Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Aplicación de la tecnología de radio cognitiva en redes de área corporal

TITULACIÓN: Grado en Ingeniería Telemática

AUTOR: Rubén Nuevo Boo

DIRECTOR: Olga León Abarca

FECHA: 10 de enero del 2017

Título: Aplicación de la tecnología de radio cognitiva en redes de área corporal

Autor: Rubén Nuevo Boo

Director: Olga León Abarca

Fecha: 10 de enero del 2017

Resumen

En los últimos años han aparecido nuevas tecnologías inalámbricas que permiten a los usuarios tener conexión a Internet de forma permanente y acceder a un sinfín de servicios, a través dispositivos móviles como *smartphones*, *tablets*, *wearables*…,

Una de las aplicaciones de estas tecnologías son las redes de área corporal. Estas redes se basan en el despliegue de un conjunto de nodos sensores que obtienen información de los parámetros corporales y la envían hacia otro dispositivo, como por ejemplo un *smartphone*, Las redes de área corporal resultan de gran utilidad para deportistas, que pueden monitorizar su rendimiento, pero resultan especialmente útiles en el campo de la medicina. En este último caso, los pacientes pueden controlar parámetros vitales de interés e informar al especialista/centro médico sobre ellos sin necesidad de desplazarse físicamente.

Las redes de área corporal tratan con datos sensibles del usuario y por dicho motivo es fundamental proporcionar un cierto nivel de fiabilidad y seguridad en la transmisión de dichos datos.

En este proyecto se ha implementado y evaluado una red de área corporal inalámbrica que ofrece comunicaciones robustas y seguras mediante el uso de herramientas criptográficas y la aplicación de la idea de radio cognitiva, que permite seleccionar de forma oportunista el canal más óptimo para las comunicaciones.

Título: Aplicación de la tecnología de radio cognitiva en redes de área corporal

Autor: Rubén Nuevo Boo

Director: Olga León Abarca

Fecha: 10 de enero del 2017

Overview

For the past few years, a number of new wireless technologies have emerged, allowing users to access the Internet and a myriad of services through mobile devices such as *smartphones*, *tablets*, *wearables*…,

Among others, body area networks is one of the applications of such technologies. These networks are based on the deployment of several sensor nodes over a body that get information about parameters such as glucose level or temperature and send it to another device, such as a smartphone. Body area networks can be very valuable for athletes, since they can monitor and track their performance but also in the medical field. In this case, a patient can get information about vital signs and report them to a medical expert/center remotely.

Body area networks deal with sensitive data and therefore it is of paramount importance to provide a given level of reliability and security to their data communications.

In this work we have implemented and evaluated a simple body area network that provides secure and robust communications by means of cryptographic tools and applies the idea of cognitive radio, which allows selecting in an opportunistic manner the optimal channel for data transmission.

ÍNDICE

ACRÓNIMOS

ACK: Acknowledgement

AES: Advanced Encryption Standard

BLE: Bluetooth Low Energy

CBC: Cipher-block Chaining

CCM: Counter CBC-MAC

CR: Cognitive Radio

CRN: Cognitive Radio Network

CTR: Counter

GTK: Group Temporal Key

HBC: Human Body Communications

IEEE: Institute of Electrical and Electronics Engineers

ISM: Industrial, Scientific and Medical

LOPD: Ley Orgánica de Protección de Datos

MAC: Media Access Control o Message Authentication Code

MICS: Medical Implant Communication Service

MK: Master Key

NB: Narrowband

PHY: Physical Layer

PTK: Pairwise Temporal Key

RTT: Round-trip Time

UWB: Ultra Wideband

WBAN: Wireless Body Area Network

WMTS: Wireless Medical Telemetry Services

INTRODUCCIÓN

Durante los últimos años, las redes de área corporal o *Body Area Networks* (BANs) han cobrado una importancia notable, puesto que permiten la monitorización continua de diferentes parámetros vitales de un paciente tales como pulso, temperatura, etc. y el envío de dichos datos a dispositivos o centros médicos donde se puede procesar esta información. La ventaja fundamental que presenta el uso de las BANs frente a los métodos tradicionales es que ofrece mayor rapidez tanto en el diagnóstico como en el tratamiento de las diferentes patologías y puede prevenir, de este modo, situaciones fatales.

El despliegue con éxito de las BANs depende en gran medida de la fiabilidad en la transmisión de los datos del paciente, así como en la protección de dichos datos. Dado que gran parte de los dispositivos que componen estas redes usan comunicaciones inalámbricas (*Wireless* BANs o WBANs) y disponen de recursos limitados, es fundamental garantizar su buen funcionamiento y la disponibilidad de las comunicaciones, es decir, asegurar que los datos del paciente puedan ser transmitidos al centro médico cuando sea necesario, sin pérdidas de información o retrasos considerables. Esto implica que la red debe ser robusta a interferencias y a degradaciones del canal. Por otro lado, los datos transmitidos en las comunicaciones contienen información sensible que debe protegerse frente a accesos no autorizados.

La tecnología de radio cognitiva representa una solución prometedora para garantizar la disponibilidad de las comunicaciones, permitiendo que los dispositivos de la red cambien de canal cuando el canal en uso no ofrece la calidad requerida, en términos de velocidad de transmisión, retardo u otros parámetros.

En cuanto a protección de las comunicaciones y de los datos transmitidos, la criptografía es una herramienta que puede ofrecer privacidad, es decir, impedir que usuarios no autorizados accedan a dicha información, e integridad, garantizar que los datos transmitidos sean modificados. Además, permite implementar mecanismos de autenticación que aseguren que el origen de los datos es legítimo.

El objetivo de este proyecto es implementar y evaluar las prestaciones de una WBAN sencilla que haga uso de la tecnología de radio cognitiva e implemente diferentes mecanismos de seguridad que permitan su buen funcionamiento.

El documento se estructura de la siguiente forma. En el Capítulo 1 se describen las diferentes tecnologías que pueden aplicarse para el despliegue de una red de área corporal. El Capítulo 2 ofrece una visión general de la tecnología de radio cognitiva y de cómo su aplicación en redes de área corporal puede ser beneficiosa. En el Capítulo 3 se describe la implementación de una red de área corporal realizada en este proyecto: dispositivos utilizados, configuración y entorno de programación, y el Capítulo 4 muestra los resultados obtenidos

durante la evaluación de las prestaciones de la red. Finalmente, el Capítulo 5 contiene las conclusiones de este trabajo.

CAPÍTULO 1. Las redes de área corporal

1.1. ¿Qué son?

Las redes de área corporal, como su propio nombre indica, son redes que cubren un área limitada como es el cuerpo de una persona. Estas redes se basan en el despliegue de nodos a lo largo del cuerpo de un usuario con los cuales poder realizar diferentes funciones dependiendo de la finalidad con la que se haya desplegado la red. Las funciones que suelen realizar estos nodos es recoger información del cuerpo del usuario para posteriormente transmitirla a un nodo más complejo conocido como coordinador.

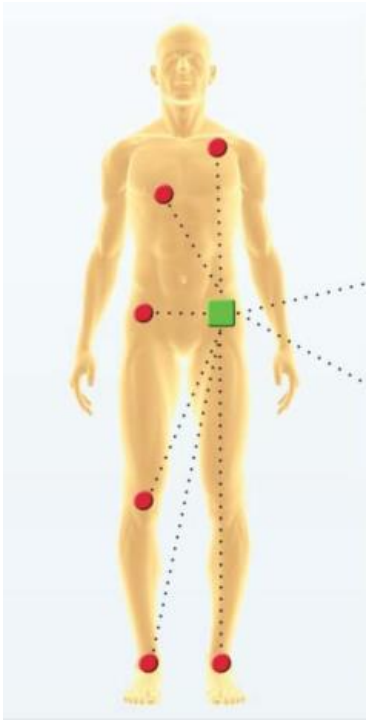

Fig. 1.1 Despliegue de una WBAN [14]

Las tecnologías de hoy en día nos permiten poder crear estas redes de forma inalámbrica, conocidas como *Wireless Body Area Networks* (WBANs), lo cual nos da la posibilidad de crear multitud de aplicaciones para este tipo de redes y poder aprovechar las nuevas tecnologías para ofrecer un servicio sin tener que depender de una sala o habitación donde ejecutar las aplicaciones.

Este tipo de redes se caracteriza por utilizar dispositivos de recursos limitados, es decir, dispositivos con una capacidad de procesamiento y de memoria reducida. Esto hace que para un óptimo funcionamiento de estos dispositivos se deban utilizar protocolos que sean capaces de extraer el máximo rendimiento de ellos.
También es importante tener en cuenta la eficiencia energética de estos dispositivos, ya que no cuentan con una gran capacidad de almacenamiento de

energía, y dependiendo de la aplicación a la que vaya destinada la WBAN puede ser un gran inconveniente el hecho de tener que cargar o cambiar las fuentes de alimentación de los diferentes nodos que componen la red.
Es por ello que los protocolos que nos encontramos en este tipo de redes son protocolos de bajo consumo como por ejemplo *Bluetooth Low Energy* (BLE), *Institute of Electrical and Electronics Engineers* (IEEE) 802.15.4 o IEEE 802.15.6.

1.2. ¿Qué aportan al mundo de la medicina?

Las nuevas tecnologías de que disponemos hoy en día, como es el caso de los *smartphones* u otros dispositivos móviles que solemos llevar siempre encima, han hecho posible que la aparición de nuevas aplicaciones para estas redes de área corporal esté en crecimiento, ya que junto estos dispositivos también han aparecido los llamados *wearables* los cuales son instrumentos como pulseras, relojes inteligentes, chips para el calzado, bandas para la cabeza, etc… que desplegamos a lo largo de nuestro cuerpo con el objetivo de recopilar información de él. Este tipo de aplicaciones, en su mayoría, están orientadas al ámbito deportivo, pero quizás el campo donde las redes inalámbricas de área corporal pueden tener una mayor utilidad e impacto sea el de la medicina.

La medicina está evolucionando hacia lo que se conoce como *eHealth*, es decir, salud electrónica. Las WBAN podrían aplicarse a los pacientes de tal forma que se pudiesen monitorizar parámetros de su cuerpo en tiempo real como por ejemplo:

- La temperatura corporal
- La presión de la sangre
- El nivel de glucosa presente en la sangre
- El número de pulsaciones por minuto

Una de las ventajas que ofrecerían estas redes y aplicaciones, tanto para pacientes como para hospitales, es el hecho de que no sería necesario el estar presente en el centro hospitalario, liberando así espacio en los hospitales y ambulatorios. Además, sin tener que hacer acto de presencia en estos centros, el paciente podría ser atendido por profesionales a partir del envío de la información corporal recolectada por los nodos sensores desplegados en el cuerpo y recibir *feedback* por parte del personal médico y, en caso de que se detectasen irregularidades en los parámetros corporales enviados, poder actuar de forma rápida y efectiva.

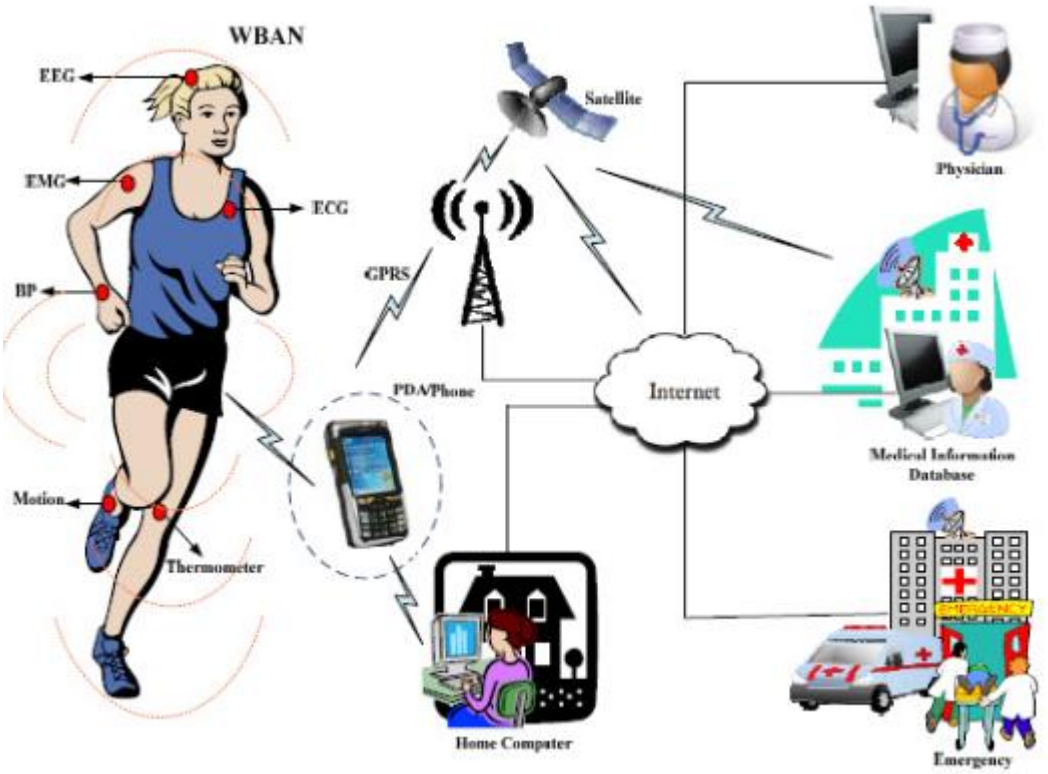

Fig. 1.2 Ejemplo de una aplicación WBAN orientada a la medicina [5]

1.3. ¿Cómo funcionan?

Siguiendo con el mundo de la medicina, que es la principal aplicación de estas redes, el funcionamiento se basa en recoger información del cuerpo de sus usuarios y enviarla a los diferentes centros médicos para ser analizada por el personal médico.

Para llevar a cabo esta tarea se despliegan diferentes nodos sensores a lo largo del cuerpo del paciente. Cada uno de estos nodos se encarga de recoger una información específica del cuerpo del usuario, es decir, un paciente puede tener en su cuerpo un nodo sensor que se encargue de medir su temperatura corporal, otro sensor que mida su ritmo cardiaco, otro nodo que se dedique a recopilar información sobre el nivel de azúcar en la sangre, etc… Estos nodos, dependiendo de la aplicación y lo que se pretenda monitorizar pueden ser externos, como por ejemplo una pulsera capaz de medir la temperatura y las pulsaciones del paciente, o internos, como por ejemplo un chip sensor que necesite estar en contacto con la sangre del usuario para medir determinados aspectos del interior del cuerpo.

La información recogida por estos sensores se envía a un nodo coordinador, un nodo dotado de mayor inteligencia y capacidad de procesamiento que los sensores, y capaz de tomar de tomar decisiones.

Una vez llegan los datos al coordinador, éste los envía hacia el centro médico a través de Internet, tal y como se puede apreciar en la Figura 1.2.

Como ya se ha comentado, la popularización de los *smartphones* entre la sociedad, que no dejan de ser pequeños ordenadores que llevamos en el bolsillo, ha hecho que el despliegue de redes de área corporal sea una realidad. Estos dispositivos constituyen un buen ejemplo de nodo coordinador: tienen una capacidad de procesamiento suficiente para coordinar y gestionar los demás nodos de la red y además cuentan con acceso directo a internet ya sea por la interfaz radio como por Wi-Fi. De hecho, actualmente, se pueden encontrar decenas de aplicaciones en estos dispositivos que sirven para medir parámetros de nuestro cuerpo a través de otros periféricos, como pulseras, aunque en estos casos no son aplicaciones médicas sino más bien deportivas, para que los usuarios puedan obtener información de su cuerpo mientras realizan deporte y analizar los resultados de su entrenamiento.

Tal y como se muestra en la Figura 1.2, la arquitectura típica en BANs consiste en tres niveles de comunicación: comunicaciones entre los nodos sensores o Intra-BAN, comunicaciones entre el nodo sensor principal y un dispositivo personal como puede ser un teléfono móvil o un ordenador, denominadas comunicaciones Inter-BAN. Estos dos niveles de comunicación constituyen las denominadas WBANs. En un tercer nivel, se encuentran las comunicaciones entre el dispositivo personal e Internet, que posibilitan que la información recogida por los nodos sensores llegue a centros médicos y se tomen las medidas pertinentes. En este proyecto, nos hemos centrado en las comunicaciones WBAN.

Los requisitos fundamentales en una WBAN son los siguientes:

- Fiabilidad: los nodos sensores transmiten información relativa a la salud del usuario y por dicho motivo es imprescindible garantizar que las comunicaciones sean fiables.
- Latencia: las aplicaciones médicas que gestionan emergencias requieren transmisiones en tiempo real.
- Seguridad: las WBAN manipulan datos personales y, en muchos casos confidenciales. La seguridad y privacidad de los datos es por tanto esencial.
- Consumo de energía. Dadas las aplicaciones de las WBAN, es importante maximizar el tiempo de vida de las WBAN y garantizar la disponibilidad. Hay que destacar, sin embargo, que el recambio de baterías o sensores es mucho más simple que en otros escenarios donde los nodos están desatendidos, como por ejemplo, una red de sensores que monitorizan el riego de una cosecha.

1.4. Tecnologías WBAN

A continuación, se describen las tecnologías que pueden usarse para las comunicaciones WBAN.

1.4.1. Bluetooth Low Energy (BLE)

Bluetooth [19] es una tecnología *wireless* que se puede encontrar implementada en la mayoría de dispositivos electrónicos de hoy en día. La versión 4.0 de esta tecnología dispone de una característica denominada *Bluetooth Low Energy* o *Bluetooth Smart* que se caracteriza por proporcionar un bajo consumo de energía en sus operaciones.
El uso de esta tecnología está enfocado hacia dispositivos que no necesitan transmitir grandes cantidades de datos y que necesitan aprovechar al máximo la eficiencia de sus recursos debido a la limitada autonomía de la que disponen.
BLE opera sobre la banda de frecuencia de 2400 MHz y ofrece un *bit rate* de hasta 1 Mbps.

1.4.2. IEEE 802.15.4 y Zigbee

El estándar IEEE 802.15.4, al igual que en el caso anterior, está pensado para ser utilizado en redes de rango limitado como es el caso de una red de área personal. Este protocolo proporciona operaciones de bajo consumo y por lo tanto esta también orientado a ser utilizado en dispositivos de recursos limitados.
Las principales características de este estándar son:

- Velocidades de transmisión que van desde los 20 Kbps hasta los 250 Kbps.
- Diferentes bandas de frecuencia de operación:
  - 868 MHz en Europa
  - 915 MHz en Norte América
  - 2.4 GHz en todo el mundo
- Topología en estrella o *peer-to-peer*

Dentro de este estándar existe una variante denominada Zigbee que se basa en el nivel físico y de control de acceso al medio definido por IEEE 802.15.4, y añade funcionalidades de red, tal y como se muestra en la Figura 1.3. Básicamente, Zigbee permite crear redes *mesh* usando como tecnología subyacente el estándar 802.15.4. *Mesh networking* se usa en aplicaciones donde la distancia entre los dispositivos involucrados en la comunicación es superior al rango de cobertura de los mismos, de modo que se usan dispositivos radio intermedios que encaminan los mensajes y posibilitan dicha comunicación.

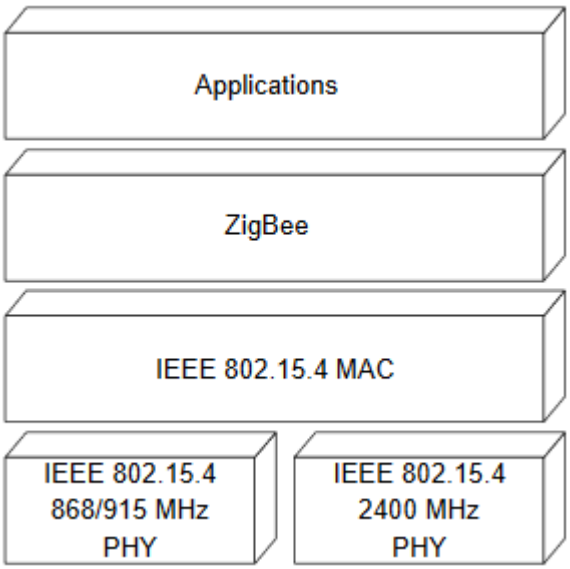

Fig. 1.3 Arquitectura del estándar 802.15.4 [20]

1.4.3. IEEE 802.15.6

El IEEE decidió establecer el *Task Group 6* del estándar IEEE 802, para estandarizar las redes de área corporal inalámbricas y poder crear un protocolo que maximizara el rendimiento de estas redes, lo cual ya se ha comentado que es un punto importante debido a la limitación de los dispositivos que la componen.

Uno de los propósitos de este estándar era el de crear una nueva capa física y una nueva capa de control de acceso al medio. Respecto a la capa física, la elección de una banda de frecuencias para este protocolo fue uno de los puntos más importantes ya que las aplicaciones médicas tratan con datos vitales del usuario y deben operar en una banda de frecuencia sólida y que permita un correcto funcionamiento del servicio.

Algunas de las bandas que se tuvieron en cuenta fueron las siguientes:

- Medical Implant Communication Service (MICS)
- Wireless Medical Telemetry Services (WMTS)
- Industrial, Scientific and Medical band (ISM)

La Figura 1.4 muestra el rango de frecuencias de cada una de las bandas. Las dos primeras bandas de frecuencia no permiten aplicaciones con una transmisión de datos de alta velocidad. Y la tercera es una banda donde ya operan una gran cantidad de dispositivos y servicios inalámbricos con lo cual hay un gran riesgo de que se produzcan interferencias.

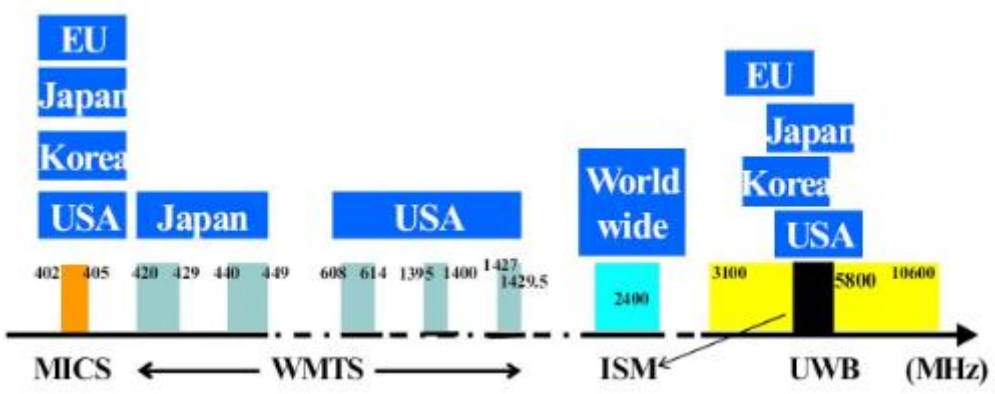

Fig. 1.4 Bandas de frecuencia para las WBAN [3]

Finalmente, el estándar IEEE 802.15.6 define una capa de control de acceso al medio que cuenta con las siguientes capas físicas:

- *Human Body Communications* (HBC)
  - Opera en el rango de frecuencia 5MHz-50MHz
- *Narrowband* (NB)
  - Opera en los rangos de frecuencia 402MHz-2400MHz, con lo cual incluye las bandas MICS, WMTS e ISM anteriormente comentadas.
- *Ultra wideband* (UWB)
  - Opera en los rangos de frecuencia 3100MHz-10600MHz.

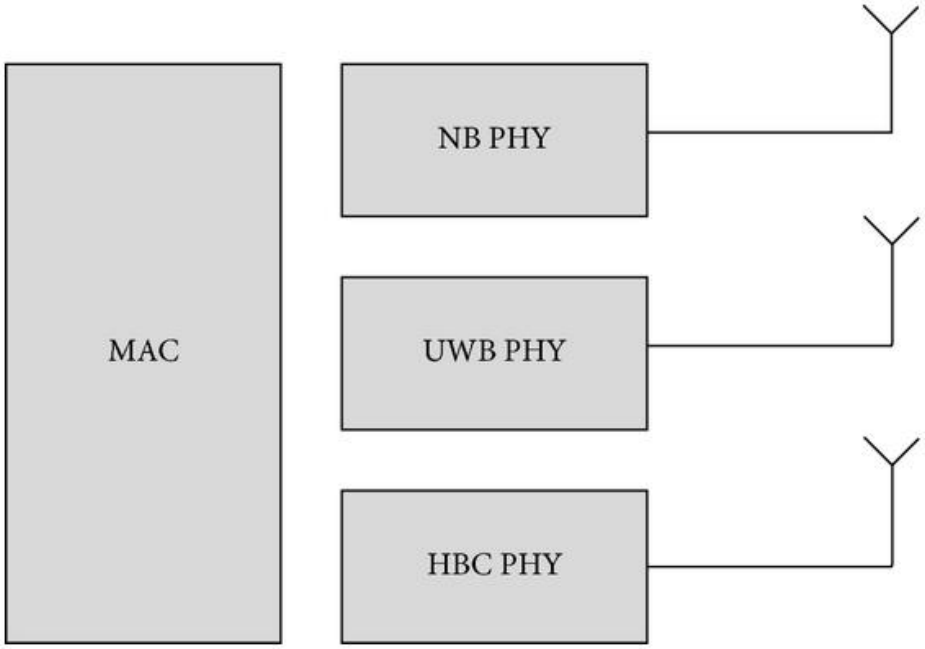

Fig. 1.5 Capas MAC y PHY de IEEE 802.15.6 [11]

Según este estándar, los nodos que componen las WBAN deben tener una topología en estrella y disponer de un coordinador que se encargue de controlar la red. De esta forma, los nodos desplegados en el área corporal que se dedican a recolectar información envían los datos a un nodo central el cual se encarga de gestionar dicha información. Los nodos de la red pueden estar directamente conectados al coordinador o llegar a él a través de algún salto en caso de que sea necesario por temas de rendimiento (redes *mesh*).

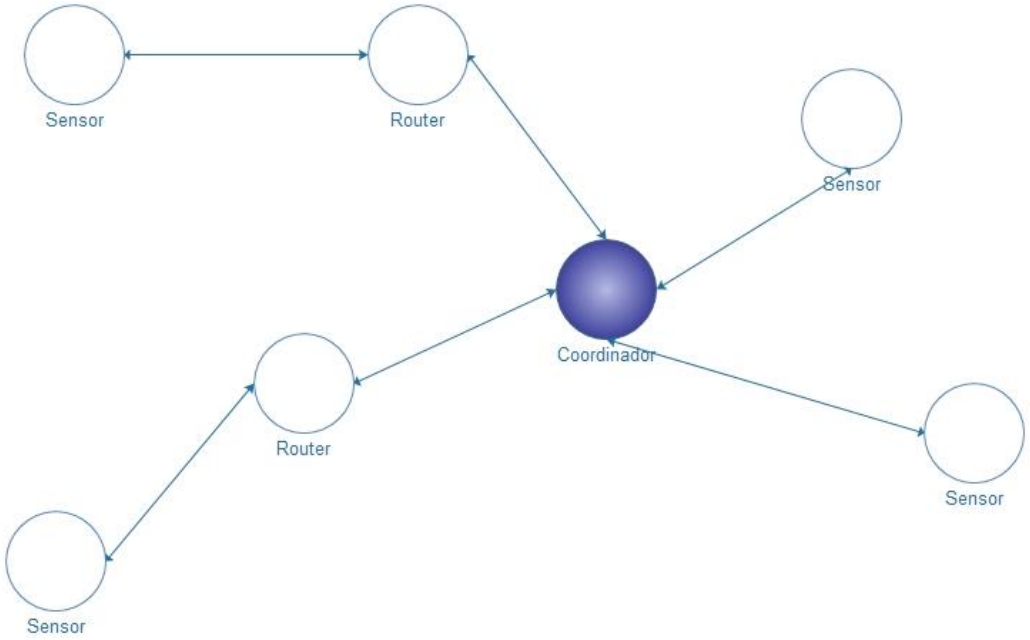

Fig. 1.6 Arquitectura WBAN según IEEE 802.15.6

1.4.4. Comparativa de las diferentes tecnologías

La Tabla 1.1 muestra un resumen de las características principales de las diferentes tecnologías WBAN.

Tabla 1.1 Comparación de los diferentes protocolos para WBAN

	Bluetooth Low Energy	IEEE 802.15.4/Zigbee	IEEE 802.15.6
Frecuencias de operación	2400 MHz	868/915/2400 MHz	400/800/900/2400 MHz
Velocidad de transmisión	Hasta 1 Mbps	Hasta 250 Kbps	Hasta 15,6 Mbps
Cobertura	60 m(BLE 4.0) – 240 m (BLE 5)	10-20 m	< 10 m
Topologías	Scatternet (master-slave)	Estrella, P2P y Mesh	Estrella

1.5. Seguridad en WBANs

Los datos recogidos por los sensores y enviados al nodo coordinador, y posteriormente al centro médico juegan un papel clave en el diagnóstico y tratamiento del paciente. Por este motivo, es fundamental proteger dichos datos frente a posibles amenazas. En WBANs, las principales amenazas son:

- Escuchas no autorizadas de las comunicaciones. Las comunicaciones inalámbricas de las WBANs usan un medio abierto, el aire, fácilmente accesible por un atacante que puede capturar los mensajes que se intercambian los distintos dispositivos.
- Acceso no autorizado a los datos almacenados en los dispositivos de la WBAN. Los datos relativos a un paciente deberían ser confidenciales.
- Modificación o inyección de datos en las comunicaciones de la WBAN o en el sistema de almacenamiento de los dispositivos. Un usuario malintencionado podría intentar falsificar los datos de un paciente. Estos ataques pueden dar lugar a falsas alarmas o incluso peor, impedir el correcto diagnóstico del paciente, con consecuencias letales.
- Denegación de servicio. Un atacante puede generar señales interferentes con el objetivo de interrumpir las comunicaciones de la WBAN. Las interferencias pueden ser generadas también, de forma no intencionada, por otros dispositivos que operen en el mismo rango de frecuencias que los dispositivos de la WBAN.

Por lo tanto, es indispensable que la WBAN proporcione los siguientes servicios de seguridad:

- Confidencialidad: los datos almacenados en los distintos dispositivos, así como los transmitidos en las diferentes comunicaciones deben estar protegidos frente a accesos no autorizados. Este servicio puede proporcionarse mediante el uso de herramientas criptográficas (cifrado de datos).
- Autenticación: debe verificarse la identidad de las fuentes que envían datos sobre el paciente y rechazar los datos que provengan de entidades no autorizadas. La criptografía es una herramienta que se usa habitualmente para proporcionar autenticación.
- Integridad: la red debe garantizar que los mensajes se reciben íntegros, es decir, que no han sido modificados a lo largo de la transmisión. Los mecanismos de autenticación proporcionan también integridad de los datos.
- Disponibilidad: es esencial garantizar el buen funcionamiento de las comunicaciones en la WBAN y que se pueda acceder a los datos en cualquier momento.

Existen numerosos mecanismos que ofrecen los servicios de seguridad mencionados anteriormente, pero su aplicación en WBANs constituye un reto debido a las limitaciones de muchos de los dispositivos de estas redes en cuanto a capacidad de procesado, memoria, consumo de energía, etc. Algunas soluciones criptográficas, como la criptografía de clave pública, tienen un coste prohibitivo para dispositivos limitados.

La mayoría de las tecnologías empleadas en redes de sensores como las WBANs incorporan el algoritmo de cifrado AES (*Advanced Encryption Standard*), puesto que existen implementaciones *hardware* del mismo optimizadas para dispositivos limitados. Este esquema criptográfico, nos permite cifrar y autenticar las comunicaciones.

1.5.1. Advanced Encryption Standard

AES [16] es un protocolo de cifrado por bloques que trabaja con bloques de 128 bits y soporta 3 tamaños de clave diferentes: 128 bits, 192 bits y 256 bits. Dependiendo del tamaño de la clave, el algoritmo realizará más o menos rondas para obtener el mensaje encriptado, concretamente 10 rondas si se utiliza una clave de 128 bits, 12 rondas para una clave de 192 bits y finalmente 14 rondas con claves de 256 bits, agregando a todas ellas una ronda adicional que será la inicial.

Dado que el tamaño del bloque de entrada al cifrador es de 128 bits, es necesario fragmentar un mensaje en bloques si el tamaño total es superior a 128 bits. En el caso de que el tamaño del mensaje no sea un múltiplo de 128 bits, el último bloque se rellena con bits de relleno o *padding* hasta completar el bloque. El mensaje cifrado o criptograma se obtiene concatenando el cifrado de todos los bloques (cada uno de 128 bits). El receptor del mensaje debe descifrar el criptograma usando el mismo esquema de cifrado y la misma clave

que el emisor. Como consecuencia, emisor y receptor deben compartir una clave secreta.

Existen diversos modos de funcionamiento que pueden usarse para el cifrado AES. Básicamente, el modo de funcionamiento define cómo se aplica el algoritmo a un mensaje determinado, y la relación que existe entre el cifrado de un bloque y el anterior.

Las tecnologías presentadas en la sección anterior usan los siguientes modos de funcionamiento:

- Modo CTR (*counter*) para el cifrado (confidencialidad).
- Modo CBC-MAC (*Cipher Block Chaining-Message Authentication Code*) para la autenticación.
- Modo CCM (*Counter* CBC-MAC) para proporcionar confidencialidad y autenticación (usa el modo CTR para el cifrado y el modo CBC-MAC para la autenticación).

La autenticación consiste en generar un código (*message authentication code* o MAC) de 32, 64 o 128 bits a partir del mensaje usando el algoritmo AES modo CBC-MAC, que se envía junto con el mensaje. El receptor debe verificar dicho código al recibir el mensaje, realizando los mismos cálculos que el emisor. Si el valor calculado por el receptor coincide con el código recibido, el mensaje queda autenticado. Tal y como ocurre en las operaciones de cifrado, para generar códigos de autenticación y verificarlos, emisor y receptor deben compartir una clave secreta.

Cabe mencionar que, en el caso en que se cifren y autentiquen mensajes, las claves usadas deben ser distintas en cada caso para evitar posibles ataques.

CAPÍTULO 2. Radio cognitiva

2.1 ¿Qué es?

Como ya se ha comentado, debido a las nuevas tecnologías móviles cada vez aparecen más servicios y aplicaciones para la sociedad. A pesar de que este hecho es positivo para los usuarios, que disponen de un rango más amplio de servicios, supone un problema a la hora de gestionarlos, puesto que el espectro de frecuencias es un recurso limitado. La mayor parte de las bandas de frecuencia están asignadas a servicios específicos que operan bajo licencia, y las bandas libres están saturadas como consecuencia de la explosión de nuevas aplicaciones inalámbricas.

Sin embargo, diversos estudios muestran que gran parte de las bandas licenciadas, como las asignadas a difusión de TV, están infrautilizadas la mayor parte del tiempo.

Una de las propuestas que ha cobrado más importancia en los últimos años para intentar hacer un uso más eficiente del espectro es la tecnología de radio cognitiva. La idea fundamental de la radio cognitiva es utilizar estos los huecos en el espectro cuando sus usuarios legítimos no los están utilizando, tal y como se muestra en la Figura 2.1. Las redes de radio cognitiva (*Cognitive Radio Networks*) actúan pues como usuarios secundarios de dicho espectro.

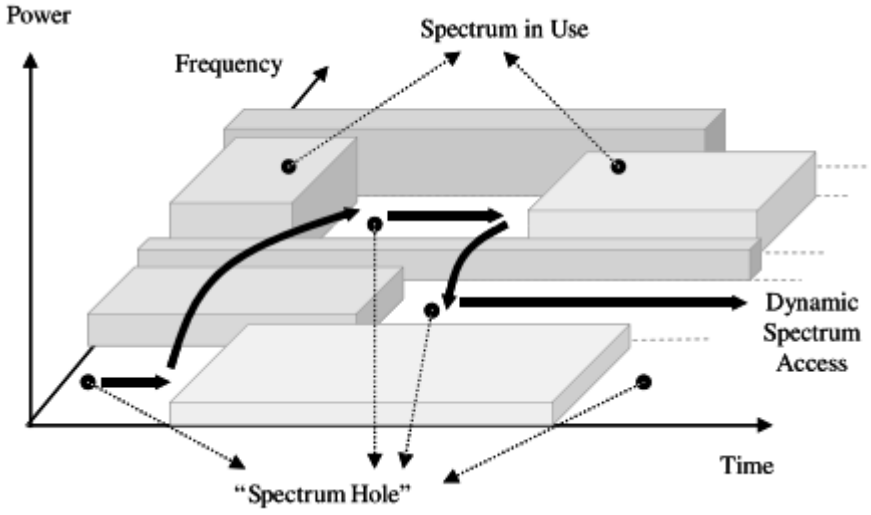

Fig. 2.1 Concepto de "hueco" en el espectro radiofrecuencial [1]

Un requisito para poder operar en frecuencias licenciadas de forma oportunista es que las redes cognitivas no interfieran las comunicaciones de los usuarios primarios del espectro. Por dicho motivo, deben implementar mecanismos de *spectrum sensing*, que consisten en observar el medio y detectar estos "huecos" en el espectro. Generalmente dichos mecanismos se basan tomar en

medidas de energía y compararlas con un umbral, de modo que si la medida para una determinada frecuencia excede el umbral se considera que el canal está ocupado.
Las radios cognitivas son dispositivos inteligentes, que implementan mecanismos de *spectrum sensing* y pueden tomar una decisión con respecto a la existencia de usuarios primarios. Tal y como se muestra en la Figura 2.2, uno de los componentes de las radios cognitivas es el módulo de aprendizaje, que permite a dichos dispositivos aprender del pasado. De este modo, las decisiones sobre el canal a usar para las comunicaciones se pueden tomar teniendo en cuenta las medidas realizadas en un momento determinado, pero también medidas anteriores.
Finalmente, estos dispositivos tienen la capacidad de reconfigurarse para poder operar en el canal seleccionado.

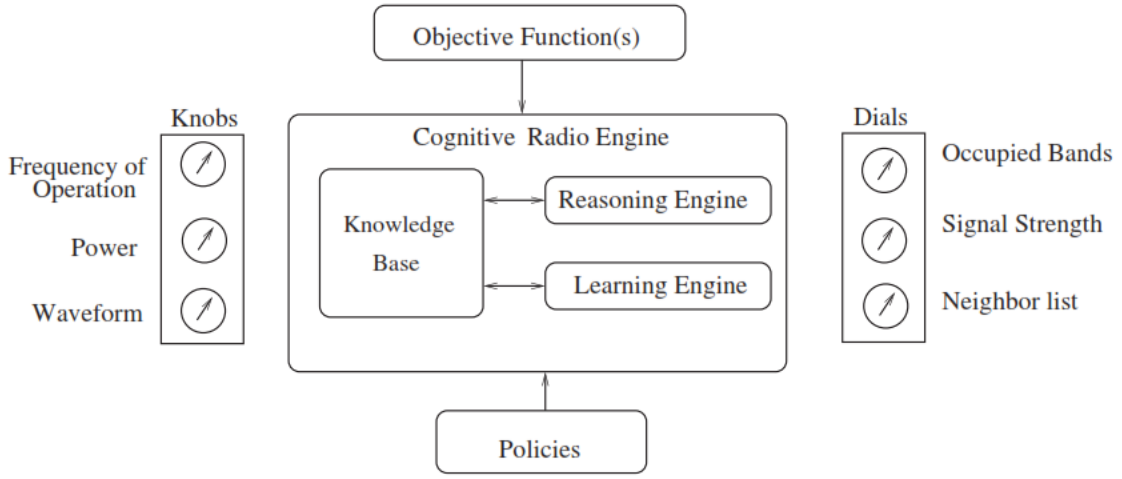

Fig. 2.2 Arquitectura genérica de una radio cognitiva [2]

En resumen, las cuatro funcionalidades básicas de una red de radio cognitiva son:

- Detección del espectro
  - Esta funcionalidad concede a la red la capacidad de detectar bandas de frecuencia que no estén siendo utilizadas por otros usuarios.
- Gestión del espectro
  - Es la capacidad de seleccionar la mejor banda de frecuencia posible que permita ofrecer de la forma más óptima el servicio a los usuarios.
- Movilidad del espectro
  - Esta funcionalidad permite a la red poder cambiar la frecuencia en la que opera a la vez que mantiene la continuidad del servicio que ofrece.
- Distribución del espectro
  - Otorga a la red la capacidad de coexistir en una banda de frecuencia junto con otros servicios de forma que puedan

compartir el espectro entre ellos de una forma óptima para que todos los servicios puedan cumplir con sus requisitos de comunicación sin interferir de forma perjudicial entre ellos.

La Fig. 2.3 muestra las diferentes funcionalidades de una CRN y la capa del modelo OSI que implementa cada una de ellas.

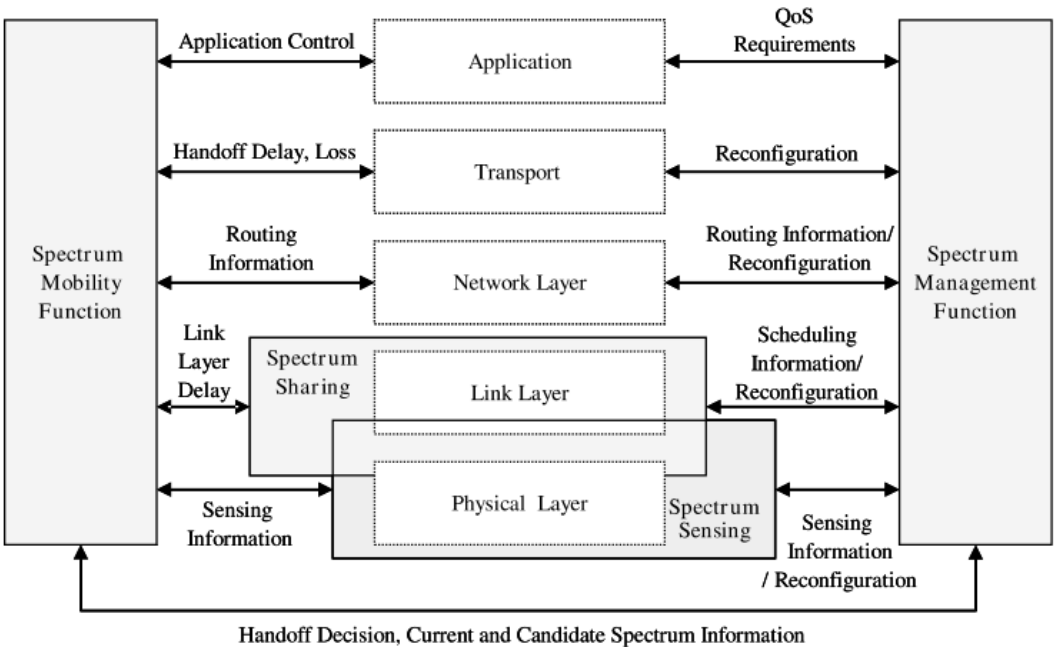

Fig. 2.3 Funcionalidades de la radio cognitiva en la red [1]

2.2 Aplicación en las redes de área corporal

Esta tecnología de la radio cognitiva puede ser muy interesante aplicada en redes de área corporal, ya que son redes cuyo uso suele ir ligado a aplicaciones médicas que monitorizan y transmiten información de nuestro cuerpo para el uso médico.

Siempre que se trabaja con datos médicos se tiene que tener precaución, ya que según la ley orgánica de protección de datos (LOPD) estos datos se consideran especialmente protegidos, es decir, el nivel de protección más alto que especifica esta ley. Esto es importante, ya que además de proteger estos datos mediante técnicas de cifrado y/o autenticación, también ayuda a proteger estos datos el hecho de que los servicios que trabajan con ellos operen en un entorno con unas condiciones lo más optimas posibles para evitar interferencias o aspectos que puedan dificultar la transmisión de la información del usuario.

Imaginemos un equipo médico de un hospital que tenga la función de monitorizar las constantes vitales de un paciente como el que se muestra en la siguiente imagen.

Fig. 2.4 Equipo médico de monitorización

¿Qué pasaría si este equipo operase en un entorno con interferencias que dificultase su tarea? Pues podría desembocar en perdida de información del paciente o en diagnósticos erróneos que complicasen la labor del personal médico que se encargue de analizar los datos obtenidos.

Algunas aplicaciones médicas implementadas a través de WBANs pueden tener la misma función que el equipo médico mostrado en la Fig. 2.4, con el hándicap de que estas redes se mueven con el usuario y los datos se transmiten *over the air,* con lo cual son más vulnerables a las condiciones del entorno.
El hecho de que sean redes móviles hace que no se pueda preparar una infraestructura en un lugar fijo (como es el caso de un hospital) para optimizar la comunicación de la red, sino que las condiciones del medio irán variando dependiendo de por donde se mueva el usuario de la WBAN, esto dificulta una configuración óptima de la red.

Es aquí donde entra la idea de la tecnología de radio cognitiva en las WBAN. El hecho de que la red pueda observar el medio y analizar sus condiciones hace que pueda reconfigurarse de forma dinámica dependiendo de los parámetros del entorno que nos interesen para conseguir una comunicación lo más fiable y robusta posible.
De esta manera las WBAN pueden escanear el espectro de frecuencias en búsqueda de la frecuencia (canal) más óptima para operar, es decir, el canal que presente menos actividad y por lo tanto que esté siendo utilizado por menos usuarios. Esto implica que al haber menos tráfico en esa frecuencia hay

menos posibilidades de interferencias con otros usuarios a la hora de utilizar la banda.

CAPÍTULO 3. Implementación

En este proyecto se pretende emular una red de área corporal muy simple en la que un paciente tiene colocado un sensor de temperatura. Dicho sensor tomará medidas de la temperatura cada 10 segundos (medida fijada en el código implementado en el nodo A) que serán transmitidas a un coordinador el cual transmitirá los datos a un PC con conexión a internet, tal y como se muestra en la Figura 3.1.

Con este objetivo se ha realizado el siguiente montaje, compuesto por dos nodos:

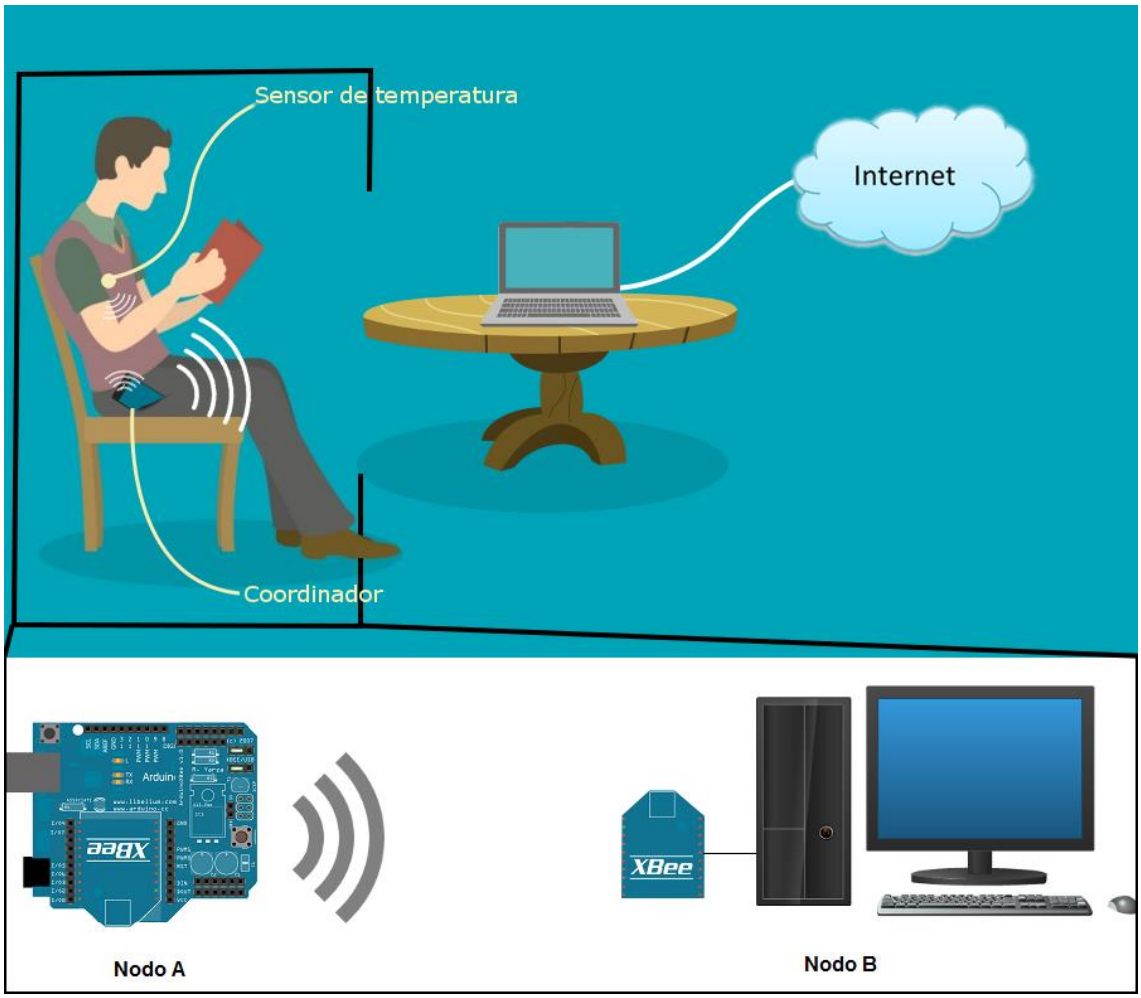

Fig. 3.1 Escenario del proyecto

- Nodo A: Actuará como nodo sensor, recogiendo datos y transmitiéndolos hacia el nodo coordinador (nodo B).

- Nodo B: Este nodo será el coordinador de la red, y se encargará de recibir los datos enviados por el Nodo A y monitorizar la calidad de las

transmisiones a partir de algunos parámetros con los cuales tomará decisiones respecto al canal de la red.

3.1 Elección de la tecnología

A pesar de que el estándar IEEE 802.15.6 está especialmente pensado para redes de área corporal, en el momento de desarrollar este proyecto no existen muchos dispositivos que implementen el protocolo, y los existentes son demasiado costosos. En este proyecto se ha optado por implementar una WBAN basada en el estándar 802.15.4, puesto que se trata del protocolo que más se asemeja al 802.15.6. A pesar de que BLE ofrece mayores velocidades de transmisión y rango de cobertura, las prestaciones de 802.15.4 son más que suficientes para implementar un servicio como el que se propone en el proyecto. También se ha descartado la tecnología Zigbee, dado que se pretende desplegar una red con topología de estrella en un espacio reducido y, por lo tanto, no es necesario implementar *mesh networking*.

3.2 Dispositivos

En esta sección se describirán los dispositivos escogidos para realizar la implementación de la WBAN.

3.2.1 Módulos Xbee de Digi

Pese a que el estándar para WBANs es el 802.15.6, se ha optado por usar módulos XBee [17] que implementan el protocolo 802.15.4, puesto que son más accesibles desde el punto económico. Las diferencias fundamentales entre ambos estándares son: 1) la velocidad de transmisión (250 Kbps para el 802.15.4 y 15.6 Mbps para 802.15.6), y 2) el rango de transmisión (75 m para 802.15.4 y 3-5 m para 802.15.6). En aquellas aplicaciones en las que no se requiere una velocidad de transmisión de datos muy alta, los dispositivos que implementan el estándar 802.15.4 son suficientes para el despliegue de una WBAN.

Los módulos XBee se presentan en dos modalidades: series 1 y series 2, tal y como puede verse en la Tabla 3.1. Para cada una de ellas existe la versión normal y la versión Pro, que principalmente aumenta el rango de cobertura a expensas de un mayor consumo de potencia.

Tabla 3.1 Versiones de los dispositivos XBee

Dispositivo	Rango	Consumo	Frecuencia	Protocolo	Potencia transmisión	Velocidad tx. datos
Serie 1	90m	50mA	2.4GHz	802.15.4	1mW	250Kbps

		@3.3v				
Serie 2	120m	40mA @3.3v	2.4GHz	Zigbee	2mW	250Kbps
Serie 1 Pro	1.6Km	215mA @3.3v	2.4GHz	802.15.4	60mW	250Kbps
Serie 2 Pro	1.6Km	295mA @3.3v	2.4GHz	Zigbee	63mW	250Kbps

En este proyecto se ha optado por usar los XBee serie 1 (ver Figura 3.2), puesto que son los que implementan el protocolo 802.15.4 y disponen de un rango de cobertura más que suficiente para el escenario planteado. Además, ofrecen la posibilidad de cambiar de canal, característica que nos permite aplicar la idea de la tecnología de radio cognitiva, y nos otorgan la posibilidad de cifrar las transmisiones a través del algoritmo AES con clave de 128 bits.

Una de las limitaciones de estos dispositivos es que no implementan los modos de cifrado definidos en el estándar. En particular, los XBee implementan el modo CBC. Por otro lado, estos dispositivos tampoco ofrecen la función de autenticación de mensajes. Por dicho motivo, en este proyecto se ha extendido la funcionalidad de los XBees para que envíen mensajes cifrados y autenticados, tal y como se explicará en la sección 3.4.1.

Finalmente, cabe destacar que los XBees son dispositivos muy sencillos y no tienen la capacidad de procesar información por sí mismos al no disponer de microprocesador. Por ello, necesitaremos de otros dispositivos como un PC y Arduino.

Fig. 3.2 Dispositivo XBee Series 1

3.2.2 Arduino Board UNO

Este dispositivo (ver Figura 3.3) cuenta con un microprocesador ATmega328P que permite suplir la incapacidad de procesamiento del dispositivo XBee y poder cargar código en memoria para posteriormente ejecutarlo, procesar información y realizar acciones.

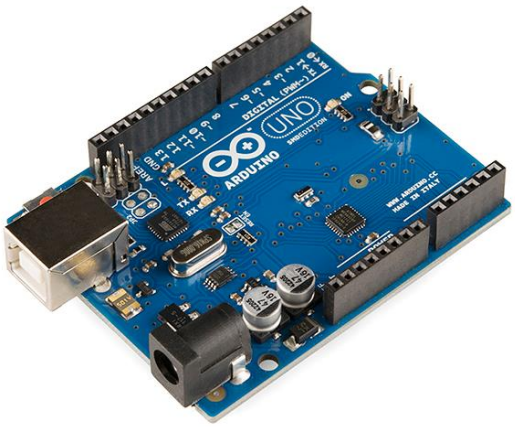

Fig. 3.3 Placa Arduino UNO

3.2.3 Arduino Xbee Shield

Para poder interconectar los dos dispositivos anteriores necesitamos un adaptador, para ello utilizaremos el XBee Shield (Figura 3.4) que nos permite realizar este cometido, de forma que se pueda procesar toda la información recibida a través de XBee con la capacidad de procesamiento que nos proporciona la placa Arduino.

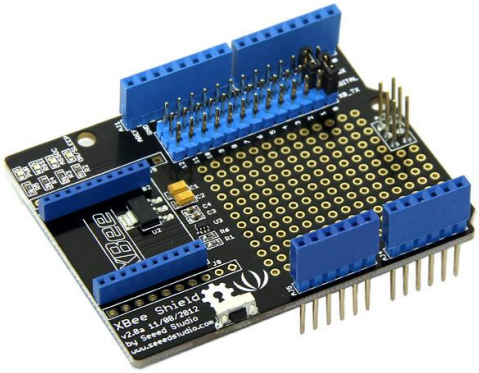

Fig. 3.4 Shield XBee para Arduino

3.2.4 Sensor TMP36

TMP36 es un sensor de temperatura el cual podremos conectar a nuestra placa Arduino UNO y obtener información de la temperatura corporal para posteriormente procesarla y transmitirla entre los dispositivos XBee.

Fig. 3.5 Sensor de temperatura TMP36

3.2.5 Xbee explorer USB

Para poder conectar los dispositivos XBee al PC necesitaremos un adaptador que permita la comunicación entre ambos bandos, para ello utilizaremos el XBee explorer USB que nos permitirá configurar los distintos parámetros de los XBee Series 1 a través de un software específico desarrollado por el fabricante.

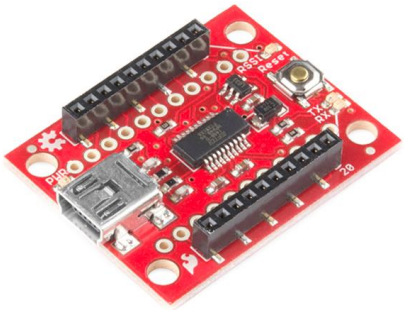

Fig. 3.6 Dispositivo XBee explorer USB

3.3 Configuración de los dispositivos

Para operar con estos dispositivos se deben configurar previamente con el fin de que se puedan comunicar entre ellos. Para ello *Digi International*, la empresa que fabrica los dispositivos XBee, proporciona un *software* llamado X-

CTU que permite configurar estos dispositivos desde una interfaz gráfica de forma sencilla.
Para poder configurar los dispositivos XBee se necesita el componente XBee explorer USB que es el que permite la comunicación con el PC. Una vez montado el XBee en el explorer USB, el software X-CTU lo reconocerá y dará la posibilidad de cambiar y configurar todos los parámetros del dispositivo que sean accesibles para el usuario.

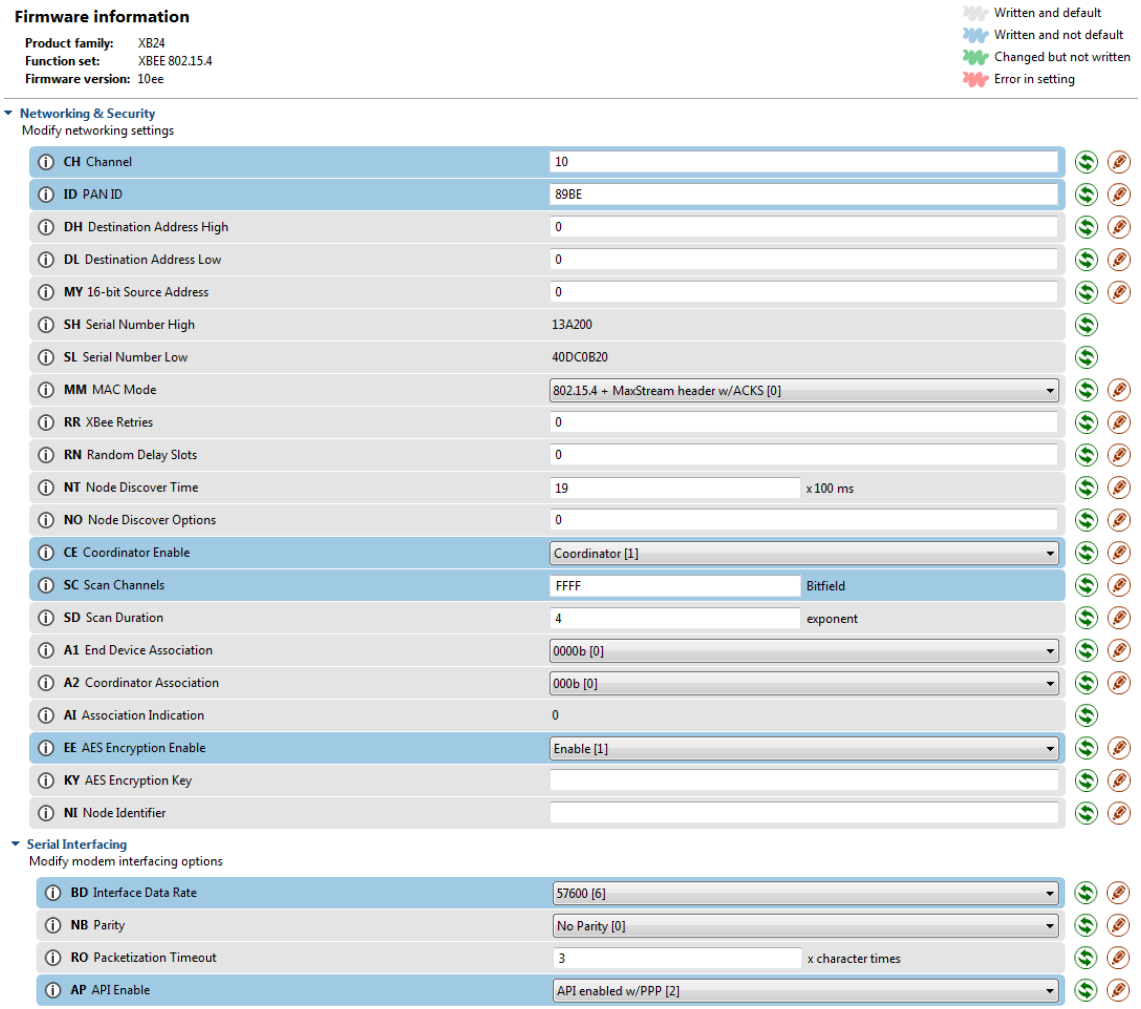

Fig. 3.7 Interfaz grafica de la aplicación X-CTU

La Figura 3.7 muestra las diferentes opciones de configuración. A continuación se describen las necesarias para este proyecto:

- *Channel*: el número del canal debe ser el mismo en ambos nodos para que puedan comunicarse.
- PAN ID: este parámetro es el identificador de la red personal, y al igual que el parámetro anterior también debe ser igual para ambos nodos.
- *Coordinator Enable*: en el caso del coordinador se debe activar esta opción (valor 1), mientras que para el nodo sensor (nodo A) se debe desactivar (valor 0).

- *Scan Channels*: número de canales escaneados por el coordinador. El máximo valor posible es (0xFFFF). En este caso se escanean los 16 canales disponibles.
- AES *Encryption Enable*: Este campo se activará o desactivará en función de si se quieren cifrar los datos transmitidos o no.
- AES *Encryption* Key: En caso de que se active la opción anterior, se debe especificar una clave de 16 bytes en este campo. Es la clave que van a compartir los dispositivos para cifrar/descifrar mensajes.
- *Interface Data Rate*: En este campo se configura el *baud rate* de transmisión de los datos. Ambos XBee deben tener el mismo valor para poder comunicarse. Los valores disponibles van desde un mínimo de 1200 bps hasta 115200 bps.
- API *Enable*: Esta opción activa o desactiva el modo API. XBee puede trabajar en dos modos, el modo AT o el modo API. El primero se basa en el envío de comandos AT a través de la consola que proporciona X-CTU. El segundo permite enviar tramas entre los dispositivos con lo cual ofrece más versatilidad y opciones.

Para poder obtener la información de la temperatura se debe conectar el sensor TMP36 a la placa Arduino, tal y como se muestra en la Figura 3.8.

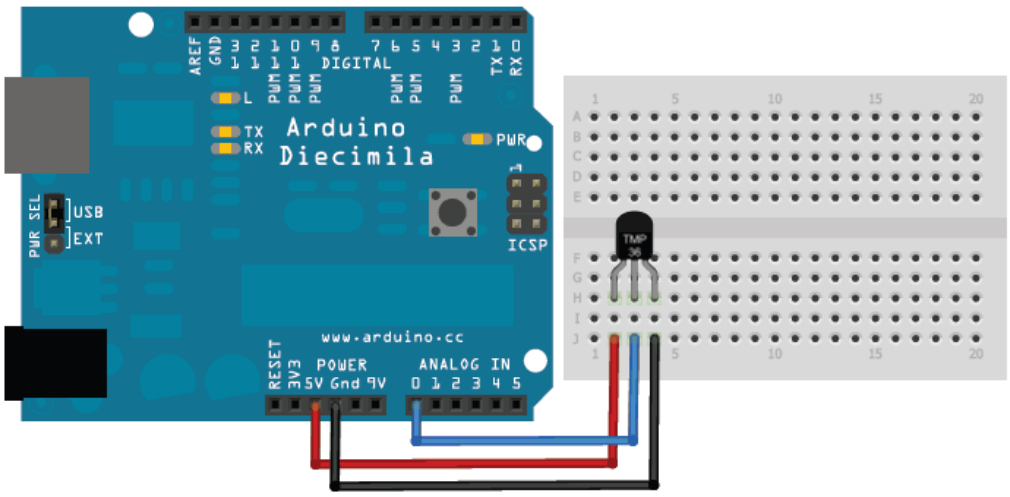

Fig. 3.8 Esquema de conexión del sensor TMP36

3.4 Funcionamiento de la WBAN

Al iniciar la red, el coordinador configura el canal en el que operarán todos los dispositivos de la red. Tal y como se ha mencionado anteriormente, los dispositivos XBee permiten que el coordinador realice un escaneo de los diferentes canales y escoja el más idóneo para transmitir.

El coordinador envía inicialmente una petición de cambio de canal (al canal especificado) en modo *broadcast*, aunque en nuestro escenario únicamente hay un nodo en la red. Seguidamente cuando el nodo A ya está operando en el nuevo canal, el propio coordinador cambia su canal de operación para poder seguir manteniendo comunicación con él.

A continuación se muestra la función utilizada por el nodo coordinador para realizar el cambio de canal:

```
public static void changeChannel(int newChannel) throws XBeeTimeoutException,
XBeeException{

	System.out.println("Realizando cambio de canal");
	RemoteAtRequest ratr = new
	RemoteAtRequest(XBeeRequest.DEFAULT_FRAME_ID,
	XBeeAddress64.BROADCAST, XBeeAddress16.BROADCAST, true, "CH",
	newChannel);
	RemoteAtResponse response = (RemoteAtResponse)
	xbee.sendSynchronous(ratr, 5000);
	System.out.println("El nuevo canal escogido es " + decToHex(newChannel));
	if (response.isOk()) {

		System.out.println("Cambio de canal realizado en el nodo sensor");
	}

	AtCommand at = new AtCommand("CH", newChannel);
	AtCommandResponse ATresponse = (AtCommandResponse)
	xbee.sendSynchronous(at, 5000);
	if (ATresponse.isOk()) {
		System.out.println("Cambio de canal realizado en el coordinador");
	}
}
```

En este caso el canal inicial escogido es el 16 (en hexadecimal su valor seria 0x10, que es como lo muestra el nodo sensor). En la sección 4.1 se presentan los resultados obtenidos durante el escaneo de frecuencias.

Una vez los dos dispositivos se pueden comunicar entre sí, el nodo A obtiene la información de la temperatura a través del sensor TMP36. Para obtener la temperatura se debe convertir a voltaje el valor que nos proporciona el sensor a través del pin analógico número 0 de Arduino. En este caso, como se ha utilizado la entrada de 5V de Arduino, multiplicamos la lectura por 5. A continuación se muestra el código utilizado para convertir el voltaje en temperatura.

```
int reading = analogRead(sensorPin);
float voltage = reading * 5.0;
voltage /= 1024.0;
float temperatureC = (voltage - 0.5) * 100
Serial.println();
Serial.print("Temperatura: ");
Serial.print(temperatureC); Serial.println(" ºC");
```

Una vez obtenida la temperatura, debemos enviarla hacia el nodo coordinador. Para ello se introducen los datos de la temperatura en el *payload* del paquete que se enviará y se transmite de la siguiente manera:

```
void sendTemp() {
        Serial.print("Enviando datos de la temperatura al coordinador. ");
        xbee.send(tx);
        xbee.readPacket(5000);
        if (xbee.getResponse().isAvailable()) {
                if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
                   xbee.getResponse().getRx16Response(rx16);
                   receivedData = rx16.getData();
                   for(int j=0;j<rx16.getDataLength();j++){
                           if(receivedData[j] == 1){
                             Serial.println("Comunicacion correcta");
                           }else{
                            Serial.println("No se ha recibido respuesta. Posible
                            perdida de la comunicacion");
                           }
                   }
         }
    }
}
```

```
Temperatura: 26.66 °C
Enviando datos de la temperatura al coordinador. Comunicacion correcta
El valor del canal actual es 10
```

Fig. 3.9 Información mostrada en el nodo A al enviar la temperatura

Tal y como se ha mencionado anteriormente, los dispositivos XBee pueden transmitir los datos en claro, o bien cifrados mediante AES-CBC. Con respecto a la autenticación de los mensajes, dado que estos dispositivos no ofrecen esta funcionalidad, se han programado los XBees para que generen un MAC que se transmita junto con el mensaje (ambos en claro o cifrados). La implementación realizada se describe con detalle en la sección 3.4.1.

Una vez transmitidos los datos, estos llegarán al nodo B (coordinador) que se encargará de procesarlos y mostrarlos, previa comprobación de la autenticidad del mensaje.

```
Autenticando mensaje...
MAC recibido: 29C8F2C42C780D478746107EDCA33AE6
MAC calculado: 29C8F2C42C780D478746107EDCA33AE6
Mensaje autenticado correctamente
```

Fig. 3.10 Información mostrada en el nodo B al autenticar el mensaje

Seguidamente realizará un escaneado de la red a partir del cual obtendrá los niveles de energía en ese instante de tiempo de todos los canales en los que puede operar. Para realizar el escaneado de la red se utiliza el siguiente método:

```
public static void scanChannels() throws XBeeTimeoutException, XBeeException{
        energy.clear();
        AtCommand  at = new AtCommand("ED");
        AtCommandResponse response = (AtCommandResponse)
        xbee.sendSynchronous(at, 5000);
        if (response.isOk()) {
                int[] array = response.getProcessedPacketBytes();
                for (int i =7; i<23; i++){
                        energy.add(array[i]);
                }
        }
}
```

Una vez realizado el escaneado se analizan los datos obtenidos y se decide si la red está operando en un canal que nos otorga una comunicación lo suficientemente buena o si por lo contrario se requiere un cambio de canal para poder asegurar el correcto funcionamiento del servicio.

```
int threshold = 66;
System.out.println("El canal actual es : " + currentChannel);
System.out.println("Comprobando la energia del canal actual... la energia es: -" +
energy.get(currentChannel - 11) + "dbm");
if(Integer.valueOf(energy.get(currentChannel - 11)) < threshold){
        System.out.println("Energia demasiado alta, requiere cambio de canal.");
        changeCHreq = true;
}

if(changeCHreq == true){
        int bestChannel = energy.get(0);
        newChannel = 11;
        currentChannel = newChannel;
        for(int i = 1; i<energy.size(); i++){
                if(energy.get(i) > bestChannel){
                        bestChannel = energy.get(i);
                        newChannel = i + 11;
                        currentChannel = newChannel;
                }
        }
        changeChannel(newChannel);
}
```

Si requiere cambio de canal, se ejecutará la función vista anteriormente (*changeChannel*) que cambia el canal tanto del nodo A como del nodo B. A continuación podemos ver la información que muestra el coordinador cuando escanea los canales y toma la decisión de si debe o no cambiar de frecuencia de operación.

```
Autenticando mensaje...
MAC recibido: 29C8F2C42C780D478746107EDCA33AE6
MAC calculado: 29C8F2C42C780D478746107EDCA33AE6
Mensaje autenticado correctamente

Temperatura recibida: 20,31

El canal actual es : 10
Comprobando la energia del canal actual... la energia es: -84dbm
--------------------------------------------------------------------------

Autenticando mensaje...
MAC recibido: C88792B65D4195E73633557E096EF709
MAC calculado: C88792B65D4195E73633557E096EF709
Mensaje autenticado correctamente

Temperatura recibida: 19,82

Comprobando la energia del canal actual... la energia es: -57dbm
Energia demasiado alta, requiere cambio de canal.
Realizando cambio de canal
El nuevo canal escogido es B
Cambio de canal realizado en el nodo sensor
Cambio de canal realizado en el coordinador
```

Fig. 3.11 Información mostrada en el nodo coordinador a cambiar de canal

Como se puede observar en la siguiente imagen, el nodo A también ha cambiado de canal tal y como ha requerido el coordinador.

```
Temperatura: 26.66 °C
Enviando datos de la temperatura al coordinador. Comunicacion correcta
El valor del canal actual es B
Temperatura: 27.64 °C
Enviando datos de la temperatura al coordinador. Comunicacion correcta
El valor del canal actual es B
```

Fig. 3.12 Información mostrada en el nodo A una vez cambiado de canal

La idea inicial era que el nodo coordinador se encargara de cambiar de canal la red en la que opera la WBAN del paciente cuando detectase que el *throughput* de las transmisiones decaía a partir de un cierto valor. Sin embargo, dado que el *throughput* observado durante las pruebas no varía significativamente con las interferencias, se ha decidido utilizar en su lugar la energía de los canales.

Se ha decidido utilizar un umbral de -66dbm, es decir, si el coordinador detecta que el canal en el que está trabajando la WBAN del usuario tiene una energía superior a -66dbm, entonces procederá a realizar un cambio de canal a uno con unas condiciones más óptimas. Se ha seleccionado este valor ya que es el valor medio obtenido en las pruebas con interferencias (ver Sección 4.2.3).

Los códigos utilizados por los dos nodos, tanto para la implementación del *spectrum sensing* explicado como para las pruebas que se explicarán en el Capítulo 4, se pueden encontrar en el anexo.

3.4.1 Autenticación de los mensajes

Con el fin de evitar la posible manipulación del mensaje o que un atacante pueda enviar datos erróneos al coordinador, se ha implementado un sistema de autenticación entre los dos nodos basado en el protocolo AES en modo CBC con clave de 128 bits. La Figura 3.13 muestra el esquema usado para generar el código MAC.

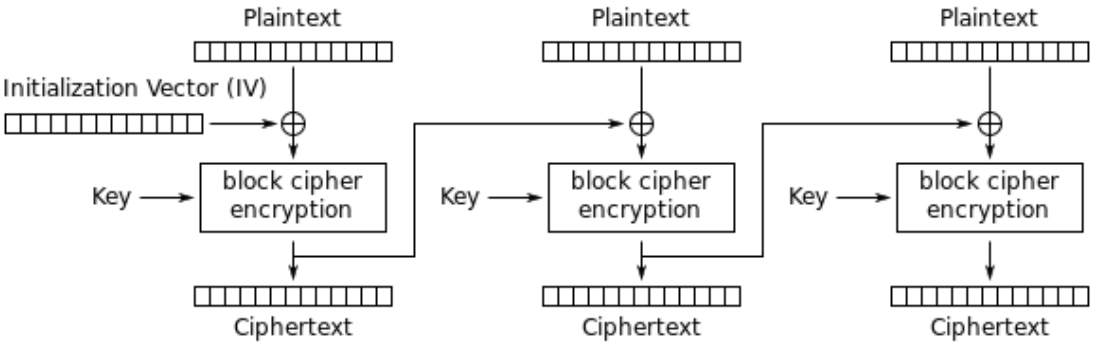

Fig. 3.13 Cifrado AES-CBC

El modo CBC requiere el uso de un vector de inicialización (*Initialization Vector* o IV), una secuencia de 128 bits que se escoge inicialmente de manera aleatoria y varía con cada paquete transmitida (por ejemplo, incrementando en una unidad su valor).

El funcionamiento de este algoritmo es el siguiente:

- El mensaje original, se divide en bloques de 16 bytes. En el caso de la aplicación de este proyecto, el mensaje es de 2 bytes (tamaño que ocupa el valor de temperatura) y por lo tanto solo tendremos un bloque al que se le aplicará un *padding* de 0's hasta obtener los 16 bytes.

```
MAC[0] = (uint8_t)temp1;
MAC[1] = (uint8_t)temp2;
int padding = 0;
for (int j=2; j<16; j++){
   MAC[j]=(uint8_t)padding;
}
```

- Se realiza la operación xor entre el mensaje y el vector de inicialización, que en este caso se ha implementado como un contador, y se cifra el resultado con la clave pre-compartida por ambos nodos. Como resultado se obtiene el MAC que se adjuntará al mensaje original cuando se transmita. En este caso para realizar dicha operación se ha utilizado la librería AESLib de Arduino [24]. Se usa como MAC la salida del cifrado de 128 bits, pero puede truncarse y obtener valores más pequeños (32 o 64 bits), para disminuir el *overhead*.

```
aes128_cbc_enc(key, iv, MAC, sizeof(MAC));
payload[0] = (uint8_t)temp1;
payload[1] = (uint8_t)temp2;
for(int j = 2; j<18; j++){
  payload[j]=MAC[j-2];

}
```

El paquete transmitido es cifrado por el dispositivo XBee ya que la opción AES *Encryption Enable*, descrita en el punto 3.3, se ha configurado para realizar dicha acción. El *payload* del paquete contiene tanto el mensaje original como el MAC del mensaje cuando es transmitido.

Fig. 3.14 Formato del *payload* cifrado

De esta manera el coordinador podrá comprobar la autenticidad del mensaje recibido mediante la repetición del proceso descrito previamente, ya que ambos nodos comparten la clave y utilizan el mismo vector de inicialización, el cual es un contador que aumenta con cada transmisión realizada.

```
calculatedMac= encrypt(receivedMsg);
for(int i = 9; i<=24; i++){
    receivedMac = receivedMac.concat(decToHex(array[i]));
}

if(calculatedMac.equals(receivedMac)){
[…]
```

```
public static String encrypt(byte[] strToEncrypt)
 {
    try
    {
        Cipher cipher = Cipher.getInstance("AES/CBC/NOPADDING");
        final SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
        cipher.init(Cipher.ENCRYPT_MODE, secretKey, new IvParameterSpec(iv));
        byte[] result = cipher.doFinal(strToEncrypt);
        return bytesToHex(result);
    }
    catch (Exception e)
    {
      e.printStackTrace();

    }
    return null;
}
```

En resumen, el proceso de autenticación sería el mostrado en la siguiente imagen donde *Sender* hace referencia al nodo A y *Receiver* al nodo B.

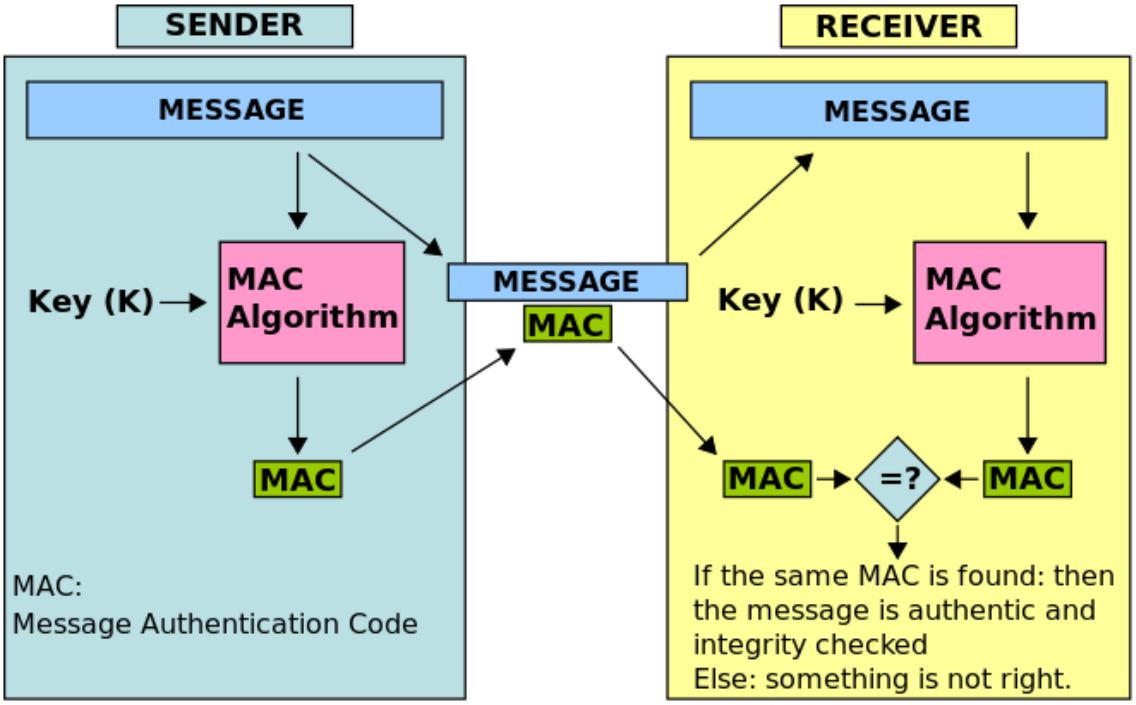

Fig. 3.15 Proceso de autenticación del mensaje [25]

CAPÍTULO 4. Evaluación de las prestaciones de la WBAN

4.1 Escaneado de frecuencias y selección del canal de operación

En una red de radio cognitiva se usan habitualmente mecanismos de *sensing* cooperativos, en los que los nodos miembros de la red toman medidas de energía que envían a un nodo central. Aplicando un determinado método de fusión de datos, dicho nodo decide qué frecuencias están libres y cuál es el mejor canal para operar.

La WBAN que se ha implementado no permite implementar dichos mecanismos, ya que el único nodo que realiza escaneo de frecuencias y decide el canal de operación es el nodo coordinador de la red.

En este apartado se ha analizado cómo el nodo coordinador realiza este proceso y el tiempo requerido para el mismo.

Los dispositivos XBee cuentan con 16 posibles canales (desde el 11 al 26) en los que trabajar. Sus frecuencias son las siguientes:

Tabla 4.1 Frecuencias de operación de XBee [18]

Canal	Hexadecimal	Frecuencia
11	0x0B	2,405 GHz
12	0x0C	2,410 GHz
13	0x0D	2,415 GHz
14	0x0E	2,420 GHz
15	0x0F	2,425 GHz
16	0x10	2,430 GHz
17	0x11	2,435 GHz
18	0x12	2,440 GHz
19	0x13	2,445 GHz
20	0x14	2,450 GHz
21	0x15	2,455 GHz
22	0x16	2,460 GHz
23	0x17	2,465 GHz
24	0x18	2,470 GHz
25	0x19	2,475 GHz
26	0x1A	2,480 GHz

Los resultados obtenidos después del escaneado realizado con el software X-CTU son los siguientes:

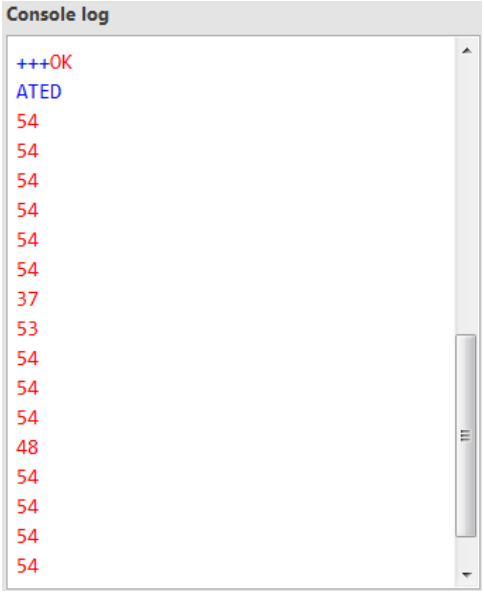

Fig. 4.1 Resultados en hexadecimal del escaneo de canales

Tabla 4.2 Resultados en decimal del escaneo de canales

Canal	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Energía (-dbm)	84	84	84	84	84	84	55	83	84	84	84	72	84	84	84	84

La Figura 4.1 muestra la energía de los 16 canales escaneados por el dispositivo XBee en valor hexadecimal. Estos valores han sido convertidos a base decimal en la tabla 4.2.
El valor mínimo de nivel de energía que se puede obtener en el escaneado de canales es -84 dbm. Como podemos ver en la tabla superior hay varios canales con el nivel de energía mínimo, y que por lo tanto proporcionan unas mejores condiciones del medio ya que cuanto menor sea el nivel de energía del canal, menor será el tráfico que tiene lugar en él y más libre estará.
En dicha tabla también se puede observar que hay algún canal que dispone de niveles de energía más altos como es el caso de los canales 17, 18 y 22. Esto puede ser debido a redes Wi-Fi cercanas (vecinos) que estén operando en esos canales.

El tiempo de escaneado de los diferentes canales se rige por la siguiente fórmula:

$$SC * 2^{ED} * 15.36ms$$

Ecuación 4.1 Tiempo de escaneado

donde SC es el número de canales a escanear (en este caso 16) y ED es un parámetro configurable por el dispositivo que puede tener un valor desde 0 a 6.

Con lo cual el tiempo mínimo de escaneado es de 245.7ms y el máximo de 15.7 segundos dependiendo de ED.
En un entorno como el usado para realizar las pruebas (hogar), las medidas de energía de los diferentes canales no varían mucho a lo largo del tiempo ni presentan diferencias significativas para diferentes tiempos de escaneado. Por ello, es suficiente con usar el tiempo mínimo de escaneado

4.2 Análisis del *throughput*

Para realizar las pruebas de *throughput* se ha seleccionado el canal número 13, ya que es uno de los que tienen una energía mínima (menos interferencias) y permitirá realizar las pruebas en unas buenas condiciones iniciales.

Pese a que, en teoría, estos dispositivos soportan una configuración de un *baud rate* máximo de 115200 bps, para estas pruebas se han configurado a 57600 bps ya que con el *baud rate* máximo las transmisiones no son estables y los dispositivos no logran comunicarse.

El nodo A se ha programado sobre la plataforma Arduino en lenguaje C de manera que transmita 100 paquetes al nodo B para posteriormente calcular el tiempo transcurrido RTT (*round trip time*), es decir, el tiempo transcurrido desde que se envían los paquetes hasta que se reciben las confirmaciones (*acknowledgements* o ACKs) correspondientes y poder obtener el *throughput*. Definimos el *throughput* como la cantidad de paquetes enviados durante un tiempo RTT.

El nodo B se ha programado en lenguaje Java utilizando una API previamente creada [10]. Esta API nos permite trabajar con paquetes XBee con lo cual podemos recibir y transmitir información entre nodos de este tipo. El código utilizado para las pruebas se puede encontrar en su totalidad en el anexo.

Las pruebas iniciales se han realizado situando a los dos nodos a una distancia de 20cm y transmitiendo paquetes de tamaño máximo desde el nodo A, es decir, de 115 bytes, 100 de los cuales corresponden a los datos a transmitir. En la siguiente imagen se puede ver la estructura del paquete transmitido:

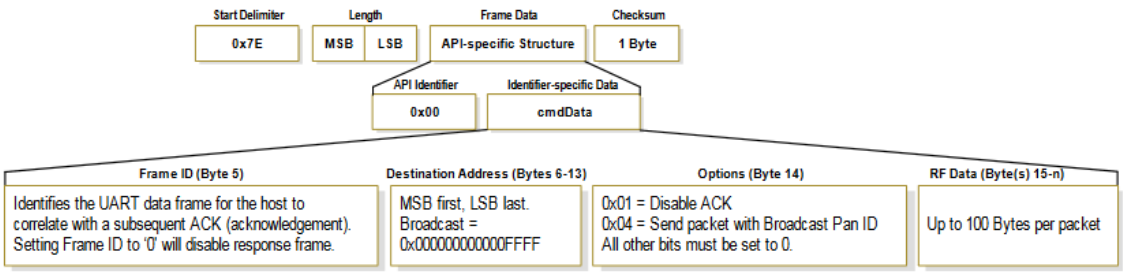

Fig. 4.2 Trama del paquete con direcciones de 64 bits

En cambio, la respuesta transmitida por el nodo coordinador contendrá únicamente como datos un identificador que actuará como ACK del paquete previamente recibido.

Para esta prueba se han trasmitido paquetes en claro, es decir, sin cifrar desde el nodo A al nodo B de tal forma que cada vez que se transmite un paquete se espera la respuesta a dicho paquete antes de transmitir el siguiente. En el siguiente diagrama podemos observar el procedimiento seguido en este test:

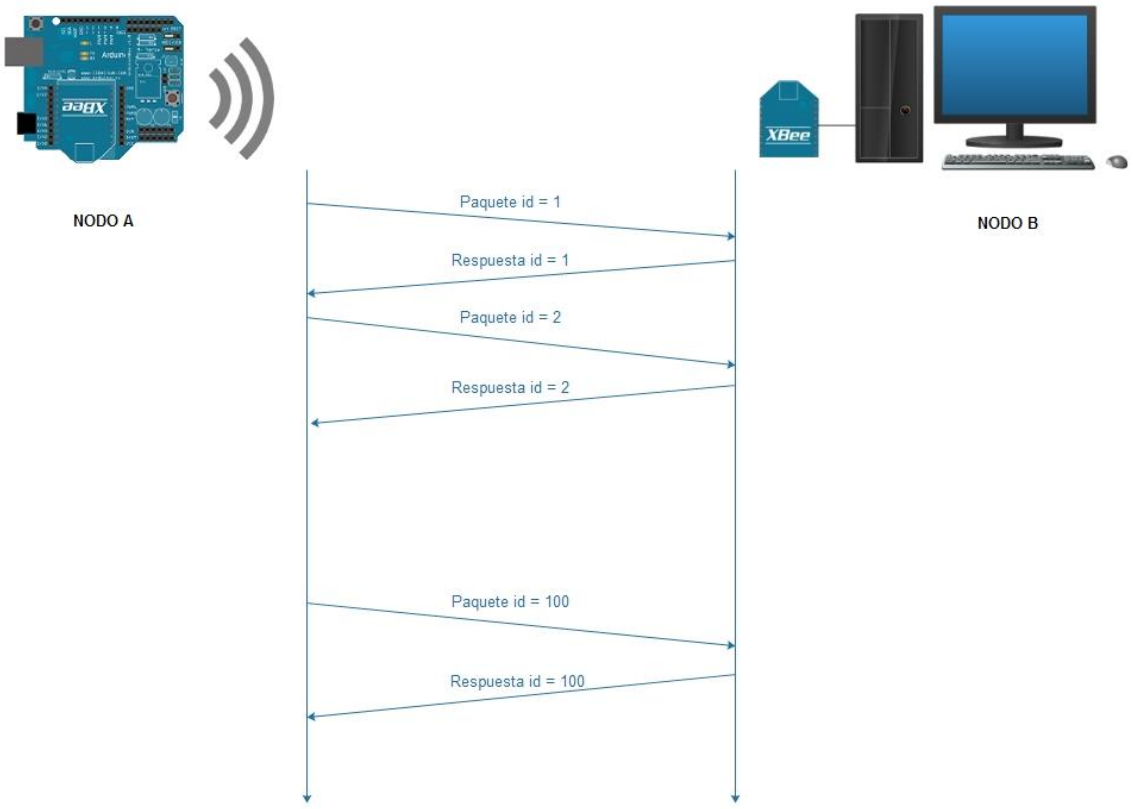

Fig. 4.3 Transmisión de paquetes sin cifrar y sin autenticar

Para realizar el test de esta manera se ha especificado un tiempo de espera máximo entre el envío de paquetes de 5000ms a través de la siguiente línea de código:

```
xbee.readPacket(5000);
```

Esto hará que cuando el nodo A envíe un paquete, espere durante un máximo de 5 segundos a recibir una respuesta a dicho paquete antes de proseguir con el siguiente. Esto quiere decir que si la respuesta llega antes de 5 segundos, que sería lo normal, continua con las acciones pertinentes para la lectura de los datos recibidos e inicia la siguiente transmisión sin esperar a agotar los 5 segundos de espera, de lo contrario, en caso de que se agoten los 5000ms y no se haya recibido una respuesta, el paquete se da como perdido.

Después de transmitir 250 paquetes y habiendo medido tanto el RTT como el número de bytes transmitidos, obtenemos los siguientes resultados del *throughput* medio:

Tabla 4.3 *Throughput* sin cifrado y sin autenticación

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
12.332	511,3	-33,66	-84

Como podemos observar, el *throughput* dista mucho del máximo teórico configurado que es de 57600 bps. El hecho de tener que esperar por una respuesta antes de transmitir un nuevo paquete hace que no estemos aprovechando ese periodo de tiempo para seguir transmitiendo paquetes. La energía del canal que obtenemos es la mínima que puede devolver XBee, lo cual nos da a entender que el canal esta libre y por lo tanto las condiciones en las que se ha realizado la prueba son favorables. La potencia de señal recibida es bastante buena teniendo en cuenta que el rango de valores disponible en los dispositivos XBee va desde -23 dbm (mejor caso) hasta -92 dbm (peor caso).

La segunda prueba realizada es muy similar a la primera con la diferencia de que esta vez no se espera a recibir una respuesta antes de enviar los siguientes paquetes, con lo cual se supone que aumentará el *throughput* de la transmisión al poder transmitir varios paquetes durante el transcurso del envío y recepción de un paquete dado. En este caso, los datos también se transmiten sin cifrar. A continuación podemos ver un diagrama de la situación:

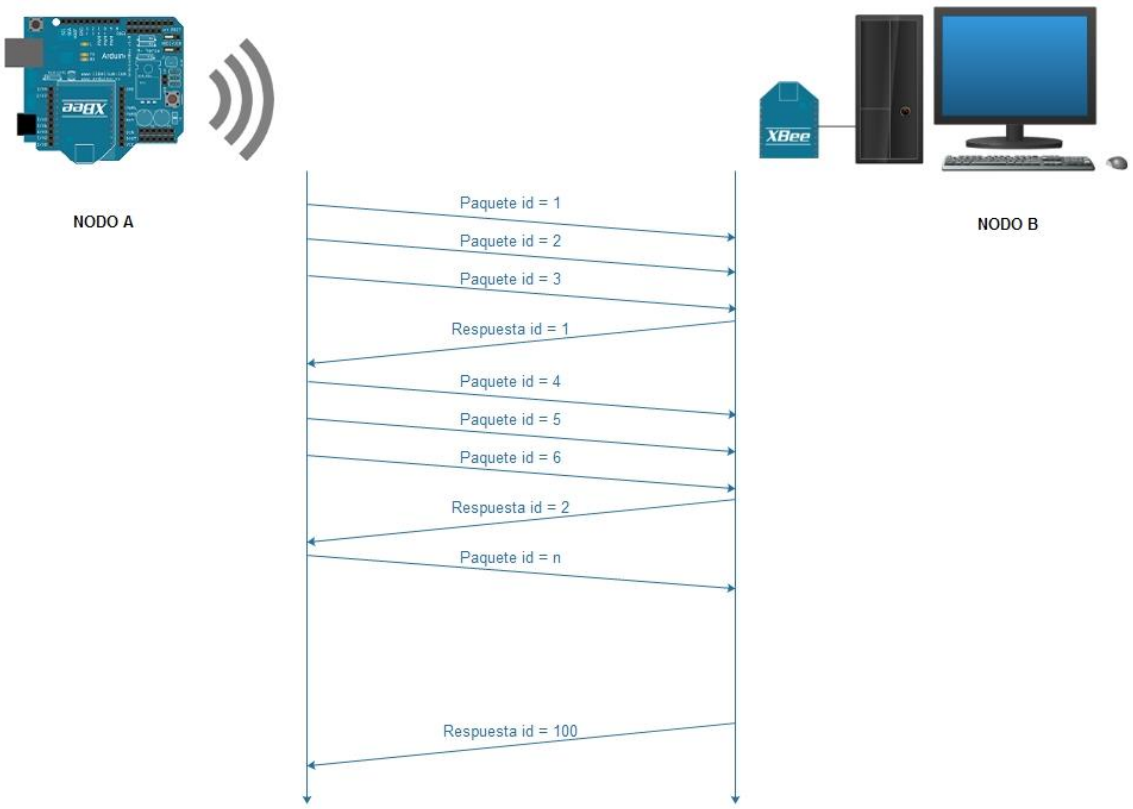

Fig. 4.4 Transmisión sin cifrado y sin autenticación con tiempo de espera 10ms

Aunque en teoría no se debería esperar nada entre el envío y recepción de un paquete, en la práctica de esta segunda prueba se ha configurado un valor de espera de 10ms entre el envío de un paquete y la recepción de la respuesta, ya que si se configura un tiempo de espera de 0ms los paquetes se darían como perdidos al ser imposible recibir una respuesta en ese tiempo. Con este nuevo valor cada vez que transmitamos un paquete únicamente se esperaran 10ms antes de transmitir el siguiente, con lo cual si la respuesta tarda más de ese tiempo especificado se enviará el próximo paquete y así sucesivamente hasta obtener una respuesta a un paquete previamente enviado. De esta forma entre el envío y recepción de un paquete específico habremos podido enviar varios paquetes en lugar de desperdiciar ese tiempo esperando.
En este caso, nos encontramos con una limitación a nivel de código que hace que no podamos procesar todas las respuestas, ya que al realizarlo de la manera previamente explicada puede que cuando el nodo A reciba una respuesta del nodo B, el código este procesando una respuesta previa o transmitiendo un paquete en lugar de estar escuchando la interfaz de entrada por donde recibe los paquetes de forma física. Esto hará que por cada ejecución del código (el cual transmite 100 paquetes) solo se puedan procesar unas pocas respuestas aunque realmente estemos recibiendo todas ellas.

Después de realizar 250 transmisiones se obtienen los siguientes resultados:

Tabla 4.4 *Throughput* sin cifrado y sin autenticación con tiempo de espera 10ms

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
30.479	960,35	-33,91	-84

Como era de esperar, el *throughput* ha aumentado respecto a la prueba anterior al haber sido capaz de transmitir varios paquetes entre el envío y la recepción de un paquete dado. El resultado obtenido prácticamente triplica el rendimiento del caso anterior, aun así está lejos del máximo teórico que es de 57600 bps.
Respecto a las condiciones del canal se puede observar que se mantienen favorables al haber obtenido una potencia de señal de -33,9 dbm y contar con un nivel de energía de -84 dbm.

4.2.1 Efecto del tamaño de paquete

En esta prueba se pretende evaluar el efecto que ejerce el tamaño de los paquetes sobre el *throughput*. En el apartado anterior se han utilizado paquetes de un tamaño de 115 bytes, es decir, 100 bytes de datos. Para esta prueba se transmitirán paquetes de 25 y 50 bytes para observar si el *throughput* obtenido varía, y en tal caso valorar si ello nos proporciona un mayor o menor rendimiento en la transmisión.

Con la idea de maximizar el *throughput*, los paquetes se transmiten con un tiempo de espera mínimo al igual que se ha hecho en la segunda prueba de la sección 4.2.

Los resultados obtenidos después de varias transmisiones son los siguientes son los siguientes:

Tabla 4.5 Efecto del tamaño del paquete en el *throughput*

Tamaño paquete (bytes)	Throughput (Kbps)	Deviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
25	11.093	302,9	-32,42	-84
50	18.197	526,3	-32,59	-84

Como se puede observar en los resultados, el *throughput* obtenido es menor cuanto más pequeño es el paquete transmitido, tanto si comparamos los

tamaños analizados en este apartado como si los comparamos respecto a los 115 bytes de la segunda prueba del apartado anterior.

4.2.2 Efecto de la distancia

En esta prueba se espera a ver cómo se comporta la calidad de la transmisión dependiendo de la distancia a la que se encuentran separados los nodos, así podremos comprobar la importancia que puede tener en una aplicación médica la colocación del nodo que recolecta la información enviada por los sensores. En nuestro caso se supone que dicho nodo siempre estará situado en alguna parte de nuestro cuerpo para que la aplicación médica pueda seguir obteniendo datos sin tener que estar en un lugar específico, es por ello que se ha considerado un rango máximo de 2 metros.

Para realizar esta prueba, los nodos empezarán en una distancia de 0 cm y se irán separando de 5 en 5 centímetros hasta alcanzar una distancia máxima de 2 metros. Para cada separación de 5 centímetros se realizaran 10 transmisiones para obtener los valores que permitan valorar la calidad de la transmisión como en los casos anteriores, y calcular una media de estas medidas.

Tabla 4.6 *Throughput* y RSSI en función de la distancia

Distancia (cm)	Throughput (Kbps)	RSSI (dbm)	Energía del canal (dbm)
0	31.150	-24,27	-84
5	31.127	-28,45	-84
10	31.038	-27,18	-84
15	30.694	-28,73	-84
20	30.587	-31,82	-84
25	30.507	-31,91	-84
30	30.619	-40,36	-84
35	30.581	-44,00	-84
40	30.464	-44,45	-84
45	30.401	-40,45	-83
50	30.404	-41,09	-83
55	30.576	-43,36	-84
60	30.527	-45,73	-84
65	30.497	-47,73	-84
70	30.508	-47,82	-84
75	30.382	-48,09	-84
80	30.472	-48,18	-84
85	30.527	-50,18	-83
90	30.448	-49,18	-84
95	30.472	-49,82	-84
100	30.475	-50,73	-84

105	30.412	-50,91	-84
110	30.399	-51,82	-84
115	30.387	-52,64	-84
120	30.375	-52,82	-84
125	30.372	-53,27	-83
130	30.371	-53,73	-84
135	30.356	-54,18	-84
140	30.358	-55,27	-84
145	30.347	-56,27	-84
150	30.355	-56,18	-84
155	30.353	-56,82	-84
160	30.350	-57,73	-84
165	30.323	-58,45	-84
170	30.305	-58,82	-84
175	30.284	-59,36	-84
180	30.259	-60,18	-84
185	30.218	-61,73	-84
190	30.219	-61,18	-84
195	30.184	-61,91	-83
200	30.151	-62,64	-84

A partir de los resultados obtenidos, se pueden generar unas gráficas que ayudarán a ver mejor la relación entre los parámetros calculados:

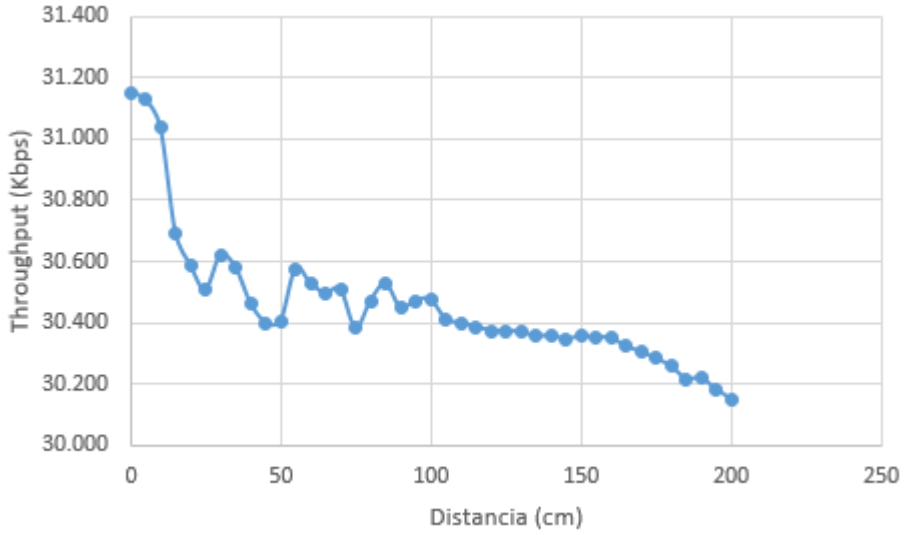

Fig. 4.5 Relación entre el *throughput* y la distancia

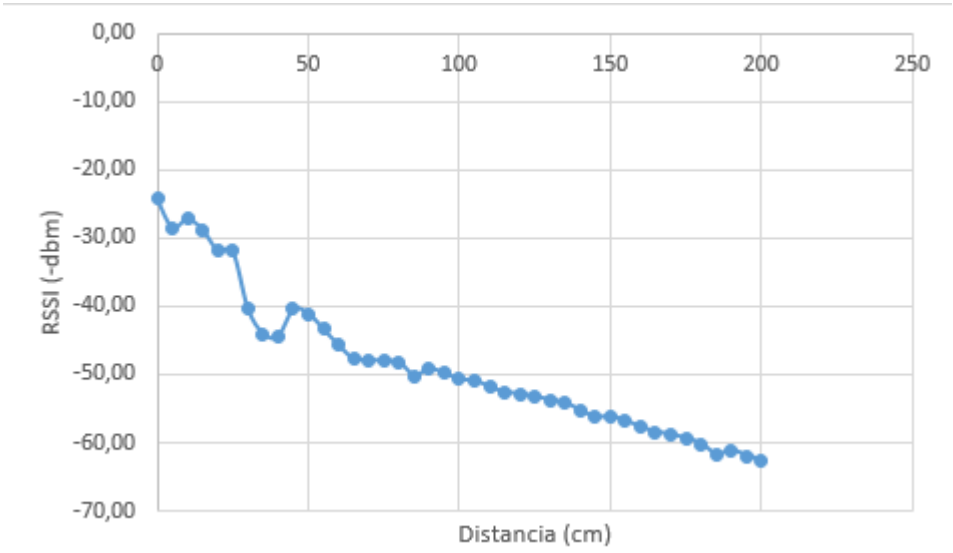

Fig. 4.6 Relación entre el RSSI y la distancia

Observando las gráficas superiores se puede ver que tienen lógica, ya que a mayor distancia, menor es el *throughput* y menor es también la potencia de señal recibida. Aun así, el valor del *throughput* apenas disminuye 1 Kbps desde distancia 0 cm hasta distancia 200 cm, con lo cual no importa mucho donde se coloque el colector que recoge los datos de los diferentes sensores que puedan desplegarse en un cuerpo humano, ya que el rendimiento apenas se verá afectado. Hay que tener en cuenta que para estas pruebas se están utilizando dispositivos 802.15.4, que pese a que son dispositivos de bajo consumo, tienen un rango de cobertura mayor que los dispositivos del protocolo 802.15.6 al estar pensados para aplicaciones a mayor escala como el despliegue de nodos en una ciudad en lugar de un cuerpo humano.

Con el objetivo de obtener unas medidas más fiables, se ha repetido la prueba realizando 30 medidas por cada distancia, que en este caso va desde los 0cm hasta los 200m en saltos de 10cm ya que, como se puede apreciar en las gráficas anteriores, el hecho de que la distancia entre medidas sea de únicamente 5cm puede provocar que otros factores tengan más peso sobre la transmisión que la propia distancia obteniendo picos e irregularidades en las medidas. Es por ello que se ha doblado la distancia entre medidas para otorgar, en la medida de lo posible, más peso en esta prueba a la distancia.

Los resultados obtenidos han sido los siguientes:

Tabla 4.7 *Throughput* y RSSI en función de la distancia

Distancia (cm)	Throughput (Kbps)	RSSI (dbm)	Energía del canal (dbm)
0	31.181	-24,27	-84
10	31.048	-27,20	-84

20	30.788	-28,63	-83
30	30.646	-40,20	-83
40	30.585	-41,23	-84
50	30.490	-42,13	-84
60	30.479	-45,70	-84
70	30.470	-47,67	-84
80	30.436	-48,27	-84
90	30.426	-49,37	-84
100	30.430	-50,77	-83
110	30.403	-51,67	-84
120	30.388	-52,70	-84
130	30.380	-53,77	-84
140	30.366	-55,20	-84
150	30.360	-56,23	-84
160	30.346	-57,80	-84
170	30.326	-58,83	-84
180	30.287	-60,24	-83
190	30.235	-61,27	-84
200	30.150	63,10	-84

A partir de estos datos, se generan las siguientes graficas:

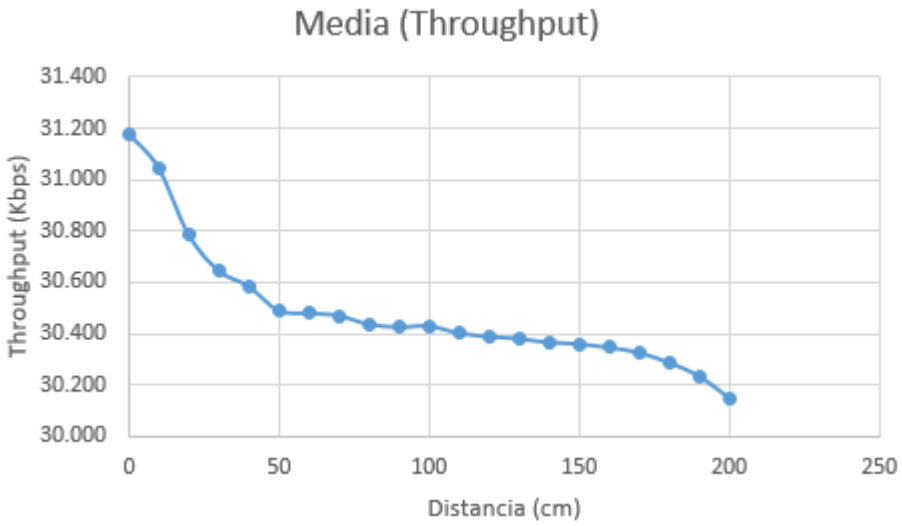

Fig. 4.7 Relación entre el throughput y la distancia

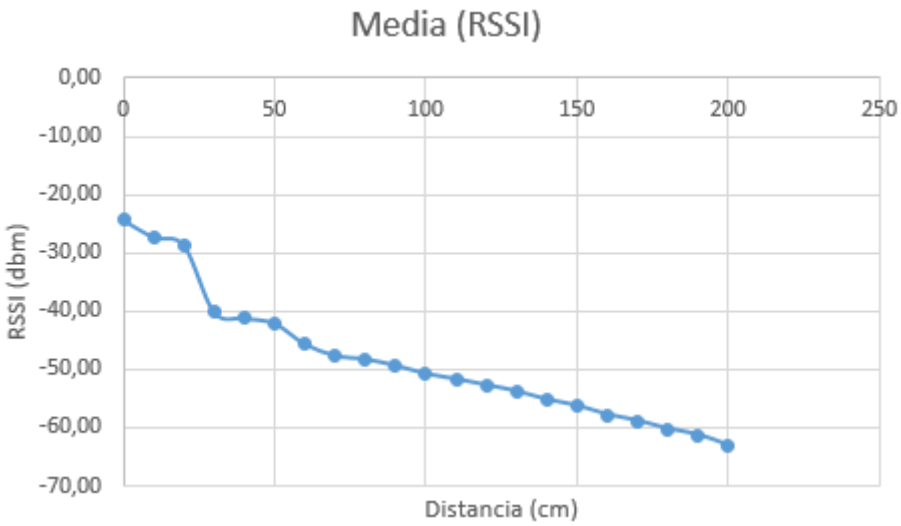

Fig. 4.8 Relación entre el RSSI y la distancia

Como se puede observar las medidas de esta segunda prueba de distancia son mas estables que en la primera, sin embargo la conclusion que se puede sacar es la misma: el rendimiento de la transmisión no se verá afectado por la distancia en un rango corporal de 0 a 2 metros.

4.2.3 Efecto de las interferencias

La diferencia de esta prueba respecto a las anteriores es que se aplicarán interferencias a la comunicación utilizando un microondas para ello, ya que también trabaja en la banda de frecuencias de 2.4MHz al igual que los dispositivos XBee [21].

Lo primero que se hará, por lo tanto, es observar cómo afecta la actividad del microondas a la energía de los canales en los que trabaja XBee. Para ello se realizará un escaneado de red con el microondas apagado y posteriormente con el microondas encendido, de esta forma podremos comparar los resultados y ver si el microondas afecta o no a los canales.

A continuación se pueden observar los resultados obtenidos de los escaneados de red:

Tabla 4.8 Energía de los canales con y sin interferencias

# Canal	Microondas OFF (dbm)	Microondas ON (dbm)
11	-84	-81
12	-84	-67
13	-78	-74
14	-84	-78

15	-77	-76
16	-84	-66
17	-80	-74
18	-82	-77
19	-84	-54
20	-84	-51
21	-84	-29
22	-84	-45
23	-84	-60
24	-84	-74
25	-84	-67
26	-84	-84

Como se puede ver en la tabla superior, cuando el microondas está apagado hay muchos canales con -84 dbm de energía, lo cual quiere decir que hay poca energía en esos canales y que por lo tanto no están ocupados por otros usuarios/dispositivos. En cambio, cuando se enciende el microondas se observa que apenas se obtienen canales con una energía de -84 dbm o que se aproxime, esto indica que el microondas está interfiriendo en la mayoría de los canales, lo cual sería similar a si un atacante intentase saturar nuestro canal y de esta forma evitar la transmisión de los datos.

Si se repite la segunda prueba de este documento pero esta vez aplicando las interferencias del microondas que provoca el aumento de energía de los canales, se obtienen los siguientes resultados:

Tabla 4.9 Efecto de las interferencias

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
29.671	1322,18	-43,352	-38,6

Lo que se puede observar a partir de estos resultados es, que como se ha visto, la energía del canal se ve afectada y obtiene un valor mayor al óptimo (-84 dbm), con lo cual las condiciones simularían el hecho de que en el canal que opera la red estaría ocupado por más redes (usuarios) que trabajan también en él y que por lo tanto interfieren en nuestras comunicaciones. La potencia de la señal recibida también se ve alterada respecto a la estabilidad que mostraba en las pruebas anteriores, en este caso se obtiene un valor menor con lo cual la señal entre ambos nodos es más débil debido a las interferencias causadas. Respecto al *throughput,* se observa que disminuye ligeramente, así que se puede concluir que pese a las interferencias causadas, el rendimiento de la comunicación no se ve muy afectado. Según se puede

observar en [22] y [23], el microondas no tiene un efecto significativo sobre el protocolo 802.15.4 en cuanto a la velocidad de transmisión.

4.3 Efecto del cifrado de datos en el *throughput*

En esta sección se quiere evaluar el impacto del cifrado de datos en el rendimiento de las comunicaciones entre el nodo A y el nodo coordinador.

Como se ha mencionado anteriormente, los dispositivos XBee permiten cifrar el *payload* de paquetes usando el algoritmo AES en modo CBC con claves de 128 bits. AES es un cifrado en bloque de 128 bits, de modo que si el mensaje que envía un dispositivo XBee tiene un tamaño superior a 128 bits, éste debe fragmentarse en diferentes bloques. El mensaje cifrado se obtiene concatenando el cifrado de todos los bloques, cada uno de los cuales tiene una longitud de 16 bytes.

En el caso en que el bloque de entrada sea inferior a 128 bits, se añaden bits de relleno o *padding,* y se obtiene el cifrado de dicho bloque, nuevamente de 128 bits. Por lo tanto, el cifrado añade un cierto *overhead* en cuanto a transmisión de bits extra en aquellos casos en los que se desea enviar unos pocos bits de información.

El tamaño máximo de los datos enviados por los dispositivos XBee es de 100 bytes, de modo que es necesario cifrar 7 bloques que dan lugar a un mensaje cifrado de 112 bytes. Esto hará que cuando se envían 100 bytes cifrados se deba hacer en 2 paquetes: uno de ellos contendrá 96 bytes de datos (6 bloques x 16 bytes) y el otro 16 bytes (último bloque). De esta forma, por cada mensaje de 100 bytes transmitidos se estarán enviando en realidad 142 bytes divididos en 2 paquetes (111 bytes el primero y 31 bytes el segundo).

El cifrado de los datos se debe activar mediante el software X-CTU, y se debe añadir una clave de 16 bytes. Todos los nodos de la red deben compartir la misma clave, que se usará para cifrar y descifrar mensajes.

4.3.1 *Throughput* de la transmisión cifrada con espera

En esta prueba se repite el mismo proceso seguido en la primera pero cifrando la comunicación entre el nodo A y el nodo B.

Una vez configurados los dispositivos y realizadas 250 transmisiones, se obtienen estos resultados:

Tabla 4.10 *Throughput* de la transmisión cifrada con espera

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
11.593	84,4	-33,7	-84

Si se comparan los resultados obtenidos en la tabla superior con los obtenidos en la primera prueba, se puede observar que el *throughput* obtenido es ligeramente menor ya que por cada paquete con el mismo identificador se envían los mismos datos útiles pero añadiendo un *overhead* propio del cifrado de los datos. Además durante el transcurso de la transmisión los XBees realizan internamente operaciones de cifrado lo cual añade un *delay*.
Las condiciones del canal siguen siendo buenas.

4.3.2 Throughput de la transmisión cifrada sin espera

Esta otra prueba se realiza con las mismas condiciones que la segunda prueba pero cifrando los datos enviados, con lo cual se transmiten 142 bytes por cada paquete con el mismo identificador en lugar de 115 bytes como se ha explicado en la anteriormente.

Con 250 transmisiones, se han obtenido los siguientes resultados:

Tabla 4.11 *Throughput* de la transmisión cifrada sin espera

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
29.433	635,61	-33,968	-84

Como era de esperar al ver los resultados de la prueba anterior, en este caso también se obtiene un *throughput* inferior al de la segunda prueba por el mismo motivo; se están enviando los mismos datos útiles en una cantidad de tiempo similar pero se está añadiendo un *overhead* a los datos y un retardo a la transmisión propio de las operaciones de cifrado realizadas por los dispositivos al tener activada la opción de transmisión cifrada.

Nuevamente, se puede observar que las condiciones del canal son favorables ya que presenta un nivel de energía mínimo y la potencia de señal recibida se mantiene estable.

4.3.3 Throughput de la transmisión cifrada con interferencias

En esta prueba se analizará el efecto de las interferencias del microondas sobre el medio al igual que en el punto 4.2.3 con la diferencia de que se cifrará la comunicación.

Los resultados obtenidos son los siguientes después de 250 transmisiones:

Tabla 4.12 *Throughput* de la transmisión cifrada con interferencias

Throughput (Kbps)	Desviación estándar (Kbps)	RSSI (dbm)	Energía del canal (dbm)
29.197	445,06	-35,8	-40,3

Después de observar los resultados obtenidos, se puede ver que a diferencia del caso de la sección 4.2.3, la potencia de señal recibida se ve menos afectada por las interferencias al realizar una trasmisión cifrada. La energía del canal se mantiene a un nivel similar al caso anterior, y el *throughput* disminuye ligeramente respecto a la prueba del apartado anterior donde se realiza la transmisión con las mismas condiciones pero sin aplicar interferencias. Nuevamente, vemos que pese a las interferencias el rendimiento no se ve afectado de forma importante.

4.4 Discusión de los resultados y análisis de seguridad

Después de realizar las distintas pruebas se ha podido observar que, tanto con cómo sin interferencias, el *throughput* obtenido es bastante similar entre sí, e independientemente de la distancia a la que se encuentren los dispositivos. Esto es debido a que, como se ha comentado previamente, los XBees no alcanzan el *throughput* teórico debido a limitaciones a la hora de procesar los datos.

Tampoco existen grandes diferencias de rendimiento entre comunicaciones cifradas y no cifradas, aunque el objetivo del cifrado es el de proteger los datos transmitidos y no el de mejorar o empeorar el rendimiento, de hecho en aplicaciones que traten con datos confidenciales siempre se debe priorizar la seguridad ante la eficiencia. Con esa idea no se han realizado pruebas de *throughput* que incluyesen la autenticación de los datos ya que de la misma forma que cuando se cifran los datos, se estaría añadiendo *overhead* a la transmisión y un *delay* propio de las operaciones realizadas por ambos nodos para calcular la MAC del mensaje, y por lo tanto esos 16 bytes de MAC no se considerarían datos útiles con lo cual el *throughput* sería inferior al obtenido en las pruebas de transmisión cifrada.

El cifrado de los datos proporciona confidencialidad en las comunicaciones, evitando que un usuario que escuche el medio inalámbrico e intercepte los mensajes enviados por el sensor pueda leer su contenido.
Por otro lado, el mecanismo de autenticación implementado evita que un atacante pueda inyectar mensajes falsos en la red.

Ambos servicios de seguridad se proporcionan mediante el algoritmo AES con claves de 128 bits en modo CBC, que es el único modo implementado en los dispositivos XBee. Para garantizar la seguridad de estos mecanismos es imprescindible cumplir con los siguientes requisitos:

- Las claves usadas para el cifrado y para la autenticación deben ser distintas.
- Se debe evitar enviar mensajes distintos cifrados o autenticados con la misma clave y el mismo IV.

El segundo requisito implica que las claves deben cambiarse cuando exista el riesgo de reutilizar un IV. El tiempo de vida de una clave dependerá del número de dispositivos que usan la misma clave y la frecuencia con la que éstos envían mensajes cifrados y/o autenticados.

La probabilidad **p** de que con *n* dispositivos se repita al menos un IV viene dada por la siguiente expresión:

p= 1 – prob(no se repita ningún IV) = 1- **q**

donde **q** es la probabilidad de que no se repita ningún IV y viene dada por la Ecuación 4.2:

$$\mathbf{q} = \frac{\frac{n!}{(n-(m_1+m_2))!}}{\frac{n!}{(n-m_1)!}\frac{n!}{(n-m_2)!}} = \frac{(n-m_1)!(n-m_2)!}{n!(n-(m_1+m_2))!}$$

Ecuación 4.2 Probabilidad de que no se repita ningún IV

Con m_1 y m_2 el número de mensajes enviados por cada uno de los dispositivos y n, el número total de posibles IVs. En este trabajo se ha considerado IVs de 128 bits, y por tanto el número total de IVs es de 2^{128}.
La ecuación puede generalizarse para una WBAN donde haya un número mayor de dispositivos.

Asumiendo que la mayor parte de los mensajes transmitidos en la WBANs son los mensajes que envía el nodo B con los datos recogidos del sensor de temperatura, y los ACKs que devuelve el nodo coordinador, podemos asumir

que ambos nodos enviarán el mismo número de mensajes. Esto permite simplificar la ecuación anterior y obtener la Ecuación 4.3.

$$\mathbf{p} = 1 - \frac{(n-m)!^2}{n!(n-2m)!}$$

Ecuación 4.3 Probabilidad de que se repita al menos un IV

Para una probabilidad de que se repita algún IV igual a p=0.6, el número total de mensajes que deben enviarse es aproximadamente 2.5e+19. Teniendo en cuenta que se envía un mensaje cada vez que el sensor toma una medida de temperatura (en las pruebas realizadas, 10s), las claves van a permanecer seguras durante 95022402415440.75 años. Por lo tanto, el esquema propuesto es totalmente seguro y no sería necesario cambiarlas.

CAPÍTULO 5. Conclusión y líneas futuras

Las redes WBAN constituyen una solución prometedora para realizar el seguimiento constante de los pacientes de forma remota. Sin necesidad de acudir a un centro médico, estas redes permiten la obtención de datos relativos a la salud de un paciente mediante un conjunto de sensores que se colocan o se implantan en el cuerpo del mismo, y que son transmitidos a un nodo coordinador, con capacidad de procesarlos y de enviarlos a través de Internet a un centro médico. Sin duda, este nuevo paradigma de monitorización de la salud puede permitir realizar diagnósticos precoces y la aplicación de tratamientos con mayor rapidez.

Sin embargo, dada la importancia de las aplicaciones de las WBANs, es necesario garantizar el buen funcionamiento de estas redes en cuanto a fiabilidad y seguridad se refiere.

El objetivo de este trabajo era implementar una WBAN robusta frente a interferencias y amenazas de seguridad haciendo uso de la tecnología IEEE 802.15.4, y evaluar sus prestaciones. Las soluciones propuestas en este trabajo para garantizar la robustez eran, por un lado, la aplicación de la tecnología de radio cognitiva y por otro, la aplicación de herramientas criptográficas.

Para llevar a cabo dicho objetivo se ha trabajado con dispositivos XBee que recogen medidas de un sensor de temperatura y que son transmitidas a un nodo coordinador, cifrando y autenticando los datos. Se han programado dichos dispositivos para periódicamente hagan un rastreo de frecuencias con el objetivo de identificar el canal óptimo para la transmisión, haciendo más robusta la red frente a interferencias. Por otro lado, se ha implementado un mecanismo de autenticación para evitar que un usuario no autorizado pueda inyectar mensajes falsos en la red, y además poder detectar si la información ha llegado íntegra o no.
Para evaluar el funcionamiento de la WBAN implementada, se ha realizado un conjunto de pruebas para medir el *throughput* bajo distintas condiciones (en función de la distancia, tamaño del paquete, *overhead* debido a cifrado de datos).

Una de las conclusiones principales de este trabajo es que los dispositivos XBee series 1 son muy limitados debido fundamentalmente a tres motivos: en primer lugar, la velocidad de transmisión de los datos es muy inferior a la que se anuncia en el *data sheet* del fabricante. Por otro lado, el único dispositivo que permite tomar medidas de energía de los canales es el nodo coordinador, impidiendo de esta forma implementar mecanismos de *sensing* cooperativo, tal y como se realiza típicamente en una red de radio cognitiva. Finalmente, estos dispositivos no implementan el estándar 802.15.4 en su totalidad. A pesar de que permiten el cifrado de datos AES, el modo de funcionamiento que usan no es el definido por el estándar. Además, no ofrecen mecanismos de autenticación.

En las pruebas realizadas se ha podido observar que el *throughput* obtenido en las comunicaciones prácticamente no varía en función de parámetros tales como la distancia o el nivel de interferencias. Tal y como se ha mencionado anteriormente, esto es debido a que el *throughput* máximo obtenido con estos dispositivos está muy por debajo del teórico debido a la limitada capacidad de estos dispositivos para procesar los datos.

Como líneas futuras, sería interesante el poder realizar las pruebas y la implementación en un escenario más real, es decir, utilizando varios nodos sensores que recolecten información más variada.
También sería interesante poder utilizar dispositivos que utilicen el protocolo IEEE 802.15.6 y que implementen los mecanismos de seguridad definidos en dicho estándar.

Respecto al impacto ambiental que puede tener este proyecto, se tiene que tener en cuenta que el uso del protocolo IEEE 802.15.4 implica un bajo consumo en los dispositivos que lo empleen ya que es un protocolo muy eficiente en términos energéticos. Además, la radio cognitiva ayudará a gestionar de una forma más efectiva el espectro de frecuencias, lo cual ya es un problema a día de hoy.
Por último, cabe destacar que un incremento del uso de las nuevas tecnologías en la medicina actual ayudaría a gestionar mejor la relación paciente-centro médico y a liberar espacio en los hospitales o ambulatorios reduciendo así diversos costes, tanto económicos como energéticos.

BIBLIOGRAFÍA

[1] Akyildiz I.F., Lee W., Vuran M. y Mohanty S. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. Computer Networks 50 (2006).

[2] Bhattacharjee S., Sengupta S. y Chatterjee M. Vulnerabilities in cognitive radio networks: A survey. Computer Communications 36 (2013).

[3] Kwak K.S., Ullah S. y Ullah N. An Overview of IEEE 802.15.6 Standard. (2011)

[4] El azhari M., Toumanari A. y Latif R. Performance Analysis of IEEE 802.15.6 and IEEE 802.15.4 for Wireless Body Sensor Networks. (2014)

[5] Somasundaram M. y Sivakumar R. Security in Wireless Body Area Networks: A survey. (2011)

[6] Ullah S. y Kwak K.S. Throughput and Delay Limits of IEEE 802.15.6. (2011)

[7] Harrison J. Study and overview on WBAN under IEEE 802.15.6 (2015)

[8] Digi. XBee/XBee-PRO S1 802.15.4 (Legacy) User Guide.
[Disponible online]:
http://www.digi.com/resources/documentation/digidocs/pdfs/90000982.pdf

[9] Rapp A. XBee API for Arduino. [Disponible online]: https://github.com/andrewrapp/xbee-arduino

[10] Rapp A. XBee API for Java. [Disponible online]: https://github.com/andrewrapp/xbee-api

[11] Ullah S., Mohaisen M. y Alnuem M. A Review of IEEE 802.15.6 MAC, PHY and Security Specifications. (2013)
[Disponible online]: http://www.hindawi.com/journals/ijdsn/2013/950704/

[12] IEEE Computer Society. IEEE Standard for Local and metropolitan area networks, Part 15.6: Wireless Body Area Networks. (2012)

[13] Ada L. Using a Temp Sensor. [Disponible online]: https://learn.adafruit.com/tmp36-temperature-sensor/using-a-temp-sensor

[14] Felisberto F., Costa N., Fdez-Riverola F. y Pereira A. Unobstructive Body Area Networks (BAN) for Efficient Movement Monitoring. (2012)

[15] WBAN Security Layer Basics as per IEEE 802.15.6 Security Specifictions. [Disponible online]: http://www.rfwireless-world.com/Tutorials/WBAN-Security-Layer.html

[16] Advanced Encryption Standard.
[Disponible online]:
https://en.wikipedia.org/wiki/Advanced_Encryption_Standard

[17] XBee device. [Disponible online]:
https://www.sparkfun.com/pages/xbee_guide. (2016)

[18] Canales Zigbee. [Disponible online]:
http://www.digi.com/wiki/developer/index.php/Channels,_Zigbee

[19] Gomez C., Oller J. y Paradells J. Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology. (2012)

[20] Lo Bello L. y Toscano E. IEEE 802.15.4 and Zigbee Overview.

[21] Horno de microondas.
[Disponible online]: https://es.wikipedia.org/wiki/Horno_de_microondas

[22] NXP. Co-existence of IEEE 802.15.4 at 2.4GHz. (2013)

[23] Sikora A., Groza V.F. Coexistence of IEEE 802.15.4 with other Systems in the 2.4GHz-ISM-Band. (2005)

[24] Landman D. Librería AES para Arduino.
[Disponible online]: https://github.com/DavyLandman/AESLib

[25] Message Authentication Code.
[Disponible online]: https://en.wikipedia.org/wiki/Message_authentication_code

ANEXOS

1.1 Código utilizado en Arduino para realizar las pruebas:

```
#include <XBee.h>
#include <SoftwareSerial.h>
#include <Time.h>

SoftwareSerial ss(3,2);
XBee xbee = XBee();
uint8_t payload[] = {'0','1','2','3','4','5','6','7','8','9','0','1','2','3','4','5','6','7','8','9',
'0','1','2','3','4','5','6','7','8','9','0','1','2','3','4','5','6','7','8','9',
'0','1','2','3','4','5','6','7','8','9','0','1','2','3','4','5','6','7','8','9',
'0','1','2','3','4','5','6','7','8','9','0','1','2','3','4','5','6','7','8','9',
'0','1','2','3','4','5','6','7','8','9','0','1','2','3','4','5','6','7','8','9'};
uint8_t option = 0;
uint8_t frame_id = 0;
XBeeAddress64 remoteAddress = XBeeAddress64(0x0013a200, 0x40dc0b20);
Tx64Request tx = Tx64Request(remoteAddress, option, payload, sizeof(payload), frame_id);
Rx16Response rx16 = Rx16Response();
double start_times[100]={};
double stop_times[100]={};
uint8_t* data = 0;
int i = 0;

void setup() {
  Serial.begin(57600);
  ss.begin(57600);
  xbee.begin(ss);
  delay(5000);
  start_time = micros();
}

void loop() {
  while(i!=99){
      sendTemp();
      i=i+1;
    if(i==99){
     calculateTxTime();
    }
  }
}


void sendTemp() {
  Serial.print("Enviando al coordinador el paquete: ");
  Serial.println(i+1);
  start_times[i+1]=micros();
  xbee.send(tx);

  xbee.readPacket(10); //5000
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
        xbee.getResponse().getRx16Response(rx16);
        data = rx16.getData();
        Serial.print("Recibida respuesta al paquete:  ");
        for(int j=0;j<rx16.getDataLength();j++){
          Serial.print(data[j]);
        }
        Serial.println("");
        stop_times[data[rx16.getDataLength()-1]] = micros();
    }
```

```
  }
}


void calculateTxTime(){
  for(int j = 1; j<=100; j++){
    Serial.print("El RTT del paquete ");
    Serial.print(j);
    Serial.print(" es: ");
    Serial.print(stop_times[j]-start_times[j]);
    Serial.print(" Tiempo start: ");
    Serial.print(start_times[j]);
    Serial.print(" y tiempo stop: ");
    Serial.println(stop_times[j]);
  }
```

1.2 Código utilizado en Java para realizar las pruebas:

```
package com.rapplogic.xbee.tfg;

import com.rapplogic.xbee.api.ApiId;
import com.rapplogic.xbee.api.XBee;
import com.rapplogic.xbee.api.XBeeAddress64;
import com.rapplogic.xbee.api.XBeeException;
import com.rapplogic.xbee.api.XBeeResponse;
import com.rapplogic.xbee.api.XBeeTimeoutException;
import com.rapplogic.xbee.api.wpan.TxRequest64;


public class App
{
        private static final int sizeOfIntInHalfBytes = 2;
        private static final int numberOfBitsInAHalfByte = 4;
        private static final int halfByte = 0x0F;
        private static final char[] hexDigits = {
        '0', '1', '2', '3', '4', '5', '6', '7',
        '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
         };
        public static String decToHex(int dec) {
                StringBuilder hexBuilder = new StringBuilder(sizeOfIntInHalfBytes);
                hexBuilder.setLength(sizeOfIntInHalfBytes);
                for (int i = sizeOfIntInHalfBytes - 1; i >= 0; --i)
                {
                  int j = dec & halfByte;
                  hexBuilder.setCharAt(i, hexDigits[j]);
                  dec >>= numberOfBitsInAHalfByte;
                }
                return hexBuilder.toString();
        }

   public static void main( String[] args ) throws XBeeException
   {

        XBee xbee = new XBee();
        xbee.open("COM3", 57600);
        int[] data ={0};
        XBeeAddress64 addr64 = new XBeeAddress64(0, 0x13, 0xa2, 0, 0x40, 0xdc, 0x0b, 0x2a);
        int j = 0;
        while(true){
                data[0]= j+1;
        TxRequest64 tx = new TxRequest64(addr64, 0, data);
                try{
                        XBeeResponse response = xbee.getResponse(5000);
```

```
                                if(j !=99 && response.getApiId()==ApiId.RX_16_RESPONSE){
                                        xbee.sendAsynchronous(tx);
                                System.out.println("Api ID " + response.getApiId());
                                System.out.println("Length: " + response.getLength().getLength());
                                System.out.println("Respuesta enviada");
                                j++;
                                }
                                int[] array = response.getProcessedPacketBytes();
                                        for (int i =0; i<array.length; i++){
                                        System.out.println(decToHex(array[i]) + "");
                                        }
                                        System.out.println("");
                        }catch (XBeeTimeoutException e){

                        }
                }

        }
}
```

1.3 Código utilizado en Arduino para la aplicación:

```
#include <XBee.h>
#include <SoftwareSerial.h>
#include <Time.h>
#include <AESLib.h>

SoftwareSerial ss(3,2);
XBee xbee = XBee();
unsigned long start_time;
uint8_t payload[18] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
uint8_t MAC[16] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};
uint8_t option = 1; //antes estaba a 0
uint8_t frame_id = 0;
int sensorPin = 0;
XBeeAddress64 remoteAddress = XBeeAddress64(0x0013a200, 0x40dc0b20);
Tx64Request tx;
Rx16Response rx16 = Rx16Response();
uint8_t* receivedData = 0;
uint8_t atCmd[] = {'C','H'};
AtCommandRequest atc = AtCommandRequest(atCmd);
AtCommandResponse atr = AtCommandResponse();
int i = 0;
int pos_iv = 15;

uint8_t key[] = {0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15};
uint8_t iv[] = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

void setup() {
  Serial.begin(57600);
  ss.begin(57600);
  xbee.begin(ss);
  delay(5000);
}

void loop() {

  int reading = analogRead(sensorPin);
  float voltage = reading * 5.0;
  voltage /= 1024.0;
  float temperatureC = (voltage - 0.5) * 100 ;
  Serial.println();
  Serial.println();
```

```
  Serial.print("Temperatura: ");
  Serial.print(temperatureC); Serial.println(" ยบC");
  int temp1 = (int)temperatureC;
  float taux = temperatureC - temp1;
  int temp2 = (int)(taux*100);
  payload[0] = (uint8_t)temp1;
  payload[1] = (uint8_t)temp2;
  MAC[0] = (uint8_t)temp1;
  MAC[1] = (uint8_t)temp2;
  int padding = 0;
  for (int j=2; j<16; j++){
     MAC[j]=(uint8_t)padding;
  }

  aes128_cbc_enc(key, iv, MAC, sizeof(MAC));
  for(int j = 2; j<18; j++){
    payload[j]=MAC[j-2];
  }

  tx = Tx64Request(remoteAddress, option, payload, sizeof(payload), frame_id);
  sendTemp();
  i++;
  if(i==16){
    pos_iv--;
    i=0;
  }

  iv[pos_iv]=(uint8_t)(iv[pos_iv]+1);
  getChannel();
  delay(10000);

}

void sendTemp() {
  Serial.print("Enviando datos de la temperatura al coordinador. ");
  xbee.send(tx);
  xbee.readPacket(5000);
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
       xbee.getResponse().getRx16Response(rx16);
       receivedData = rx16.getData();
       for(int j=0;j<rx16.getDataLength();j++){
         if(receivedData[j] == 1){
           Serial.println("Comunicacion correcta");
         }else{
           Serial.println("No se ha recibido respuesta. Posible perdida de la comunicacion");
         }
       }
    }
  }
}


void sendTemp() {
  Serial.print("Enviando datos de la temperatura al coordinador. ");
  xbee.send(tx);
  xbee.readPacket(5000);
  if (xbee.getResponse().isAvailable()) {
    if (xbee.getResponse().getApiId() == RX_16_RESPONSE) {
       xbee.getResponse().getRx16Response(rx16);
       receivedData = rx16.getData();
       for(int j=0;j<rx16.getDataLength();j++){
         if(receivedData[j] == 1){
           Serial.println("Comunicación correcta");
         }else{
           Serial.println("No se ha recibido respuesta. Posible pérdida de la
           comunicación");
```

```
        }
      }
    }
  }
}

void getChannel(){
  xbee.send(atc);
   if(xbee.readPacket(5000)){
    if(xbee.getResponse().getApiId() == AT_COMMAND_RESPONSE) {
      xbee.getResponse().getAtCommandResponse(atr);
      if(atr.isOk()){
        if(atr.getValueLength() > 0){
          Serial.print("El valor del canal actual es ");
          for(int i = 0; i < atr.getValueLength(); i++){
            Serial.print(atr.getValue()[i], HEX);
            Serial.print(" ");
          }
          Serial.print(" ");
        }
      } else {
        Serial.print("Error al obtener el canal. Estado: ");
        Serial.println(atr.getStatus(), HEX);
      }
    } else {
      Serial.print("En lugar de una respuesta AT se ha obtenido:  ");
      Serial.println(xbee.getResponse().getApiId(), HEX);
    }
  } else if (xbee.getResponse().isError()){
    Serial.print("Error leyendo el paquete. Mensaje: ");
    Serial.println(xbee.getResponse().getErrorCode());
  }else{
    Serial.print("No hay respuesta");
  }
}
```

1.4 Código utilizado en Java para la aplicación:

```
package com.rapplogic.xbee.tfg;



import com.rapplogic.xbee.api.ApiId;
import com.rapplogic.xbee.api.AtCommand;
import com.rapplogic.xbee.api.AtCommandResponse;
import com.rapplogic.xbee.api.RemoteAtRequest;
import com.rapplogic.xbee.api.RemoteAtResponse;
import com.rapplogic.xbee.api.XBee;
import com.rapplogic.xbee.api.XBeeAddress16;
import com.rapplogic.xbee.api.XBeeAddress64;
import com.rapplogic.xbee.api.XBeeException;
import com.rapplogic.xbee.api.XBeeRequest;
import com.rapplogic.xbee.api.XBeeResponse;
import com.rapplogic.xbee.api.XBeeTimeoutException;
import com.rapplogic.xbee.api.wpan.TxRequest64;

import java.io.ByteArrayOutputStream;
import java.util.ArrayList;



import javax.crypto.Cipher;
import javax.crypto.spec.IvParameterSpec;
```

```
import javax.crypto.spec.SecretKeySpec;

import org.apache.commons.codec.binary.Base64;

public class App
{

	private static XBee xbee = new XBee();
	private static XBeeAddress64 addr64 = new XBeeAddress64(0, 0x13, 0xa2, 0, 0x40, 0xdc, 0x0b,
0x2a);
	private static ArrayList<Integer> energy = new ArrayList<Integer>();
	private static int currentChannel = 16;
   private static byte[] key = {
      0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08, 0x09, 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F};
   private static byte[] iv = {
      0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00};


	 private static final int sizeOfIntInHalfBytes = 2;
	 private static final int numberOfBitsInAHalfByte = 4;
	 private static final int halfByte = 0x0F;
	 private static final char[] hexDigits = {
	    '0', '1', '2', '3', '4', '5', '6', '7',
	'8', '9', 'A', 'B', 'C', 'D', 'E', 'F'
	  };

	final protected static char[] hexArray = "0123456789ABCDEF".toCharArray();

	public static void scanChannels() throws XBeeTimeoutException, XBeeException{
		energy.clear();
		AtCommand  at = new AtCommand("ED");
		AtCommandResponse response = (AtCommandResponse) xbee.sendSynchronous(at,
5000);
		if (response.isOk()) {
			int[] array = response.getProcessedPacketBytes();
			for (int i =7; i<23; i++){
				energy.add(array[i]);
			}
		}
	}

	public static void changeChannel(int newChannel) throws XBeeTimeoutException,
XBeeException{
		System.out.println("Realizando cambio de canal");
		RemoteAtRequest ratr = new RemoteAtRequest(XBeeRequest.DEFAULT_FRAME_ID,
XBeeAddress64.BROADCAST, XBeeAddress16.BROADCAST, true, "CH", newChannel);
		RemoteAtResponse response = (RemoteAtResponse) xbee.sendSynchronous(ratr,
5000);
		System.out.println("El nuevo canal escogido es " + decToHex(newChannel));
		if (response.isOk()) {
			System.out.println("Cambio de canal realizado en el nodo sensor");
		}
		AtCommand at = new AtCommand("CH", newChannel);
		AtCommandResponse ATresponse = (AtCommandResponse)
xbee.sendSynchronous(at, 5000);
		if (ATresponse.isOk()) {
			System.out.println("Cambio de canal realizado en el coordinador");
		}
	}

  public static String decToHex(int dec) {
	    StringBuilder hexBuilder = new StringBuilder(sizeOfIntInHalfBytes);
	    hexBuilder.setLength(sizeOfIntInHalfBytes);
	    for (int i = sizeOfIntInHalfBytes - 1; i >= 0; --i)
	    {
	      int j = dec & halfByte;
	      hexBuilder.setCharAt(i, hexDigits[j]);
```

```
            dec >>= numberOfBitsInAHalfByte;
          }
          return hexBuilder.toString();
        }


private static byte[] hexStrToByteArray(String hex) {
          ByteArrayOutputStream baos = new ByteArrayOutputStream(hex.length() / 2);

          for (int i = 0; i < hex.length(); i += 2) {
             String output = hex.substring(i, i + 2);
             int decimal = Integer.parseInt(output, 16);
             baos.write(decimal);
          }
          return baos.toByteArray();
       }

 public static String decrypt(String strToDecrypt)
 {
    try
    {
       Cipher cipher = Cipher.getInstance("AES/CBC/NOPADDING");
       final SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
       cipher.init(Cipher.DECRYPT_MODE, secretKey, new IvParameterSpec(iv));
       byte[] strByteArray = hexStrToByteArray(strToDecrypt);
       byte[] result = cipher.doFinal(strByteArray);
       return bytesToHex(result);
    }
    catch (Exception e)
    {
      e.printStackTrace();

    }
    return null;
 }

 public static String encrypt(byte[] strToEncrypt)
 {
    try
    {
       Cipher cipher = Cipher.getInstance("AES/CBC/NOPADDING");
       final SecretKeySpec secretKey = new SecretKeySpec(key, "AES");
       cipher.init(Cipher.ENCRYPT_MODE, secretKey, new IvParameterSpec(iv));
       byte[] result = cipher.doFinal(strToEncrypt);
       return bytesToHex(result);
    }
    catch (Exception e)
    {
      e.printStackTrace();

    }
    return null;
 }


 public static String bytesToHex(byte[] bytes) {
          char[] hexChars = new char[bytes.length * 2];
          for ( int j = 0; j < bytes.length; j++ ) {
             int v = bytes[j] & 0xFF;
             hexChars[j * 2] = hexArray[v >>> 4];
             hexChars[j * 2 + 1] = hexArray[v & 0x0F];
          }
          return new String(hexChars);
       }

  public static void main( String[] args ) throws XBeeException
  {
```

```
		xbee.open("COM3", 57600);
		System.out.println("Dispositivo conectado");
		changeChannel(16);
		int[] ok = {1};
		int newChannel = 0;
		int tx_num = 0;
		int pos_iv = 15;
		while(true){
		boolean changeCHreq = false;
				TxRequest64 tx = new TxRequest64(addr64, 0, ok);
			try{
				XBeeResponse response = xbee.getResponse(5000);
				if(response.getApiId()==ApiId.RX_16_RESPONSE){
				String receivedMac = "";
				String calculatedMac = "";
					xbee.sendAsynchronous(tx);
					int[] array = response.getProcessedPacketBytes();
					byte[] receivedMsg = {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

					for(int i = 7; i<9; i++){
						receivedMsg[i-7]=(byte)array[i];
					}

					receivedMac = receivedMac.concat("00000000");
					calculatedMac= encrypt(receivedMsg);
					receivedMac = "";
					for(int i = 9; i<=24; i++){
						receivedMac = receivedMac.concat(decToHex(array[i]));
					}
					System.out.println("----------------------------------------------------------------------
-----\n");
					System.out.println("Autenticando mensaje...");
					System.out.println("MAC recibido: " + receivedMac);
					System.out.println("MAC calculado: " + calculatedMac);
					if(calculatedMac.equals(receivedMac)){
						System.out.println("Mensaje autenticado correctamente");
						tx_num ++;
						if(tx_num==16){
							pos_iv --;
							tx_num=0;
						}
						iv[pos_iv] = (byte)(iv[pos_iv] + 1);
						if(array.length > 24){
							if(array[8] >= 10){
								System.out.println("\nTemperatura
recibida: " + array[7] + "," + array[8]);
							}else{
								System.out.println("\nTemperatura
recibida: " + array[7] + ",0" + array[8]);
							}

							}
							scanChannels();

							int threshold = 66;
							System.out.println("\nEl canal actual es : " +
decToHex(currentChannel));
							System.out.println("Comprobando la energia del
canal actual... la energia es: -" + energy.get(currentChannel - 11) + "dbm");
							if(Integer.valueOf(energy.get(currentChannel - 11))
< threshold){
								System.out.println("Energia demasiado
alta, requiere cambio de canal.");
								changeCHreq = true;
							}

							if(changeCHreq == true){
```

```
                                                        int bestChannel = energy.get(0);
                                                        newChannel = 11;
                                                        currentChannel = newChannel;
                                                        for(int i = 1; i<energy.size(); i++){
                                                                if(energy.get(i) > bestChannel){
                                                                        bestChannel =
energy.get(i);
                                                                        newChannel = i + 11;
                                                                currentChannel = newChannel;
                                                                }
                                                        }
                                                        changeChannel(newChannel);
                                                }
                                        }
                                }
                        }catch (XBeeTimeoutException e){

                        }
                }
        }
}
```