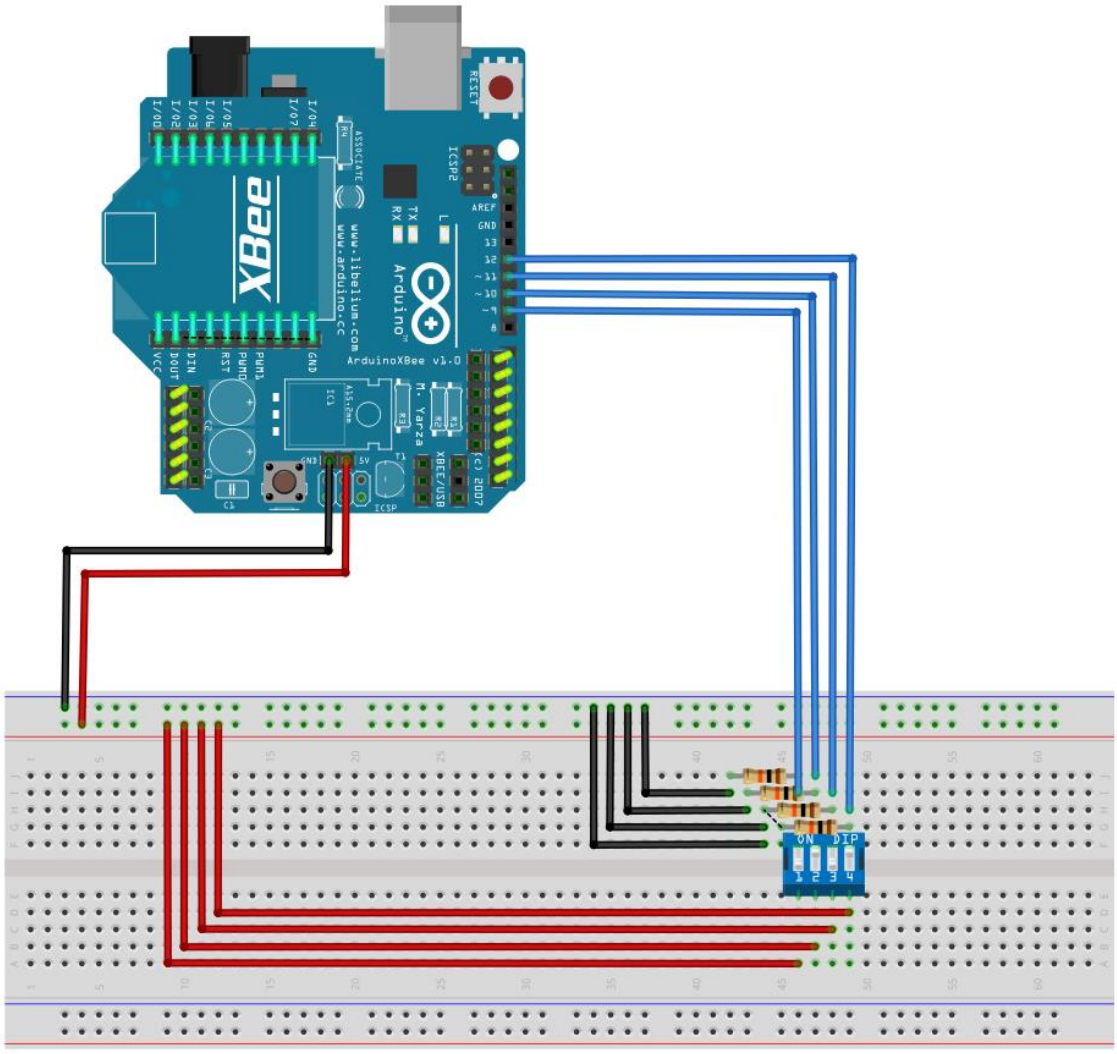


Figura 44. Circuito Emisor: montaje en protoboard modo diseño.

Dispondremos de una protoboard para colocar el DIP switch y los diferentes componentes pasivos necesarios para su funcionamiento y cableado correspondiente.

Al utilizar el adaptador XBee Shield simplemente conectamos el módulo XBee® en los pines habilitados que nos facilita la placa y posteriormente conectamos el adaptador junto con el módulo XBee® al módulo Arduino. En la Figura 45 se muestra una fotografía real donde se aprecia el shield ubicado encima de la placa Arduino y el coordinador en el zócalo especial correspondiente. Para ello también se nos facilita su conexión ya que únicamente deberemos conectar las patillas del adaptador en los pines hembra habilitados. El voltaje y la masa irán conectados en los pines del

adaptador disponibles para ello mientras que la conexión del DIP Switch 4 tiene que efectuarse en los pines correctos para activar su funcionamiento y así poder alternar entre las distintas opciones que integremos en su programación. Para ello se han utilizado los pines 9, 10, 11 y 12 del módulo Arduino que irán conectados al switch programados como entradas digitales.

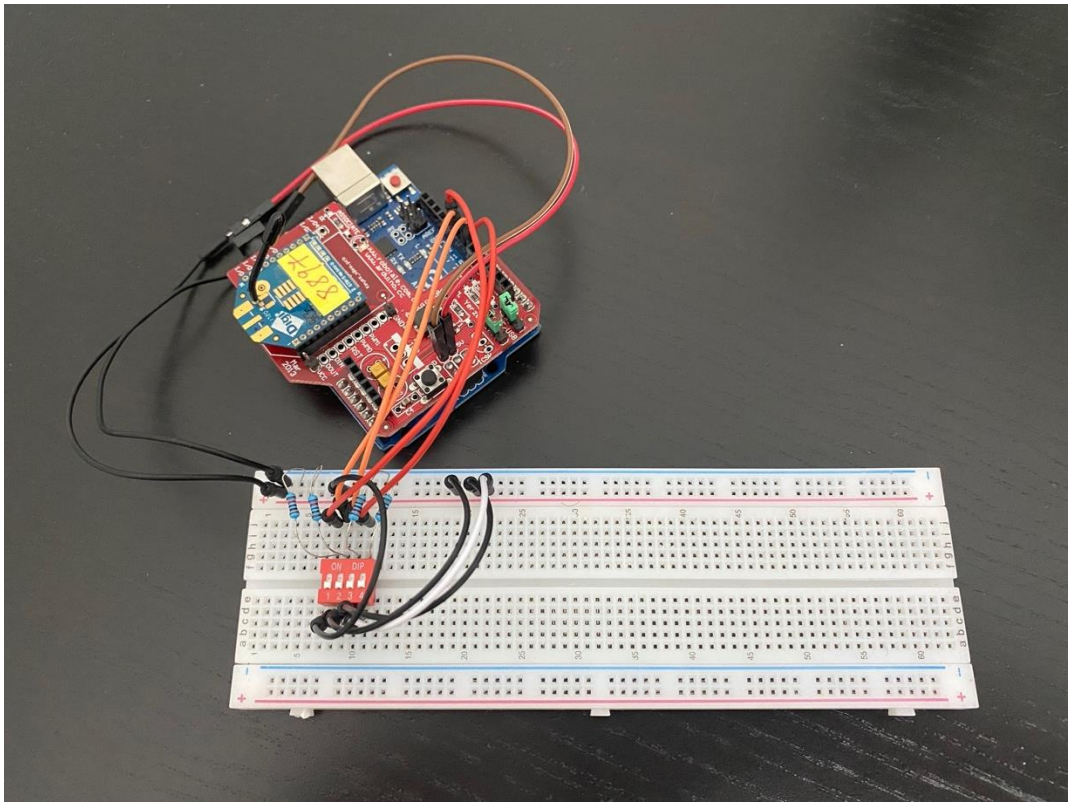


Figura 45. Circuito Emisor: montaje real en protoboard.

4.1.2 Circuito Receptor

Una vez sea enviada toda la información desde el nodo emisor, el circuito receptor será el encargado de recibir los datos y procesarlos para realizar una u otra acción según se especifique en los mensajes. Estará formado por un módulo Arduino UNO conectado a un equipo informático, que servirá como fuente de alimentación, y el Arduino a su vez conectado al módulo XBee®. Para conectar ambas partes se utilizará la placa XBee Shield, la cual permite acoplar el módulo XBee® sobre el Arduino. En la Figura 46 se puede ver el diseño de este circuito.

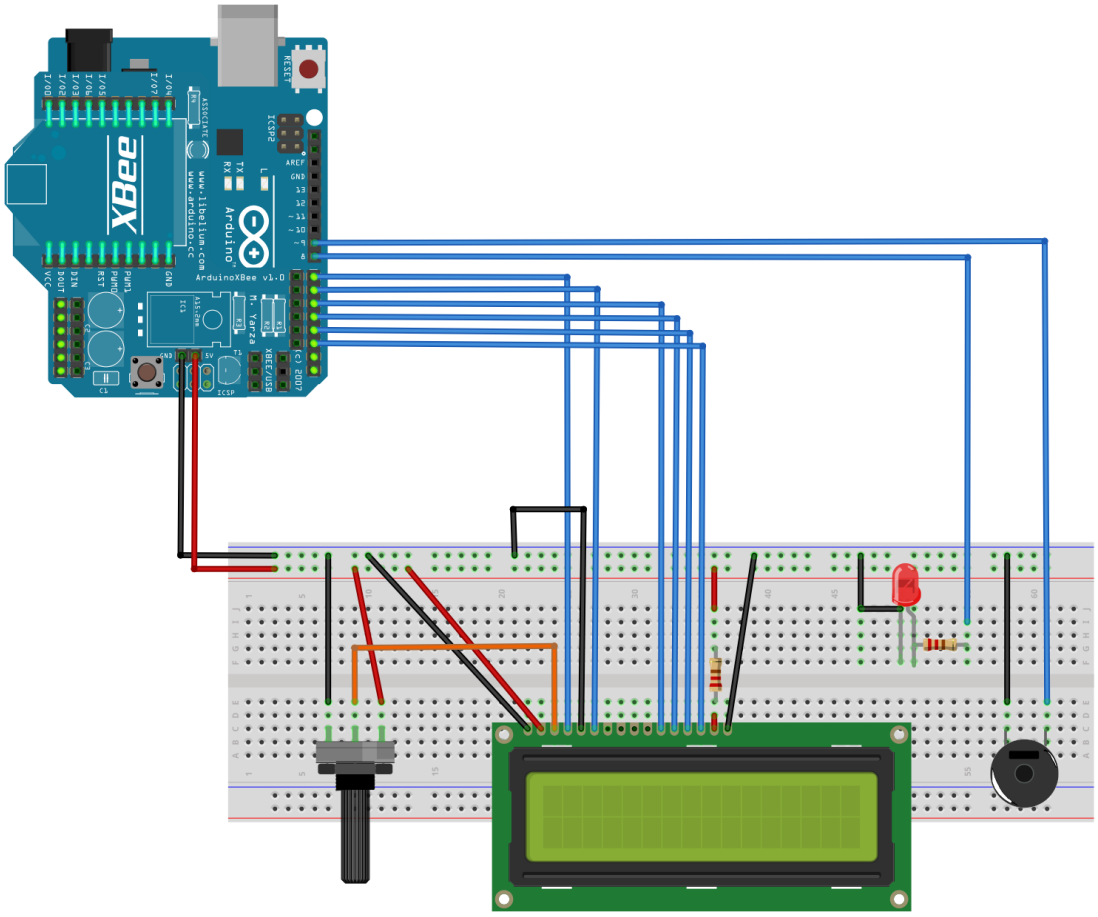


Figura 46. Circuito Receptor: montaje en protoboard.

Al ser una implementación de configuración completa, en la protoboard conectaremos los diferentes componentes, así como el led, el display LCD y zumbador, que nos mostrarán los resultados deseados según la acción implementada desde el nodo emisor.

La conexión del adaptador Shield XBee y el módulo XBee® a la placa Arduino se hará de forma similar a la del circuito emisor detallada anteriormente. A la hora de conectar el display LCD con la placa se tendrá que hacer en los pines correctos para así poder activar el display y visualizar los mensajes (véase Tabla 8).

Tabla 8. Conexiones del LCD 16x2.

Pines LCD	Conexión
1	GND
2	VCC
3	Potenciómetro 10 KΩ
4	Pin 7 Arduino
5	GND
6	Pin 6 Arduino
7	-
8	-
9	-
10	-
11	Pin 2 Arduino
12	Pin 3 Arduino
13	Pin 4 Arduino
14	Pin 5 Arduino
15	GND
16	VCC

En su conexión, los pines 1 y 2 corresponden a los de masa y voltaje (GND y VCC). El pin 3 está conectado a un potenciómetro de 10KΩ que se utilizará para variar el contraste del LCD hasta ajustarlo donde se pueda observar una imagen correcta. El pin 4 (RS) irá conectado al pin 7 del módulo y corresponde con el selector del registro, es decir, para mandar información al LCD de lo que se quiere hacer. El pin 5 (RW) se conecta a masa para indicar que queremos escribir. El pin 6 (EN) corresponde a Enable, con esto se activa la pantalla para que reciba la información, va conectado al pin 6 de la placa Arduino. Los pines 7 al 14 se utilizan para datos de 8 bit, en nuestro caso solo necesitamos 4 bits, por lo que utilizamos del pin 11 al 14, que irán conectados al pin 2, 3, 4 y 5 de la placa Arduino. Los pines 15 y 16 se utilizan para la luz de fondo, estos van conectados a GND y VCC.

En cuanto a la conexión del led, este irá conectado al pin 8 de la placa que habrá sido programa como salida digital. Por último, conectaremos el zumbador, que al tratarse de un buzzer activo, simplemente conectaremos el terminal negativo a masa, mientras

que la parte positiva se conectará al pin 9 correspondiente del Arduino e igualmente programado como salida digital. Al estar alimentado entre 5V y GND, este sonará a una frecuencia fija. En la Figura 47 se muestra una fotografía real.

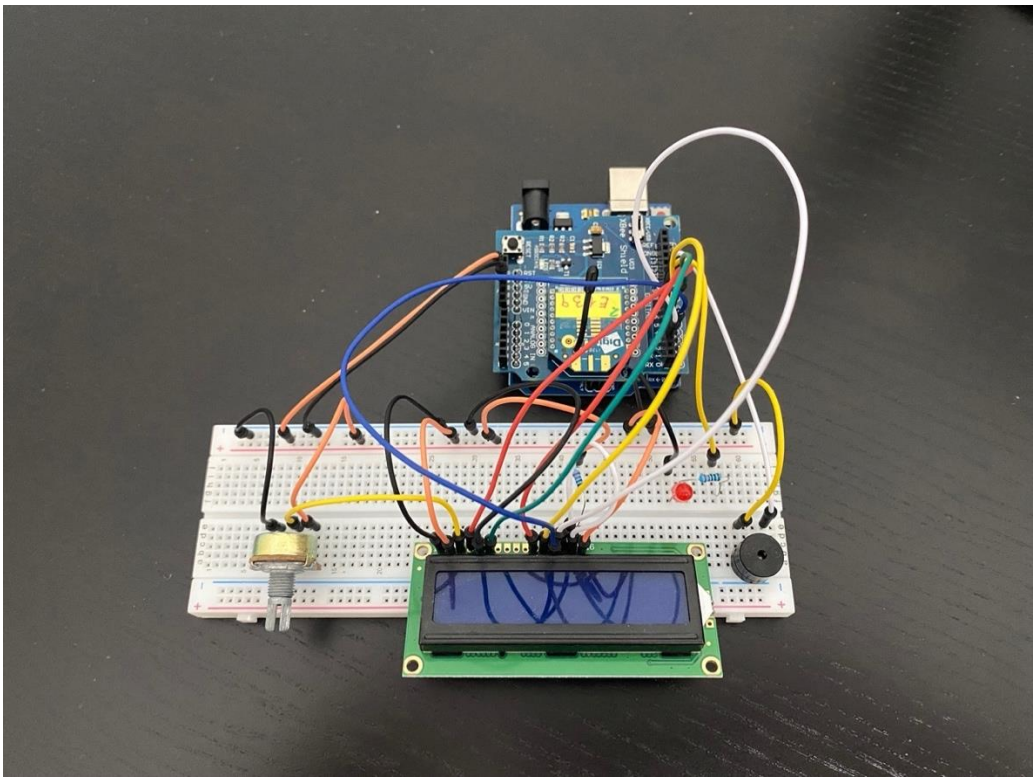


Figura 47. Circuito Receptor: montaje real en protoboard.

4.2 Configuración de los módulos XBee® S2C

A continuación, estableceremos la comunicación entre los módulos XBee®, por lo tanto, se procederá a configurar cada uno de los módulos para su correcta emisión y recepción por medio de radiofrecuencia y su vinculación IEEE 802.15.4. En este proyecto disponemos de tres módulos XBee® S2C, por lo tanto, para simular posibles casos, la red estará configurada en un principio por dos módulos en los que uno tendrá el rol de coordinador de la red y el otro módulo desempañará la función de router y posteriormente habilitaremos el tercer módulo para su configuración en modo básico, combinando dos configuraciones. Con la primera combinación (solo configuración completa) se harán las pruebas necesarias para lograr el intercambio favorable de

mensajes. En una segunda fase de pruebas, se utilizará un tercer módulo configurado como coordinador en la misma PAN de nuestro sistema coexistiendo con el anterior. Con ello se podrá comprobar si se reciben diferentes mensajes de diferentes coordinadores en celdas aisladas o solapadas. Además, se llevará a cabo simular que el nodo receptor sale del área de alcance del coordinador inicial y posteriormente entrar en el área del nuevo coordinador, es decir, a efectos prácticos, establecer contacto con otro nodo emisor (coordinador y celda) emitiendo sus datos y aceptar sus nuevos avisos o anuncios. Con esta idea trataremos de simular el efecto de un router en movilidad y su comportamiento. La Tabla 9 muestra cómo están asignados los dispositivos en esta segunda fase.

Tabla 9. Roles de los dispositivos XBee® S2C

Dispositivo	Función
Módulo 1	Coordinador 1
Módulo 2	Router/Dispositivo final
Módulo 3	Coordinador 2

Para la primera fase de pruebas, se tendrá que comprender la configuración de los dispositivos, para ello se realizará una descripción de los parámetros que tienen que estar establecidos en los módulos para asegurar el correcto funcionamiento de la red. A continuación, se especifican los parámetros dispuestos a configurar en los módulos XBee® S2C.

- *Parámetro ID*: Identifica o configura una Red de Área Personal específica (Tabla 10). Si el valor se establece en 0, el coordinador selecciona un PAN ID aleatorio, mientras que el router/dispositivo final se una a cualquier red.

Tabla 10. Parámetro ID

Parámetro ID	
Rango	0-0xFFFF
Valor establecido	1234

- *Parámetro SC*: Configura y lee la lista de canales que son escaneados y determina cuales están activos. Este parámetro influye cuando se establece un

coordinador antes de iniciar la red y en los dispositivos finales a la hora de encontrar un coordinador/router para unirse. Cada bit representa un canal, teniendo desde el canal 11 hasta el 26 (véase Tabla 11). El XBee® S2C puede escanear hasta 16 canales (SC=0xFFFF). El valor establecido es 0x4.

Tabla 11. Parámetro SC

Bit	Canal (Dec)	Canal (Hex)	Parámetro
0	11	0x0B	0x1
1	12	0x0C	0x2
2	13	0x0D	0x4
3	14	0x0E	0x8
4	15	0x0F	0x10
5	16	0x10	0x20
6	17	0x11	0x40
7	18	0x12	0x80
8	19	0x13	0x100
9	20	0x14	0x200
10	21	0x15	0x400
11	22	0x16	0x800
12	23	0x17	0x1000
13	24	0x18	0x2000
14	25	0x19	0x4000
15	26	0x1A	0x8000

- *Parámetro SD*: Establece o muestra el tiempo de duración para escanear el canal. En un coordinador dictamina la duración por el cual es examinado un canal para mostrar las redes que existen. Igualmente, dictamina la duración por la cual es examinado el nivel de energía de dicho canal, y precisar cual canal operar. En un dispositivo final, es el tiempo que dura el examinar cada canal para localizar un coordinador/router y unirse durante la asociación. El parámetro SD difiere entre 0 y 15. El tiempo de escaneo se mide como:

(nº de canales a examinar) x ((2^{SD}) x 15.36ms) + (38ms x (nº de canales a examinar)) + 20ms

La tabla siguiente muestra el valor establecido:

Tabla 12. Parámetro SD

Parámetro SD	
Rango	0-0x0F
Valor establecido	3

- *Parámetro OP*: Lee el PAN ID extendido de 64 bits. Este valor refleja el ID PAN operativo que esté ejecutando el dispositivo. Si el ID es mayor a cero, el OP es igual al ID.

Tabla 13. Parámetro OP

Parámetro OP	
Rango	0x01-0xFFFF

- *Parámetro OI*: Lee el PAN ID de 16 bits. Este valor refleja el ID PAN de 16 bits operativo que esté ejecutando el dispositivo.

Tabla 14. Parámetro OI

Parámetro OI	
Rango	0-0xFFFF

- *Parámetro CH*: Muestra y permite asignar el canal que será utilizado para transmitir y recibir datos entre los dispositivos (Tabla 15). Si el valor es 0 significa que el dispositivo no se ha unido a una red y por lo tanto no está operando en ningún canal.

Tabla 15. Parámetro CH

Parámetro CH	
Rango	0x0B-0x1A
Valor establecido	0x0D

- *Parámetro JV:* Permite configurar o leer la verificación de un canal. Si JV=1, verifica que el coordinador esté en un canal operativo. Si no lo detecta, el router/dispositivo final abandonará el canal actual e intenta unirse a una nueva PAN. Si JV=0, el router/dispositivo final continúa operando en el canal actual, aunque no detecte un coordinador.

Tabla 16. Parámetro JV

Parámetro JV	
Rango	0-1

- *Parámetro CE*: Este valor corresponde con la función que desempeña el dispositivo en la red. Establece o muestra si el dispositivo es un coordinador y se ha establecido según la tabla siguiente.

Tabla 17. Parámetro CE

Parámetro CE	**Configuración**
0	Router
1	Coordinador

- *Parámetro II*: Este valor define el PAN ID de 16 bits a utilizar que está preconfigurado al formar la red. Si se asigna un valor predeterminado, el dispositivo forma una red en un PAN ID aleatorio de 16 bits. Por otro lado, este parámetro puede ser utilizado para reemplazar un nodo coordinador en una red existente.

Tabla 18. Parámetro II

Parámetro II	
Rango	0-0xFFFF

- *Parámetro SH*: Permite leer los 32 bits más significativos que conforman la dirección de origen de 64 bits asignada de forma única por el IEEE en cada módulo XBee®.

- *Parámetro SL*: Permite leer los 32 bits menos significativos que conforman la dirección de origen de 64 bits asignada de forma única por el IEEE en cada módulo XBee®.

- *Parámetro MY*: Asigna una dirección de origen de 16 bits. Un valor de 0xFFFE indica que el módulo no se ha unido a una red ZigBee.

- *Parámetro DH*: Lee y configura los primeros 32 bits más significativos que conforman la dirección de destino de 64 bits. Este valor, junto con el DL, forman la dirección de destino.

Tabla 19. Parámetro DH

Parámetro DH	
Rango	0-0xFFFFFFFF

- *Parámetro DL*: Lee y configura los primeros 32 bits menos significativos que conforman la dirección de destino de 64 bits. Este valor, junto con el DH, forman la dirección de destino. Un valor de 0x0000FFFF en el coordinador indica que la transmisión es hacia todos los dispositivos de la red, es decir, en broadcast. Mientras que un valor de 0x00000000 indica que la transmisión es únicamente hacia el coordinador,

Tabla 20. Parámetro DL

Parámetro DL	
Rango	0-0xFFFFFFFF

- *Parámetro BD*: A continuación, se recogen los valores que pueden ser asignados.

Tabla 21. Parámetro BD

Valor	Configuración (bps)
0x1	2400
0x2	4800
0x3	9600
0x4	19200
0x5	38400
0x6	57600
0x7	115200
Valor asignado	0x3

- *Parámetro AP*: Permite configurar el modo de operación del módulo. Un valor de AP=0 entra en modo de operación transparente y AP=1 en modo de operación API. Se ha seleccionado el modo de operación API.

- *Parámetro PL:* Asigna o muestra el nivel de potencia a transmitir del módulo RF. En la Tabla 22 se muestra las configuraciones disponibles, así como el valor asignado.

Tabla 22. Parámetro PL

Valor	Nivel de potencia
0	-5dBm
1	-1dBm
2	+1dBm
3	+3dBm
4	+5dBm
Valor asignado	4

- *Parámetro AO:* Permite configurar las opciones del modo API. Según que opción establece el tipo de trama API a recibir para enviar por el UART para los paquetes de datos RF recibidos.

Tabla 23. Parámetro AO.

Parámetro AO	
Rango	0x00-0x0B
Valor establecido	0x00. Indicador API Rx habilitado.

En la Tabla 24 se resume la configuración seleccionada en los dispositivos XBee® S2C, que forman la red de este proyecto.

Tabla 24. Resumen de los parámetros asignados en los módulos XBee® S2C

Descripción	Parámetro	Módulo 1	Módulo 2	Módulo 3
Canal operativo	CH	13	13	13
Identificador de PAN	ID	1A1A	1A1A	1A1A
Lista de canales para examinar	SC	4	0x3FFF	4
Tiempo para examinar el canal	SD	3	3	3
Verificación del canal	JV	0	1	0
PAN ID operativo	OP	1A1A	1A1A	1A1A
PAN ID de 16 bits operativo	OI	63B2	63B2	63B2
Dirección de Destino	DH	0	0	0
Dirección de Destino	DL	0xFFFF	0	0xFFFF
Función de coordinador	CE	1	0	1
Nivel de potencia	PL	4	4	4
Velocidad de transmisión	BD	3(9600bps)	3(9600bps)	3(9600bps)
Modo de operación	AP	1	1	1
Opciones API	AO	0	0	0

Los módulos que aparecen en Figura 48 son los utilizados en los que aparecen los dos últimos bytes de sus MAC. Las funciones asignadas son expresadas en la Tabla 25.

Figura 48. Módulos XBee® S2C.

Tabla 25. Roles asignados y direcciones MAC de los módulos XBee®.

Dispositivo	Función	Dirección MAC	
		SH	SL
Módulo 1	Coordinador 1	0013A200	4108641F
Módulo 2	Router	0013A200	417CE139
Módulo 3	Coordinador 2	0013A200	415D8897

Se procederá a utilizar el software XCTU para instalar y configurar los módulos XBee® S2C, haciendo uso del XBee USB Adapter el cual permite conectar y utilizar el módulo XBee® S2C directamente mediante un puerto USB como se muestra en la Figura 49. La conexión se realizará con el acceso a los pines TX/RX del XBee® S2C. Dado que la instalación y configuración de los tres módulos se hace de manera similar, a continuación, se detalla los pasos a seguir para implementar el módulo 1, que será el primer coordinador a usar. Este proceso será implementado de la misma forma en el resto de los módulos restantes ajustando los parámetros correspondientes a su configuración dada.

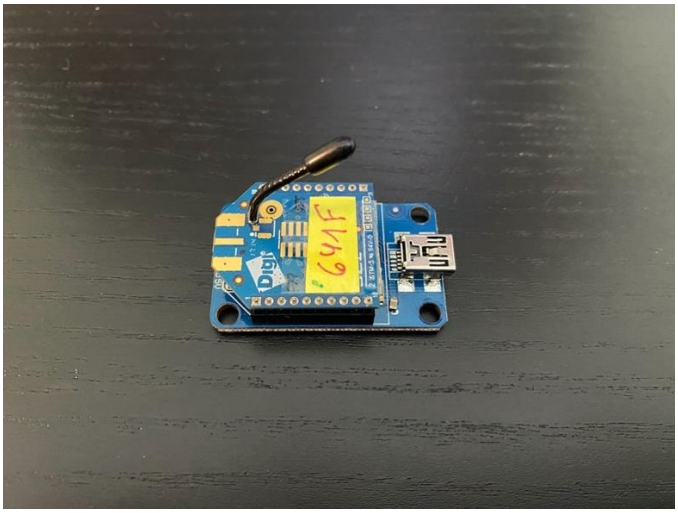


Figura 49. Módulo XBee® S2C conectado sobre XBee USB Adapter.

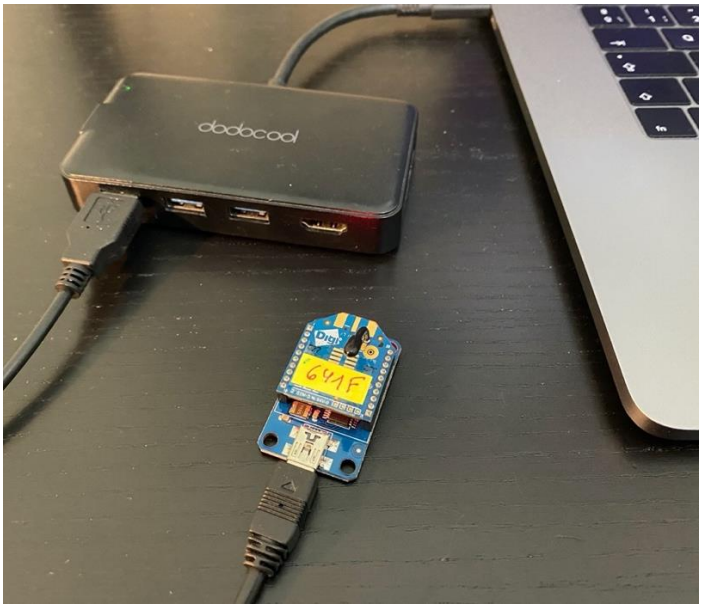


Figura 50. Módulo XBee® conectado a ordenador portátil.

Iniciado el software XCTU, como se observa en la Figura 51, vemos que cuenta con una barra de herramientas superior compuesta por varios iconos divididos en tres secciones, los cuales proporcionan soporte y control sobre los módulos de comunicación. La primera sección está formada por dos iconos que permiten añadir los módulos al XCTU. La segunda sección está compuesta por cuatro iconos que contiene las herramientas XCTU, las preferencias, un formulario de comentarios y la función de ayuda y soporte. La última sección contiene tres iconos que corresponden a los tres modos de trabajo con el XCTU. Cada modo determina que operación

específica se puede realizar con el módulo. *Configuration*, configuración de los parámetros del módulo. *Consoles*, la consola que permite la comunicación con los módulos agregados. *Network*, para descubrir y visualizar la topología y las distintas interconexiones que hay en la red. Solo se puede seleccionar un modo de trabajo a la vez. Para estos tres modos de trabajo, es indispensable haber añadido al menos un módulo XBee® al XCTU.

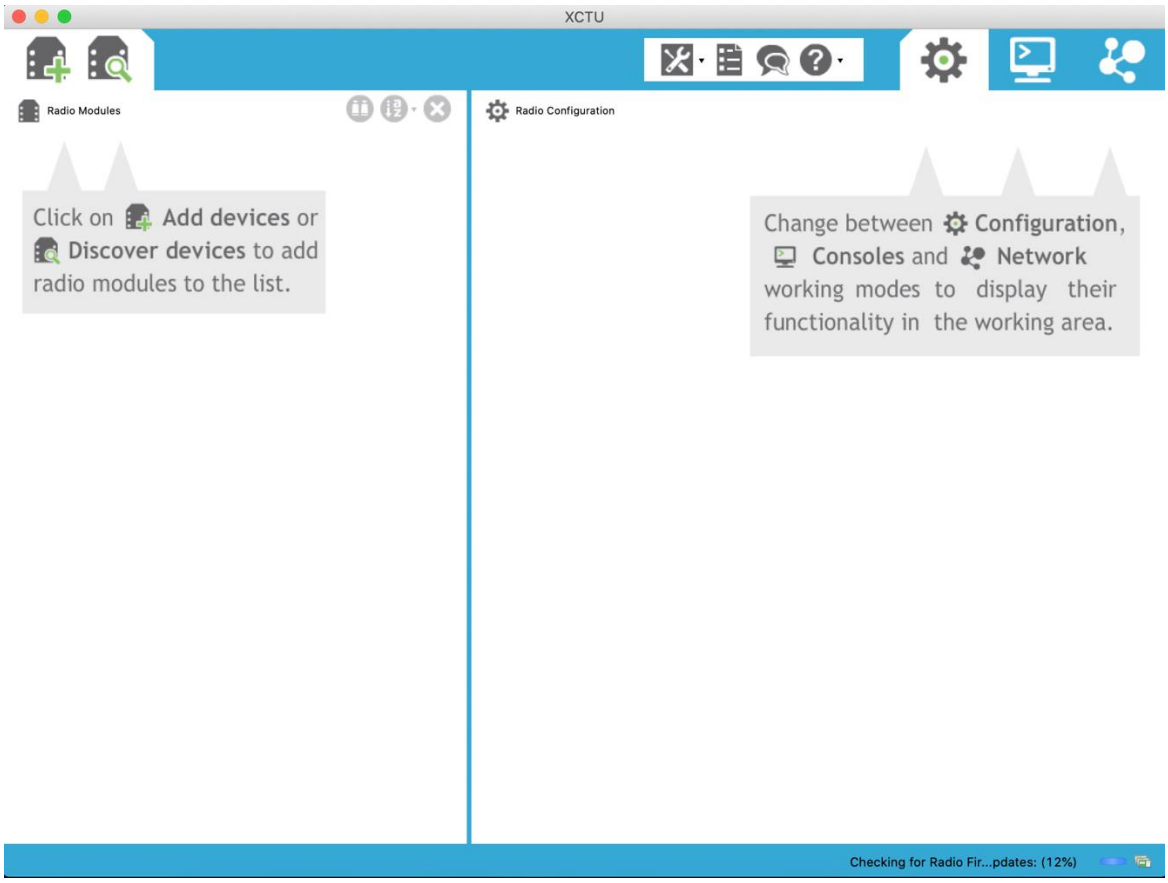


Figura 51. XCTU: Entorno del programa.

Seleccionamos el icono de *Discover Radio modules* de la primera sección, el cual buscaremos los puertos serie en los que se deseen encontrar los módulos conectados y asignamos las características del puerto del computador con el que se va a configurar el XBee® S2C, características como: velocidad de transmisión, control de flujo de bits, bits de información, entre otras opciones. Todas estas se aprecian en la Figura 52. Generalmente, vienen unas opciones por defecto preestablecidas y únicamente bastaría con verificar la velocidad a la que es establecida la conexión que sería a 9600 baudios.

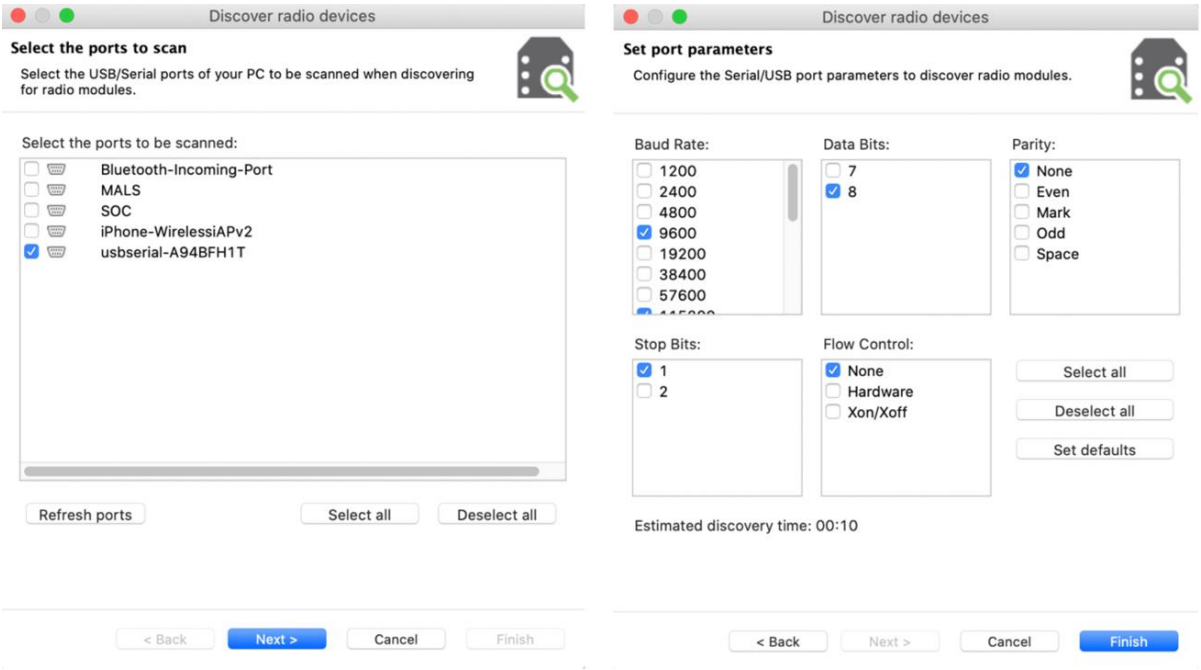


Figura 52. XCTU: Discover radio devices

En este punto, hacemos clic sobre *Finish*, la cual comienza la búsqueda del módulo y se despliega una ventana como la que se muestra en la Figura 53. Dicha venta confirma la existencia de una comunicación entre el módulo XBee® y el computador, indicando el puerto establecido y la dirección MAC del dispositivo. Todos los módulos XBee® traen de fábrica una dirección MAC única, colocada en la parte trasera de la placa. En la Figura 54, podemos apreciar dicha dirección, que es la misma que la que reconoce el software XCTU una vez encontrado el módulo.

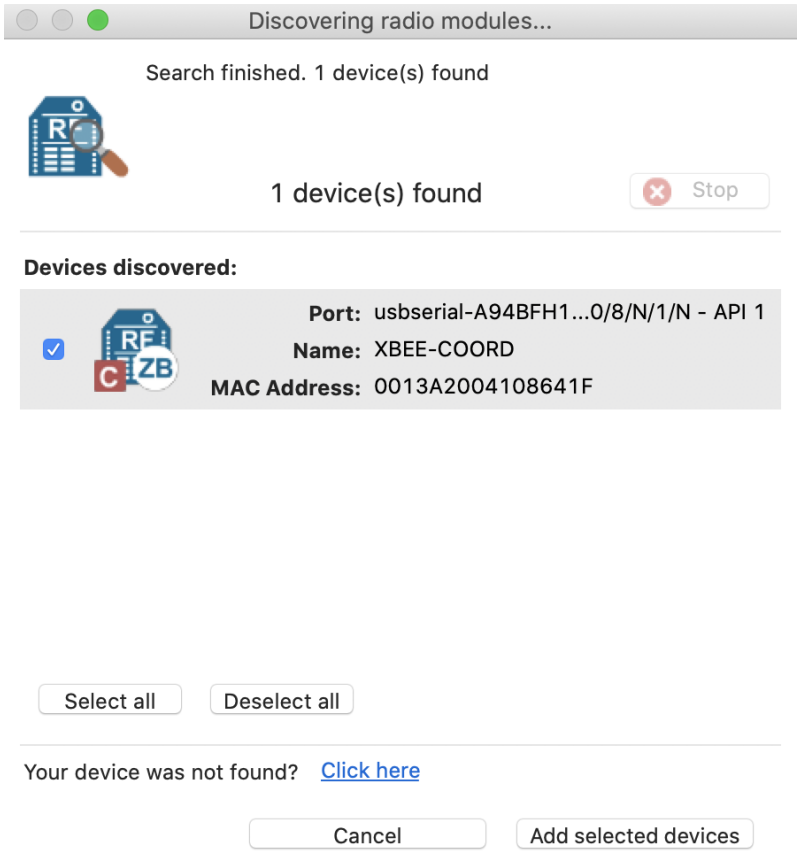


Figura 53. XCTU: Descubriendo módulos radio.

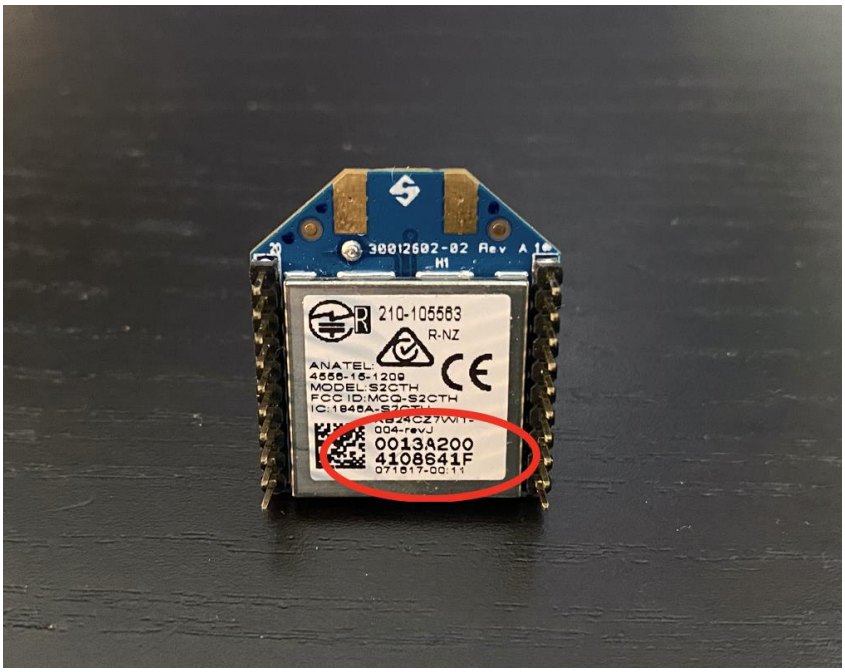


Figura 54. Dirección MAC del módulo XBee® S2C coordinador.

En este punto en el que ya tenemos añadido el módulo, nos aparece dentro de *Configuration* una barra de menú (ver en Figura 55), la cual nos muestra diferentes acciones donde destacamos las siguientes que vamos a usar para configurar. *Read,* que es para leer los ajustes del firmware del módulo seleccionado. *Write* para escribir nuevos valores del firmware del módulo seleccionado. *Default*, carga los valores del firmware que vienen por defecto en el módulo seleccionado. *Update*, que actualiza el firmware del módulo seleccionado, pero sin escribir nuevos valores.

Figura 55. XCTU: Menú superior.

Para esta última opción comprobamos si está actualizado a la versión más reciente, de lo contrario siempre es recomendable actualizarlo. La información respecto al firmware actualizado se detalla a continuación.

Familia del producto	Función	Versión del firmware
XB24C	ZIGBEE TH Reg	4061

Una vez actualizado, nos dirigimos nuevamente a la parte de *Configuration*. El programa XCTU muestra la configuración de los parámetros del módulo RF debajo del panel de información del firmware. Están divididos en secciones o categorías con una breve descripción en cada una. Para aplicar los cambios hechos en las configuraciones se deberá seleccionar la opción *Write*. La Figura 56 muestra una captura de pantalla del programa XCTU, en donde se configuraron los parámetros del módulo XBee® S2C que actúa con el rol de coordinador (alguno de estos parámetros es de solo lectura). Las configuraciones de los parámetros en todos los módulos XBee® S2C a través del programa XCTU han sido las indicadas en la Tabla 24.

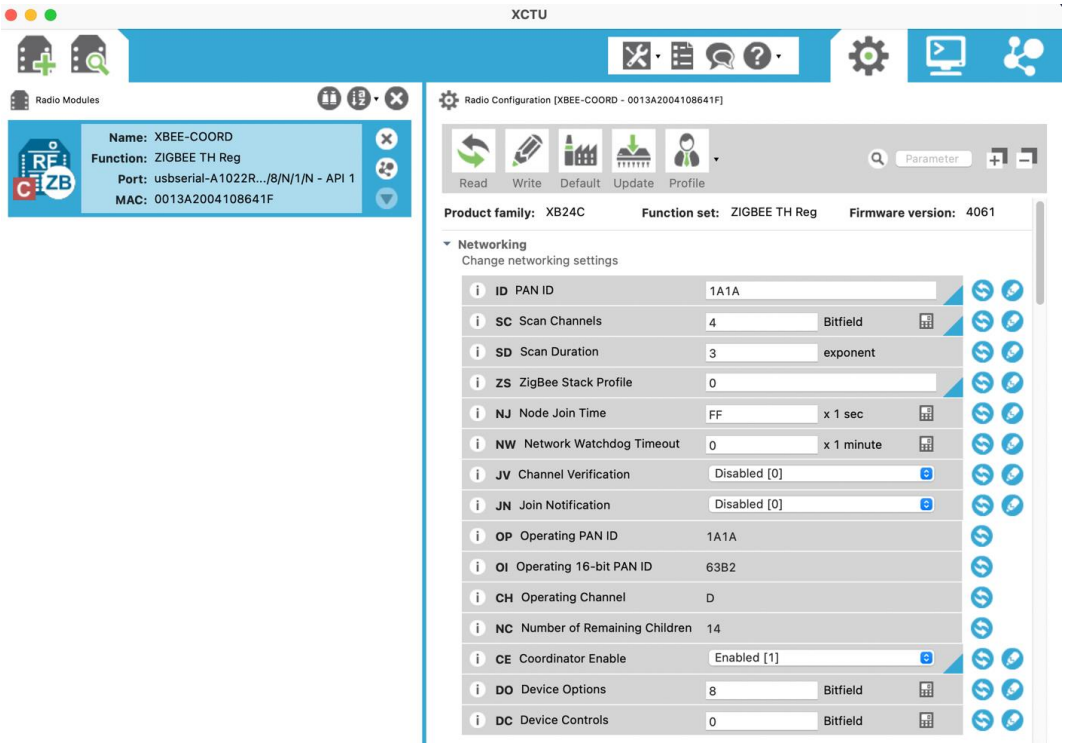


Figura 56. XCTU: Configuración de parámetros.

4.2.1 Configuración de varios coordinadores en misma red

Como se indicó, para analizar la disponibilidad de varios coordinadores que usen la misma PAN (misma red) y evaluar el comportamiento de la red, se habilitará otro coordinador emitiendo las mismas o diferentes notificaciones. Se pretende que este nuevo coexista en la red (misma o diferente celda) con el anteriormente probado, y teniendo la misma configuración de la red. Esto se hace para detectar posibles problemas relacionados con la potencial coexistencia de coordinadores con la misma PAN. Se procederá a configurar el nuevo XBee coordinador utilizando el PAN ID (16 bits y 64 bits), el canal y la configuración del perfil de pila de una red en ejecución para configurar dicho coordinador existente. En el caso de celdas aisladas los routers se asocian directamente al coordinador que detecten de forma automática con solo el PAN ID de 64 bits.

Para configurar el nuevo dispositivo XBee como coordinador, hay que indicar que algunos parámetros de configuración vistos anteriormente en el XCTU son de solo

lectura, por lo que se utilizará un tipo de trama que permita la configuración de dichos parámetros en el dispositivo local. El tipo de trama a utilizar corresponde a la trama tipo 0x08: *Local AT Command Request*. Para configurar el coordinador se realizará la configuración detallada a continuación:

Tabla 26. Comandos AT

Comando	Descripción
OP	Lee el PAN ID operativo de 64 bits.
OI	Lee el PAN ID operativo de 16 bits.
CH	Lee el canal operativo.
ZS	Lee el perfil de pila.

Cada uno de los comandos enumerados anteriormente se puede leer desde cualquier dispositivo en la red y los parámetros asignados serán los mismos en todos los dispositivos de la red. Después de leer los comandos del dispositivo coordinador, programaremos los valores de los parámetros en el nuevo coordinador usando los siguientes comandos.

Tabla 27. Comandos AT a programar para el nuevo coordinador

Comando	Descripción
ID	Se configura el PAN ID de 64 bits para que coincida con el valor de OP leído.
II	Se configura el PAN ID inicial de 16 bits para que coincida con el valor de OI leído.
SC	Se configura la máscara de bits de los canales de exploración para habilitar la lectura del canal operativo (comando CH). Por ejemplo, si el canal operativo es 0x0B, configura SC en 0x0001. Si el canal operativo es 0x17, establezca SC en ox1000.
ZS	Se configura el perfil de pila para que coincida con el valor de ZS leído.

El parámetro II es el PAN ID inicial de 16 bits. Bajo ciertas condiciones, la pila ZigBee puede cambiar el ID PAN de 16 bits de la red. Por esta razón, no se puede guardar el

comando II usando el comando *Write*. Una vez que se establece II, el coordinador abandona la red y comienza con el PAN ID de 16 bits especificado por II.

Como se comentaba, el parámetro II al ser de solo lectura recurriremos a la trama 0x08 para configurarlo (ID PAN inicial de 16 bits) en el nuevo coordinador y que coincida con el valor de OI leído en el coordinador existente. Este tipo de trama se utiliza para consultar o establecer parámetros de comando en el dispositivo local. Cualquier parámetro que se establezca con este tipo de trama aplicará el cambio inmediatamente. En la Tabla 28 se muestra los campos de la trama 0x08, configurando el parámetro "II" y estableciendo su valor a "63B2". Se envía los siguientes datos en hexadecimal:

"*7E 00 06 08 00 49 49 63 B2 50*".

Tabla 28. Tipo de trama = 0x08. Trama TX: "Local AT Command Request".

Campo	Byte	Ejemplo	Descripción
Delimitador de inicio	0	0x7E	Identifica el inicio de una nueva trama
Longitud	1 2	0x00 0x06	Número de bytes entre longitud y checksum
Tipo de trama	3	0x08	Identifica el tipo de trama
ID de trama	4	0x00	Identifica la trama de datos UART del host para relacionarla con el correspondiente ACK. Al ser 0x00, no se envía respuesta (ACK).
Comando AT	5 6	0x49 0x49	Nombre del comando. Siempre en dos caracteres ASCII que identifican el comando AT. Comando II.
Valor del parámetro (opcional)	7 8	0x63 0xB2	Parámetro del comando II.
Checksum	9	0x50	Checksum.

Enviaremos esta trama a través de la consola API proporcionada por el XCTU. Nos dirigimos a ella, y una vez en el panel seleccionamos la opción *Open* permitiendo así abrir la conexión Serial con el módulo que tenemos agregado. Usaremos la herramienta *Frame Generator* para crear la trama vista anteriormente y la añadimos al panel de envío de tramas. A continuación, seleccionamos la opción *Send selected Frame* para finalmente enviar la trama. La Figura 57 muestra una captura de pantalla del proceso realizado en la que se muestra el monitor de tráfico de tramas viéndose el registro de tramas enviadas en este caso. Cuando el módulo envía o recibe tramas API, se agregan al registro de tramas. Dependiendo de si la trama se envía o se recibe, el color de los campos de la trama es azul o rojo, respectivamente.

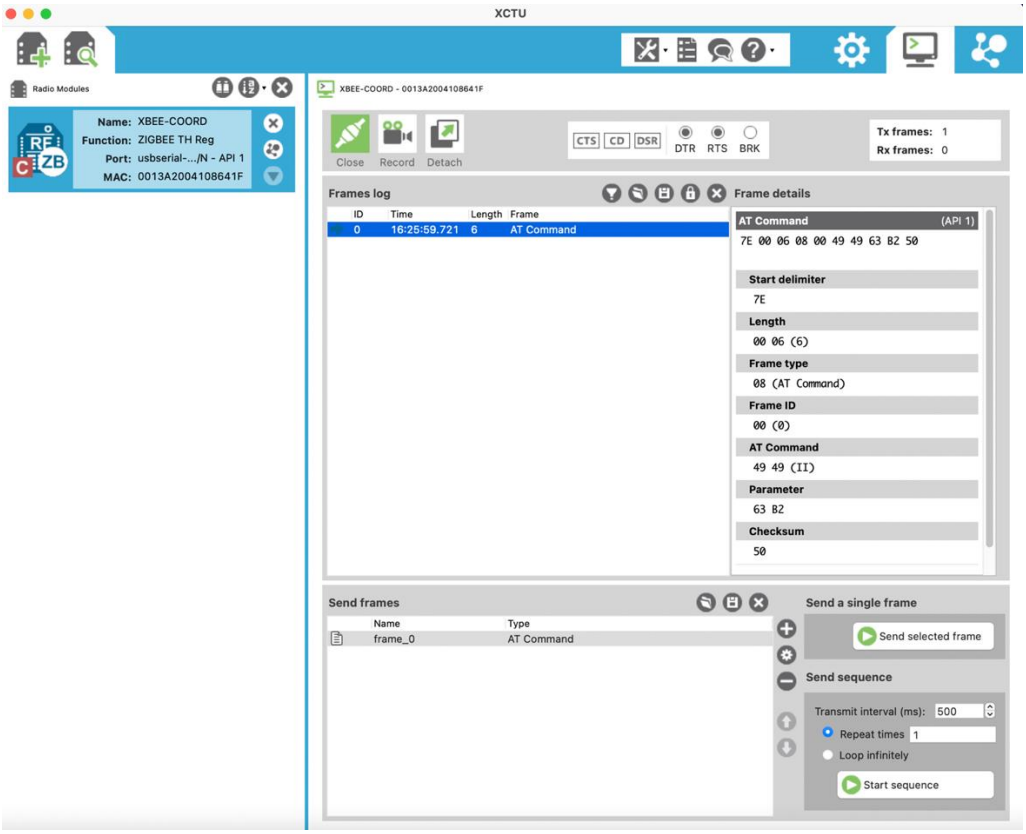


Figura 57. XCTU: Registro de tramas enviadas.

Los parámetros restantes, CE, ID, SC y ZS se configuraron en el nuevo coordinador para que coincidan con el coordinador existente, seguido del comando *Write* para guardar estos valores de parámetros.

4.3 Formato de la Trama API

Antes de profundizar en la escritura del código, se describirá el tipo de trama API a utilizar y cómo será su formato para posteriormente realizar la construcción y retransmitir los paquetes que envía y recibe el código.

Como sabemos, el modo API tiene un protocolo predefinido y se comunica mediante paquetes, denominadas tramas, las cuales los datos ya están ordenados. Tal y como se ha visto en la sección 2.2.5.2 de este documento, los primeros 4 bytes de las tramas son siempre del mismo tipo de información, como se describe en la Tabla 5.

Hasta este punto, sabemos cómo será la estructura de nuestra trama, pero no sabemos qué tipo de trama debemos utilizar para construir y enviar el paquete. Existen varios tipos de tramas [9], dependiendo si es para transmisión o recepción de datos y comandos. En nuestro caso y habiendo evaluado los tres tipos de configuraciones que teníamos para el diseño, se ha procedido a valorar las tramas tipo 0x10 y 0x17. A continuación se detalla su estructura.

4.3.1 Remote AT Command – 0x17

Este tipo de trama se utiliza para la configuración o lectura de manera remota de los parámetros de otro módulo XBee® [23]. Aunque no se usará este tipo de trama para llevarla a la fase de pruebas en este prototipo, resulta de especial interés utilizar esta trama para el caso en el que el tipo de configuración en el nodo receptor esté compuesto únicamente por un módulo XBee® (sin el µC Arduino), ya que permite utilizar la salidas digitales y salidas por PWM del módulo XBee® para poder controlar los dispositivos externos que estén conectados a él. Para que la trama pueda aplicar los cambios una vez ejecutado el comando (los cambios que se efectúen en los parámetros del módulo sólo son tenidos en cuenta una vez son aplicados), se tiene que activar el bit correspondiente (campo Remote command options = 0x02), o tener

que enviar posteriormente una trama con el comando "AC" (Apply Changes) al mismo nodo remoto.

En la Tabla 29 se muestra un ejemplo de los campos de la trama 0x17, configurando de forma remota el parámetro “D3” (DIO3/AD3) como salida digital, con valor alto por defecto (0x5), indicando que se apliquen directamente los cambios (campo Remote command options = 0x02). El dispositivo coordinador envía la trama y se recibe en el en el dispositivo router. Los siguientes datos son enviados en hexadecimal:

“7*E 00 10 17 00 00 00 00 00 00 00 FF FF FF FE 02 44 33 05 6F* “.

Tabla 29. Tipo de trama = 0x17. Trama TX: "Remote Command Request".

Campo	Byte	Ejemplo	Descripción
Delimitador de inicio	0	0x7E	Identifica el inicia de una nueva trama
Longitud	1 2	0x00 0x10	Número de bytes entre longitud y checksum
Tipo de trama	3	0x17	Identifica el tipo de trama
ID de trama	4	0x00	Identifica la trama de datos UART del host para relacionarla con el correspondiente ACK. Al ser 0x00, no se envía respuesta (ACK).
Dirección 64-bit destino	5 6 7 8 9 10 11 12	0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFF	Identifica la dirección MAC del nodo/dispositivo de origen. Se admite la dirección “00 00 00 00 00 00 FF FF” para indicar la dirección broadcast.
Dirección 16-bit destino	13 14	0xFF 0xFE	Identifica dirección 16-bit.
Opción comando remoto	15	0X02	Valor por defecto. Aplicar cambios inmediatamente en el dispositivo remoto.

Comando AT	16 17	0x44 0x33	Nombre del comando. Siempre en dos caracteres ASCII que identifican el comando AT.
Valor del parámetro (opcional)	18 19	0x05	Parámetro del comando D3. Activo salida digital con valor alto.
Checksum	20	0x6F	Checksum.

4.3.2 Transmit Request – 0x10

Este tipo de trama hace que el módulo XBee® conectado localmente envíe por RF los datos contenidos en el campo Payload data empaquetados al nodo de destino [24]. En nuestro caso este tipo de trama se usará para la configuración completa, que será el prototipo a construir visto anteriormente, ya que al disponer del µC Arduino podemos ampliar las lecturas digitales recibidas y configurarlas.

En la Tabla 30 se muestra un ejemplo de los campos de la trama tipo 0x10, realizando una transmisión al nodo destino mediante RF para enviar los datos contenidos en el campo Payload data “Rojo”. Se envía los siguientes datos en hexadecimal:

“*7E 00 12 10 00 00 00 00 00 00 00 FF FF FF FE 00 00 52 6F 6A 6F 5A*”.

Tabla 30. Tipo de trama = 0x10. Trama Tx: "Transmit Request".

Campo	Byte	Ejemplo	Descripción
Delimitador de inicio	0	0x7E	Identifica el inicia de una nueva trama
Longitud	1 2	0x00 0x12	Número de bytes entre longitud y checksum
Tipo de trama	3	0x10	Identifica el tipo de trama
ID de trama	4	0x00	Identifica la trama de datos UART del host para relacionarla con el correspondiente ACK. Al ser 0x00, no se envía respuesta (ACK).

Dirección 64-bit destino	5 6 7 8 9 10 11 12	0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFF	Identifica la dirección MAC del nodo/dispositivo de origen. Se admite la dirección “00 00 00 00 00 00 FF FF” para indicar la dirección broadcast.
Reservado	13 14	0xFF 0xFE	Identifica dirección 16-bit
Broadcast radius	15	0x00	Valor por defecto. Número máximo de saltos,
Transmit options	16	0x00	Nombre del comando. Siempre en dos caracteres ASCII que identifican el comando AT.
RF Payload data	17 18 19 20	0x52 0x6F 0x6A 0x6F	Datos enviados al nodo/dispositivo de destino. En este ejemplo se envía el valor “52 6F 6A 6F” (en ASCII: Rojo).
Checksum	21	0x5A	Checksum.

4.3.3 Receive Packet – 0x90

Este tipo de trama se emite cuando el módulo XBee® recibe un paquete de datos por RF. El paquete con los datos contenidos en él se envía a través de la UART del módulo (si está configurado con el parámetro A0=0). Por lo general, esta trama se emite como resultado de un dispositivo en la red que envía datos en serie utilizando la trama tipo 0x10: *Transmit Request* o la trama tipo 0x11: *Explicit Addressing Command Request*.

En la Tabla 31 se muestra un ejemplo de los campos de la trama tipo 0x90, recibiendo una transmisión de un dispositivo con la dirección de 64 bits correspondiente a la MAC “*0013A2004108641F”* mediante RF para recibir los datos contenidos en el campo

Payload data “Rojo”. La siguiente trama se emite si el destino está configurado con AO=0. Se reciben los siguientes datos en hexadecimal:

“*7E 00 12 90 00 13 A2 00 41 08 64 1F 56 14 02 52 6F 6A 6F E8*”.

Tabla 31. Tipo de trama = 0x90. Trama Rx: "Receive Packet".

Campo	Byte	Ejemplo	Descripción
Delimitador de inicio	0	0x7E	Identifica el inicia de una nueva trama.
Longitud	1 2	0x00 0x0D	Número de bytes entre la longitud y el checksum.
Tipo de trama	3	0x90	Identifica el tipo de trama
Dirección 64-bit de origen	4 5 6 8 9 9 10 11	0x00 0x13 0xA2 0x00 0x41 0x08 0x64 0x1F	Identifica la dirección MAC del nodo/dispositivo de origen.
Dirección 16-bit de origen	12 13	0x56 0x14	Identifica dirección 16-bit
Receive options	14	0x02	Campo de bits de opciones que se aplican al mensaje recibido: Bit 0: Paquete reconocido [0x01] Bit 1: Paquete enviado como transmisión [0x02] Bit 2: Reservado Bit 3: Reservado
Receive data	15 16 17 18	0x52 0x6F 0x6A 0x6F	Datos recibidos del nodo/dispositivo de origen. En este ejemplo se envía el valor “52 6F 6A 6F” (en ASCII: Rojo).
Checksum	19	0xE8	Checksum.

La Figura 58 muestra los intercambios de las tramas API como resultado de enviar el tipo de trama 0x10 con los datos encapsulados a su destino. Los datos inalámbricos son recibidos en el dispositivo remoto encapsulados en una trama de recepción del tipo 0x90.

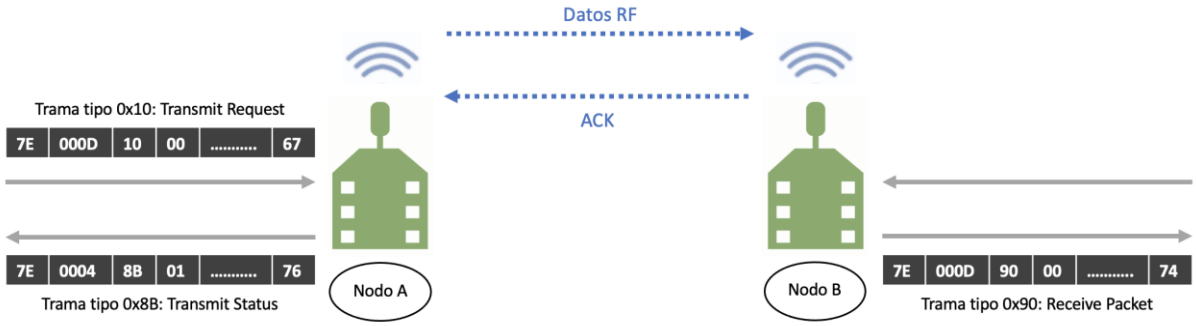


Figura 58. Intercambio de tramas API

El tener diferentes configuraciones del sistema receptor (sin µC y con µC) se ha necesitado habilitar varios tipos de tramas según si el nodo receptor está formado únicamente por el módulo XBee® o se dispone del µC Arduino. Con ello se posibilita el intercambio de mensajes de texto o comandos, y los nodos receptores interpretarán correctamente los mismos para mostrar la salida correspondiente (encender led, activar zumbador y/o mensaje en display LCD). Aunque no se haya implementado en el prototipo se ha evaluado la configuración y el envío de las tramas API específicas para el intercambio de mensajes simples a modo de prueba, comprobando el envío y recepción de las tramas entre los nodos. Todo ello, a través del software XCTU teniendo los módulos conectados a la computadora portátil en el momento de su configuración.

Finalizada la configuración de los módulos XBee®, montaje tanto del circuito emisor como el circuito receptor y habiendo garantizado la comunicación entre ellos y la creación del formato de tramas según nuestra implementación concreta del sistema, el siguiente paso es la implementación del programa que será el encargado de controlar las distintas acciones tanto el emisor como receptor. Dicha configuración se hará para la configuración completa.

4.4 Programación del µC Arduino

El lenguaje de programación que hemos utilizado para crear el programa o sketch ha sido el estándar para Arduino, que es C++, utilizando para ello el software de programación de código abierto Arduino IDE, proporcionado por el fabricante en su página web.

Según las especificaciones de nuestro sistema, el nodo coordinador se encargará, actuando de controlador, de tomar la decisión de enviar una trama o comando al nodo remoto según las indicaciones del µC Arduino emisor. Por lo tanto, este último interactuará con el exterior para cada propósito y mediante el programa desarrollado procesar la información a enviar, y cuando se disponga de receptor procesar los mensajes recibidos y entregarlos a los dispositivos de salida correspondiente que estén disponibles. A continuación, se procederá a explicar la primera parte del programa que corresponde al µC Emisor.

4.4.1 Programa del µC Emisor

Nuestro sistema parte de que el código ejecutado en el Arduino emisor tomará los valores entrantes del switch DIP para ensamblar el paquete y transmitirlo a la fuente destinataria. Esto se ha pensado hacer así, para que sea configurable pues de otra manera tendría que estar todo pregrabado sin posibilidad de portar para diferentes propósitos o tener que depender de otro computador externo para indicarle como proceder. Como se comentó se permiten hasta ocho posibles modos de operación/mensajes, aunque hemos implementado tres de ellas.

En el diagrama de flujo mostrado en la Figura 59, se describe el organigrama aproximado con la secuencia básica de acciones que deberían regir el funcionamiento que tiene el programa y la secuencia que sigue en base al protocolo de comunicación del Arduino a partir de la configuración estática de los switchs que dirigen el formato de los mensajes contenidos en los paquetes (tramas) a enviar.

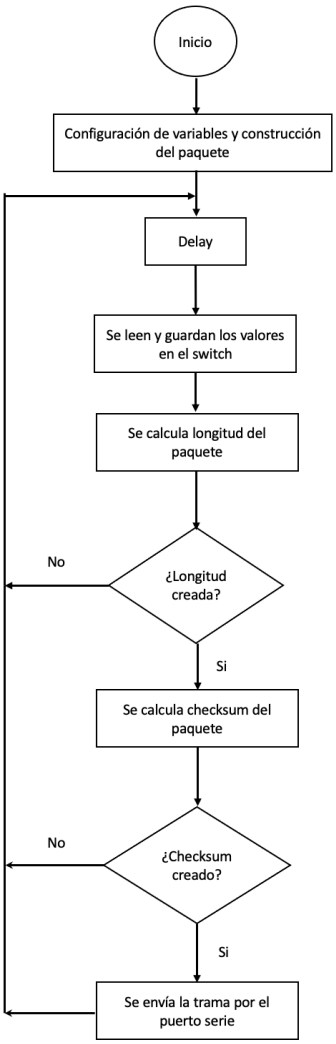


Figura 59. Diagrama de flujo del µC Arduino en nodo transmisor.

El código del programa se compone de varias partes. La primera parte configura las variables y toda la inicialización de las conexiones seriales de Arduino antes de ingresar a la función *loop*. Comienza declarando las variables que forman la trama a enviar. Dado que el transmisor desconoce a los potenciales receptores, la transmisión se va a realizar de forma broadcast (0x000000000000FFFF) garantizando que cualquier nodo receptor en las misma PAN capte el mensaje/anuncio. El código actualmente transmite un tipo de paquete formado correspondiente al tipo de trama 0x10.

```
// Definiciones:
#define tamtipoTrama 1
#define tamIDTrama 1
#define tamDir64Dest sizeof(dir64Dest)/sizeof(byte)
#define tamDir16Dest sizeof(dir16Dest)/sizeof(byte)
#define tamBroadcastRadius 1
#define tamOption 1
#define tamDatoRF 1

// Declaración de variables que forman la trama.
byte inicioTrama = 0x7E;
byte longMSB = (byte)0x00;
byte longLSB;
byte tipoTrama = 0x10;
byte IDTrama = (byte)0x00;
byte dir64Dest[] = {0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xFF, 0xFF};
byte dir16Dest[] = {0xFF, 0xFE};
byte broadcastRadius = (byte)0x00;
byte option = (byte)0x00;
byte datoRF;
byte checksum;
```

Se tendrá que asegurar de que cada trama que se envíe esté bien ensamblada. Recordemos que en cada paquete a enviar tenemos el byte de longitud y el byte de checksum. El primero nos indica la longitud de la trama y el segundo la suma de comprobación. Estos estarán sin asignar un valor en un primer momento, ya que aunque siempre se envíe el mismo tipo de trama, dependiendo de los datos a enviar, el byte Checksum será diferente en cada caso y si estos dos bytes son erróneos el XBee® recibirá una trama incorrecta y no se podrá acceder al campo Payload de los datos. Por consiguiente, es imprescindible asegurar a través de una función un método que se encargue de calcular estos parámetros correctamente.

A continuación, se declaran las variables correspondientes a las lecturas de los pines 9, 10, 11 y 12 del Arduino correspondientes a la ubicación del switch. Seguido de las variables a imprimir.

```
// Variables de los pines.
const int pin9 = 9;
const int pin10 = 10;
const int pin11 = 11;
const int pin12 = 12;

// Variables a imprimir.
char comando[20];
char imprimir[20];
char imprimir2[20];
```

La función *setup* se ejecuta una sola vez al iniciar el programa, en ella establecemos los pines como entradas digitales a través del método *pinMode* con las variables anteriores. Inicializamos la velocidad de comunicación serial a 9600 baudios

```
void setup() {
  pinMode(pin9, INPUT);      // pin 8 como entrada
  pinMode(pin10, INPUT);     // pin 10 como entrada
  pinMode(pin11, INPUT);     // pin 11 como entrada
  pinMode(pin12, INPUT);     // pin 12 como entrada
  Serial.begin(9600);        // Inicializa monitor Serial con una velocidad
de 9600 baudios.
}
```

Mientras tengamos la placa Arduino encendida, la función *loop* se ejecutará constantemente en bucle. Dado que la tecnología ZigBee no soporta grandes velocidades, es necesario distribuir la información anunciada en un tiempo razonable y opcionalmente configurable. Para ello hemos introducido que para un caso genérico un tiempo de anuncio de 2 segundos entre trama y trama en el nodo transmisor es aceptable para cualquier mensaje. Esto se materializa en el delay (2000) indicado en el comienzo del código. En ella recogeremos los valores de los pines digitales conectados al switch uniéndolos en un solo byte y dependiendo de la combinación que establezcamos invocaremos al método *enviarTrama* pasándole por parámetros el valor de dichos estados. Se imprime el byte del estado del switch, junto a los bytes de longitud y checksum a través del Monitor Serie.

```
void loop() {
  delay(2000);
  int bit2 = digitalRead(pin9);           // Lee el estado del pin 9
  int bit1 = digitalRead(pin10);          // Lee el estado del pin 10
  byte numComando = bit1 | (bit2 << 1);   // Coloca la entrada binaria en
un byte
  enviarTrama (numComando);
  Serial.println();

  sprintf(comando, "COMANDO NUMERO: %02d", numComando);
  Serial.println(comando);

  sprintf(imprimir, "LONGITUD DE LA TRAMA: %02X", longLSB);
  Serial.println(imprimir);

  sprintf(imprimir2, "CHECKSUM CALCULADO: %02X", checksum);
  Serial.println(imprimir2);
  Serial.println();
}
```

La segunda parte del programa contiene la función *enviarTrama*. Dentro de ella, se hace una llamada a los métodos *calcLongLSB* y *calcChecksum*, con ello se calcula el tamaño del paquete y la suma de comprobación asegurándonos de que el paquete esté bien formado antes de enviarlo y almacenando el valor de estas operaciones en sus respectivas variables. Posteriormente se envía el paquete con todos sus datos a través del puerto serie en forma de bytes.

```
/*
 * Función que envía los datos de la trama a través del puerto Serial.
 */
void enviarTrama(byte datoRF) {
  longLSB = calcLongLSB();             // Byte longitud
  checksum = calcChecksum(datoRF);     // Byte checksum
  Serial.write(inicioTrama);
  Serial.write(longMSB);
  Serial.write(longLSB);
  Serial.write(tipoTrama);
  Serial.write(IDTrama);
  Serial.write(dir64Dest, tamDir64Dest);
  Serial.write(dir16Dest, tamDir16Dest);
  Serial.write(broadcastRadius);
  Serial.write(option);
  Serial.write(datoRF);
  Serial.write(checksum);
}
```

La función *calcLongLSB* devuelve la longitud del paquete del tercer byte sumando todos los bytes después del de inicio y longitud hasta antes del byte del checksum.

```
/*
 * Función que calcula la longitud de la trama a enviar.
 */
byte calcLongLSB() {
  return tamtipoTrama + tamIDTrama + tamDir64Dest + tamDir16Dest +
tamBroadcastRadius + tamOption + tamDatoRF;
}
```

La función *calcChecksum* extrae la longitud de los paquetes de direcciones de red y las añade a una variable que va sumando todos los bytes del paquete y finalmente se calcula la suma de comprobación.

```
/*
 * Función que calcula la suma de verificación o checksum.
 */
byte calcChecksum(byte datoRF) {
  unsigned int suma = 0;
  for (int i = 0; i < tamDir64Dest; i ++) {
    suma += dir64Dest[i];
  }
  for (int i = 0; i < tamDir16Dest; i ++) {
    suma += dir16Dest[i];
  }
  suma = suma + tipoTrama + IDTrama + broadcastRadius + option + datoRF;
  suma = (0xFF - (suma)) & 0xFF;
  return suma;
}
```

4.4.2 Programa del µC Receptor

Como se ha indicado, en la parte del nodo receptor el µC Arduino, destino último de los datos, se encargará de procesar los paquetes que el dispositivo router le entrega. Sabemos que la trama recibida que debemos esperar es la del tipo 0x90 como resultado del dispositivo coordinador en la red que envía los datos en serie utilizando la trama tipo 0x10. El código actualmente identifica y muestra un tipo de paquete (trama tipo 0x90) tomando los datos entrantes del coordinador existente en la red y extrayendo dichos datos para efectuar las acciones indicadas a través de los valores enviados desde el coordinador. A modo de depurado, se ha utilizado el monitor Serie para que mientras el paquete se procesa imprima datos relativos, como una notificación cuando se haya recibido un paquete entrante, el paquete sin procesar en sí, las direcciones de la fuente de origen y la comprobación de la longitud y checksum recibidos.

En la Figura 60 se muestra el diagrama de flujo de los pasos que sigue el algoritmo y la toma de decisiones en base al protocolo de comunicación del Arduino con los periféricos.

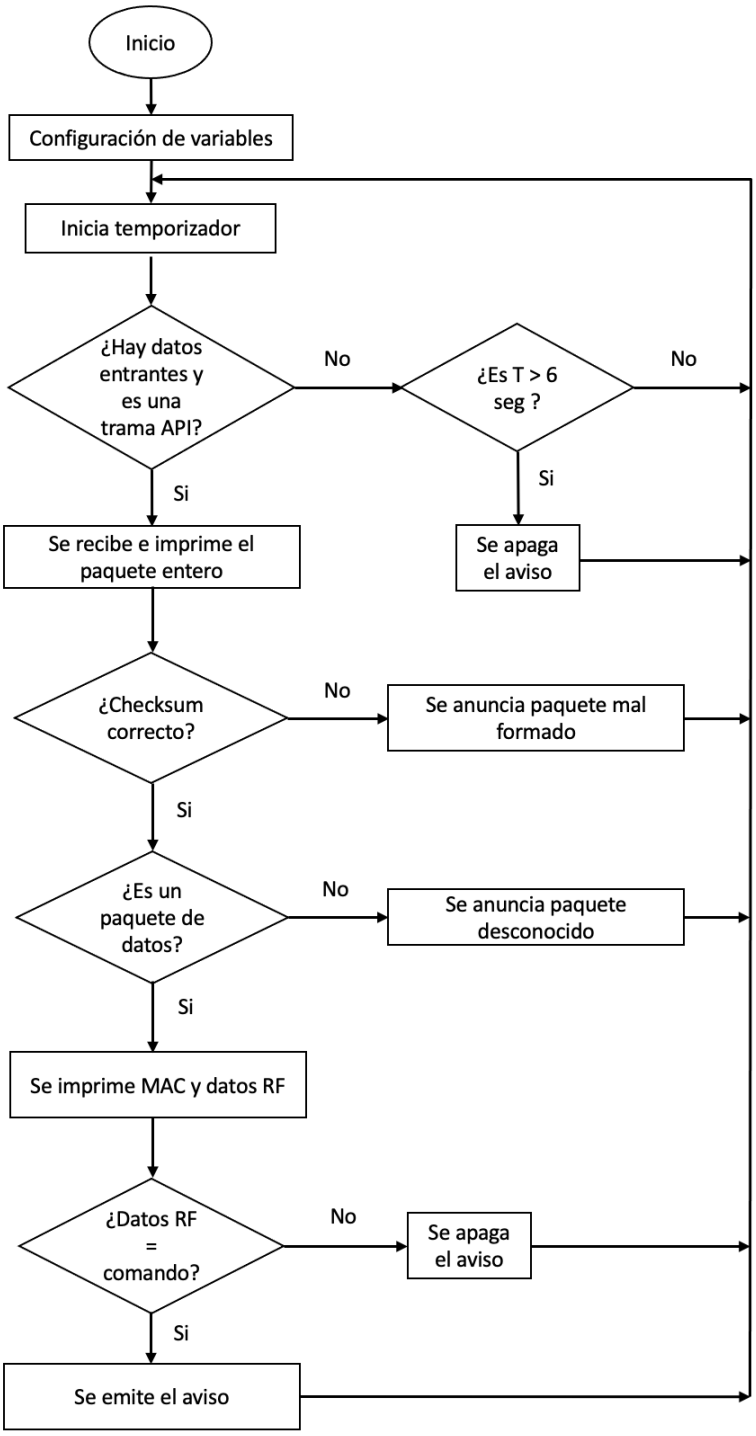


Figura 60. Diagrama de flujo del nodo receptor.

El código del programa está compuesto de varias partes. La primera parte configura las variables y toda la inicialización de las conexiones seriales de Arduino antes de ingresar a la función *loop*. Comienza declarando las variables que forman la trama a recibir.

```
// Se incluye la librería LiquidCrystal.
#include <LiquidCrystal.h>

// Declaración de variables.
byte paqRecibido[20];          // Buffer para el paquete recibido
char datoRecibido[20];         // Phrased data holder
byte srcAddr[10];              // Dirección de origen
int longitudRF;                // Longitud solo de los datos RF recibidos
int longTotal;                 // Longitud total del paquete recibido
int longPaq;                   // Longitud del paquete recibido (MSB y LSB)
byte calcsum;                  // Checksum
byte datoRF;                   // Datos RF

int led = 8;                   // pin 8 para led
int buzzer = 9;                // pin 9 para buzzer activo
int i = 0;                     // Contador LCD
char mensaje[256];             // Variable a mprimir del LCD
char mensaje2[256];            // Variable a imprimir LCD
char trama[20];                // Variable a imprimir de la trama

// Se crea objeto llamado LCD de la clase LiquidCrystal.
// Parametros: (RS, E, D4, D5, D6, D7):
LiquidCrystal lcd (7, 6, 5, 4, 3, 2);

// Variables millis()
int estado = 0;
unsigned long tiempoanterior = 0;
long espera = 6000; // Espera de 6 segundos
```

En la función *setup* se Inicializa la velocidad de comunicación serial a 9600 baudios y establecemos los pines 8 y 9 como salidas digitales a través del método *pinMode* con las variables anteriores. Como en este prototipo, contamos con un display LCD como periférico de salida, se definen las dimensiones del display LCD para el texto a mostrar.

```
void setup() {
  delay(2000);
  Serial.begin (9600);      // Inicializa monitor Serial con una velocidad
de 9600 baudios.
  pinMode (led, OUTPUT);    // pin 8 como salida
  pinMode (buzzer, OUTPUT); // pin 9 como salida
  lcd.begin (16, 2);        // Definición de las dimensiones del LCD
(16x2).
  lcd.clear ();             // Limpia cualquier texto o cursor
que  haya  impreso en la pantalla.
}
```

La función *loop* espera a que el puerto serie esté disponible y comprueba el byte de inicio del paquete, correspondiente al byte 0x7E, con ello se asegura de lo que se

recibe es una trama de datos API. Un bucle captura el paquete y cuenta los bytes entrantes mientras el software Serial está disponible. Cuando se recibe el paquete, se informa al usuario del paquete entrante junto con el contenido del paquete sin procesar. Como en el programa anterior, podremos hacer uso del monitor Serie imprimiendo los detalles antes de procesar el paquete. El funcionamiento de una implementación real no se vería alterado si no existiese esa conexión con el monitor Serie. La primera parte del procesamiento del paquete es calcular la suma de comprobación llamando a la función *calcChecksum*. Si la suma de comprobación es correcta, el programa continúa analizando el paquete invocando al método *procesarPaq* que se encargará de procesar el paquete recibido. En caso de ser incorrecta, se informa a través del monitor Serie y se vacía el buffer para seguir recibiendo nuevos datos.

```
void loop () {
  if (Serial.available () > 16 && 0x7E == Serial.read ()) { // Comprueba
byte de inicio
    paqRecibido[0] = 0x7E;
    longTotal = 1;
    while (Serial.available ()) {
      paqRecibido[longTotal] = Serial.read(); // Se recibe el paquete
entrante
      longTotal ++; // Sigue rastreando bytes entrantes
    } // Fin del while (Serial.available())
    Serial.println ();
    Serial.println ("Recibido un nuevo paquete");
    Serial.print ("El paquete recibido es: ");
    for (int i = 1 ; i < longTotal - 1 ; i++) { // Imprime el paquete
      sprintf(trama, "%02X ", paqRecibido[i]); // Imprime en hexadecimal
      Serial.print(trama);
    }
    Serial.println (paqRecibido[longTotal - 1], HEX); // Último byte del
paquete.
    calcChecksum ();
    if (calcsum == paqRecibido[longTotal - 1]) {
      procesarPaq ();
    }
    else {
      Serial.println ("Error, el paquete está mal formado");  // Imprime
error cuando un paquete llega mal formado
      while (Serial.available ()) {
        Serial.read ();  // En caso de error se vacía el buffer.
      }
    }
  } // Fin de búsqueda de byte de inicio.

  if (millis() - tiempoanterior > espera) {
    digitalWrite (led, LOW);
    digitalWrite (buzzer, LOW);
    lcd.noDisplay ();
```

```
  }
}  // fin del loop
```

La segunda parte del programa contiene las funciones para calcular la suma de comprobación y analizar los datos de los paquetes recibidos. La función *calcChecksum* extrae la longitud del paquete de los primeros dos bytes después del de inicio, y luego se calcula la suma de comprobación antes de volver a sintonizar la función de bucle.

```
/*
 * Función que calcula la suma de verificación o checksum.
*/
void calcChecksum () {
  calcsum = 0;
  longPaq = paqRecibido[1] + paqRecibido[2];
  for (int i = 3 ; i <= longPaq + 2 ; i++) {
    calcsum = calcsum + paqRecibido[i];
  }
  calcsum = 0xFF - calcsum;
}
```

Cuando se llama a la función *procesarPaq*, se informa al usuario que el paquete tiene la suma de comprobación correcta; el código luego determina el tipo de paquete usando la cuarta posición del paquete (correspondiente al tipo de trama). La declaración de conmutación actualmente responde a un paquete de recepción de datos (trama tipo 0x90), dejando abierto el código para incorporar futuros posibles casos de conmutar entre diferentes paquetes. El paquete de datos se maneja analizando la dirección del XBee emisor y extrayendo los datos RF del campo Payload correspondiente a la trama que se utilizarán para ejecutar las distintas situaciones que se definan. Posteriormente, se definen las instrucciones en las que se pueden ejecutar o no los distintos casos en función del valor de los datos recibidos. Durante todo el proceso, nos ayudaremos del Monitor Serie para imprimir la información relativa.

```
/*
 * Función que procesa el paquete recibido extrayendo:
 * Dirección de 64 bits.
 * Datos RF del campo Payload.
 * Posteriormente según que datos RF nos llega entra o no en las
condiciones definidas.
*/
void procesarPaq () {
  Serial.println ("Checksum del paquete correcto ");
```

```
  switch (paqRecibido[3]) {  // comprueba  el tipo de paquete y realiza
cualquier acción
    case 0x90:
      Serial.println ("El paquete es un paquete de datos"); // anuncia el
tipo de paquete
      for (int i = 4 ; i <= 13 ; i++) {   // se obtiene ambas direcciones
del dispositivo de origen
        srcAddr[i - 4] = paqRecibido[i];
      }
      longitudRF = longTotal - 16 ;    // Reducimos solo hasta la longitud
de los datos RF para obtenerlos.
      for (int i = 15 ; i < longitudRF + 15 ; i++) {
        datoRecibido [i - 15] = paqRecibido[i]; // phrase out the data
      }
      Serial.print ("La dirección de 64 bits es: ");   // se imprime la
dirección de 64 bits de origen
      for (int i = 0 ; i < 7 ; i++) {
        sprintf(trama, "%02X ", srcAddr[i]); // hexadecimal
        Serial.print(trama);
      }
      Serial.println (srcAddr[7], HEX);   // fin de la dirección de 64 bits
      Serial.print ("Los datos RF del paquete son: ");   // comenzamos a
imprimir los datos RF
      for (int i = 0 ; i < longitudRF ; i++) {
        Serial.print (datoRecibido[i], HEX);
        datoRF = datoRecibido[i];
      }

      // DATO RF RECIBIDOS //

      // OPCION 0:
      if (datoRF == 0x00) {
        digitalWrite (led, LOW);
        digitalWrite (buzzer, LOW);
        lcd.noDisplay ();
        Serial.println ();
      }
      // OPCION 1:
      if (datoRF == 0x01) {
        digitalWrite (led, HIGH); // Encendido de led
        Serial.println ();
        Serial.println ("LED ON");
        tiempoanterior = millis();
      } else {
        digitalWrite (led, LOW); // Apagado del led
      }
      // OPCION 2:
      if (datoRF == 0x02) {
        digitalWrite (buzzer, HIGH); // Encendido del buzzer
        Serial.println ();
        Serial.println ("BUZZER ON");
        tiempoanterior = millis ();
      } else {
        digitalWrite (buzzer, LOW); // Apagado del buzzer
      }
      // OPCION 3:
      if (datoRF == 0x03) { //11
        i = i + 1;
        lcd.display (); // Enciende la pantalla LCD y muestra cualquier
texto o cursor que haya impreso en la pantalla
```

```
        //Columna 0, fila 0
        lcd.setCursor (0, 0); // Seleccionamos en que columna y en que
linea empieza a mostrar el texto.
        Serial.println ();
        Serial.print ("LCD Fila 1 mostrando: ");
        sprintf (mensaje, "AVISO %d", i);
        Serial.println (mensaje);
        lcd.print (mensaje);
        lcd.setCursor (0, 1);
        Serial.print ("LCD Fila 2 mostrando: ");
        Serial.println ("PELIGRO");
        lcd.print ("PELIGRO");
        tiempoanterior = millis ();
      } else {
        i = 0;
        lcd.clear(); // Borra cualquier texto y posiciona el cursor en la
esquina superior izq.
        lcd.noDisplay (); // Desactiva cualquier texto o cursor impreso en
la pantalla.
      }
      break;  // Terminamos con el paquete recibido
      default: // Anuncia un paquete desconocido
      Serial.println ("Error: el tipo de paquete no se reconoce");
  } // fin del switch
}   // fin de procesarPaq
```

4.5 Pruebas y experimentos realizados

En este apartado se ha procedido a la puesta en marcha del prototipo realizando las pruebas de funcionamiento con los distintos periféricos y en distintas condiciones. Como se ha mencionado en el apartado 3.1 de este documento, el prototipo realizado tiene la finalidad de abarcar de una manera global distintos usos y no un caso final concreto, sino que sirva de modelo de implementación multipropósito. Como un ejemplo práctico a modo de experimentación, se ha decidido incluir todos los escenarios y alimentar tanto el nodo coordinador como el nodo router a través de dos computadoras portátiles, también serán usadas para imprimir datos relativos al código.

Para la realización de las pruebas se han utilizado tres dispositivos, uno comportándose como nodo coordinador y el otro como nodo router. Posteriormente se incluirá un tercer dispositivo como segundo coordinador. Al ser un sistema en el que el emisor está constantemente transmitiendo y el receptor siempre está en espera de una actividad, y dada la flexibilidad de nuestro proyecto, se han incluido diferentes

situaciones de ir simulando los posibles casos de aviso y anuncio para conseguir el funcionamiento deseado en este prototipo desarrollado.

En un primer momento se han realizado pruebas en el interior de una vivienda familiar de un edificio y así evaluar que el prototipo desempeñaba los casos prácticos creados sin ningún problema para posteriormente llevarlo a una zona exterior en el que se podrán evaluar distancias mayores. A continuación, se detallan el conjunto de pruebas realizadas.

4.5.1 Pruebas realizadas en interiores

Con la realización de estas pruebas se puso el prototipo final en acción. Para ello se han recreado varias situaciones con las que hemos ido probando la finalidad de nuestro sistema y ver que cumplía con los objetivos establecidos. El nodo coordinador, que estará fijado en un sitio, estará emitiendo las alertas o anuncios deseados en su rango de red establecido. Por otro lado, el nodo remoto, al estar dentro de este rango, se asociará a su red (según la PAN especificada) y recibirá las instrucciones del coordinador teniendo como resultado la acción deseada. En la siguiente tabla se muestra como está establecida la configuración del prototipo con respecto al número de comandos y el aviso o anuncio resultante.

Tabla 32. Información de comandos y su resultado.

Comando	Notificación
00	Sin acción
01	Encender Led
02	Activar Zumbador
03	Notificación mediante display LCD

Cuando se puso en marcha el prototipo, se observó que el nodo remoto al encontrarse en cobertura con el coordinador se asociaba perfectamente a la red creada por el coordinador y empezaba a recibir las tramas enviadas por este tal y como se esperaba. En las primeras pruebas, se activaron los valores del switch en el µC del

nodo emisor con todos los bits a cero (0000), esta opción consiste en enviar la trama sin ningún tipo de mensaje o aviso. En el µC del nodo receptor se puede observar cómo se reciben los datos correctamente. Esto lo corroboramos a través del Monitor Serie del IDE Arduino del receptor (véase Figura 61). En el podemos ver los detalles la trama recibida con los mensajes que hemos configurado, así como la dirección de 64 bits de origen, correspondiente a la dirección MAC del nodo coordinador y el valor del campo Payload de los datos RF entre otros.

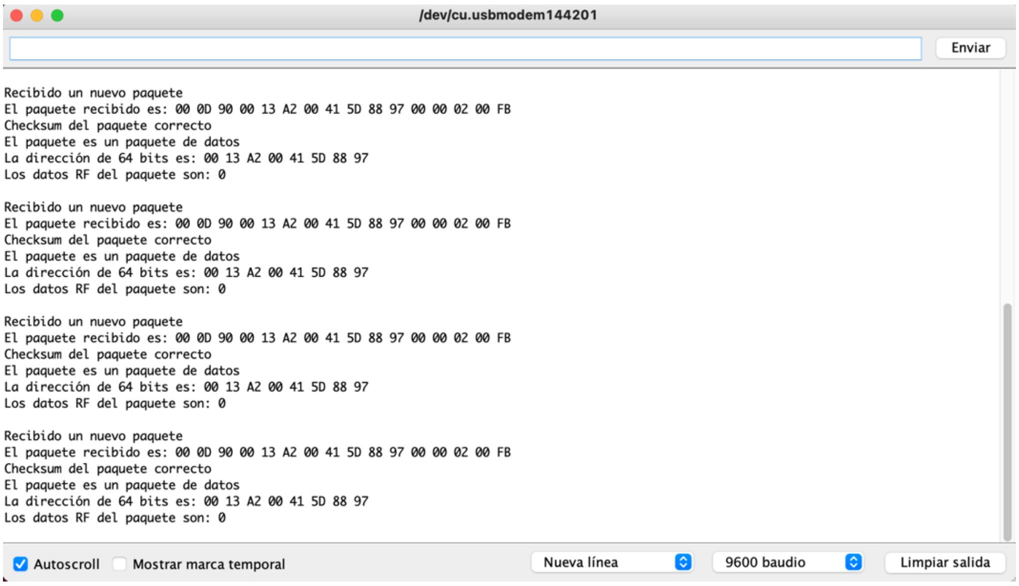


Figura 61. Monitor Serie en µC receptor.

- **Prueba de notificación con led**

  Con esta prueba se configuró el nodo emisor mediante el switch con la combinación 0001, por lo tanto, se especificó que el µC y consecuentemente el coordinador emitiese periódicamente la trama con el valor del comando 01 correspondiente al aviso de encendido del led ubicado en el µC receptor. El nodo remoto, una vez asociado a la red del coordinador recibía sin problemas la trama correspondiente cada 2000 milisegundos como estaba previsto. Una vez se dejaba de emitir la trama por parte del coordinador, al cabo de 6 segundos la notificación led se apagaba tal y como estaba programado. Esto fue necesario dado que si se pierde la conexión con el coordinador debemos retornar al estado inicial de los dispositivos de salida en el receptor y poder actualizar los mismos frente a los mensajes recibido nuevamente (los mismos

o diferentes) del mismo coordinador si se recupera conexión o si se detecta un coordinador diferente adaptarse a las nuevas condiciones y mensajes. Mientras se realizaba la prueba, se activó el Monitor Serie (ver Figura 62) para ver la información obtenida. En la Figura 63 se muestra el resultado visual del led rojo activado al haber recibido la trama con el comando que indica encender el led correspondiente.

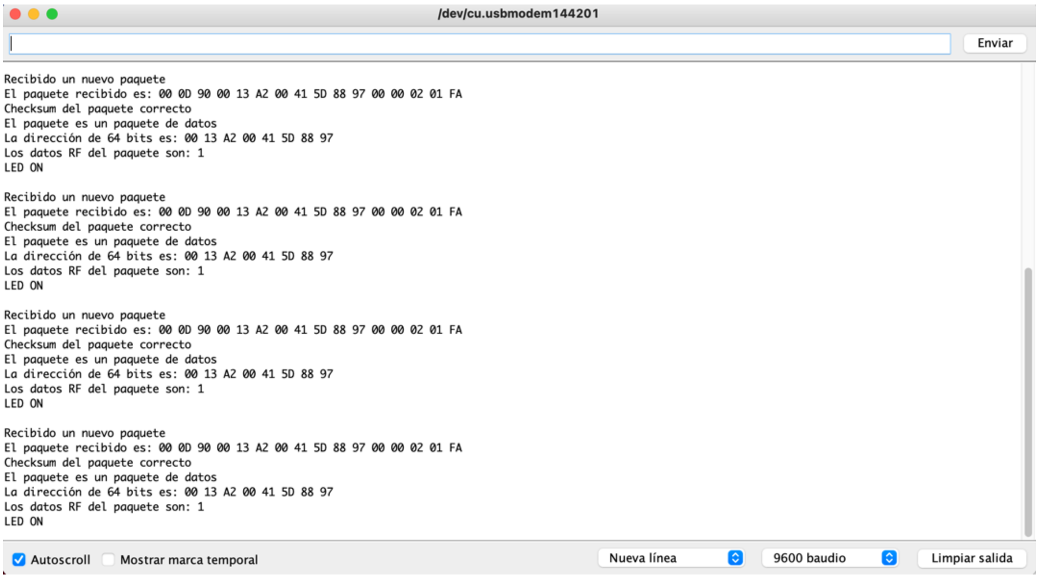


Figura 62. Monitor Serie en µC receptor con salida a LED ON.

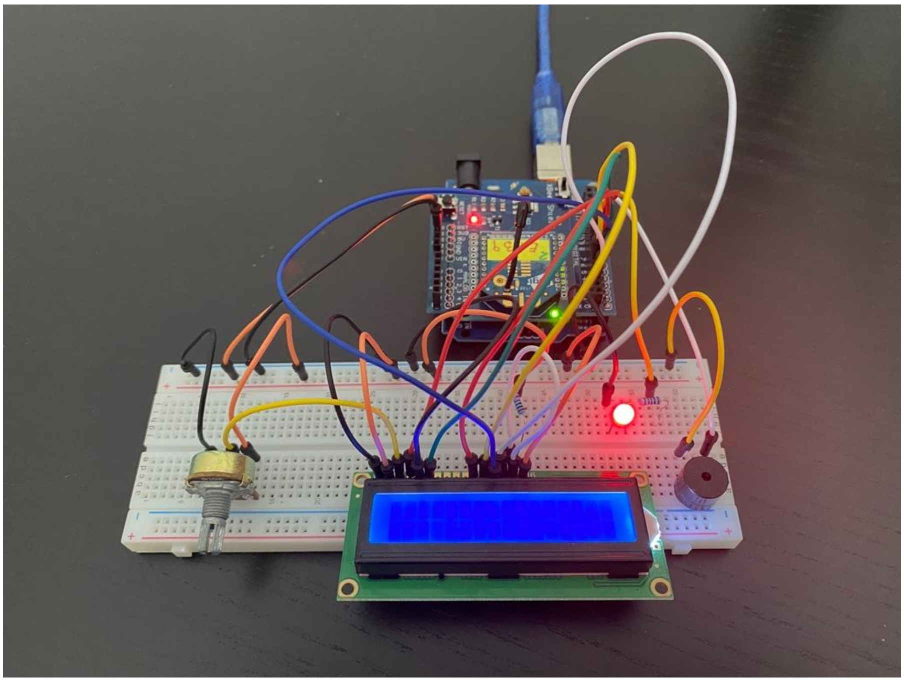


Figura 63. Fotografía del receptor recibiendo notificación led.

- **Prueba de notificación con buzzer activo**

  Con esta prueba se puso al nodo emisor a emitir periódicamente la trama con el valor del comando 02, mediante la combinación del switch 0010, correspondiente al aviso sonoro de activar el zumbador. El nodo remoto, una vez asociado a la red del coordinador recibía sin problemas la trama correspondiente cada 2000 milisegundos como estaba previsto. Una vez se dejaba de emitir la trama por parte del coordinador, al cabo de 6 segundos la notificación sonora se apagaba tal y como estaba programado. La Figura 64 muestra el monitor Serie con las tramas correspondientes.

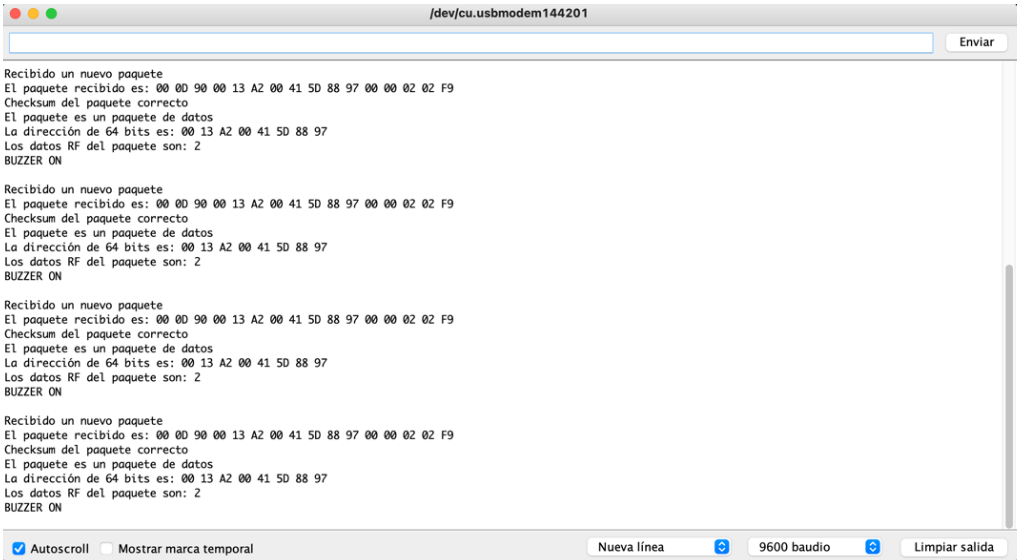


Figura 64. Monitor Serie en µC receptor con notificación por zumbador.

- **Prueba de notificación con display LCD.**

  Con esta prueba se puso al µC emisor a emitir periódicamente la trama con el valor del comando 03, mediante la combinación del switch 0011, correspondiente al aviso de notificación mediante display LCD. El nodo remoto, una vez asociado a la red del coordinador recibía sin problemas la trama correspondiente cada 2000 milisegundos como estaba previsto. En este caso, se observa que el anuncio está codificado para que, en el nodo receptor, más concretamente, en su display LCD se visualice un mensaje basado en caracteres (ver Figura 66), concretamente “AVISO 1 PELIGRO”. Una vez se

dejaba de emitir la trama por parte del coordinador, al cabo de 6 segundos la notificación mediante display LCD se apagaba tal y como estaba programado. Mientras se realizaba la prueba, se activó el Monitor Serie (ver Figura 65) para ver la información obtenida. En la Figura 66 se muestra el resultado visual del display LCD activado al haber recibido la trama con el comando que indicado.

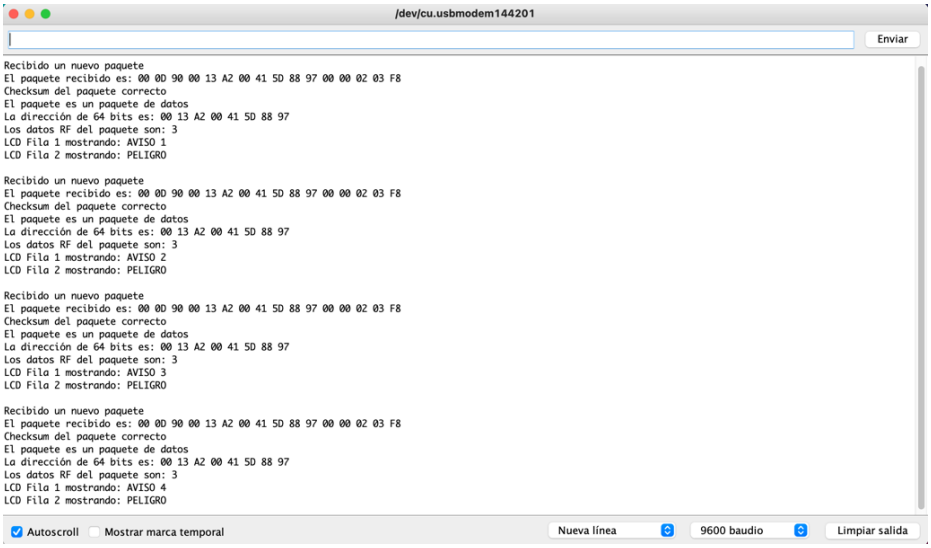


Figura 65. Monitor Serie en µC receptor con mensaje en LCD.

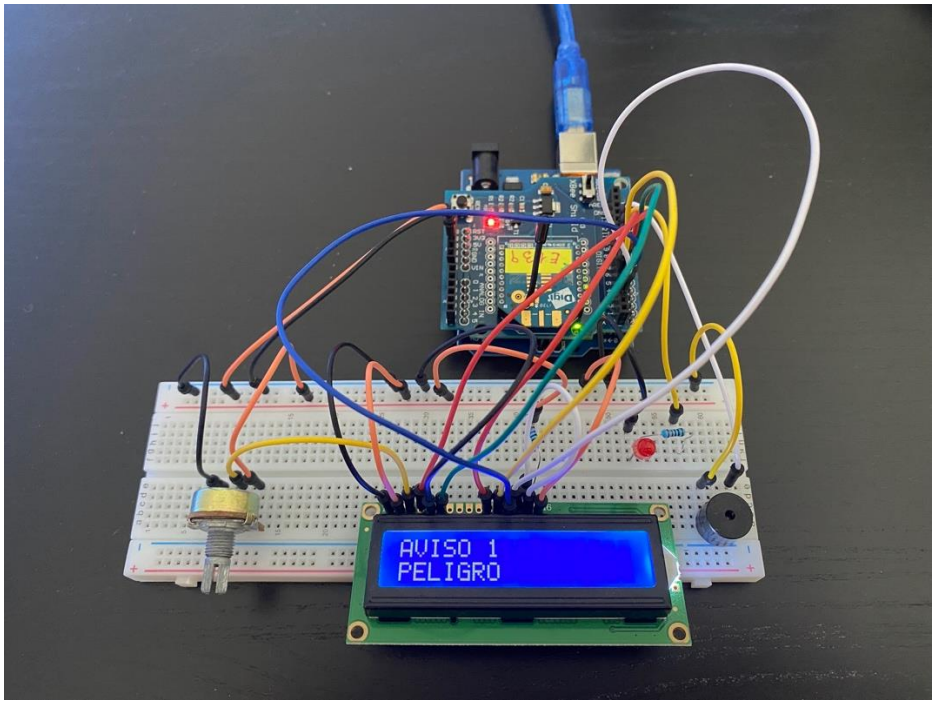


Figura 66. Fotografía del receptor recibiendo notificación en display LCD.

Una vez realizada las diferentes pruebas con las tres posibilidades de información en destino (visual luminosa, sonora o mensaje textual), se comprobó la capacidad de configuración dinámica del µC emisor alternando los valores del switch, como consecuencia el nodo remoto recibía los comandos pertinentes e iba cambiando de notificación a medida que los recibía mediante el encendido y apagado de los correspondientes periféricos del prototipo. Cuando se cambiaba de comando, inmediatamente recibida la trama en el nodo remoto, este apagaba el periférico anterior dando paso al encendido del nuevo (ver Figura 67). Si bien estos cambios no parece que tengan que hacerse en una implementación real y estática si fue necesario hacerla para comprobar la portabilidad del nodo emisor para diferentes usos y aplicaciones (multipropósito).

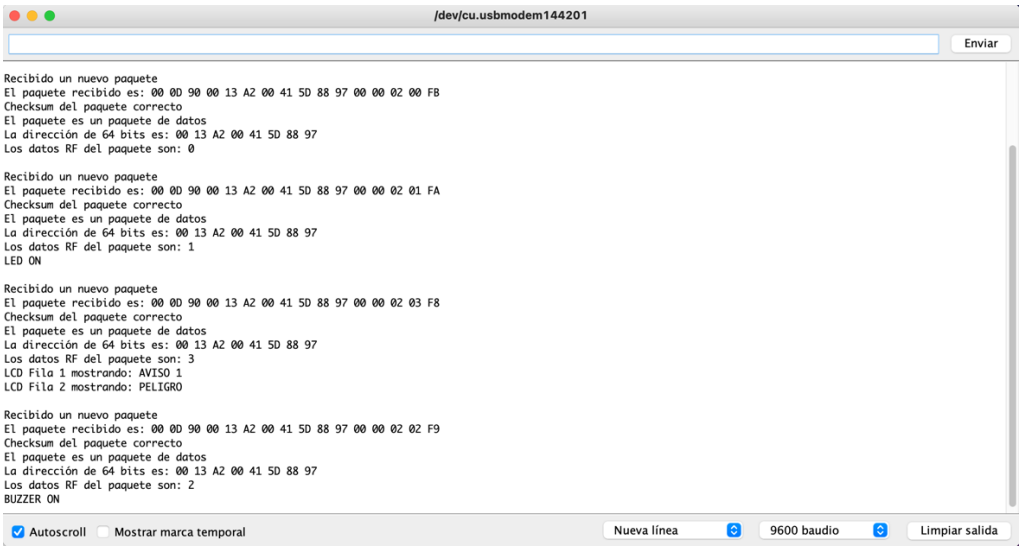


Figura 67. Monitor Serie en µC receptor recibiendo diferentes combinaciones de tramas.

4.5.1.1 Pruebas de conexión, movilidad y cobertura

Si bien no es un objetivo de este Trabajo Fin de Grado, se ha considerado interesante medir y analizar la conectividad y asociación entre el nodo coordinador y los nodos router (aspectos no controlables ya que forman parte del módulo, antena, y restricciones de la tecnología). Concretamente tratamos de determinar las distancias a las que los nodos son capaces de establecer la comunicación. Para la realización se ha procedido de la siguiente manera: 1) En cada simulación el nodo emisor se

mantenía fijo en un sitio, mientras que el nodo receptor se situaba en un primer momento a una distancia cercana, lo alimentábamos y comprobábamos que se establecía la comunicación. 2) Posteriormente se movía el nodo receptor aumentando la distancia hasta la deseada para volver a realizar la comprobación. De esta forma se comprobó la capacidad de conexión que tienen los nodos desde diferentes distancias. En la Tabla 33 se recogen los datos realizados en el interior.

Tabla 33. Conexión entre nodos a distintas distancias (interior).

Distancia entre nodos	Conexión	Transmisión de datos
3 metros	Establecida	Sin pérdidas
6 metros	Establecida	Sin pérdidas
8 metros	Establecida	Sin pérdidas
13 metros	Establecida	Con pérdidas

Como se puede observar, se evaluaron distintas distancias y en todas ellas se logró asociar el nodo remoto con la red del coordinador estableciéndose la conexión y transmisión de los datos. Si bien es cierto que, a los 13 metros y por consiguiente en adelante, aunque la asociación estaba establecida algunas veces no llegaba el paquete dando como resultado que esa ubicación era una situación límite no recomendable si el número de paquetes perdidos es alto por captar un nivel de señal muy bajo. Esta situación límite es debida al no haber visión directa entre los nodos dado que la señal tenía que atravesar varias paredes, ya que nos encontrábamos dentro de una vivienda familiar.

Como sabemos, la comunicación entre los módulos XBee® se realiza a través del aire, por lo que la calidad de la señal inalámbrica puede verse afectada por muchos factores tales como absorción, reflejo de ondas, problemas de línea de visión, ubicación de la antena, etc. Para determinar el alcance de la comunicación y saber si es fiable, es necesario interpretar el parámetro de indicador de fuerza de la señal recibida (RSSI, del inglés *Received Signal Strength Indicator*). Siendo este una escala de referencia que nos sirve para estimar el nivel de potencia de la señal recibida [25]. La Tabla 34 muestra los valores recomendados de RSSI en los que un rango normal, sería de -45

dB a -87 dB. Si este está por debajo de -85 dB suele considerarse como inutilizable debido a la perdida de fiabilidad.

Tabla 34. Valores recomendados de RSSI.

Rango RSSI	Calidad de la Señal
Mayor a -40 dB	Perfecto
-40 dB a -55 dB	Muy bueno
-55 dB a -70 dB	Bueno
-70 dB a -80 dB	Marginal
Menor a -80 dB	Intermitente o no operacional

Una vez conocidas las recomendaciones, se ha utilizado la herramienta *Range Test* proporcionada por el software XCTU para testar estos aspectos en el prototipo implementado. Esta prueba demuestra el rango de RF y la calidad del enlace entre los dos módulos XBee® en la misma red. La realización de esta prueba de alcance dará una indicación inicial del rendimiento de comunicación esperado en el lugar de las pruebas. Evidentemente esto debería realizarse allí donde se desplegase una instalación real definitiva. Los datos aquí obtenidos son muy dependientes de la ubicación no pudiendo ser extrapolables.

En la prueba realizada, al menos un módulo XBee® tiene que ser conectado directamente al computador por medio del XBee USB Explorer. En este caso se ha escogido el coordinador ya que es el nodo que siempre estará fijo en un lugar. A continuación, en la Figura 68 se muestran los resultados del test realizado en interior poniendo el nodo receptor a la distancia más lejana del coordinador, siendo esta los trece metros evaluados anteriormente.

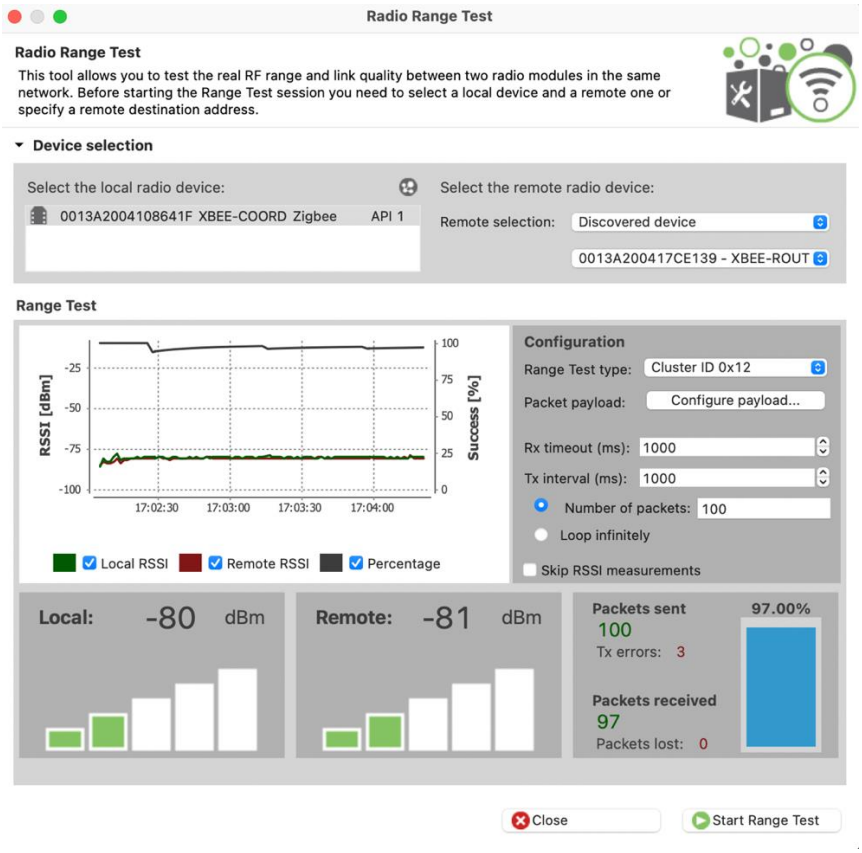


Figura 68. Pruebas en interiores: niveles de potencia de señal.

Observamos que se obtienen niveles potencia desde -80 dBm hasta los -81 dBm teniendo como resultado una calidad de señal en el límite de lo recomendado; lo que era de esperar, ya que al estar en el interior de una vivienda la señal tiene que atravesar varios obstáculos y como se ha comentado anteriormente, a la máxima distancia realizada en el interior, el envío de los paquetes al nodo remoto sufría de intermitencias. También se puede apreciar en el test que el envío de paquetes al nodo remoto a veces era fallido dando como resultado una fiabilidad del 97%,

4.5.2 Pruebas realizadas en exteriores

A continuación, se llevó el equipo a una zona exterior para realizar los ensayos y poder cubrir distancias mayores, veremos cómo se comporta el prototipo y que capacidad tienen los nodos de transmitir datos en diferentes escenarios.

Según la ficha técnica del módulo XBee® S2C [22] se sabe que el alcance exterior puede ser de hasta 1200 m. Si bien es cierto que este valor depende de las condiciones y puede variar en función de la potencia de transmisión, la orientación del transmisor y el receptor, altura de la antena transmisora y receptora, fuentes de interferencia, etc. Por lo tanto, para garantizar un correcto establecimiento de la red pudiendo asegurar la fiabilidad de los datos y evitar lo menos posible cualquier tipo de interferencia se procedió a situar el equipo en una zona con visión directa en todo momento entre los nodos, situándolos a la misma altura y únicamente teniendo como posibles fuentes de interferencia obstáculos eventuales (tránsito de coches, personas, etc.). El lugar en concreto es una calle con 255 metros de longitud totalmente plana situada en Maspalomas (véase Figura 69).

Figura 69. Pruebas en exteriores: situación geográfica real con trayectoria.

Se puso en escena el prototipo evaluando las pruebas realizas en exterior. En un primer momento se estableció la conexión de los nodos desde una distancia cercana a los 2 metros y posteriormente se fue alejando el nodo receptor hasta los 255 metros,

al mismo tiempo se iba alternando entre los diferentes comandos del switch. Otra prueba que se realizó fue inicialmente tener el nodo receptor no asociado a la red del coordinador y situarlo en un primer momento fuera del alcance del coordinador. Una vez que el nodo receptor entraba dentro del rango del coordinador se asociaba a su red y empezaba recibir las tramas de este. Por la situación geográfica, el límite de este rango se comprobó que era de 265 metros ya que a partir de esa distancia ya no había visión directa entre los nodos (dado que nos encontrábamos en una pendiente) y por lo tanto, se perdía la comunicación entre los nodos.

A continuación, se muestran los resultados de la prueba de rango realizado en exterior poniendo el nodo remoto a la distancia más lejana del coordinador, siendo esta los 255 metros evaluados anteriormente (ver Figura 70).

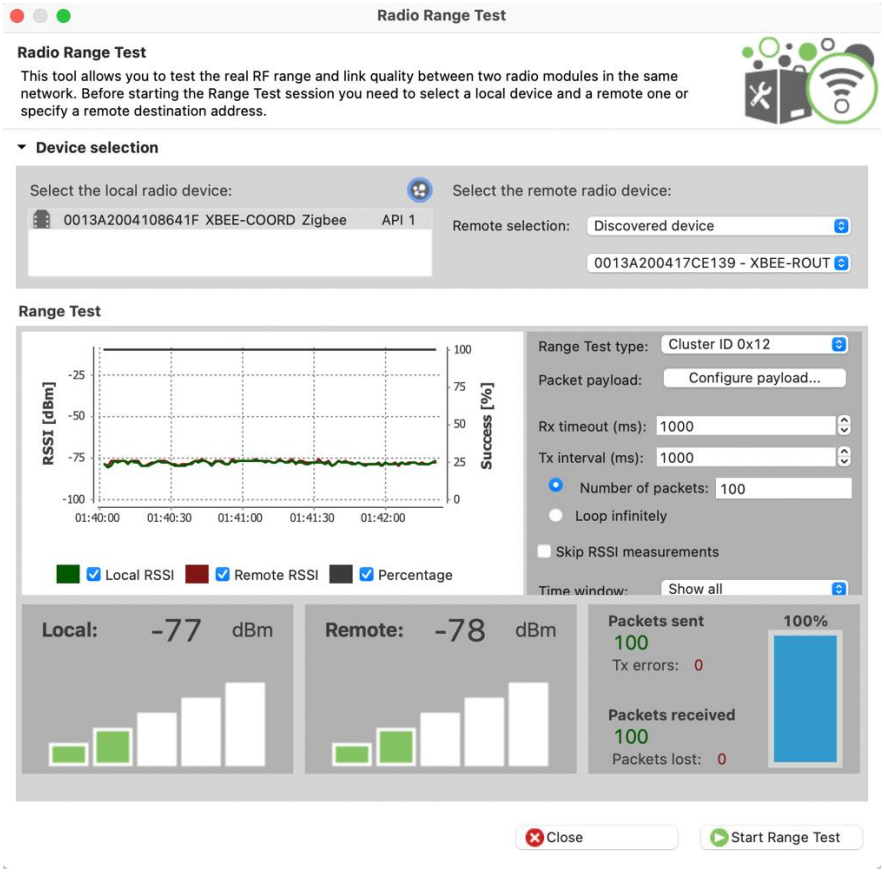


Figura 70. Pruebas en exteriores: niveles de potencia de señal.

Se puede observar que obtenemos unos niveles de potencia entre los -77 dBm y -78 dBm, por lo que aun teniendo una calidad de señal media los nodos eran capaces de

entregar y recibir el 100% de los paquetes, teniendo como resultado una conexión de la red estable y operativa como se comprobó.

En los dos procesos evaluados, se comprobó por los resultados obtenidos que el funcionamiento de la red, así como el intercambio de la información entre nodos era totalmente confiable mientras hubiera visión directa.

4.5.3 Pruebas de conexión, movilidad y cobertura con varios coordinadores

Para determinar las condiciones que debe cumplir el sistema para un despliegue generalizado y su aplicabilidad práctica, es necesario analizar el comportamiento de los receptores ante la detección simultánea de múltiples coordinadores. El objetivo de esta prueba es ver si es posible que los receptores puedan recibir datos de varios coordinadores, como se comportan y si son capaces de interpretar los mensajes procedentes de cada uno de ellos. Por consiguiente, será necesario configurar con el mismo rol al menos a dos coordinadores, con sus mismos parámetros y su configuración requerida. En el caso de celdas aisladas ha sido probado que no hay problema alguno pues los receptores se asocian al coordinador que detecten en cada momento.

Tenemos que recordar que ZigBee según sus especificaciones, cada coordinador gestiona una red (con su correspondiente PAN ID y el canal de trabajo), por eso se recomienda usar uno solo por cada red. Además, cuando un router o dispositivo final se asocia a una red establecida por un coordinador, este guarda y conserva los parámetros de la red, asignados por el coordinador, independiente de los ciclos de energía (apagado o encendido del dispositivo). Por último, hay que indicar que esta tecnología no está orientada a cambios de celdas y menos aún de redes, a diferencia de otras tecnologías, como WiFi o Bluetooth. La información que el router o dispositivo final conserva es la siguiente:

- PAN ID de 64 bits y 16 bits.
- Canal operativo.
- Política de seguridad y valores de contador de tramas.
- Tabla de hijos, es decir, dispositivos finales que se unen al coordinador.
- Tabla de vinculación.
- Tabla de grupos.

Por tanto, en nuestro sistema los dispositivos router conservarán esta información de forma indefinida hasta que abandone la red (deje de estar asociado). Cuando el dispositivo abandona la red, este pierde el PAN ID anterior, su canal operativo y los demás datos comentados anteriormente.

Tras las pruebas realizadas, se ha detectado que para que varios dispositivos coordinadores puedan coexistir en la misma zona de cobertura de nuestro sistema y puedan ser detectados por cualquier receptor, deben tener completamente fijados e igualados los mismos parámetros de configuración de la red según se especificó en el apartado 4.2.1 y a efectos prácticos se comporten como un único coordinador.

Solo de la forma descrita, el nodo receptor tendrá la capacidad, desde un punto de vista de un sistema en movilidad, de que allá donde se haya instalado un nodo coordinador, de seguir operando en la misma red y, por lo tanto, tener la capacidad de recibir los datos de este sin desasociarse y asociarse a otra nueva red, incluso aún cuando los dos nodos emisores se encuentren en el mismo área de cobertura y el nodo receptor se halle en los rango de red de estos.

De forma más detallada, el experimento consistió en simular que se tienen dos coordinadores (coordinador 1 y coordinador 2), con la misma información de la red. Para ello, el coordinador 2 (módulo XBee® únicamente) se conectó a la computadora portátil, configurado a través del software XCTU, y programado para emitir indefinidamente el mismo tipo de trama API, similar a la usada en la configuración completa del prototipo (µC Arduino+XBee). En un primer momento cada uno de ellos se situó de tal forma que no interfiriesen en su área de cobertura y emitiendo sus tramas correspondientes, estando el “nodo receptor” primeramente en el rango de

cobertura del coordinador 1, para posteriormente abandonar el rango de red de este y entrar en el rango del coordinador 2, recibiendo sus datos correctamente. La Figura 71 muestra la situación práctica experimental.

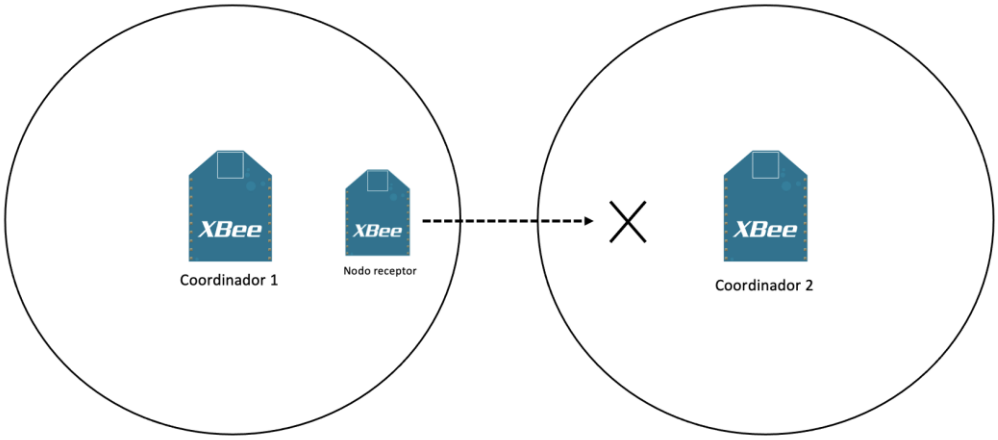


Figura 71. Pruebas de movilidad con múltiples coordinadores en celdas no solapadas.

De la prueba realizada se comprobó que el nodo receptor recibía sin problemas en un primer momento las tramas emitidas por el coordinador 1 con el comando correspondiente al encendido del led, y cuando se situó dentro del área de cobertura del coordinador 2, comenzó a recibir de inmediato las tramas emitidas por este. Durante el proceso se pudo observar como a través el Monitor Serie (véase Figura 72) activado en la computadora del nodo receptor se cambiaban las direcciones MAC de los correspondientes módulos XBee® como estaba previsto, al ser la dirección de origen distinta predefinida en cada módulo necesariamente.

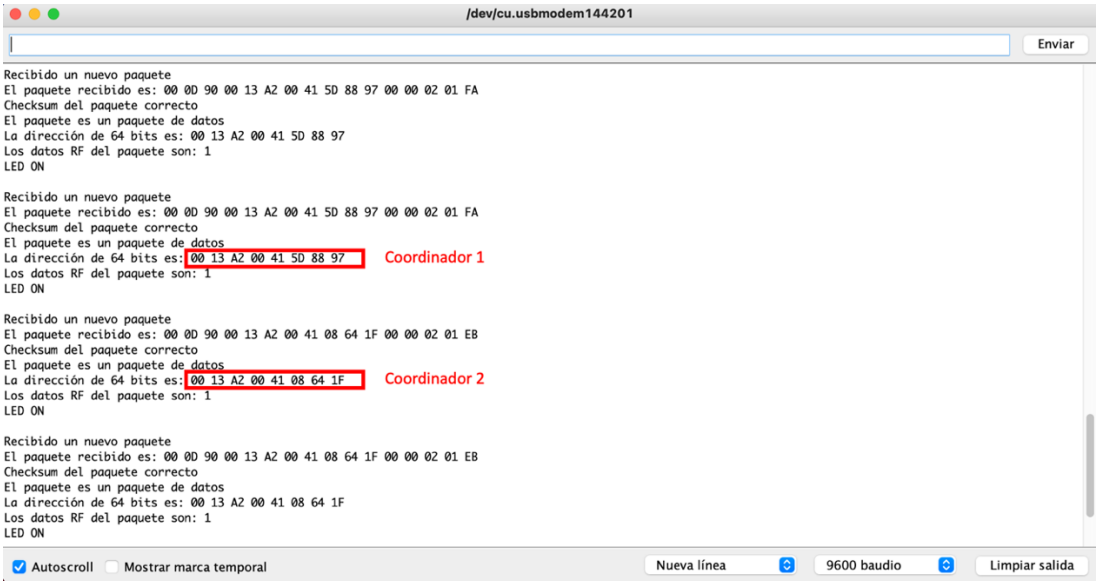


Figura 72. µC receptor recibiendo tramas de varios coordinadores con celdas no separadas.

Otra situación simulada fue hacer que los dos coordinadores tuviesen una zona de cobertura compartida y el dispositivo router estuviese en dicha zona (véase Figura 73). Esta situación se puede dar por la hipotética movilidad del nodo receptor como por un despliegue. Para esta prueba, se decidió que ambos coordinadores emitieran diferentes notificaciones (coordinador 1: activar buzzer y coordinador 2: encender LED).

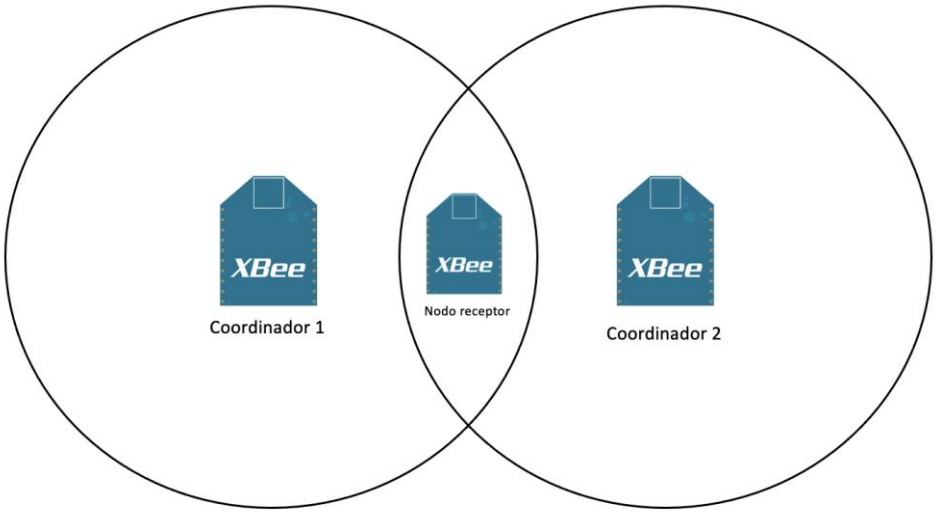


Figura 73. Pruebas de movilidad con múltiples coordinadores en celdas solapadas.

Con la realización de esta prueba se confirmó que el nodo receptor puede recibir mensajes emitidos por ambos dispositivos coordinadores según lo explicado, como se observa en la Figura 74.

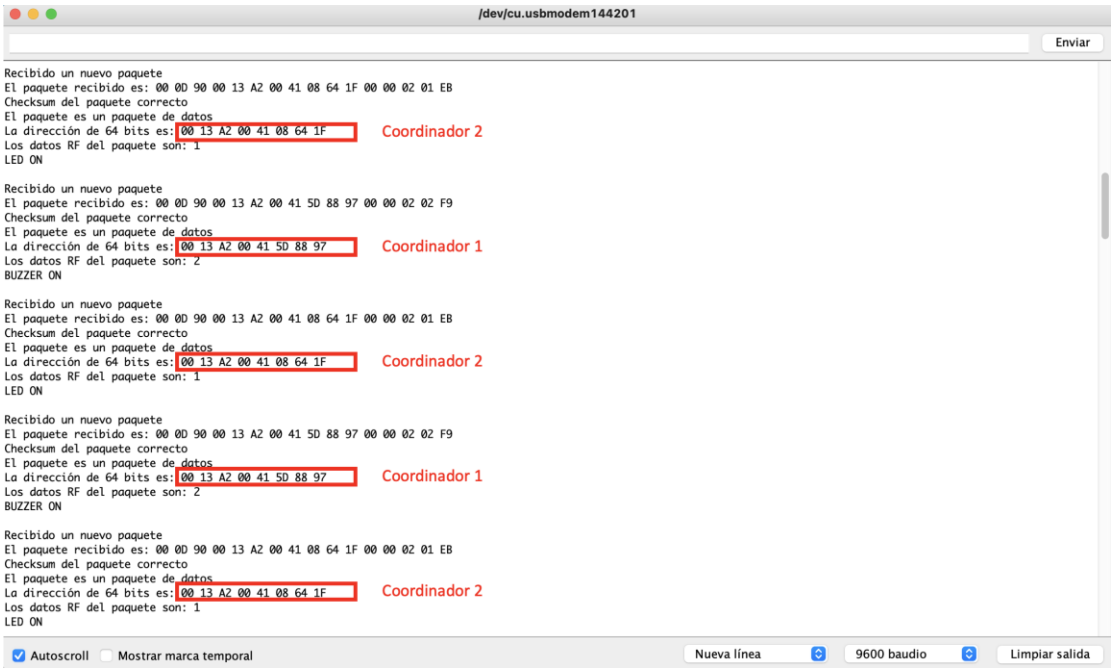


Figura 74. µC receptor recibiendo tramas de varios coordinadores con celdas solapadas.

Como resumen tenemos que destacar la limitación que presenta instalar varios coordinadores en la mima zona de cobertura para una misma red; no basta con que tengan el mismo PAND ID de 64 bits, sino que requieren se programados de forma similar a las configuraciones realizadas para que los dispositivos router puedan recibir mensajes de ambos simultáneamente. Como esta programación se realiza en tiempo de configuración, ya que de lo contrario los parámetros se asignan aleatoriamente por cada coordinador, no parece que forzarlos sea una solución práctica para una red desplegada, por tanto es recomendable evitar dicha coexistencia y evitar zonas solapadas y solo unificar el PAN ID de 64 bits.

4.6 Resultados

Con las diferentes pruebas realizadas se ha pretendido mostrar que el sistema propuesto y especialmente de forma experimental con el prototipo implementado cumple con la finalidad desarrollada. Se ha propuesto y elaborado un sistema que permite enviar y recibir diferentes notificaciones o prealertas para nodos o sistemas que están físicamente distanciados gracias a tecnologías inalámbricas. Estos elementos se pueden comunicar, consiguiendo desde una o varias estaciones emisoras emitir los avisos a los posibles receptores, tanto en cuanto a la forma y dispositivos requeridos como en cuanto a la funcionalidad y aplicabilidad del sistema. Se ha enfocado que el prototipo abarque de manera general distintos ejemplos prácticos y permitiendo una flexibilidad a la hora de configurarlo, ya sea desde la fuente emisora como desde la receptora; esto permite que el sistema sea desde lo más simple posible hasta lo más complejo según la utilidad que se le quiera otorgar en futuros usos.

El uso o aplicación de este sistema en ningún momento se pensó que pretendiera cubrir grandes distancias ni tampoco soportar grandes velocidades o ancho de banda (véase apartado 2.1.2), para ello está LR-WPAN, WiFi u otras; usar ZigBee con sus limitaciones, que aprovecha el uso del espectro no licenciado se considera acertado para el objetivo propuesto. El medir la cobertura y calidad de los módulos XBee® se consideró necesario pues nos permite comprobar el rango de cobertura del fabricante de estos módulos, así como su comportamiento en interior y exteriores; de los resultados obtenidos nos podemos hacer una idea más real de las capacidades de estos módulos así delimitar su uso, según se desee en un despliegue real.

Capítulo 5. CONCLUSIONES Y LÍNEAS FUTURAS

A lo largo de este documento se ha descrito el procedimiento seguido para el desarrollo de un sistema de aviso y anuncio dando lugar a un prototipo formado por microcontroladores Arduino y módulos XBee®. Como conclusión, en este capítulo se hará un sumario de las conclusiones extraídas de este desarrollo, en relación a los objetivos planteados para el mismo y a los aspectos que requieren de esfuerzos adicionales más allá del alcance de este proyecto.

5.1 Conclusiones

El objetivo de este proyecto ha sido implementar un sistema de comunicación inalámbrico multipropósito de aviso y anuncio mediante alertas o notificaciones. Para ello se plantea una red de nodos diseñada para que los datos se envíen de forma elemental uno a uno y de forma más general uno a varios receptores en función de la aplicación o uso asignado.

Al ser una red orientada al bajo consumo se ha decidido utilizar un conjunto de protocolos de comunicación ZigBee. En esta, se destaca la existencia de un nodo especial, el coordinador, encargado de realizar la configuración de la red y gestionarla en todo momento. Al ser este el nodo central de la red, se plantea un sistema de comunicación con el resto de los nodos y sea este el que emita las órdenes/mensajes adecuados que recibirán los demás dispositivos que indicarán los correspondientes avisos y anuncios. En cuanto al prototipo desarrollado a modo de experimentación real del sistema propuesto, se ha utilizado los módulos XBee® S2C ZigBee de Digi.

Se han descrito tres tipos de configuraciones posibles de cara a plantear diferentes configuraciones del sistema: configuración básica, ampliada y completa para según necesidades o posibilidades de implementar uno u otras. La primera de estas configuraciones está caracterizada por el uso único de los módulos XBee® sin la necesidad de utilizar microcontroladores, si bien es la configuración que menos consumo de energía produce, tiene la limitación de que la funcionalidad otorgada se reduce a la que trae el propio hardware/firmware. La segunda y tercera configuración consiste en incorporar el uso de microcontroladores permitiendo con ello posibles tratamientos más complejos de datos y ejecuciones de código, gobernando los nodos según nuestra necesidad y posibilitando la integración de distintos recursos hardware como periféricos, actuadores, sensores... en los transmisores y receptores.

Para el desarrollo del prototipo se ha implementado la opción más completa incluyendo el microcontrolador Arduino junto con el módulo XBee® en el nodo transmisor y nodo receptor. De esta manera se ha podido abordar el caso más

completo, con un mayor número de pruebas y permitir al sistema ofrecer una gran versatilidad.

Se ha estudiado el protocolo empleado, sus funcionalidades y en base a ello se desarrolló la lógica para un programa de control y gestión tanto para la fuente emisora como la receptora utilizando el lenguaje propio de Arduino, C++. Dicho programa cumple con su correcto funcionamiento asegurando la correcta formación de los datos a enviar y recibir entre los módulos.

Se ha probado el prototipo desarrollado en interiores y llevándolo a zonas de exterior donde se realizaron las pruebas que han podido confirmar que el prototipo cumple con los objetivos establecidos para este proyecto llegando a extrapolarse a situaciones reales según la necesidad.

Además, cabe destacar, que la fácil configuración de los módulos, el poder implementar redes de hasta 65535 nodos además de la creación de ellas, así como la de asegurar la comunicación y el envío de los datos a través del modo API mediante tramas, garantizando la confiabilidad de los datos enviados y recibidos en la red, y pudiendo dotar a cada módulo XBee® de un microcontrolador Arduino convierten esta tecnología en una de las ideales para realizar este tipo de aplicaciones de bajo coste, reducido alcance y velocidad.

Por otra parte, este proyecto me ha significado una experiencia muy enriquecedora ya sea a nivel académico como en lo personal. Durante la elaboración de este, he tratado con dificultades que pueden surgir de manera habitual y he tenido que saber desenvolverme para buscar las soluciones adecuadas y completarlo. Además, al no haber interactuado nunca con microcontroladores como Arduino y tener que aprender su ámbito de funcionamiento que desconocía, me ha dado la oportunidad de adquirir nuevos conocimientos para mi formación. También, el poder profundizar más en el área de la tecnología ZigBee y en este tipo de redes me han permitido tener una comprensión más amplia de cómo funcionan y de las posibilidades que presentan en cuanto a su aplicación.

Finalmente, y a modo de resumen, podemos afirmar que el sistema propuesto cumple con los requisitos y que el prototipo desarrollado satisface la necesidad del usuario ofreciendo a su vez una gran versatilidad ya que su funcionamiento se puede extrapolar a diferentes situaciones a través de su configuración y de la capacidad de añadir elementos hardware respondiendo a las necesidades del entorno. Además, añadiendo la tecnología ZigBee, la competencia del sistema para crear redes de distintas topologías y el manejo de las mismas se concluye que este proyecto supone una buena base de cara a futuras ampliaciones mucho más ambiciosas.

5.2 Líneas futuras

Durante el desarrollo de este Trabajo Fin de Grado, se han identificado varias áreas de mejora a las que sería interesante dar una solución satisfactoria. Algunas de las áreas sobre las que se debería trabajar en esfuerzos futuros se resumen a continuación.

- Dotar al sistema y particularmente para el prototipo, de un sistema de alimentación autónoma mediante baterías. Sabemos que la alimentación es un parte importante en el ámbito de la movilidad, por ello se considera que en futuras mejoras de este proyecto se debería dotar de baterías e integrarlo en una caja para hacerlo verdaderamente portable, quedando implementado en. un sistema integrado.

- Implementar el modo Sleep en los módulos XBee®. Actualmente al no trabajar con baterías en este prototipo no se ha abordado la funcionalidad de este modo. Evidentemente en un sistema que este inactivo cierto tiempo entre transmisión y transmisión debería pasar a modo ahorro de energía. Por otro lado, esto puede estar reñido con la necesidad de que los anuncios de alerta o aviso no pueden detenerse por estar el sistema dormido. Sería interesante incorporarlo pues las ventajas de este modo suponen un ahorro de energía minimizando el consumo cuando los nodos no son usados. De esta forma el nodo permanece

inactivo con un mínimo de energía prologando notablemente la duración de las baterías en el tiempo, además se disponen de distintos funcionamientos dentro de este modo.

- Ampliación de la red incorporando dispositivos que actúen como nodos intermedios. De esta manera se aprovecha los beneficios de la topología en malla, aumentando el rango de cobertura, creando una red dinámica que se autoconfigure y elija la mejor ruta para comunicarse y transmitir los datos.

- Implementar nuevas funcionalidades en el código, como puede ser la posibilidad de enviar y recibir multi-tramas, es decir, tramas de distintos tipos o incorporando el envío de datos desde el nodo destino al coordinador, actualmente la comunicación es un solo sentido, siendo esta del coordinador a los demás nodos, sería interesante que lo nodos remotos pudieran enviar avisos al coordinador también, dando lugar a notificaciones en ambas direcciones según las necesidades. Cabe mencionar que cualquier ampliación o mejora del programa estará condicionada a la capacidad de procesamiento de los µC incorporados y autonomía de los nodos sobre el que se ejecute el código (en caso de poderse aplicar).

REFERENCIAS

[1] «WPAN ('Wireless Personal Area Network') - CCM». https://es.ccm.net/contents/821-wpan-wireless-personal-area-network (accedido jun. 09, 2021).

[2] «6LowPAN | Official Website of ERNET India Education & Research Network». https://ernet.in/content/6lowpan (accedido jun. 09, 2021).

[3] C. Wang, T. Jiang, y Q. Zhang, *ZigBee Network Protocols and Applications*. 2016.

[4] «Home - Zigbee Alliance». https://zigbeealliance.org/ (accedido jun. 09, 2021).

[5] «IEEE 802.15.4-2003 - IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 15: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area». https://standards.ieee.org/standard/802_15_4-2003.html (accedido jun. 09, 2021).

[6] «Time to consider ZigBee in your next design? ZigBee vs. Bluetooth vs. WiFi - Embedded blog - System - Arm Community». https://community.arm.com/developer/ip-products/system/b/embedded-blog/posts/time-to-consider-zigbee-in-your-next-design-zigbee-vs-bluetooth-vs-wifi (accedido jun. 09, 2021).

[7] Z. Alliance, «ZigBee Specification», *Stand. Oct*, 2015.

[8] «Industrial IoT (IIoT) Devices and Services for M2M Networking | Digi International». https://www.digi.com/ (accedido jun. 11, 2021).

[9] «Digi XBee and XBee-PRO Zigbee RF Modules | Digi International». https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-zigbee (accedido jun. 11, 2021).

[10] «XBee®/XBee-PRO S2C Zigbee® RF Module User Guide», 2020. [En línea]. Disponible en: https://www.digi.com/resources/documentation/digidocs/pdfs/90002002.pdf.

[11] «XCTU - Download and Install the Configuration Platform for XBee/RF Solutions | Digi International». https://www.digi.com/products/embedded-systems/digi-xbee/digi-xbee-tools/xctu (accedido jun. 11, 2021).

[12] «XCTU Configuration and Test Utility Software User Guide», 2020. [En línea]. Disponible en: https://www.digi.com/resources/documentation/digidocs/PDFs/90001458-13.pdf.

[13] «AT commands». https://www.digi.com/resources/documentation/Digidocs/90001456-13/concepts/c_at_commands.htm?TocPath=XBee transparent mode%7CCommand mode%7C_____1 (accedido jun. 09, 2021).

[14] «Arduino - Home». https://www.arduino.cc/ (accedido jun. 09, 2021).

[15] «Smart | Connected | Secure | Microchip Technology». https://www.microchip.com/ (accedido jun. 09, 2021).

[16] «Arduino - Libraries». https://www.arduino.cc/en/Reference/Libraries (accedido jun. 11, 2021).

[17] «Arduino Boards & Modules - Arduino». https://store.arduino.cc/arduino-genuino/boards-modules (accedido jun. 11, 2021).

[18] «When we share, everyone wins - Creative Commons». https://creativecommons.org/ (accedido jun. 10, 2021).

[19] «ATmega328P - 8-bit AVR Microcontrollers». https://www.microchip.com/wwwproducts/en/ATmega328p (accedido jun. 11, 2021).

[20] «Arduino Uno Rev3 | Arduino Official Store». https://store.arduino.cc/arduino-uno-rev3 (accedido jun. 11, 2021).

[21] «Pull-up Resistor and Pull-down Resistor Explained». https://www.electronics-tutorials.ws/logic/pull-up-resistor.html (accedido jun. 10, 2021).

[22] «Digi XBee Zigbee Datasheet | Digi International». https://www.digi.com/resources/library/data-sheets/ds_xbee_zigbee (accedido jun. 10, 2021).

[23] «Remote AT Command Request - 0x17». https://www.digi.com/resources/documentation/Digidocs/90000982/reference/r_frame_0x17.htm%3FTocPath%3DAPI%2520operation%7CAPI%2520types%7C_____5 (accedido may 22, 2021).

[24] «Transmit Request - 0x10». https://www.digi.com/resources/documentation/Digidocs/90001480/reference/r_frame_0x10.htm (accedido may 22, 2021).

[25] J. Cruz Núñez Pérez, A. Bonilla Rodriguez, y A. Calvillo Téllez, «Estimación del alcance de radiotransmisores Xbee Estimation of Range of Xbee Radio Transmitters», 2017.

ANEXOS

ANEXOS

En esta sección se recopilan otra documentación o recursos utilizados durante este trabajo. Estos anexos consisten en las hojas de datos empleadas para los dispositivos y módulos, así como los esquemáticos de los circuitos eléctricos del prototipo construido.

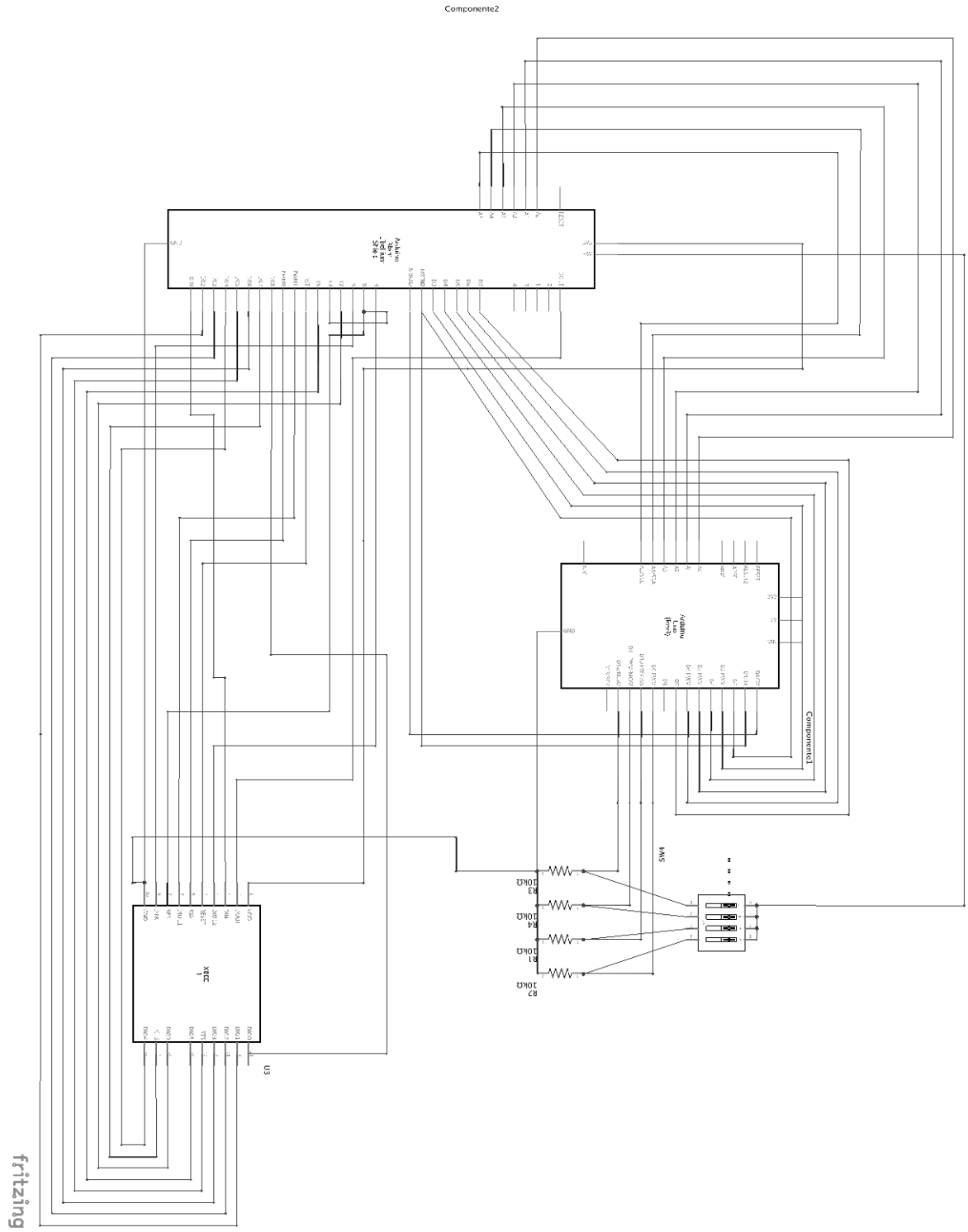
A1. Datasheet Digi XBee® S2C ZigBee

SPECIFICATIONS	Digi XBee® S2C Zigbee Standard	| Programmable
PERFORMANCE		
TRANSCEIVER CHIPSET	Silicon Labs EM357 SoC	
DATA RATE	RF 250 Kbps, serial up to 1 Mbps	
INDOOR/URBAN RANGE*	Up to 60 m (200 ft)	
OUTDOOR/RF LINE-OF-SIGHT RANGE*	Up to 1200 m (4000 ft)	
TRANSMIT POWER	3.1 mW (+5 dBm) / 6.3 mW (+8 dBm) boost mode	
RECEIVER SENSITIVITY (1% PER)	-100 dBm / -102 dBm boost mode	
FEATURES		
SERIAL DATA INTERFACE	UART, SPI	
CONFIGURATION METHOD	API or AT commands, local or over-the-air (OTA)	
FREQUENCY BAND	ISM 2.4 GHz	
FORM FACTOR	Through-hole, surface mount	
INTERFERENCE IMMUNITY	DSSS (Direct Sequence Spread Spectrum)	
ADC INPUTS	(4) 10-bit ADC inputs	
DIGITAL I/O	15	
ANTENNA OPTIONS	Through-hole: PCB antenna, U.FL connector, RPSMA connector, or integrated wire SMT: RF pad, PCB antenna, or U.FL connector	
OPERATING TEMPERATURE	-40° C to 85° C (-40° F to 185° CF)	
DIMENSIONS (L X W X H) AND WEIGHT	Through-hole: 2.438 x 2.761 cm (0.960 x 1.087 in) SMT: 2.199 x 3.4 x 0.305 cm (0.866 x 1.33 x 0.120 in)	
PROGRAMMABILITY		
MEMORY	N/A	32 KB flash / 2 KB RAM
CPU/CLOCK SPEED	N/A	HCS08 / up to 50.33 MHz
NETWORKING AND SECURITY		
PROTOCOL	ZigBee PRO 2007, HA-Ready with support for binding/multicasting	
ENCRYPTION	128-bit AES	
RELIABLE PACKET DELIVERY	Retries/Acknowledgements	
IDS	PAN ID and addresses, cluster IDs and endpoints (optional)	
CHANNELS	16 channels	
POWER REQUIREMENTS		
SUPPLY VOLTAGE	2.1 to 3.6 V	
TRANSMIT CURRENT	33 mA @ 3.3 VDC / 45 mA boost mode	47 mA @ 3.3 VDC / 59 mA boost mode
RECEIVE CURRENT	28 mA @ 3.3 VDC / 31 mA boost mode	42 mA @ 3.3 VDC / 45 mA boost mode
POWER-DOWN CURRENT	<1 µA @ 25° C (77° F)	1.5 µA @ 25° C (77° F)
REGULATORY APPROVALS		
FCC, IC (NORTH AMERICA)	Yes	
ETSI (EUROPE)	Yes	
RCM (AUSTRALIA AND NEW ZEALAND)	Yes	

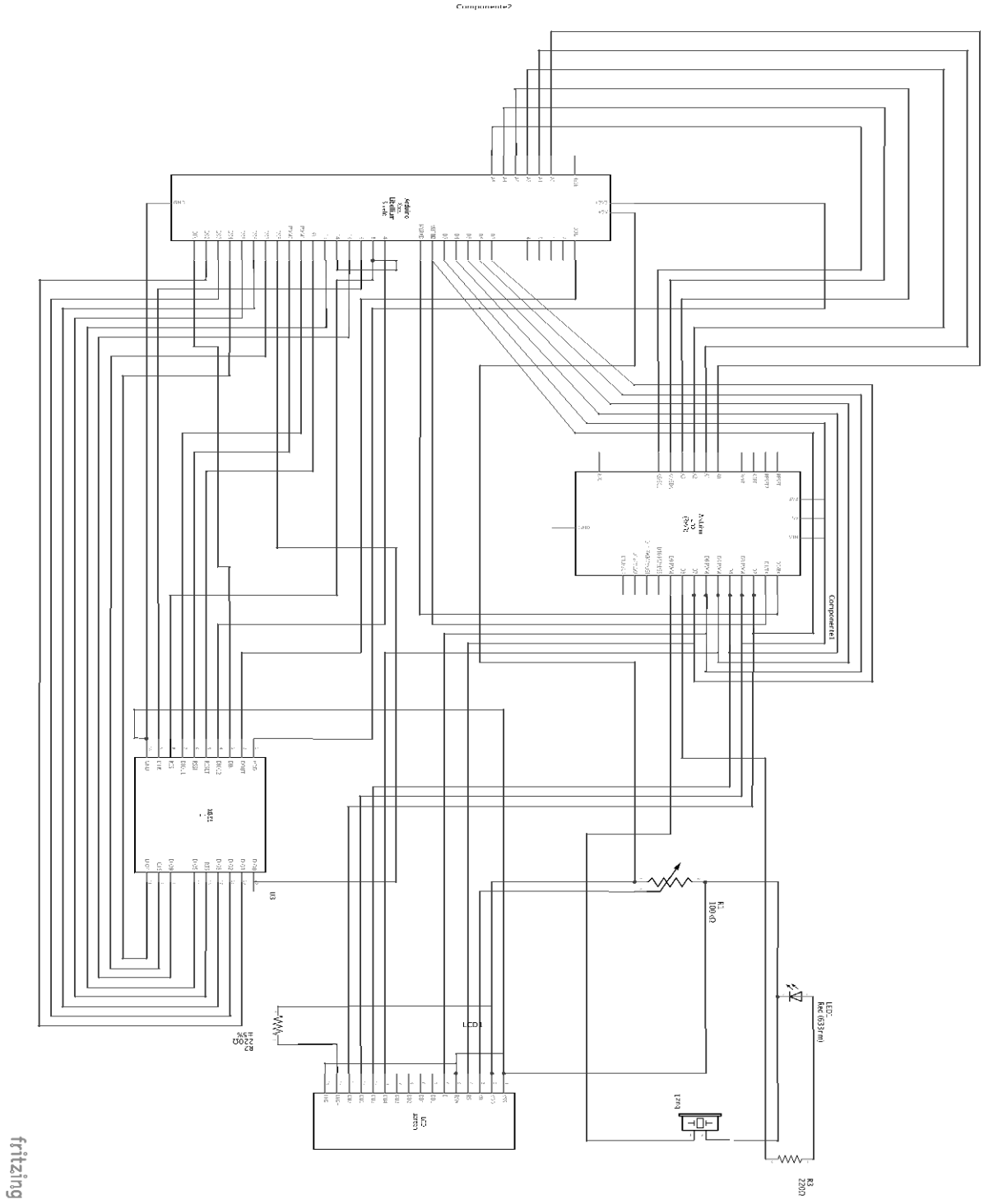
*Range figure estimates are based on free-air terrain with limited sources of interference. Actual range will vary based on transmitting power, orientation of transmitter and receiver, height of transmitting antenna, height of receiving antenna, weather conditions, interference sources in the area, and terrain between receiver and transmitter, including indoor and outdoor structures such as walls, trees, buildings, hills, and mountains.

WWW.DIGI.COM DIGI

A2. Esquemático Circuito Emisor



A3. Esquemático Circuito Receptor



PLIEGO DE CONDICIONES

PLIEGO DE CONDICIONES

A continuación, se especifican los recursos hardware y software empleados para la realización de este proyecto:

- **Recursos de hardware**

  - Ordenador portátil MacBook Pro. Este ordenador incorpora un procesador Intel® Core I7-6820HQ a. 2,7 GHz, 16 GB de memoria RAM y un almacenamiento interno de 500 GB.
  - Kit de desarrollo Arduino Starter Kit. Se han utilizado 2 unidades de este kit de desarrollo.
  - Módulo XBee® S2C. Se han empleado 3 unidades de estos módulos.

- **Recursos de software**

  - MacOS Catalina 10.15.7. Sistema operativo utilizado para el desarrollo del proyecto.
  - Arduino IDE 1.8.12. Utilizado para la programación del código de este proyecto.
  - Microsoft Office 365. La redacción de esta memoria se ha hecho con el programa Microsoft Word.
  - XCTU. Utilizado para la configuración de los módulos XBee® y pruebas mínimas.
  - Fritzing. Empleado en el diseñado de los esquemas y conexionados.

PRESUPUESTO

PRESUPUESTO

En esta sección se recogen los costes asociados a la realización de este proyecto. Los conceptos incluidos en el presupuesto se dividen en las siguientes partes:

- Trabajo tarifado por tiempo empleado.
- Recursos materiales o amortización del inmovilizado material.
- Redacción de la memoria.
- Derechos de visado del COITT.
- Gastos de tramitación y envío.

Trabajo tarifado por tiempo empleado

En este apartado se presentan los honorarios correspondientes según las horas que se han destinado a la realización de este proyecto. Según las recomendaciones del COITT se ha calculado el importe de las horas de trabajo realizadas. Mediante la expresión (I).

$$Honorarios\ (€) = H_n * 14{,}48€ + H_e * 20{,}27€ \quad (I)$$

Siendo:

- H_n: horas realizadas dentro de la jornada laboral.
- H_e: Horas realizadas fuera de la jornada normal de trabajo.

La duración total invertida para el desarrollo de este proyecto ha sido de 420 horas, de las cuales 400 horas dentro del horario normal de trabajo y las 20 horas restantes

fuera de este horario. Por lo tanto, sustituyendo estos datos en la ecuación (II) obtenemos que el coste total asciende a:

Honorarios (€) = 400 * 14,48€ + 20 * 20,27€ = 6.197,4€ (II)

Amortización del inmovilizado material

Este concepto comprende los recursos hardware y software empleado para la consecución del proyecto, que se encuentran especificadas en el pliego de condiciones del presente documento.

Este coste es estipulado para un período de tres años, utilizando un sistema de amortización lineal o constante en el que se establece que el material se desprecia de forma constante a lo largo de su vida útil. El cálculo de la cuota de amortización viene expresado por:

$$Cuota\ de\ amortización\ lineal = \frac{\text{Valor de adquisición} - \text{Valor residual}}{\text{Número de años de vida útil}}$$

Donde el valor residual indica el valor que tendrá el material cuya cuota de amortización se está calculando una vez finalizada su vida útil. La duración del proyecto ha sido de 6 meses y teniendo en cuenta que el coste de amortización se estipula en tres años, se indican los costes derivados del primer año en las dos siguientes tablas.

Los costes de amortización asociados al material software, han sido provistos por la entidad universitaria, dado que se considera que ha proporcionado todas las licencias, por lo que no ha tenido repercusión en ningún tipo de coste a nosotros.

Tabla 35. Costes de amortización del software.

Coste de las herramientas software			
Software	**Valor de adquisición**	**Valor residual (3 años)**	**Valor amortizado (1 año)**
MacOS Catalina	0 €	0 €	0 €
Microsoft Office 365	0 €	0 €	0 €
Arduino IDE	0 €	0 €	0 €
XCTU	0 €	0 €	0 €
Fritzing	8 €	0 €	2,66 €
Total			2,66 €

Tabla 36. Costes de amortización del hardware.

Coste de las herramientas hardware			
Hardware	**Valor de adquisición**	**Valor residual (3 años)**	**Valor amortizado (1 año)**
Ordenador portátil	2.200 €	0 €	733,33 €
Arduino Starter Kit	0 €	0 €	0 €
XBee® S2C	0 €	0 €	0 €
Total			733,33 €

Por lo tanto, la suma del coste del material *software* y *hardware* asciende a *setecientos treinta y cinco euros con noventa y nueve céntimos* (735,99 €).

Redacción de la memoria

El coste de la redacción viene expresado como el 5% del presupuesto de ejecución:

$$R = 0{,}05 \cdot P$$

Donde R es el honorario por la redacción de la memoria y P corresponde al presupuesto en ejecución. Este viene dado por la suma del trabajo tarifado por tiempo empleado y la amortización del inmovilizado material:

$$R = 0{,}05 \cdot P = 0{,}05 \cdot (6.197{,}4 + 735{,}99) = 346{,}66\ €$$

Por lo tanto, el coste asociado a la redacción de la memoria asciende a *trescientos cuarenta y seis euros con sesenta y seis céntimos*.

Derechos de visado del COITT

Según la normativa del COITT se establece que los gastos del visado para proyectos técnicos de carácter general se calculan en función de la siguiente expresión:

$$V = 0{,}0035 \cdot P \cdot C$$

Donde P es la suma de los costes de ejecución, redacción y material. Mientras que C corresponde a un coeficiente reductor en función del presupuesto del proyecto. En este caso, el presupuesto acumulado del proyecto se calcula en la Tabla 37 y vemos que se encuentra por debajo de los 30.050 €, por lo que el coeficiente C es igual a la unidad.

Tabla 37. Presupuestos totales con la redacción del trabajo.

Concepto	Coste
Trabajo tarifado por tiempo empleado	6.197,4 €
Amortización del material	735,99 €
Redacción del trabajo	346,66 €
Total	7.279,99 €

Teniendo estos datos en cuenta, aplicamos la fórmula y se obtiene que el valor para los derechos del visado del COITT es de:

$$V = 0{,}0035 \cdot 7.279{,}99 = 25{,}47 \text{ €}$$

Por lo tanto, el coste de los derechos del visado asciende a *veinticinco euros con cuarenta y siete céntimos.*

Gastos de tramitación y envío

Los costes de tramitación y envío se establecen en 6,00 € por cada documento visado de forma telemática.

Aplicación de impuestos

La cuantía a pagar en impuestos supondría un 7% del subtotal derivado de los conceptos desglosados anteriormente, dado que debe aplicar el Impuesto General Indirecto Canario (IGIC). Por tanto, el coste definitivo del proyecto se detalla en la siguiente tabla.

Tabla 38. Presupuesto total del proyecto.

Coste total del proyecto	
Descripción	**Coste**
Trabajo tarifado por tiempo empleado	6.197,4 €
Amortización del material	735,99 €
Redacción del trabajo	346,66 €
Derechos del visado del COITT	25,47 €
Coste de tramitación y envío	6 €
Subtotal	7311,52 €
Aplicación e impuestos (7% I.G.I.C.)	511,8 €
Total	7.823,32 €

El presupuesto total del proyecto "*Sistema multipropósito de aviso y anuncio basado en ZigBee. Prototipo formado por µC Arduino y módulos XBee*" asciende a *siete mil ochocientos veintitrés euros con treinta y dos céntimos (7.823,32€)*.

Las Palmas de Gran Canaria, a 24 de junio de 2021.

El autor del proyecto,

Juan Trujillo González.