ZIGBEE APLICADO A LA TRASNMISIÓN DE DATOS DE SENSORES BIOMÉDICOS

Memoria del proyecto de
Ingeniería Técnica en
Telecomunicaciones -especialidad
en Sistema Electrónicos-
realizado por

Iván Barneda Faudot

y dirigido por

Jordi Aguiló Llobet (Director)
Vicenç Soler Ruiz (Co-Director)

Bellaterra, 17 de Setiembre de 2008

El infrascrito, Jordi Aguiló Llobet

Profesor/a de la Escuela Técnica Superior de Ingeniería de la UAB,

CERTIFICA:

Que el trabajo al que corresponde esta memoria ha sido realizado bajo su dirección por Iván Barneda Faudot

Y para que conste firma el presente.

Firmado: .....

Bellaterra, 5 de setiembre de 2008

Agradecimientos

La realización de este trabajo no habría sido posible sin la colaboración de los miembros del grupo de Diseño de Circuitos y Sistemas Integrados (DCSI) del departamento de Microelectrónica y Sistemas Electrónicos de la Universitat Autònoma de Barcelona (UAB).

Un agradecimiento especial a Anna Meresta y Andrés Peñalver por su orientación y apoyo en el desarrollo de este proyecto.

Índice

Lista de figuras

Lista de tablas

Abreviaciones y acrónimos

ACK	Reconocimiento (*Acknowledgement*)
AP	Punto de acceso (*Access Point*)
APS	Subnivel de soporte a la aplicación (*Application Support Sublayer*)
CPU	Unidad central de procesamiento (*Central Processing Unit*)
CSMA/CA	*Carrier Sense Multiple Access with Collision Avoidance*
DSSS	*Direct Sequence Spread Spectrum*
ED	Dispositivo final (*End Device*)
FFD	Dispositivo de funcionalidad completa (*Full-Function Device*)
HR	Ritmo cardíaco (*Heart Rate*)
IDE	*IAR Embedded Workbench Integrated Development Environment*
ISM	Banda frecuencial usada para fines industriales, científicos y médicos
LLC	Control de enlace lógico (*Logical Link Control*)
LSB	Bit menos significativo (*Less Significant Bit*)
MAC	Capa de control de acceso al medio (*Medium Access Control layer*)
MCU	*Multipoint Control Unit*
MSB	Bit más significativo (*Most Significant Bit*)
MSComm	Control Personalizado Microsoft COMM
OEM	*Original Equipment Manufacturer*
OQPSK	*Offset Quadrature Phase Shift Keying*
PAN	Red de área personal (*Personal Area Network*)
PCB	Tarjeta del circuito impreso (*Printed Circuit Board*)
PHY	Capa física (*Physical layer*)
PSU	Suministro de potencia (*Power Supply*)
RE	Extensores de rango (*Range Extenders*)
RF	Radio frecuencia (*Radio Frequency*)
RFD	Dispositivo de funcionalidad reducida (*Reduced-Function Device*)
RSSI	Indicador de fuerza/potencia de señal recibida (*Received Signal Strength Indicator*)
SPI	*Serial Peripheral Interface*
SpO2	Nivel de oxígeno en sangre

TI	*Texas Instrument*
UART	*Universal Asynchronous Receiver-Transmitter*
UWB	Banda ultra-amplia (*Ultra-Wide Band*)
VB	Visual Basic
WPAN	Red de área personal inalámbrica (*Wireless Personal Area Network*)
ZC	Coordinador ZigBee (*ZigBee Coordinator*)
ZDO	Objetos de dispositivo ZigBee (*ZigBee Device Objects*)
ZED	Dispositivo final ZigBee (*ZigBee End Device*)
ZigBee	Estándar específico para redes de sensores inalámbricas de bajo consumo
ZigBee Stack	Pila de protocolos que agrupa todos los niveles y especificaciones que añade el estándar ZigBee
ZR	Router ZigBee (*ZigBee Router*)

Introducción

- **Contexto del proyecto:**

Pensado en un ámbito de tecnología médica, se pretende hacer uso de las nuevas tecnologías para fines médicos. Para ello, se debe plantear desde un principio los objetivos que se pretenden alcanzar y los beneficios que con ellos queremos obtener. Sin más dilación, hablaré ya del propósito del proyecto.

Según médicos y psicólogos, a causa del ritmo frenético de la vida moderna, las personas tienen tendencia a estresarse y posteriormente a caer en depresión. El ideal que se pretende alcanzar es el de establecer unos límites, unos valores máximos, que al ser sobrepasados se detecte que el individuo que está siendo observado empieza a padecer estrés.

El método que nos permita generar los baremos de estrés será aquel que nos consiga realizar un conjunto de medidas de diferentes variables del cuerpo en un paciente. Estas medidas deberán ser realizadas, en un comienzo, con el paciente totalmente relajado y posteriormente deberá ser estimulado y estresado para que sus constantes corporales se vean afectadas. De este modo, después de una larga sucesión de medidas en diferentes partes del cuerpo, pueda llegarse a unas conclusiones que determinen los valores máximos que definen que una persona aún no está estresada. Sobrepasando estos valores será cuando se dictamine que al paciente le aumentan las posibilidades de estrés.

Dejando de lado el propósito general del proyecto, seguidamente se tratará el propósito específico. Nuestro proyecto ha sido pensado para realizar la tarea de transmisión de datos desde el sensor que genera la medida hasta la monitorización y almacenamiento de las variables.

El proyecto puede desglosarse en dos grandes bloques: uno de los bloques consistiría en la elección y utilización de una gran gama de sensores para realizar diferentes medidas corporales, y el otro bloque sería aquel en que el objetivo principal es el de la transmisión de los datos del sensor hacia una base. Hacer constancia desde un principio que todo el trabajo relacionado con el estudio y manipulación de sensores queda fuera de este proyecto.

Definida la parte que se pretende trabajar, mencionar el porqué de la utilización de un medio inalámbrico. Si lo que se pretende es hacer un seguido de medidas de diferentes variables del cuerpo para establecer unos niveles máximos que al ser

superados nos notifique que el individuo se está estresando, la utilización de cableado para la transmisión de datos puede hacer alterar las medidas a causa del agobio que pueda sufrir el paciente en el momento de la medición. Por este principal motivo es por el cual se ha decidido utilizar un medio inalámbrico. El porqué de la utilización del medio inalámbrico ZigBee ya se argumentará más adelante, ya que ésta es una de las tareas principales del proyecto.

Detallado cual es el propósito del proyecto, descrita la parte que se trabajará y aquella que queda fuera, y asimilado el contexto en el que se ubica, ya solo queda valorar que éste es el inicio de un proyecto a largo plazo y que, por lo tanto, todos aquellos resultados que se obtengan serán beneficiosos para llegar a conclusiones y para seguir trabajando en ellas.

- **Objetivos del proyecto:**

Se ha creído oportuno detallar de forma concisa los objetivos que se pretenden conseguir con el trabajo de este proyecto para que, de este modo, queden claros desde un principio.

Como ya se ha argumentado anteriormente, nuestro proyecto abarcará únicamente el proceso de transmisión de datos. No se trabajará la manipulación de los sensores. Puesto que se trabajará con todo lo relacionado con la transferencia de datos, el principal objetivo que se pretende alcanzar es el de conseguir un correcto envío y recepción de datos desde un emisor hacia un receptor, y en nuestro caso de forma inalámbrica. Consecuentemente deberán realizarse diferentes tareas con el fin de cumplir un seguido de objetivos secundarios para que se pueda llegar a una correcta implementación del proyecto.

Uno de estos objetivos secundarios será el de una adecuada elección del medio inalámbrico. Para ello se ha realizado un pertinente estudio del medio a utilizar para poder llegar a la conclusión de que el mejor medio es el estudiado. Al igual que la elección del medio es de gran importancia en nuestro proyecto, la elección de los dispositivos que se usarán también requiere de su importancia. De esta elección se extrae otro objetivo que se pretende solventar.

Finalmente, y una vez cumplidos los objetivos previos, sólo quedará la implementación y puesta en marcha del proyecto.

- **Estructura de la memoria:**

La memoria está constituida por tres grandes bloques: estudio del medio de transmisión que se utilizará, diseño y análisis de los módulos que se harán servir, e implementación del proyecto.

En un principio se ha estudiado y buscado información acerca del medio inalámbrico ZigBee. La importancia de este bloque es la de saber el porqué de la utilización de ZigBee y no la de cualquier otro medio inalámbrico. Para ello nos hemos informado de en qué consiste ZigBee, cuales son las prestaciones que puede aportar al proyecto y, por contra, cuales son las limitaciones de éste. Finalmente se ha realizado una comparativa con distintos medios inalámbricos para cerciorarnos de que la mejor opción es la utilización de ZigBee.

Seguidamente se ha pasado al diseño del proyecto. El trabajo a desempeñar en este bloque es el de la búsqueda y análisis del mejor módulo compatible con ZigBee que se encuentre en el mercado. Se han consultado distintos fabricantes con sus respectivos dispositivos, pero finalmente se ha decidido que la mejor opción es la utilización de los módulos de la casa Texas Instrument.

Una vez estudiado el medio de transmisión y encontrados los dispositivos que se utilizarán, ya solo queda implementar el proyecto. Ésta es la finalidad del tercer y último bloque de la memoria. En este tercer bloque podemos encontrar tanto la configuración y programación de los dispositivos como la realización de una placa adaptadora entre el sensor biomédico y nuestro dispositivo emisor.

Finalmente, y como síntesis de todo lo que se ha trabajado a lo largo del proyecto, se detallan las conclusiones a las que se han llegado una vez finalizado el proyecto planteado desde un principio. Mencionar que no solo se detallan las conclusiones finales, sino que también una propuesta de posibles líneas futuras.

1 ZigBee

ZigBee constituye una de las grandes partes de nuestro proyecto puesto que será el medio de transmisión elegido para el envío de nuestros datos, una vez medidos con los sensores. Como veremos más adelante, ZigBee puede ser usado para una gran variedad de aplicaciones, como podemos observar en la Figura 1, pero en nuestro caso nos interesa su utilización para fines de monitorización corporal.

Figura 1. Aplicaciones de ZigBee [1]

En este apartado se tratará de forma amplia y específica el funcionamiento de este tipo de conexión, partiendo de una definición de ZigBee y posteriormente estudiando tanto sus principales características como su funcionamiento en una red ZigBee.

Otro punto a tener en cuenta es el porqué de su elección teniendo otros tipos de posibilidades. Para ello se realizará una comparación con otros medios de transmisión muy parejos a ZigBee. A partir de dicha comparativa podremos entender y asimilar que un medio ZigBee para este tipo de proyecto es lo más adecuado.

1.1 ¿Qué es ZigBee?

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica. Esta especificación define una solución para comunicaciones inalámbricas de bajo coste y consumo. El principal objetivo que pretende satisfacer una red de comunicación ZigBee es la de comunicar aplicaciones que requieren comunicaciones seguras, con baja tasa de envío y maximización de la vida útil de sus baterías. La red, en su conjunto, utilizará una cantidad muy pequeña de

energía de forma que cada dispositivo individual pueda tener una autonomía de hasta 5 años antes de necesitar un recambio en su sistema de alimentación.

La ZigBee Alliance [2] es el grupo encargado de su desarrollo. La primera versión 1.0 fue aprobada el 14 de diciembre de 2004. En diciembre de 2006 se aprobó el protocolo ZigBee 2006, y actualmente se está trabajando en nuevas versiones.

El medio de transmisión ZigBee trabaja sobre la banda ISM para usos industriales, científicos y médicos; en concreto, 868MHz en Europa, 915MHz en Estados Unidos y 2.4GHz en todo el mundo. Al ser éste último libre en todo el mundo, las empresas optan por esta opción a la hora de diseñar. En el rango de frecuencias de 2.4GHz se definen hasta 16 canales, cada uno de ellos con un ancho de banda de 5MHz.

La pila de protocolos ZigBee, también conocida como ZigBee Stack, se basa en el nivel físico (PHY) y el control de acceso al medio (MAC) definidos en el estándar IEEE 802.15.4, que desarrolla estos niveles para redes inalámbricas de área personal de baja tasa de transferencia (LR-WPAN, *Low Rate - Wireless Personal Area Network*). La especificación ZigBee completa este estándar añadiendo cuatro componentes principales:

- Nivel de red.
- Nivel de aplicación.
- Objetos de dispositivo ZigBee (ZDO, *ZigBee Device Objects*).
- Objetos de aplicación definidos por el fabricante.

Además de añadir dos capas de alto nivel (nivel de red y de aplicación) a la pila de protocolos, el principal cambio es la adición de los ZDO ya que son los responsables de llevar a cabo una serie de cometidos, entre los que se encuentran el mantenimiento de los roles de los dispositivos, la gestión de peticiones de unión a una red, el descubrimiento de otros dispositivos y la seguridad. Dichos ZDO’s los podemos ver definidos en el apartado 1.4.

También hacer referencia a los objetos de aplicación definidos por el fabricante puesto que permiten la personalización y adaptación, y favorecen la integración total.

Dicha ZigBee Stack será un elemento a tener en cuenta más adelante puesto que los dispositivos que se usen para implementar nuestro proyecto deberán ser totalmente compatibles con ZigBee y de este modo poder programar la pila de protocolos.

En la Figura 2 vemos de forma esquematizada los distintos niveles del estándar 802.15.4 y de la especificación ZigBee.

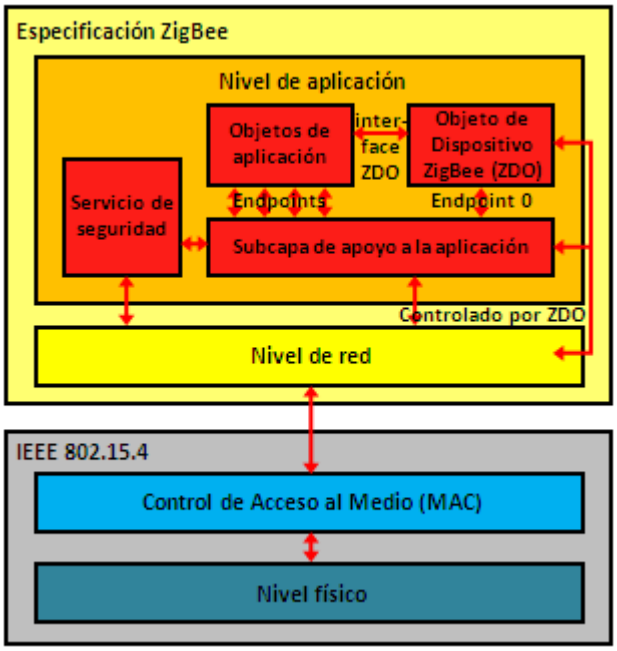

Figura 2. Pila de protocolos (ZigBee Stack)

1.2 IEEE 802.15.4

Antes de estudiar la especificación ZigBee, trataremos los niveles definidos por el estándar IEEE 802.15.4 específico para conexiones inalámbricas tales como la nuestra.

El estándar no define niveles superiores ni subcapas de interoperabilidad. Existen extensiones, como la especificación ZigBee, que complementan al estándar en la propuesta de soluciones completas.

IEEE 802.15.4[1] es la base sobre la que se define la especificación de ZigBee, cuyo propósito es ofrecer una solución completa para este tipo de redes construyendo los niveles superiores de la pila de protocolos que el estándar no cubre.

El propósito del estándar es definir los niveles de red básicos para dar servicio a un tipo específico de red inalámbrica de área personal (WPAN) centrada en la habilitación de comunicación entre dispositivos con bajo coste y velocidad. Se enfatiza el bajo coste de comunicación con nodos cercanos y sin infraestructura, o con muy poca, para favorecer aún más el bajo consumo.

En su forma básica se concibe un área de comunicación de 10 metros con una tasa de transferencia de 250kbps. Como se ha indicado, la característica fundamental de 802.15.4 entre las WPAN's es la obtención de costes de fabricación excepcionalmente

[1] El grupo de trabajo IEEE 802.15 es el responsable de su desarrollo.

bajos por medio de la sencillez tecnológica, sin perjuicio de la generalidad o la adaptabilidad.

Entre los aspectos más importantes se encuentra la adecuación de su uso para tiempo real por medio de *slots*[2] de tiempo garantizados, evitación de colisiones por CSMA/CA y soporte integrado a las comunicaciones seguras. También se incluyen funciones de control de consumo de energía como calidad del enlace y detección de energía.

1.2.1 Arquitectura de los protocolos

La definición de los distintos niveles se basa en el modelo OSI (Figura 3). Aunque los niveles inferiores (físico y enlace de datos) se definen en el estándar 802.15.4, se prevé la interacción con el resto de niveles por medio de un subnivel de Control de Enlace Lógico basado en IEEE 802.2 (LLC, *Logical Link Control*), que acceda al Control de Acceso al Medio (MAC) a través de un subnivel de convergencia. La implementación puede basarse en dispositivos externos o integrarlo todo en dispositivos autónomos.

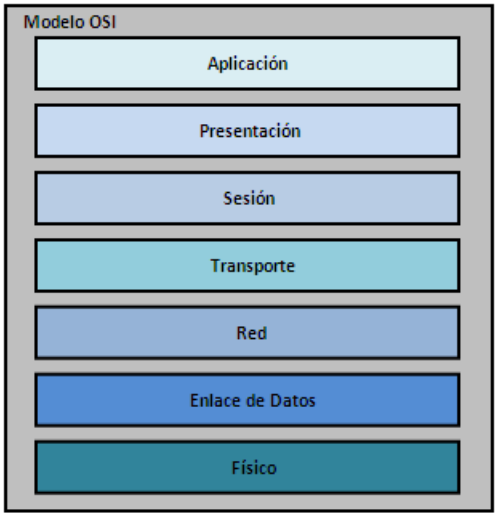

Figura 3. Pila de protocolos del modelo OSI

1.2.1.1 Nivel físico

El nivel físico (PHY) proporciona el servicio de transmisión de datos sobre el medio físico propiamente dicho, así como la interfaz con la entidad de gestión del nivel físico, por medio de la cual se puede acceder a todos los servicios de gestión del nivel y que mantiene una base de datos con información de redes de área personal relacionadas. De

[2] Utilización de balizas. Véase capítulo 1.6 .

esta forma, PHY controla el transceptor de radiofrecuencia y realiza la selección de canales junto con el control de consumo y de la señal.

Como ya se ha comentado con anterioridad, se opera en una de las tres posibles bandas de frecuencia siguientes:

- 868-868.8MHz: Europa. Permite de uno a tres canales.
- 902-928MHz: Norte América. Permite de diez a treinta canales.
- 2400-2483.5MHz: Uso en todo el mundo. Permite hasta dieciséis canales.

La versión original del estándar (2003) especifica dos niveles físicos basados en Espectro Ensanchado por Secuencia Directa (DSSS, *Direct Sequence Spread Spectrum*). Uno de los dos niveles físicos trabaja en las bandas de 868/915MHz con tasas de transferencia de entre 20 y 40kbps; el otro nivel trabaja en la banda de 2450MHz con hasta 250kbps. Posteriormente se realizó una revisión en el 2006 incrementando las tasas de datos máximas de las bandas de 868/915MHz, pasando a transmitir hasta 100 y 250kbps.

En el nivel físico podemos localizar cuatro niveles físicos distintos en base al método de modulación usado. Tres de estos cuatro niveles conservan el mecanismo DSSS: las bandas de 868-915MHz que usan modulación en fase binaria o por cuadratura en offset (OQPSK, *Offset Quadrature Phase Shift Keying*). En la banda de 2450MHz se usa la técnica OQPSK.

Adicionalmente, se define una combinación opcional de modulación binaria y en amplitud para las bandas de menores frecuencias, basadas por lo tanto en una difusión de espectro paralela, no secuencial (PSSS). Si se usan éstas bandas de menor frecuencia, se puede cambiar dinámicamente el nivel físico usado entre los soportados.

1.2.1.2 Nivel enlace de datos

En este nivel encontramos el Control de Acceso al Medio (MAC). Éste transmite tramas MAC usando para ello el canal físico. Además del servicio de datos, ofrece un interfaz de control y regula el acceso al canal físico y al balizado de la red. También controla la validación de las tramas y las asociaciones entre nodos, y garantiza *slots* de tiempo. Por último, ofrece puntos de enganche para servicios seguros.

En el nivel de enlace de datos podemos localizar el Control de Enlace Lógico (LLC) que hace la función de interfaz con los niveles superiores de la pila de protocolos. En la Figura 4 podemos observar los niveles definidos por el estándar 802.15.4 y las capas

intermedias que posibilitan la comunicación con los niveles superiores definidos por un tipo de especificación WPAN, como ZigBee.

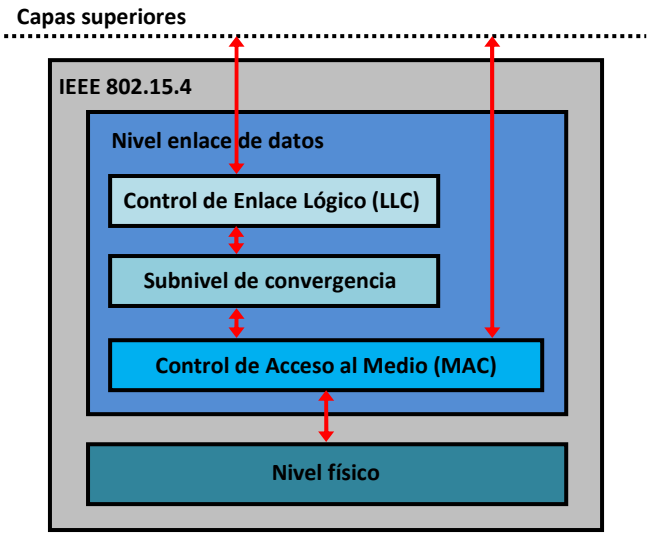

Figura 4. Pila de protocolos IEEE 802.15.4

1.3 Especificación ZigBee

1.3.1 Nivel de red

Los cometidos principales del nivel de red son permitir el correcto uso del subnivel MAC, definido por el estándar 802.15.4, y ofrecer un interfaz adecuado para su uso por parte del nivel inmediatamente superior, el nivel de aplicación. Sus capacidades son las típicas de un nivel de red clásico.

Por una parte, la entidad de datos crea y gestiona las unidades de datos del nivel de red a partir del *payload* del nivel de aplicación y realiza el ruteo en base a la topología de la red en la que el dispositivo se encuentra. Por otra parte, las funciones de control del nivel de red controlan la configuración de nuevos dispositivos y el establecimiento de nuevas redes; puede decidir si un dispositivo colindante pertenece a la red e identifica nuevos routers y vecinos. El control puede detectar así mismo la presencia de receptores, lo que posibilita la comunicación directa y la sincronización a nivel MAC.

1.3.2 Nivel de aplicación

El nivel de aplicación es el más alto definido por la especificación y, por tanto, la interfaz efectiva entre el nodo ZigBee y sus usuarios. En él se ubican la mayor parte de

los componentes definidos por la especificación: tanto ZDO's, como sus procedimientos de control y los objetos de aplicación.

1.3.2.1 Componentes principales

El ZDO se encarga de la definición del rol de un dispositivo como Coordinador ZigBee o Dispositivo Final (véase capítulo 1.4). Además, el ZDO identifica los dispositivos que se encuentran a un salto en la red (dispositivos vecinos) y los servicios que ofrecen. Tras ello, puede proceder a establecer enlaces seguros con dispositivos externos y responder peticiones.

El subnivel de soporte a la aplicación (APS, *Application Support Sublayer*) es el segundo componente básico del nivel. Como tal, ofrece una interfaz bien definida y servicios de control asociados. Trabaja como nexo de unión entre el nivel de red y el resto de componentes del nivel de aplicación. Mantiene actualizadas las tablas de asociaciones en forma de base de datos, que puede utilizarse para encontrar dispositivos adecuados en base a los servicios demandados y ofrecidos. Como puente entre dos niveles, encamina los mensajes a lo largo de la pila de protocolos.

1.4 Dispositivos que constituyen ZigBee

En una red ZigBee podemos encontrar y detectar tres tipos de dispositivos ZigBee diferentes, según el papel que desarrollen en nuestra red. Estos dispositivos son los siguientes:

- Coordinador ZigBee (*ZigBee Coordinator*, ZC): Consiste en el dispositivo más completo de los tres, puesto que sus funciones son las de controlar y coordinar la red y los caminos que deben seguir los dispositivos para conectarse entre ellos. Debemos encontrar obligatoriamente un ZC en cada red ZigBee.
- Router ZigBee (*ZigBee Router*, ZR): Su función es la de interconectar los dispositivos separados en la topología de la red, además de ofrecer un nivel de aplicación para la ejecución de código de usuario.
- Dispositivo Final (*ZigBee End Device*, ZED): En este dispositivo queda representado las principales características de ZigBee, como son el bajo consumo y el bajo coste. Los ZED poseen la funcionalidad necesaria para comunicarse con su nodo padre, que ya puede ser el Router ZigBee o el Coordinador ZigBee, pero no puede transmitir información destinada a otros

dispositivos. Es por ello, que este tipo de dispositivo puede estar "dormido" la mayor parte del tiempo aumentando así la vida media de sus baterías. Un ZED tiene requerimientos mínimos de memoria y es por ello significativamente más barato.

Vistas las funciones realizadas por los distintos tipos de dispositivos que forman una red ZigBee podemos clasificarlos, según su funcionalidad, en dos tipos de nodo definidos por el estándar 802.15.4:

- Dispositivo de Funcionalidad Completa (FFD, *Full-Function Device*): Conocido también como nodo activo. Gracias a la memoria adicional y a la capacidad de computar puede funcionar como Coordinador o Router ZigBee de una red de área personal (PAN) o como un nodo normal. Implementa un modelo general de comunicación que le permite establecer un intercambio con cualquier otro dispositivo pudiendo encaminar mensajes, en cuyo caso se le denomina coordinador (coordinador de la PAN si es el responsable de toda la red y no sólo de su entorno). Puede ser usado en dispositivos de red que actúen de interface con los usuarios.
- Dispositivo de Funcionalidad Reducida (RFD, *Reduced-Function Device*): Conocido también como nodo pasivo. Posee una capacidad y funcionalidad limitada para garantizar un bajo coste y una gran simplicidad, por ello sólo pueden comunicarse con FFD's y nunca pueden ser coordinadores. Básicamente constituyen los sensores de la red.

1.5 Comunicación y descubrimiento de dispositivos

Para que los dispositivos que forman una aplicación puedan comunicarse, deben utilizar un protocolo de aplicación compartido. Estos acuerdos se agrupan en perfiles. En el momento de establecer una comunicación entre los distintos dispositivos se realiza mediante pares de identificadores fuente y destino (identificadores de *cluster*), agrupando las parejas en tablas de asociaciones. Dichas tablas estarán correctamente almacenadas en los Coordinadores ZigBee.

En base a la información disponible, el descubrimiento de dispositivos puede efectuarse utilizando varios métodos. En caso de conocer la dirección de red, se pide la

dirección IEEE utilizando *unicast*[3]. Si no es así, se pide por *broadcast*[4]. Los Dispositivos Finales ZigBee responden a estas peticiones con sus direcciones propias, mientras que los Routers y Coordinadores ZigBee envían también las direcciones de todos los dispositivos asociados a ellos.

Los identificadores de *cluster* favorecen la asociación entre entidades complementarias por medio de tablas de asociación, mantenidas en los Coordinadores ZigBee, ya que estas tablas siempre han de estar disponibles en una red. Los Coordinadores son los encargados de guardar las tablas ya que, de entre todos los nodos, son los que con mayor seguridad dispondrán de una alimentación continua. Para establecer asociaciones entre diferentes dispositivos se necesita que se haya formado un enlace de comunicación, tras ello se decide si es necesario adjuntar un nuevo nodo a la red en base a la aplicación y las políticas de seguridad.

Nada más establecerse la asociación, pueden iniciarse las comunicaciones.

Existen dos modos de direccionamientos. Por una parte, el direccionamiento directo utiliza la dirección de radio y el número de *endpoint*[5]; por otra parte, el direccionamiento indirecto necesita toda la información relevante (dirección, *endpoint*, *cluster* y atributo), y la envía al Coordinador de la red. El Coordinador es el encargado de traducir su petición y proporcionarle los datos deseados. Este último direccionamiento es especialmente útil para favorecer el uso de dispositivos muy sencillos y minimizar el almacenamiento interno.

1.6 Modos de funcionamiento de ZigBee

El funcionamiento de ZigBee debe cumplir la premisa del bajo consumo de sus nodos. Para ello un nodo ZigBee, tanto activo como pasivo, reduce su consumo gracias a que puede permanecer “dormido” la mayor parte del tiempo, incluso muchos días seguidos. Cuando decimos que el nodo permanece “dormido” nos referimos a que está a la espera de ser activado por parte del Router o del Coordinador ZigBee.

[3] Envío de información desde un único emisor a un único receptor.

[4] Envío de información a todas las estaciones de la red. Envío por difusión.

[5] Identificador de los distintos nodos de una red.

Cuando se requiere su uso, el nodo ZigBee es capaz de despertar en un tiempo ínfimo, para volverse a "dormir" cuando deje de ser requerido. El tiempo que tarda un nodo cualquiera en despertarse es de aproximadamente 15ms.

Las redes ZigBee han sido diseñadas para conservar la potencia en los nodos esclavos (ZED), de esta forma se consigue el bajo consumo de potencia. La idea del funcionamiento ZigBee consiste en que los dispositivos esclavos en todo momento permanecen en modo "dormido" a no ser que sean activados, de tal forma que solo se "despiertan" por una fracción de segundo para confirmar que siguen en nuestra red de dispositivos de la que forman parte, es decir, que siguen "vivos".

En las redes ZigBee, se pueden usar dos modos de funcionamiento diferentes: con balizas o sin balizas.

1.6.1 Con balizas

En este modelo de funcionamiento, el camino de transmisión y recepción está permanentemente controlado por un distribuidor que se encarga de controlar el canal y dirigir las transmisiones. El distribuidor permite a todos los dispositivos saber cuándo pueden transmitir.

Para el control del canal se utilizan las balizas, elementos que se usan para poder sincronizar todos los dispositivos que conforman la red. Los intervalos de las balizas son asignados por el coordinador de la red (Coordinador ZigBee) y pueden variar desde los 15ms hasta los 4 minutos.

Este modo es más recomendable cuando el coordinador de red trabaja con una batería. Los dispositivos que conforman la red escuchan a dicho coordinador durante el balizamiento (envío de mensajes a todos los dispositivos -*broadcast*-, entre 0.015 y 252 segundos). Un dispositivo que quiere intervenir, lo primero que tendrá que hacer es registrarse para el coordinador, y es entonces cuando mira si hay mensajes para él. En el caso de que no haya mensajes, este dispositivo vuelve a "dormir", y se despierta de acuerdo a un horario que ha establecido previamente el coordinador. En cuanto el coordinador termina el balizamiento, todos los dispositivos de la red vuelven a "dormirse".

Como vemos, se trata de un mecanismo de control del consumo de potencia en la red.

1.6.2 Sin balizas

En este tipo cada dispositivo es autónomo, pudiendo iniciar una conversación en la cual los otros dispositivos pueden interferir. A veces puede ocurrir que el nodo destino puede no oír la petición o que el dispositivo emisor pretenda transmitir cuando el canal esté ocupado, ocasionando posibles colisiones. Es por ello que se debe utilizar un mecanismo de control de acceso al medio. Las redes sin balizas acceden al canal por medio del CSMA/CA. El CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*) se basa en la escucha del canal por parte del nodo que pretende transmitir, y de esta forma detecta si algún otro nodo que compone la red está transmitiendo o tiene intención de ello. Al tratarse de un medio inalámbrico, la detección de colisiones no es posible, por lo tanto para evitar dichas colisiones la estación que quiera transmitir, si no hay tráfico en el canal, podrá hacerlo pasado un cierto instante de tiempo; en caso de existir tráfico, deberá esperar un cierto tiempo de espera más otro cierto intervalo de tiempo aleatorio.

Este sistema se usa típicamente en los sistemas de seguridad, en los cuales sus dispositivos duermen prácticamente todo el tiempo. Para que se les tenga en cuenta, estos elementos se “despiertan” de forma regular para anunciar que siguen en la red. Cuando se produce un evento, como por ejemplo que un sensor detecte algún movimiento, el sensor “despierta” instantáneamente y transmite a la alarma correspondiente. Es en ese momento cuando el coordinador de red recibe el mensaje enviado por el sensor y activa la alarma pertinente. En este caso, el coordinador de red se alimenta de la red principal durante todo el tiempo.

Los *routers* suelen estar activos todo el tiempo, por lo que requieren una alimentación estable en general. Esto permite redes heterogéneas en las que algunos dispositivos pueden estar transmitiendo todo el tiempo, mientras que otros sólo transmiten ante la presencia de estímulos externos.

En general, los protocolos ZigBee que no hacen uso de las baliza, minimizan el tiempo de actividad para evitar el uso de energía. En las redes con balizas, los nodos sólo necesitan estar despiertos mientras se transmiten las balizas (además de cuando se les asigna tiempo para transmitir); si no hay balizas, el consumo de cada dispositivo será distinto ya que encontraremos nodos activos permanentemente y otros que sólo lo están esporádicamente.

1.6.3 Acceso al medio

El medio físico es un recurso al que se accede utilizando CSMA/CA. Las redes que no utilizan las balizas hacen uso de una variación del mismo basada en la escucha del medio, temporizada por un algoritmo de *backoff*[6], salvo en el caso de las confirmaciones (ACK, *Acknowledgement*).

Estos mensajes de confirmación pueden ser opcionales en algunos casos. La recepción de una confirmación certifica el éxito de nuestro envío. En cualquier caso, si un dispositivo es incapaz de procesar una trama en un momento dado, no confirma su recepción. Pueden realizarse reintentos basados en *timeout* un cierto número de veces, tras lo cual se decide si seguir intentándolo a dar error de transmisión.

El entorno de funcionamiento previsto para este tipo de redes exige que se maximice la vida de la fuente de energía, por lo que se favorecen los protocolos que conducen a estos fines. Para ello, se programan comprobaciones periódicas de mensajes pendientes, más o menos frecuentes según la aplicación concreta.

En lo que respecta a la seguridad en las comunicaciones, el subnivel MAC ofrece funcionalidades que los niveles superiores pueden utilizar para lograr alcanzar el nivel de seguridad deseado. Estos niveles superiores pueden especificar claves simétricas para proteger los datos y restringir éstos a un grupo de dispositivos o a un enlace punto a punto. Estos grupos se especifican en listas de control de acceso. Además, MAC realiza comprobaciones de frescura (*freshness check*) entre recepciones sucesivas para asegurar que las tramas viejas cuyo contenido no se considera útil o válido ya, no transcienden a los niveles superiores.

1.7 Servicios de fiabilidad y seguridad

Uno de los aspectos más característicos de ZigBee son los servicios que ofrece para el soporte de comunicaciones seguras. Se protege el establecimiento y transporte de claves, el cifrado de trama y el control de dispositivos. La seguridad depende de la correcta gestión de las claves simétricas y de la adecuada implementación de los métodos y políticas de seguridad.

[6] Tiempo de espera exponencial aleatorio.

1.7.1 Modelo básico de seguridad

La piedra angular de la confidencialidad es la adecuada protección de todo el material de cifrado. Debe asumirse que se confía en la instalación inicial de las claves, así como en el procesado de la información de seguridad. Para que la implementación funcione en su conjunto, se asume su corrección.

Las claves son la base de la arquitectura de seguridad y, como tal, su protección es fundamental para la integridad del sistema. Las claves nunca deben transportarse utilizando un canal inseguro, exceptuando la conexión momentánea de la fase inicial que consiste en la unión de un dispositivo desconfigurado a una red. La red ZigBee debe tener gran cuidado con este aspecto, puesto que una red *ad hoc* puede ser accesible físicamente a cualquier dispositivo externo, y el entorno de trabajo no se puede conocer de antemano. Las aplicaciones que se ejecutan utilizando el mismo transceptor deben, así mismo, confiar entre sí, ya que por motivos de coste no se asume la existencia de un cortafuegos entre las distintas entidades del nivel de aplicación.

Los distintos niveles definidos dentro de la pila de protocolos no están separados criptográficamente, por lo que se necesitan políticas de acceso, que se asumen correctas en su diseño. Este modelo de confianza abierta (*open trust*) posibilita la compartición de claves, disminuyendo de este modo el coste de forma significativa. No obstante, el nivel que genera una trama es siempre el responsable de su seguridad. Todos los datos de las tramas del nivel de red han de estar cifrados, ya que podría haber dispositivos maliciosos, de forma que el tráfico no autorizado se previene de raíz. De nuevo, la excepción es la transmisión de la clave de red a un dispositivo nuevo, lo que dota a toda la red de un nivel de seguridad único. También es posible utilizar criptografía en enlaces punto a punto.

1.7.2 Arquitectura de seguridad

ZigBee utiliza claves de 128 bits en sus mecanismos de seguridad. Una clave puede asociarse a una red (utilizable por los niveles de ZigBee y el subnivel MAC) o a un enlace (en tal caso, adquirida por preinstalación, acuerdo o transporte). Las claves de enlace se establecen en base a una clave maestra que controla la correspondencia entre claves de enlace. Como mínimo la clave maestra inicial debe obtenerse por medios seguros (transporte o preinstalación), ya que la seguridad de toda la red depende de ella.

Los distintos servicios usarán variaciones unidireccionales (*one-way*) de la clave de enlace para evitar riesgos de seguridad.

Es claro que la distribución de claves es una de las funciones de seguridad más importantes. Una red segura encarga a un dispositivo especial la distribución de claves: el denominado centro de confianza (*trust center*). En un caso ideal los dispositivos llevarán precargados de fábrica la dirección del centro de confianza y la clave maestra inicial. Si se permiten vulnerabilidades momentáneas, se puede realizar el transporte como se ha descrito. Las aplicaciones que no requieran un nivel especialmente alto de seguridad utilizarán una clave enviada por el centro de confianza a través del canal inseguro transitorio.

Por lo tanto, el centro de confianza controla la clave de red y la seguridad punto a punto. Un dispositivo sólo aceptará conexiones que se originen con un clave enviada por el centro de confianza, salvo en el caso de la clave maestra inicial. La arquitectura de seguridad está distribuida entre los distintos niveles de la siguiente manera:

- El subnivel MAC puede llevar a cabo comunicaciones fiables de un solo salto. En general, utiliza el nivel de seguridad indicado por los niveles superiores.
- El nivel de red gestiona el ruteo, procesando los mensajes recibidos y pudiendo hacer *broadcast* de peticiones. Las tramas salientes usarán la clave de enlace correspondiente al ruteo realizado, si está disponible; en otro caso, se usará la clave de red.
- El nivel de aplicación ofrece servicios de establecimiento de claves al ZDO y las aplicaciones, y es responsable de la difusión de los cambios que se produzcan en sus dispositivos a la red. Estos cambios podrían estar provocados por los propios dispositivos (un cambio de estado sencillo) o en el centro de confianza, que puede ordenar la eliminación de un dispositivo de la red, por ejemplo. También encamina peticiones de los dispositivos al centro de seguridad y propaga a todos los dispositivos las renovaciones de la clave de red realizadas por el centro. El ZDO mantiene las políticas de seguridad del dispositivo.

1.8 Modelo de redes de ZigBee

Las redes están compuestas por grupos de dispositivos separados por distancias suficientemente reducidas. Cada dispositivo posee un identificador único de 64 bits, aunque si se dan ciertas condiciones de entorno en éste pueden utilizarse identificadores

cortos de 16 bits. Probablemente éstos se utilizarán dentro del dominio de cada PAN separadas.

Un aspecto a tener muy en cuenta son los tipos de topologías de red que permite el estándar que soporta ZigBee. Su nivel de red permite tres topologías distintas:

- Topología en estrella.
- Topología en árbol.
- Topología de malla.

Toda red necesita al menos un dispositivo coordinador (FFD), encargado de su creación, mantenimiento básico y control de sus parámetros. Seguidamente estudiaremos de forma detallada los distintos tipos de topologías de ZigBee, dónde se sitúa en cada uno de ellos el coordinador de la red y cuál es la topología más adecuada.

1.8.1 Topología en estrella

En redes en estrella el coordinador se sitúa en el centro, y toda conexión que se quiera realizar entre los distintos nodos de la red debe pasar por éste.

En la Figura 5 vemos ilustrada la típica estructura de una red en estrella.

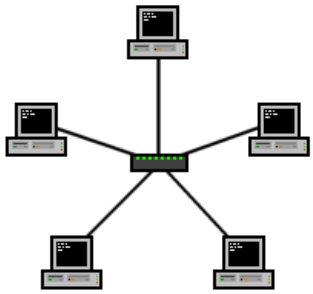

Figura 5. Estructura de una red en estrella

Una red en estrella activa tiene un nodo central activo que normalmente tiene los medios para prevenir problemas. Se utiliza sobre todo para redes locales. La mayoría de las redes de área local que tienen un *router*, un *switch* o un *hub*[7] siguen esta topología. El nodo central, en el caso de utilizar uno de estos dispositivos, sería el router, el switch o el *hub* por el que pasan todos los paquetes.

Las ventajas que nos puede aportar una red en estrella sería la facilidad a la hora de implementarla, adecuada para redes temporales, el fallo de un nodo periférico no influiría en el comportamiento del resto de la red y no hay problemas con colisiones de datos ya que cada estación tiene su propia conexión al coordinador central.

[7] Dispositivo encargado del encaminamiento de la información a través de una red de comunicación.

En contrapartida, la utilización de una red en estrella nos limita tanto el número de nodos que pueden estar conectados a la red, como la longitud del cableado (en caso de ser una conexión cableada). También se debe tener muy en cuenta que los costes de mantenimientos pueden aumentar a largo plazo, y que el fallo del nodo central puede echar abajo la red entera.

A causa de todo ello, podemos confirmar que una red en estrella puede ser poco fiable en el momento de realizar transferencias de información.

1.8.2 Topología en árbol

Topología de red en la que los nodos están colocados en forma de árbol (Figura 6). Para este tipo de topología el coordinador será la raíz del árbol. Desde una vista topológica, la conexión en árbol es parecida a una serie de redes en estrella interconectadas, salvo en que no tiene un nodo central. En cambio, tiene un nodo de enlace troncal, generalmente ocupado por un *hub* o *switch*, desde el que se ramifican los demás nodos.

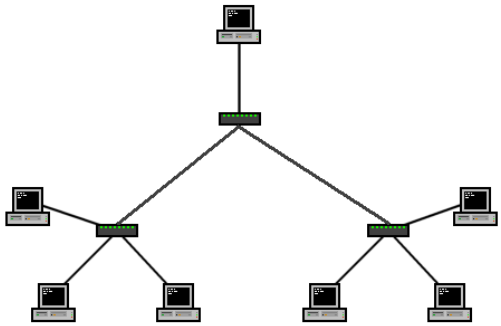

Figura 6. Estructura de una red en árbol

En redes de árbol se permite el uso de Router ZigBee para interconectar los distintos nodos de la red. Así mismo, la comunicación en árbol es estrictamente jerárquica con lo que puede utilizar opcionalmente tramas balizas (véase capítulo 1.6). Una falla de un nodo no implica la interrupción en las comunicaciones. Se comparte el mismo canal de comunicaciones.

Los problemas asociados a este tipo de topología radican en que los datos son recibidos por todas las estaciones sin importar para quien vayan dirigidos. Por lo tanto es necesario dotar a la red de un mecanismo que permita identificar al destinatario de los mensajes. Además, debido a la presencia de un medio de transmisión compartido entre muchas estaciones, pueden producirse interferencias entre las señales cuando dos o más estaciones transmiten al mismo tiempo. La solución al primero de estos

problemas aparece con la introducción de un identificador de estación destino. Para darle solución al segundo problema, hay que mantener una coordinación entre todas las estaciones, y para eso se utiliza cierta información de control en las tramas que controlan quien transmite en cada momento (utilización de tramas balizas mencionadas anteriormente).

1.8.3 Topología en malla

La topología más interesante es la topología de malla. Consiste en que al menos uno de los nodos tendrá más de dos conexiones. Con ello conseguimos que si, en un momento dado, un nodo del camino falla y se cae, pueda seguir la comunicación entre todos los demás nodos debido a que se rehacen todos los caminos.

El establecimiento de una red de malla es una manera de encaminar datos, voz e instrucciones entre los nodos. Las redes de malla se diferencian de otras redes en que las piezas de la red (los nodos) están conectadas unas con otras por uno u otro camino. Esta configuración ofrece caminos redundantes por toda la red, de modo que si falla un cable, otro se hará cargo del tráfico.

Esta topología, a diferencia de las vistas en apartados anteriores, no requiere de un servidor o nodo central, con lo que se reduce el mantenimiento. Un error en un nodo, sea importante o no, no implica la caída de toda la red.

Como ya se ha comentado anteriormente, las redes de malla son autogenerables. La red puede funcionar incluso cuando un nodo desaparece o la conexión falla, ya que el resto de nodos evitan el paso por ese punto. Consecuentemente, se forma una red muy confiable.

Gracias a las estructuras arbitrarias que permite la topología en malla (Figura 7), es posible llevar los mensajes de un nodo a otro por diferentes caminos. A consecuencia de este tipo de estructura no se pueden usar tramas balizas.

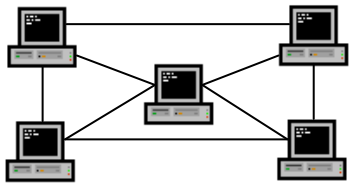

Figura 7. Estructura de una red en malla

Al igual que en la topología en árbol, las redes en malla permiten el uso de Routers ZigBee para habilitar la comunicación en el nivel de red. Éstos no son Coordinadores

ZigBee, pero pueden serlo de sus respectivos espacios de operación personal definidos por 802.15.4.

Aunque la facilidad de solución de problemas y el aumento de la fiabilidad son ventajas muy interesantes, estas redes resultan caras de instalar, ya que utilizan mucho cableado. Es por ello que su uso se centra en redes inalámbricas, como es el caso de nuestro proyecto.

1.8.4 Árbol de *clusters*

Vistos los tipos de topologías que podemos manipular, encontramos que nuestro estándar menciona un tipo de red definido como "árbol de *clusters*". Para ello se usan redes punto a punto.

Las redes punto a punto pueden formar patrones arbitrarios de conexionado, donde su extensión se ve limitada únicamente por la distancia existente entre cada par de nodos. Forman la base de redes *ad hoc*[8] autoorganizativas. El estándar no define un nivel de red, por lo que no se soportan funciones de ruteo de forma directa, aunque si ha dicho nivel se le añade se pueden realizar comunicaciones en varios saltos.

Las estructuras árbol de *clusters* (Figura 8) están formadas por el conexionado entre nodos FFD y RFD. Puesto que se necesita de al menos un nodo FFD para poder conectar diferentes RFD's, se aprovecha de ello para generar estructuras donde los nodos RFD's simbolizan las hojas de un árbol, y donde la mayoría de los nodos son FFD's. A partir de los árboles de *clusters* podemos generar grandes redes de malla, cuyos nodos sean árboles de *clusters* con un coordinador local para cada *cluster*, junto con un coordinador global.

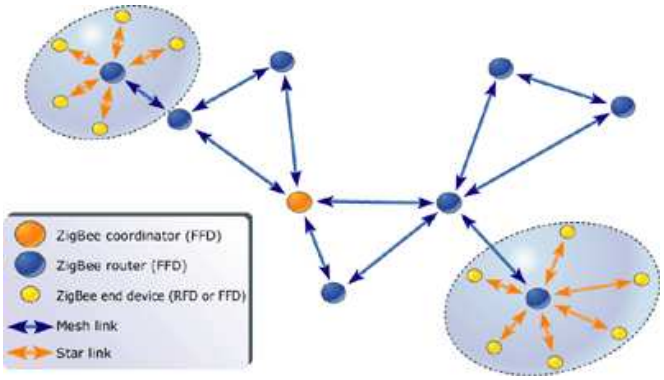

Figura 8. Árbol de *clusters* [6]

[8] Redes, especialmente inalámbricas, en las que no hay un nodo central, y todos los nodos de la red están en igualdad de condiciones.

1.9 Otros medios de transmisión de datos

1.9.1 ZigBee vs. Bluetooth

Como ya hemos podido comprobar, el desarrollo de la tecnología de ZigBee se centra en la sencillez y bajo coste de sus nodos y en el bajo nivel de transmisión de información. Es por ello que su utilización para nuestro proyecto es la más adecuada en comparación con otras redes inalámbricas semejantes de la familia WPAN, como por ejemplo Bluetooth.

ZigBee es muy similar a Bluetooth, pero con algunas diferencias:

- Una red ZigBee puede constar de más de 65000 nodos distribuidos en subredes de 255 nodos, frente a los 8 máximos de una subred (piconet[9]) Bluetooth.
- Menor consumo eléctrico que el de Bluetooth. Este menor consumo se debe a que el sistema ZigBee, como ya se ha mencionado anteriormente, se queda la mayor parte del tiempo dormido, mientras que en una comunicación Bluetooth esto no se puede dar ya que suelen estar siempre transmitiendo y/o recibiendo. En la Tabla 1 podemos ver en términos exactos el consumo de ZigBee y de Bluetooth.

	En transmisión	**En reposo**
ZigBee	30mA	3µA
Bluetooth	40mA	0.2mA

Tabla 1. Consumo ZigBee vs Bluetooth

- La velocidad ZigBee es menor a la de Bluetooth. ZigBee tiene una velocidad de hasta 250kbps, mientras que en Bluetooth es de hasta 1Mbps. Esta diferencia de valores es lógica si tenemos en cuenta que ZigBee se basa en una transmisión de datos baja, y por lo tanto no nos afecta en gran medida en el momento de transmisión y recepción de datos.

Vistas las principales diferencias entre ZigBee y Bluetooth, podemos llegar a la conclusión de que uno es más apropiado que el otro para ciertas aplicaciones. Haciendo referencia a la diferencia de velocidades, Bluetooth se usa para aplicaciones con mayor

[9] Se conoce como piconet a una red de dispositivos informáticos que se conectan utilizando Bluetooth. Un piconet puede constar de dos a ocho dispositivos, y siempre habrá un "maestro" y los demás serán "esclavos".

carga de información para transmitir, como por ejemplo para teléfonos móviles e informática casera; la velocidad de ZigBee se hace insuficiente para estas tareas, desviándolo a usos tales como la domótica, los productos dependientes de la batería, artículos de juguetería y sensores médicos, en los cuales la transferencia de datos es menor. Éste último caso de utilización de ZigBee, los sensores médicos, nos garantiza que ZigBee será un medio idóneo para nuestro proyecto que se basa en la transmisión de datos a partir de medidas capturadas por parte de un conjunto de sensores.

1.9.2 Wireless USB

Wireless USB es un protocolo de comunicación inalámbrica por radio con gran ancho de banda, que combina la sencillez de uso de USB con la versatilidad de las redes inalámbricas.

Utiliza como banda frecuencial la plataforma UWB (*Ultra-Wide Band*), operando en los rangos de frecuencia de 3.1 a 10.6GHz. Puede lograr tasas de transmisión de hasta 480Mbps en rangos de tres metros y 110Mbps en rangos de diez metros.

Gracias a su alta tasa de transferencia, Wireless puede usarse para aplicaciones que requieren un flujo de transferencia elevado. Wireless USB se utiliza en mandos de videoconsola, impresoras, escáneres, transmisión y visualización de vídeos, etc.

Haciendo una comparativa de los principales aspectos de Wireless USB, Bluetooth, Wi-Fi y ZigBee, como podemos comprobar en la Tabla 2, el rango de frecuencia en el que trabaja Wireless es superior al de los otros medios de transmisión, al mismo tiempo que su tasa de transferencia es más elevada. Centrándonos en los valores de transferencia y utilizando la lógica, a mayor tasa de transferencia mayor será el consumo, con lo cual no nos beneficia para el tipo de proyecto que queremos llevar a cabo. ZigBee sigue siendo la mejor opción.

	ZigBee	Bluetooth	Wireless USB	Wi-Fi
Banda frec. (Hz)	2.4G / 868M / 915M	2.4G	3.1-10.6G	2.4-5G
Tasa de transf. (bps)	250k	1M	110-480M	11-108M
Cobertura (metros)	1-75	1-10	3-10	20-100

Tabla 2. Comparativa ZigBee – Bluetooth – Wireless USB – Wi-Fi

1.9.3 Wi-Fi

Wi-Fi, al igual que el resto de medios que estamos analizando, consiste en un sistema de envío de datos sobre redes que utiliza ondas en lugar de cables (*wireless*). Se basa en el estándar IEEE 802.11, y es por ello que podemos encontrar diversos tipos de Wi-Fi:

- Los estándares IEEE 802.11 son fácilmente aceptados debido a que usan la banda frecuencial de 2.4GHz.
- La velocidad de transferencia del medio depende del estándar que se haga uso. Puede proporcionarnos una velocidad de 11Mbps (IEEE 802.11b) hasta 108Mbps (IEEE 802.11n), pasando por los 54Mbps del estándar IEEE 802.11g.
- En la actualidad ya se utiliza el estándar IEEE 802.11a, conocido como Wi-Fi5 ya que opera en la banda de 5GHz. Dicha banda frecuencial (5GHz) ha sido recientemente habilitada. Al no existir otras tecnologías como ZigBee o Bluetooth que hagan uso de ella, nos garantiza un mínimo de interferencias vecinas. Su alcance es algo menor que el de los estándares que trabajan a 2.4GHz.

La principal ventaja que nos proporciona Wi-Fi, aparte de las altas velocidades de transferencia, es la capacidad de suministrar cobertura en un gran rango de distancia (capaz de alcanzar los 100 metros).

Por otra parte, la desventaja fundamental de estas redes existe en el campo de la seguridad. Existen algunos programas capaces de capturar paquetes enviados y calcular la contraseña de la red, y de esta forma acceder a ella. Al igual que el resto de medios de transmisión *wireless*, la velocidad de transferencia se ve disminuida al no utilizar cables.

Vistos los principales puntos que nos puede aportar la utilización de Wi-Fi (altas velocidades de transmisión y gran cobertura), podemos concluir que este tipo de medio quedaría fuera del tipo de transmisión que queremos llevar a cabo al necesitar una mayor fuente de energía para poder proporcionar dichas tasas de velocidades.

1.9.4 Wibree

Wibree resultaría ser una opción a tener en cuenta para su empleo en nuestro proyecto puesto que sus principales características y sus principios básicos son parejos a los ya vistos en ZigBee.

Wibree es una nueva tecnología digital de radio interoperable para pequeños dispositivos. Consiste en una tecnología de comunicación inalámbrica que nos ofrece conexión y comunicación entre dispositivos móviles o computadores y otros dispositivos más pequeños. Su diseño está pensado para que funcione con un consumo de energía mínimo.

Al igual que en ZigBee, Wibree opera a los 2.4GHz (banda ISM) haciendo posible la comunicación entre dispositivos de pila de botón y dispositivos Bluetooth. Su tasa de transferencia es superior respecto a ZigBee: 250kbps de ZigBee frente a 1Mbps de Wibree. Puesto que en nuestro proyecto la cantidad de información a transmitir será mínima (datos medidos por nuestros sensores), la tasa de transferencia tampoco nos interesa, en primera instancia, que sea muy elevada.

Wibree se diseñó para dos modos distintos de implementación:

- Wibree de implementación única: Esta implementación está pensada para el funcionamiento de dispositivos que requieren un consumo bajo de energía, pequeños y de bajo costo, como por ejemplo relojes, sensores deportivos, teclados inalámbricos, etc.
- Wibree de implementación modo dual (Bluetooth – Wibree): Se diseña para su uso en dispositivos Bluetooth donde Wibree se integra con Bluetooth y BluetoothRF, utilizando los dispositivos existentes y dirigidos especialmente a dispositivos como teléfonos móviles y computadoras personales.

1.9.5 TinyOS

TinyOS es un sistema operativo *open source* basado en componentes para redes de sensores inalámbricas. Está diseñado para incorporar nuevas innovaciones rápidamente y para funcionar bajo las importantes restricciones de memoria que se dan en las redes de sensores.

Las aplicaciones para TinyOS se escriben en nesC, un dialecto del lenguaje de programación C optimizado para las limitaciones de memoria de las redes de sensores; donde el lenguaje de programación nesC consiste en un conjunto de tareas y procesos que colaboran entre sí. Existen, además, varias herramientas que completan y facilitan su uso, escritas en su mayoría en Java y en Bash.

TinyOS proporciona interfaces, módulos y configuraciones específicas, que permiten a los programadores construir programas como una serie de módulos que hacen tareas

específicas. Los módulos de TinyOS proporcionan interfaces para los tipos estándar de entradas y salidas de hardware y sensores.

A pesar de que TinyOS no sea una tecnología para comunicaciones inalámbricas, se puede tener en cuenta su utilización para posibles configuraciones y programaciones de redes inalámbricas.

1.9.6 Comparativa general

	ZigBee	**Bluetooth**	**Wireless USB**	**Wi-Fi**	**Wibree**	**TinyOS**
Aplicaciones	Monitori-zación y control	Reemplazo de cable	Web, email, video	Web, email, video	Reemplazo de cable	Soluciones de muy bajo consumo energético
Frecuencia de radio	868MHz, 915MHz, 2.4GHz	2.4GHz	3.1 – 10.6GHz	2.4 – 5GHZ	2.4GHz	868MHz, 915MHz, 2.4GHz
Ancho de banda	20 – 250kbps	1Mbps	110 – 480Mbps	11 – 108Mbps	1Mbps	20 – 250kbps
Dimensión de red	>65000	7 - 8	-	30	-	>65000
Rango de transmisión (metros)	1 - 75	1 - 10	1 - 10	1 - 100	1 -10	1 - 100
Topología de red	Estrella, mallado y árbol de *cluster*	Ad hoc, piconet	Point-to-point, multipoint	Point-to-point, multipoint	-	Mallado puro
Complejidad	Baja	Media - alta	Alta	Alta	Media	Baja
Consumo energético	Bajo	Medio	Alto	Alto	Bajo	Muy bajo
Puntos fuertes	Consumo, coste, robustez, seguridad	Coste, seguridad	Velocidad	Velocidad, flexibilidad	Consumo, coste, robustez, seguridad	Robustez, consumo, coste, flexibilidad

Tabla 3. Comparativa entre los distintos medios de transmisión de datos

Una vez analizados los distintos medios de transmisión inalámbricos podemos concluir diciendo que cada uno de los medios se adapta mejor a un tipo de necesidades, pero en nuestro caso son primordiales el bajo consumo y la baja transmisión de datos, por lo tanto el medio que mejor se adapta es ZigBee.

2 Diseño del proyecto

Una vez estudiado de forma detallada el medio de transmisión que se usará para llevar a cabo este tipo de proyecto, se debe realizar una búsqueda exhaustiva de los dispositivos que se encuentran en el mercado y que compondrán nuestro proyecto. Para ello debemos tener muy claro cuáles son los dispositivos que nos interesan y que deben formar parte de nuestro trabajo, y cuáles son las principales características que deben cumplir para ser aptos.

Antes de empezar la búsqueda, especificaremos cuáles son los principales componentes y sus características que nos serán beneficiosas para la implementación, y de forma esquemática una ilustración inicial de cómo debe estar compuesto nuestro circuito impreso (PCB, *Printed Circuit Board*).

Los componentes imprescindibles para la implementación del proyecto son los siguientes:

- Transceiver ZigBee: Consiste en el dispositivo que realizará la transmisión y recepción de la comunicación. Para ello deberá ser compatible con el protocolo ZigBee, es decir, que nuestro transceiver sea capaz de transmitir en la banda de frecuencia utilizada por ZigBee (2.4GHz). Lógicamente deberá llevar integrada una antena para hacer posible la transmisión y recepción vía RF.
- Microcontrolador: Estará compuesto por una CPU (*Central Processing Unit*), una memoria y puertos de E/S. El microcontrolador hará las tareas de recepción de los datos extraídos por nuestros sensores, procesarlas y mandarlas hacia el transceiver. Dentro de este componente podemos encontrar la operación de conversión A/D, es decir, pasar las posibles señales analógicas transmitidas por los sensores a señales digitales para que el transceiver pueda hacer el envío de los datos. Mencionar también que el "micro" deberá ser compatible con ZigBee, con ello queremos decir que sea apto para instalar y programar el ZigBee Stack.
- Sensores: Serán los elementos encargados de capturar las medidas. La selección y estudio de estos dispositivos queda fuera de este proyecto a pesar de ser nuestra fuente de datos, por lo tanto no entraremos en detalles.
- SPI (*Serial Peripheral Interface*): Será el bus que usaremos para conectar el microcontrolador con el transceiver. Este elemento vendrá ya integrado en el circuito impreso.

- PSU (*Power Supply*): Será necesaria la presencia de una pila para mantener operativos los componentes detallados anteriormente.

Una vez detallados los dispositivos necesarios, debemos especificar las características y condiciones que deben cumplir para poder utilizarlos. Las condiciones impuestas van acorde con el tipo de proyecto que se pretende llevar a cabo.

Dejando de lado que deben ser compatibles con ZigBee, condición indiscutible para poder plantearse su utilización, los principales requisitos impuestos son los siguientes:

- Tamaño reducido: Tratándose de un proyecto que se basa en la realización de medidas en diferentes partes del cuerpo, los dispositivos que deben efectuar las operaciones de medida y envío de datos deben ser lo más pequeños posibles. Por ello se han de descartar elementos de gran tamaño (hablamos de elementos que no deben superar los 2-3cm) y por consiguiente de mayor peso (se prefieren dispositivos lo más ligeros posible). Se trata de una de las condiciones primordiales ya que componentes voluminosos incomodarían al paciente y a su vez dificultarían la tasca de medición.
- Solución completa: Con ello nos referimos a encontrar una PCB que nos incluya tanto el transceiver como el microcontrolador. A parte de facilitarnos el trabajo en el momento de búsqueda de los distintos dispositivos compatibles unos con otros, una solución completa nos ayudaría a cumplir con la condición anterior puesto que vendrían integrados de fábrica todos los componentes y probablemente el tamaño sería el más reducido posible.
- Bajo consumo: No se trata de una de las condiciones más importantes para cumplir ya que, como hemos podido estudiar, ZigBee se caracteriza por su bajo consumo. De todos modos, este punto se suple con la pila incorporada a la PCB.

Seguidamente podemos visualizar, de forma esquemática, el aspecto que debe tener nuestra PCB (Figura 9).

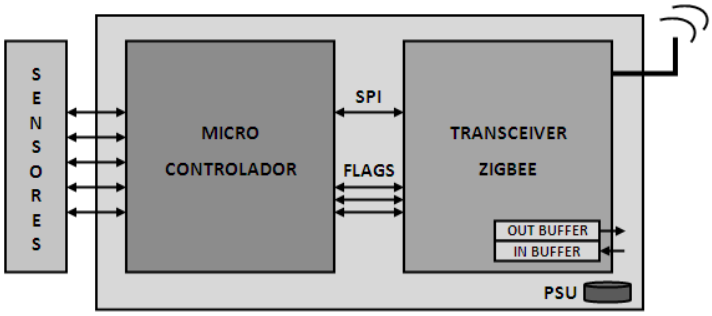

Figura 9. Diseño esquemático del circuito impreso (PCB)

2.1 Búsqueda de los módulos ZigBee

Una vez conocidos los dispositivos que debemos encontrar para poder implementar el proyecto y planteadas las condiciones que deben cumplir éstos, podemos empezar la búsqueda de los módulos compatibles con ZigBee.

Para ello se ha realizado una búsqueda y estudio de diferentes dispositivos ZigBee (transceiver y microcontroladores) que proporcionan un listado de fabricantes. Por lo tanto, en este apartado quedan detallados los distintos componentes que han sido analizados para poder encontrar la mejor solución posible.

Una vez finalizado el estudio de los distintos módulos se realizará una tabla donde quedarán detalladas todas las especificaciones técnicas de todos los dispositivos y de este modo realizar la selección del módulo más apto para nuestro proyecto

2.1.1 XBee ZNet 2.5 RF Module

En la familia de módulos XBee podemos encontrar, como veremos más adelante, dos tipos de módulos muy similares: XBee ZNet y XBee-PRO ZNet. En este caso tratamos con el módulo XBee ZNet 2.5, mostrado en la Figura 10.

Figura 10. Transceiver XBee ZNet 2.5 RF Module

El módulo XBee ZNet 2.5 RF satisface las necesidades de bajo coste y bajo consumo exigido en redes de sensores inalámbricas. El módulo diseñado para ZigBee es de fácil uso, requiere niveles mínimos de potencia y proporciona una entrega fiable de datos entre dispositivos. Su pequeño tamaño ayuda a su integración en una PCB.

Este módulo, totalmente compatible con ZigBee, opera a una frecuencia de 2.4GHz dentro de la banda ISM. Está diseñado para su uso en US, Canadá, Australia y Europa.

No precisa de ninguna configuración para comunicaciones RF. XBee ZNet 2.5 no está configurado para ningún nivel de aplicación específico, esto proporciona que se pueda usar en una amplia gama de sistemas.

El fabricante proporciona únicamente el transceiver, por lo tanto se debe encontrar un microcontrolador compatible con este módulo.

Las principales especificaciones del módulo XBee ZNet 2.5 RF[10] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 2mW (+3dBm)
- Consumo Rx: 40mA
- Consumo Tx: 40mA
- Pila: 2.1 – 3.6V
- Temperatura: -40ºC a 85ºC
- Distancia: Interior: 40m / Exterior: hasta 120m
- Tamaño (cm): 2.43 x 2.76
- Peso: 3gramos
- Uso para redes avanzadas y soporte de módulos de bajo consumo.
- Conectores RPSMA y U.FL, antena Chip o antena Wired Whip.

2.1.2 XBee-PRO ZNet 2.5 RF Module

Este nuevo módulo proviene del mismo fabricante que el anterior, por lo tanto las aportaciones de uno y otro son muy parejas. XBee-PRO se diferencia por una mayor potencia de salida y por lo tanto mayor cobertura, mayor tamaño de dispositivo y mayor consumo en potencia.

Al igual que en el caso anterior, XBee-PRO es solamente el transceiver, como podemos ver en la Figura 11. Este es uno de los motivos por el que probablemente se puedan descartar los módulos proporcionados por XBee.

Figura 11. Transceiver XBee-PRO ZNet 2.5 RF Module

[10] Para más información, véase el Anexo: "Ficha técnica XBee ZNet 2.5 RF Module"

Las principales especificaciones del módulo XBee-PRO ZNet 2.5 RF[11] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 50mW (+17dBm)
- Consumo Rx: 45mA
- Consumo Tx: 295mA
- Pila: 3.0 – 3.4V
- Temperatura: -40ºC a 85ºC
- Distancia: Interior: 120m / Exterior: hasta 1.6km
- Tamaño (cm): 2.43 x 3.29
- Peso: 3gramos
- Uso para redes avanzadas y soporte de módulos de bajo consumo.
- Conectores RPSMA y U.FL, antena Chip o antena Wired Whip.

2.1.3 EasyBee ZigBee Transceiver Module

EasyBee consiste en un módulo transceiver útil para RF ZigBee dentro del IEEE 802.15.4. Esto permite facilitar el diseño de productos inalámbricos que posean ZigBee/IEEE 802.15.4 sin la necesidad de RF o de diseño de antenas grandes.

El módulo EasyBee, que podemos ver en la Figura 12, contiene toda la circuitería pertinente de RF, incluyendo una antena integrada y el regulador, el enchufe de unión o el módulo de montaje superficial.

Figura 12. EasyBee ZigBee Transceiver Module

Contiene un SPI de 4 cables que funcionan de interface con el microcontrolador. La capa PHY incluye una impedancia al igual que la antena integral. Posibilita la opción de una pantalla o de un conector externo de antena.

La capa MAC incluye la generación CRC-16, la evaluación de canal clara, la detección de señal de energía, la seguridad, encriptación y autentificación.

[11] Para más información, véase el Anexo: "Ficha técnica XBee-PRO ZNet 2.5 RF Module"

EasyBee es un dispositivo totalmente capaz de comunicarse en una red y no precisa de una capa de red ZigBee. Si se precisa ZigBee, podemos encontrar ZigBee Stacks libres.

Este módulo es una solución rápida para aplicar ZigBee y IEEE 802.15.4 a las comunicaciones del mercado.

Nos encontramos con otro fabricante que nos proporciona únicamente el dispositivo transceiver para hacer la conexión mediante ZigBee, pero en este caso el propio fabricante nos recomienda el uso de un microcontrolador de la casa MICROCHIP. Sería una posible solución en caso de no encontrar ningún módulo que posibilite una solución completa.

Las principales especificaciones del módulo EasyBee ZigBee Transceiver Module[12] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 1mW (+0dBm)
- Consumo Rx: 20mA
- Consumo Tx: 18mA
- Pila: 2.1 – 3.6V
- Temperatura: -40ºC a 85ºC
- Distancia: 120m
- Tamaño (cm): 2.6 x 2
- Posibilidad de antena integrada o externa.
- Uso para redes ZigBee, sustitución de cables, automatización del hogar, control y conexionado industrial, redes inalámbricas de sensores.

2.1.4 ZB-21 ZigBee OEM Module

Es uno de los mejores módulos con capacidad de utilización de ZigBee. El ZB-21 ZigBee OEM Module ha sido diseñado para aportar una mayor flexibilidad en las conexiones. El módulo ZB-21 incluye un procesador OKI ARM7TDMI y funciones RF de ZigBee.

El procesador ARM7 está disponible para el desarrollo de conexiones flexibles, esto lo hace un dispositivo controlador inalámbrico.

[12] Para más información, véase el Anexo: “Ficha técnica EasyBee ZigBee Transceiver Module”

A lo que refiere a la potencia de consumo de este módulo, ha sido cuidadosamente optimizada (23µA en estado dormido).

El módulo está listo para ser precargado con programas y posteriormente ser probado de modo que estén listos a instalar sin procedimientos adicionales.

Como podemos ver en la Figura 13, el ZB-21 consiste en un montaje superficial sobre una PCB, donde se encuentra el módulo totalmente integrado (microcontrolador + transceiver) listo para ser utilizado en tecnologías sin cables ZigBee.

Figura 13. ZB-21 ZigBee OEM Module

Visto lo que nos proporciona AMPEDRF, casa origen del módulo ZB-21, hay que tener muy en cuenta esta opción puesto que nos suministran de forma conjunta e integrada los dispositivos transceiver y microcontrolador.

Las principales especificaciones del módulo ZB-21 ZigBee OEM Module[13] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 1mW (+0dBm)
- Consumo Rx: 25µA
- Consumo Tx: 50mA
- Pila: 3.3V
- Temperatura: -20ºC a 70ºC
- Distancia: Interior: 30m / Exterior: 100m
- Tamaño (cm): 1.5 x 2.7
- Apto para estándar ZigBee.
- Módulo completo de RF.
- Antena Integrada.
- Interface SPI.

[13] Para más información, véase el Anexo: "Ficha técnica ZB-21 ZigBee OEM Module"

2.1.5 MICAz Module

El fabricante de MICAz, Crossbow, proporciona módulos inalámbricos que satisfacen todo tipo de necesidades de los usuarios para sus aplicaciones o incluso para el diseño de proyectos a gran escala. Crossbow proporciona una gran variedad de kits diseñados para proporcionar las herramientas necesarias para el diseño y desarrollo de redes de sensores inalámbricas.

El módulo MICAz que vemos en la Figura 14 hace uso de la banda frecuencial de 2.4GHz y es usado para habilitar redes de sensores inalámbricos de bajo consumo.

Figura 14. MICAz Module

La PCB del módulo MICAz se conoce como MPR2400CA, y está basado en un microcontrolador de Atmel (ATmega1281). ATmega128L consiste en un “micro” de bajo consumo en el cual se puede instalar en su memoria flash el ZigBee Stack usado por el fabricante (MoteWorks).

Su único procesador (MPR2400) puede ser configurado para ejecutar nuestra aplicación con sensores, procesar nuestra red o realizar comunicaciones simultáneas.

Incluye una expansión de conectores de 51-pin que soporta entradas analógicas, entradas/salidas digitales, I2C, SPI e interface UART. Esta interface es muy útil para poder conectar el módulo a una gran variedad de periféricos. Crossbow ofrece una gran variedad de conexiones para sensores vía el estándar de la expansión de conectores 51-pin y de este modo la posibilidad de adquisición de datos muy útiles.

MICAz (MPR2400), basado en el estándar 802.15.4, transfiere a una velocidad de 250kbps e incluye un hardware de seguridad (AES-128).

Este módulo puede ser una muy buena opción para la implementación de nuestro proyecto, ya que incluye microcontrolador y transceiver y ambos compatibles con ZigBee. El *handicap* que puede acusar este dispositivo es el dimensionado de su PCB puesto que, como se puede apreciar en su ficha técnica, sobrepasaría los límites deseados.

Las principales especificaciones del módulo MICAz Module[14] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 1mW (+0dBm)
- Consumo Rx: 19.7mA
- Consumo Tx: 17.4mA
- Pila: 3.3V
- Distancia: Interior: 20-30m / Exterior: 75-100m
- Tamaño (cm): 3.2 x 5.8 x 0.7
- Apto para estándar ZigBee.
- Módulo completo de RF.
- Antena Integrada.
- Interface SPI.
- Expansión de conectores para una gran variedad de sensores.

2.1.6 ATMEL Module

ATMEL es el nombre del fabricante que proporciona dispositivos compatibles con la especificación ZigBee.

En este caso se nos detalla, por separado, cual es el transceiver y el microcontrolador utilizados para la implementación del módulo. Como se puede comprobar en la Figura 15, nos encontramos con otra solución completa al integrar en su PCB todos los componentes requeridos y compatibles con ZigBee, por lo tanto debemos de analizar dichos componentes para conocer los beneficios que pueden aportar al proyecto y de este modo llegar a la mejor solución posible.

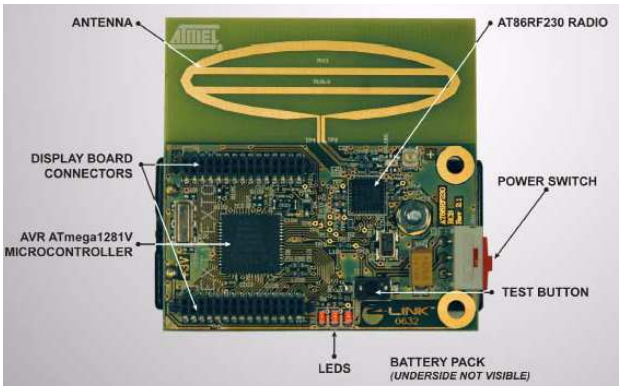

Figura 15. ATMEL Module

[14] Para más información, véase el Anexo: "Ficha técnica MICAz Module"

- ***AT86RF230:***

El AT86RF230 es un transceiver que de bajo consumo, como es requerido en ZigBee, y que trabaja en la banda frecuencial de los 2.4GHz. Éste ha sido diseñado especialmente para las aplicaciones de bajo coste de ZigBee/IEEE 802.15.4.

Junto al AT86RF230 viene incorporada la antena aunque ésta no está integrada dentro de la tarjeta PCB, como se puede apreciar en la Figura X. Todos los componentes necesarios para RF, exceptuando la antena, vienen implementados en el chip.

Por lo tanto, se puede decir que el transceiver AT86RF230 es muy útil para las siguientes aplicaciones:

- Redes de sensores inalámbricas.
- Control industrial.
- Automatización del hogar.
- Consumo electrónico.
- Periféricos del PC.

Este transceiver, como ya se ha comentado al inicio del apartado, puede ser usado juntamente con otro microcontrolador compatible y de la misma casa.

Las principales especificaciones del módulo ATMEL Module[15] son las siguientes:

- Frecuencia: 2.4GHz
- Potencia de salida: 4mW (+3dBm)
- Consumo Rx: 15.5mA
- Consumo Tx: 16.5mA
- Pila: 3.6V
- Temperatura: -40ºC a 85ºC
- Apto para estándar ZigBee.
- Módulo completo de RF.
- Antena Integrada.
- Interface SPI.

- ***ATmega1281:***

ATmega1281 es el microcontrolador utilizado en el módulo ATMEL, compatible con ZigBee y con el transceiver anterior.

[15] Para más información, véase el Anexo: “Ficha técnica ATMEL Module: AT86RF230”

El ATmega1281 es un microcontrolador CMOS 8-bit de bajo consumo basado en las mejoras AVR de la arquitectura RISC. Este "micro" ha sido muy optimizado en el aspecto de la velocidad de procesamiento, por lo tanto es muy útil en casos de gran transferencia de datos y posterior procesamiento de ellos.

Otro aspecto que se ha tenido muy en cuenta a la hora del diseño del microcontrolador es su consumo, cumpliendo de este modo una de las principales premisas del estándar ZigBee.

Las principales especificaciones del módulo ATMEL Module[16] son las siguientes:

- Memoria flash: 128kB
- Consumo modo activo: 500µA
- Consumo modo pasivo: 0.1µA
- Pila: 3.6V
- Temperatura: -40ºC a 85ºC
- Apto para estándar ZigBee.
- Interface SPI.

Una vez estudiado y analizado el módulo que propone ATMEL podemos llegar a un seguido de conclusiones a partir de las cuales podemos determinar si es apto o no para nuestro proyecto.

En primera instancia todos sus componentes son los adecuados para la implementación de una red de sensores inalámbrica ZigBee, puesto que el microcontrolador y el transceiver utilizados son compatibles con el estándar. Las características de ambos dispositivos son óptimas para su utilización ya que se adaptan al bajo consumo de energía.

De todos modos, la utilización del módulo de ATMEL parece más propicia para aplicaciones a mayor escala [13] (entornos industriales o automatización). Me explico. En nuestro caso se requiere de una tasa de transferencia de datos muy baja, al igual que el consumo; y el dimensionado de la placa debe de ser lo menor posible. ATMEL proporciona servicios que quedarían muy por encima de los que nosotros haríamos servir, por lo tanto, a pesar de ser una muy buena opción, no entra en el perfil de dispositivos que queremos utilizar para trabajar en nuestro proyecto.

[16] Para más información, véase el Anexo: "Ficha técnica ATMEL Module: ATmega1281"

2.2 eZ430-RF2500 Development Tool

eZ430-RF2500 es la herramienta de trabajo elegida para trabajar y realizar las primeras pruebas de implementación del proyecto.

Una vez finalizada la búsqueda y el estudio de los diferentes módulos compatibles con ZigBee que podemos encontrar en el mercado, se ha decidido que la mejor opción es la de hacer uso del kit que proporciona el fabricante Texas Instrument: eZ430-RF2500.

En este apartado se plasmará el estudio que se ha realizado a dicha herramienta de trabajo, desglosando el kit en los diferentes módulos que lo componen; y haciendo una mención especial al tipo de ZigBee Stack que utiliza nuestro dispositivo (SimpliciTI).

2.2.1 Contenido y descripción del kit eZ430-RF2500

En primera instancia, detallaremos todo el material que se nos ha proporcionado con la compra[17] del kit [14]. En la Figura 16 podemos apreciar de forma visual los componentes principales que constituyen la herramienta de trabajo.

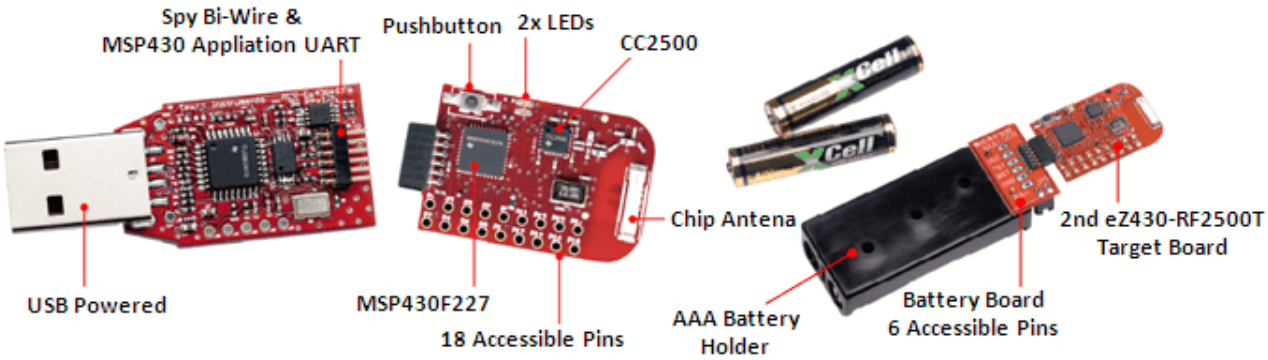

Figura 16. Kit eZ430-RF2500 [14]

En formato *hardware* incluye:

- Dos tarjetas eZ430-RF2500T.
- Una interface USB eZ430-RF.
- Un soporte para las baterías de los módulos.

A su vez, el kit nos incluye un CD donde podemos encontrar documentación y herramientas de desarrollo relacionadas con el eZ430-RF2500:

[17] Véase anexo: "Compra del kit eZ430-RF2500"

- La guía de usuario para el eZ430-RF2500.
- Una guía de usuario para la familia de MSP430F2xx.
- Las herramientas de trabajo Code Composer Essentials (CCE) y IAR Embedded Workbench.
- Una demo que visualiza una red inalámbrica de sensores eZ430-RF2500.

El eZ430-RF2500 consiste en una completa herramienta de desarrollo inalámbrico para la creación de proyectos *wireless*, haciendo uso de una interface USB para una mayor comodidad a la hora de interactuar, por ejemplo, con un ordenador y suministrando tanto el *hardware* como el *software* necesario para manipular el microcontrolador MSP430F2274 y el transceiver inalámbrico CC2500 a 2.4GHz. La herramienta, como ya se ha detallado anteriormente, incluye dos tarjetas inalámbricas a 2.4GHz que incluyen un microcontrolador de muy bajo consumo (MSP430F2274). Los proyectos que se realicen con estos dispositivos pueden gozar de una autonomía gracias a la plataforma de batería que incluye el kit. Se debe hacer especial mención a SimpliciTI, tipo de ZigBee Stack que hace uso nuestra herramienta, puesto que posee las propiedades necesarias para generar redes inalámbricas de bajo consumo y robustas.

Las características principales del eZ430-RF2500 son:

- Una interface USB que permite la programación de los dispositivos.
- Un microcontrolador MSP430 de ultra-bajo consumo a 16MHz.
- Un transceiver CC2500 a 2.4GHz.
- El aplicativo UART.
- 18 pins disponibles para la realización de pruebas.
- Dos LEDs (rojo/verde) para indicar, de forma visual, el desarrollo de la comunicación.

El eZ430-RF2500 puede hacer uso de dos entornos de programación para poder escribir, descargar y programar nuestras aplicaciones en el dispositivo ZigBee. Estos dos entornos son:

- IAR Embedded Workbench Integrated Development Environment (IDE)
- Code Composer Essentials (CEE)

Estas dos herramientas nos serán muy útiles en el momento que queramos probar y programar nuestros módulos.

El término eZ430-RF2500T hace referencia exclusivamente a la tarjeta inalámbrica adicional para la herramienta de desarrollo eZ430-RF2500, y ésta puede ir conectada

indistintamente a la interface USB o al soporte de batería. Este componente puede realizar dos papeles diferentes en nuestra red inalámbrica, dependiendo del tipo de programación que se le haya instalado en su microcontrolador, ya que podemos utilizarlo como punto de acceso (AP, *Access Point*) o como dispositivo final (ED, *End Device*). Las tarjetas de eZ430-RF2500T incluyen los siguientes componentes:

- Microcontrolador MSP430F2274: 32kB de memoria Flash, 1kB de RAM, USCI (UART, 2xSPI, I2C, IrDA), 10-bit 200ksps ADC, 2 amplificadores operacionales.
- Transceiver CC2500: 2.4GHz, banda multicanal ISM de baja potencia.
- Dos LEDs.
- Un pulsador.

La interface USB permite al eZ430-RF2500 recibir y enviar, de forma remota, información a nuestro ordenador utilizando la aplicación UART del MSP430. El corazón del sistema de comunicaciones serie es la UART.

El chip UART controla los puertos y dispositivos serie, y se encuentra integrado en la placa base o en la tarjeta adaptadora del dispositivo. Las funciones principales de UART son las de manejar las interrupciones de los dispositivos conectados al puerto serie, convertir los datos en formato paralelo y transmitirlos al bus del sistema en formato serie para que puedan ser transmitidos a través de los puertos, y viceversa. La UART es un dispositivo programable en el que pueden establecerse las condiciones que se utilizarán para la transmisión (velocidad, paridad, longitud y bits de parada). En el apartado 3.1 se tratará la configuración de los dispositivos y del mencionado chip UART.

Gracias al conector Spy Bi-Wire, la herramienta eZ430-RF2500 puede ser compatible con otros microcontroladores de la familia de MSP430 (series de F22xx y F20xx), y con las tarjetas correspondientes a eZ430-F2013 y eZ430-T2012.

- ***Transceiver CC2500:***

El CC2500 [15] consiste en un transceiver a 2.4GHz de bajo coste diseñado para aplicaciones de bajo consumo. Su circuitería trabaja dentro de la banda frecuencial ISM (entre 2400MHz a 2483.5MHz), necesario para ser compatible con ZigBee, y SRD (*Short Range Device*).

El RF transceiver está integrado con un módem configurado en banda base, el cual soporta varios formatos de modulación y puede ser configurado para realizar una transferencia de datos por encima de los 500kBaud.

El transceiver CC2500 viene provisto del *hardware* que soporta el manejo de paquetes, el almacenamiento de datos, el inicio de la transferencia, la evaluación de limpieza en el canal e indicadores de calidad del enlace.

Los parámetros principales de las operaciones realizadas en el CC2500 pueden ser controlados vía una interface SPI. La gestión de cola de datos en el buffer del transceiver es gestionada mediante la política FIFO[18]. En un sistema típico, el CC2500 será usado junto con un microcontrolador y componentes adicionales pasivos.

- ***Microcontrolador MSP430F2274:***

La familia de microcontroladores de potencia ultra-baja MSP430 [16] de *Texas Instrument* consiste en una gran gama donde éstos pueden ser usados en una gran variedad de aplicaciones. La arquitectura, combinada con cinco modos de conservación de energía, optimiza la durabilidad de sus baterías, ayudando, de este modo, a que su utilización se extienda a un gran abanico de posibilidades.

En el dispositivo destaca una CPU de 16 bits RISC, registros de 16 bits y generadores constantes que contribuyen a maximizar la eficiencia del código. El oscilador controlado digitalmente (DCO, *Digitally Controlled Oscillator*) permite "despertar" a los módulos del modo de bajo consumo al modo activo en menos de 1µs.

La serie de MSP430x22x se basa en microcontroladores de señal mixada y de consumo ultra-bajo, con dos temporizadores de 16 bits, un interface de comunicación universal, un conversor analógico-digital de 10 bits, un controlador de datos de transferencia (DTC, *Data Transfer Controller*), dos amplificadores operacionales de uso general en los dispositivos de MSP430x22x4 y 32 pins I/O.

Típicamente, en las aplicaciones que se hace uso de estos microcontroladores aparecen sensores que capturan señales de forma analógica, convirtiendo estas señales a formato digital y posteriormente procesar los datos para mostrarlos, o bien transmitirlos a otro equipo a través de radiofrecuencias (RF).

[18] FIFO: *First In First Out*. Política de cola que se basa en que los primeros datos en entrar serán los primeros en salir.

2.2.2 SimpliciTI Network Protocol

Como ya se ha mencionado con anterioridad, SimpliciTI constituye una de las partes más importantes de nuestra red de sensores inalámbrica puesto que engloba todos los protocolos necesarios para poder implementar una red ZigBee.

El protocolo de red SimpliciTI consiste en un protocolo de radiofrecuencia de baja potencia capaz de contener redes de <100 nodos. La propiedad de los nodos que forman parte de nuestra red para poder permanecer dormidos durante largos intervalos de tiempo hacen que SimpliciTI sea un protocolo de baja potencia; a su vez, dicho protocolo es de bajo coste al hacer uso de memorias *flash* inferiores a los 4kbytes y RAM menores de los 512 bytes. Por lo tanto, podemos comprobar cómo SimpliciTI aporta las principales propiedades que caracteriza ZigBee (Figura 17).

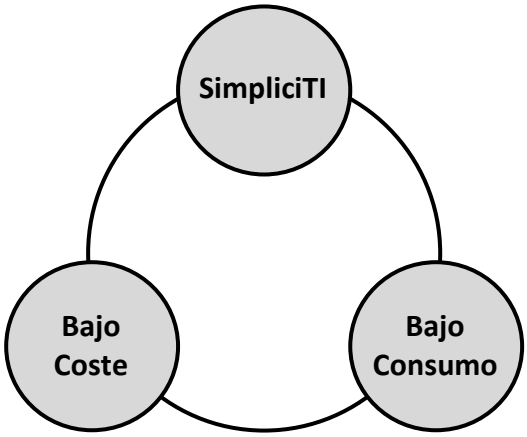

Figura 17. Propiedades que proporciona SimpliciTI

SimpliciTI ha sido diseñado para una fácil implementación, usando los mínimos recursos requeridos para los microcontroladores. El protocolo funciona inmediatamente en microcontroladores MSP430 de muy baja energía de TI (como los usados en nuestra red) y múltiples transceivers RF.

El protocolo de red apoya a una amplia gama de aplicaciones con bajo uso de energía como, por ejemplo, seguridad y activación de alarmas (detectores de humo, sensores de luz y de movimiento,…), lectura de mediciones automatizada (medición de gas o agua), o automatización del hogar.

A pesar de los recursos requeridos, SimpliciTI soporta dispositivos finales (ED) en una topología de red entre iguales (*peer-to-peer*), la opción de usar un punto de acceso (AP) para almacenar mensajes, y extensores de rango (RE) que nos permite ampliar el rango de nuestra red hasta cuatro saltos. Como se puede apreciar en la Figura 18, gracias a los dispositivos programados como RE somos capaces de extender nuestro

rango de cobertura y de este modo implementar redes inalámbricas mucho más amplias y útiles.

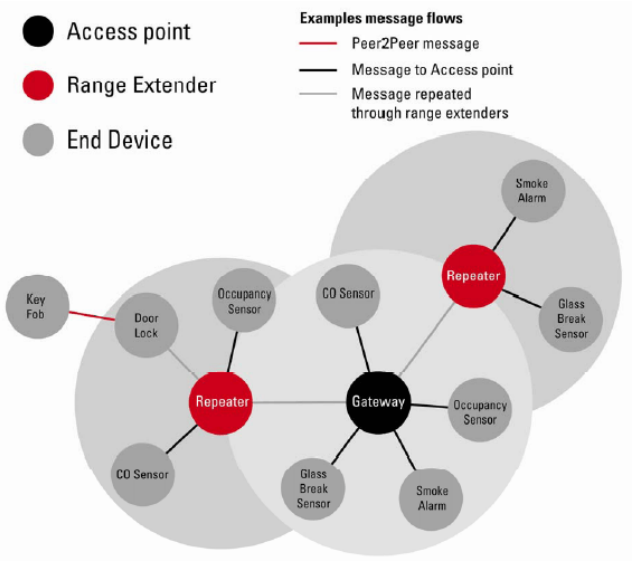

Figura 18. Red de cobertura que proporciona SimpliciTI [18]

2.3 Comparativa general

En la Tabla 4 podemos comprobar las principales características de los diferentes dispositivos estudiados; y a su vez llegar a un seguido de conclusiones y observaciones que nos permitirán comprobar que la mejor opción es la de utilizar los módulo de Texas Instrument.

	XBee Pro		EasyBee	AMPEDRF	MICAz	ATMEL	Texas Instrument
	XBee	**XBee-PRO**					
Tamaño (cm) [LxWxH]	2.43 x 2.76	2.43 x 3.29	2.6 x 2	2.7 x 1.5	6 x 3.2 x 0.7	-	3 x 2 x 0.1
Peso	3gr	3gr	-	-	18gr	-	-
Transceiver	Sí	Sí	Sí	Sí	Sí	Sí (AT86RF230)	Sí (CC2500)
Micro	No	No	No	Sí	Sí	Sí (ATmega1281)	Sí (MSP430F2274)
Consumo Rx	40mA	45mA	20mA	25mA	19.7mA	15.5mA	13.3mA
Consumo Tx	40mA	295mA	18mA	50mA	17.4mA	16.5mA	21.2mA
Pila	2.1-3.6V	3.0-3.4V	2.1-3.6V	3.3V	1.5V	3.6V	3.6V
Potencia salida	1.25-2mW	10-50mW	1mW	1mW	1-3.98mW	4mW	1mW

Tabla 4. Comparativa entre los distintos dispositivos compatibles con ZigBee

Como se mencionó a principios de este segundo bloque, aquellos fabricantes que no nos proporcionasen módulos con una integración conjunta de los dos dispositivos con mayor peso (transceiver y microcontrolador) será desestimada la utilización de éstos. Por lo tanto, los dispositivos XBee, XBee-PRO y EasyBee se ha creído oportuno no utilizarlos puesto que complicaría el trabajo al tener que buscar un microcontrolador compatible con los transceivers de cada uno de ellos.

Otra de las condiciones que se impuso en un principio fue el dimensionado de las PCB de los módulos. Como ya se dijo en su momento, es un factor muy importante ya que nos facilitará su posterior manipulación. Por ello podemos descartar la utilización de los módulos de MICAz y ATMEL. A partir del estudio realizado a estos dos dispositivos, parecen ser más útiles para proyectos a mayor escala y, por lo tanto, mayor complejidad.

Para la elección de los módulos de Texas Instrument, no se contempló tanto los *handicaps* del resto de dispositivos sino las ventajas que nos podía aportar la compra del kit eZ430-RF2500.

Dejando de lado la compatibilidad con ZigBee, la solución completa y la reducción de tamaño; la interface USB que viene incorporada nos hace muy interesante la utilización del eZ430-RF2500. Con ello nos posibilita una comunicación fácil y sencilla con el puerto serie del ordenador, y por lo tanto una monitorización de los datos recibidos. Teniendo presente que también se nos proporcionan dos tarjetas inalámbricas y el soporte de batería, el kit del fabricante Texas Instrument es el más adecuado para la implementación del proyecto.

Con ello no quiero decir que con la utilización de los otros módulos no sea posible su implementación, al contrario, yo propondría que una vez finalizado este proyecto existiera la posibilidad de implementar el mismo diseño pero en módulos de otro fabricante. Esto sería bueno para perfeccionar el proyecto realizado anteriormente y poder comparar resultados a partir de la utilización de otros materiales.

3 Implementación del proyecto

En este bloque de la memoria se tratará toda la parte práctica que se ha llevado a cabo para la implementación de nuestro proyecto.

Una vez conocido el medio de transmisión y la herramienta de trabajo que se utilizará, ya estamos dispuestos a trabajar con los dispositivos. Para ello, previamente, debe realizarse una correcta configuración y programación de los dispositivos y posteriormente una adaptación de los módulos al sensor para hacer posible la transferencia de datos de un componente al otro.

3.1 Configuración de los dispositivos

La configuración de los dispositivos es muy sencilla puesto que únicamente debemos establecer los parámetros necesarios al puerto que haremos servir.

En el momento de la configuración nos aparece el término “MSP430 Application UART”, el cual se refiere al ya mencionado chip UART. Como ya se ha detallado con anterioridad, la UART es la encargada de controlar los puertos y dispositivos serie, por lo tanto debemos establecer las condiciones de transferencia de los datos. En la Figura 19 podemos visualizar los valores establecidos.

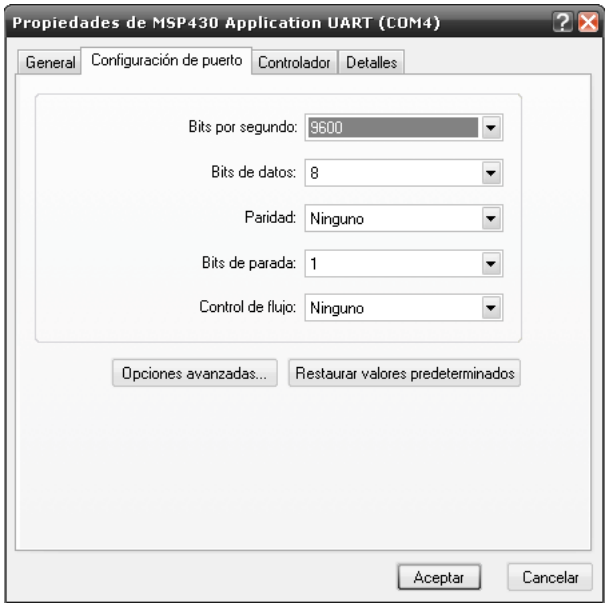

Figura 19. Configuración de los dispositivos (MSP430 Application UART)

Los ajustes que se deben realizar son los siguientes:

- Bits por segundo: Este parámetro nos indica la velocidad, en baudios, de transferencia de nuestros datos. En nuestro caso establecemos que la velocidad sea de 9600bps.
- Bits de datos: Determinamos la cantidad de bits que se envían y se reciben por trama de información. En nuestro caso establecemos 8 bits de información (1 byte).
- Paridad: Tenemos la opción de utilizar la paridad para comprobar si nuestro envío ha sido realizado de forma correcta. Cabe la posibilidad de realizar un envío y comprobación de paridad con el criterio de paridad impar, o bien con el criterio de paridad par; o, como en nuestro caso, no se envía bit de paridad ni hace comprobación de paridad en la recepción.
- Bits de parada: Determinamos el número de bits de parada que creemos convenientes. En nuestro caso establecemos únicamente un bit de parada.
- Control de flujo: En este último apartado establecemos que no deseamos ningún tipo de control de flujo.

3.2 Programación de los dispositivos

Una vez realizado el estudio y la configuración de nuestra herramienta de desarrollo eZ430-RF2500 para la generación de redes de sensores inalámbricas, podemos pasar a la programación de las tarjetas que nos proporcionan para que éstas desempeñen las tareas y funciones que nosotros deseemos.

El entorno de programación que usaremos para programar los dispositivos es el *IAR Embedded Workbench Integrated Development Environment* (IDE). Antes de empezar a manipular el entorno de programación y de conocer el tipo de lenguaje que se deberá utilizar, debemos de pensar qué papel jugarán nuestros nodos en la red que generaremos.

Al suministrarnos únicamente dos tarjetas eZ430-RF2500T, nuestra red constará solamente de dos nodos (con la compra de más tarjetas del mismo perfil es posible la ampliación de la red). Recordamos que uno de los principales objetivos de nuestro proyecto, por no decir el principal, se basa en conseguir una transferencia de datos de forma inalámbrica entre dos terminales (emisor-receptor), por lo tanto con dos tarjetas ya seremos capaces de trabajar en ello.

Más allá de la transmisión de datos de forma inalámbrica, se desea la posibilidad de adaptar un sensor a una de las tarjetas para hacer posible la transferencia inalámbrica de los datos medidos por el sensor. La tarjeta que irá conectada directamente al sensor la conoceremos como *End Device* (ED) y actuará como emisor. Con los datos ya almacenados en el emisor, éste se encargará de la transmisión RF de los datos hacia la otra tarjeta. Esta segunda tarjeta, que irá conectada directamente a la interface USB, la conoceremos como *Access Point* (AP) y actuará como receptor. La tarjeta receptora irá conectada al puerto serie del ordenador para que, finalmente, los datos puedan ser monitorizados.

Una vez planteados los roles de cada tarjeta y el principal funcionamiento de la red, debemos tener presente la organización que suele tener una red basada en ZigBee. Como se ha podido estudiar en apartados anteriores, una red ZigBee está basada en un nodo central (AP) que actúa como coordinador de la red (ZC) y en un seguido de dispositivos finales (ED), los cuales se encargan de mandarle paquetes de información al punto de acceso. En nuestro caso nos encontramos con una situación de un punto de acceso y de un único dispositivo final, por lo tanto no nos tendremos que plantear la situación de una red más compleja al encontrarse diversos dispositivos finales en ella.

Antes de empezar con la programación de los dispositivos detallaremos las principales funciones y procedimientos que deben realizar el *Access Point* y el *End Device*:

- Una de las tarjetas será programada como *Access Point*. Ésta será la encargada de hacer el papel de nodo central. Sus funciones principales son las de inicializar la red, crear las direcciones de identificación para los posibles nodos de la red y estar escuchando el canal por la posibilidad de llegada de paquetes.
- La otra tarjeta será programada como *End Device*. Ésta hará el papel de nodo de la red y emisora de paquetes de información. Su función principal es la de recibir los datos provenientes del sensor y posteriormente transmitirlos hacia el nodo central (AP).

A continuación se comentarán los principales bloques de los programas realizados para la tarjeta *Access Point* y para la tarjeta *End Device*.

Antes de comentar los códigos de programación, mencionar que éstos están basados en un programa de medida de temperatura y del nivel de energía realizado por Texas Instrument. Lo que se hizo fue suprimir aquellas partes del código que no fuesen

necesarias en el proyecto, mantener aquellas que por norma general siempre deben de estar presentes en la programación de una red ZigBee (inicialización de la red, generación de direcciones,…), y añadir procedimientos y funciones específicas para el tipo de proyecto que se pretende llevar a cabo.

3.2.1 Código de programación del Access Point

En este apartado comentaré las principales[19] funciones que debe realizar el punto de acceso de mi red ZigBee, y a su vez aquellos procedimientos que se han programado específicamente para este proyecto.

Las funciones básicas que debemos encontrar en la programación de una red ZigBee son las siguientes:

```
void createRandomAddress(); // Creación de las direcciones
void MCU_Init(void); // Inicialización de la MCU
```

En primera instancia nos encontramos con el procedimiento de creación de direcciones. Esta función es fundamental a la hora de inicializar la red, puesto que para que sea posible la comunicación entre un dispositivo final (ED) y un punto de acceso (AP) ambos deben estar identificados de algún modo. También hacer mención especial a la variable que almacena estas direcciones:

```
__no_init volatile char Flash_Addr[4] @ 0x10F0; // Vector de 4 posiciones, puesto que
                                                // las direcciones son de 4 bytes
```

En segunda instancia, y con ello no menos importante, tenemos el procedimiento de inicialización de la MCU. Como se puede comprobar en la parte de código destinada a la inicialización de la MCU, se realiza la configuración y calibración del micro, tanto la velocidad de transferencia como la velocidad de procesamiento.

Una vez realizados los dos procedimientos anteriores, nuestro nodo central está dispuesto a inicializar la red. Nos informa de ello a través del siguiente mensaje:

```
TXString("\r\nInicializando Red...",22); // Inicializando la red
```

Para que se finalice la inicialización de la red, previamente debe ejecutarse el siguiente comando que está relacionado con el ZigBee Stack SimpliciTI:

```
SMPL_Init(sCB); // Inicialización de la red a través de SimpliciTI
```

Este comando es específico de SimpliciTI, y su ejecución se encarga de toda la inicialización de la red. Una vez ejecutado se nos muestra el mensaje de finalización:

[19] Para más información, véase el Anexo: “Código de programación del Access Point”

TXString(" FINALIZADO!!\r\n",15); // Red inicializada

Una vez comentados todos los procedimientos que debe realizar el *Access Point* antes de poder interactuar con los nodos de la red, trataré las dos otras funciones que se han programado específicamente para el proyecto. Estas dos funciones son las siguientes:

void transmitByte(char pre, unsigned char val); // Preparación del envío
void TXString(char* string, int length); // Transmisión del mensaje (mensaje+longitud)

La primera de estas dos funciones se encarga de la preparación del envío. En este procedimiento se genera el mensaje que saldrá por pantalla.

La segunda función realiza la transmisión del mensaje. Lo único que hay que introducirle es la cadena de caracteres a enviar y la longitud de ésta.

Comentadas las funciones programadas en el MSP430F2274 del *Access Point*, y para finalizar con el código de programa destinado a éste, hacer referencia a las dos posibles situaciones en las que se puede encontrar nuestro punto de acceso:

1. Si la situación es que uno de los nodos de la red pretende conectarse a nuestro nodo central, esta situación hará que se active la variable siguiente y por consiguiente todos los procesos que le preceden:

   static uint8_t sJoinSem; // Posible conexión con otro dispositivo

2. La otra hipotética situación es que nuestro nodo central reciba un paquete proveniente de uno de sus nodos. Esta situación hará que se active la variable siguiente y, al igual que en el caso anterior, todos los procesos que le preceden:

   static uint8_t sPeerFrameSem; // Recibimos el paquete de un dispositivo de la red

3.2.2 Código de programación del End Device

Al igual que en el apartado anterior, comentaré únicamente los procedimientos más importantes[20] programados en el microprocesador del *End Device*.

Gran parte del código de programación del ED es muy similar al del AP, ya que los procedimientos de inicialización de red y de transmisión de mensajes son los mismos. Por lo tanto comentaré únicamente la función declarada específicamente para el dispositivo final:

void linkTo(void); // Creación del enlace

[20] Para más información, véase el Anexo: "Código de programación del End Device"

Como hemos podido comprobar en el apartado anterior, los primeros procedimientos que debemos realizar son los de generación de direcciones e inicialización de la MCU. Una vez realizadas estas dos operaciones, el programa ejecutará el comando mencionado antes para crear un enlace con el nodo central de la red (AP). Creado el enlace ya seremos capaces de enviar paquetes de datos.

Lo que se ha hecho en esta función ha sido declarar dos buffers de almacenamiento, y a su vez dos variables que nos indican cual de los dos buffers se está utilizando y cual de los dos buffers se está enviando:

```
RXUartBuf[MAXRXBUFSIZE]; // Primer buffer
RXUartBuf[MAXRXBUFSIZE]; // Segundo buffer
qui_guarda // Identificador del buffer que se está utilizando
pot_enviar // Identificador del buffer que se está enviando
```

El motivo ha sido el poder asegurar que mientras se envían datos hacia el AP, los datos que siguen llegando a nuestro ED desde el sensor no se pierden. Tanto las variables “qui_guarda” como “pot_enviar” pueden adoptar los valores 1 y 2, indicando de este modo a cual de los dos buffers se hace referencia.

Al iniciarse el proceso de utilización de uno de los buffers, es decir, almacenamiento de datos, si la variable “qui_guarda” no dice lo contrario se empezarán guardando los valores en el primer buffer. Cuando el buffer que se esté utilizando llegue a su tope (MAXRXBUFSIZE), se indicará mediante la variable “pot_enviar” el número del buffer lleno y de este modo se enviará su contenido. Mientras se hace el envío los datos que siguen llegando pasarán a almacenarse en el otro buffer, modificándose así el valor del indicador “qui_guarda”. Y así sucesivamente.

Mencionar que durante el proceso de envío de datos hacia el AP el LED verde permanece encendido, puesto que está recibiendo datos por parte del sensor, y el rojo permanece parpadeado. Una vez finalizado el envío del buffer, éste es inicializado a 0.

3.3 Adaptación al sensor

Dejando de lado los módulos compatibles con ZigBee (eZ430-RF2500), otro elemento muy importante en la implementación del proyecto es el sensor. El sensor será la fuente que nos proporcione los datos que enviaremos de forma inalámbrica.

En un caso ideal, nuestros dispositivos ZigBee deben ser compatibles con una gran gama y variedad de sensores para que, de este modo, se puedan hacer diferentes medidas de diferentes variables del cuerpo (ritmo cardíaco, temperatura de la piel,…).

Como se puede suponer pocas cosas son ideales, y nuestro caso no es una excepción, por lo tanto el proyecto ha sido diseñado e implementado para ser compatible con un único tipo de sensor. Este sensor es un oxímetro[21] del fabricante Nonin. En la Figura 20 podemos observar el oxímetro que se ha utilizado.

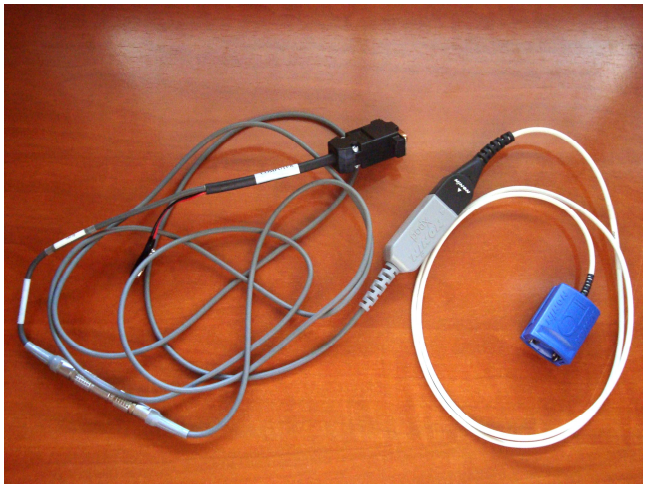

Figura 20. Sensor utilizado para la implementación del proyecto (Oxímetro Nonin)

Una de las partes importantes del proyecto consiste en una buena adaptación entre el sensor y la tarjeta programada como *End Device*. Si la adaptación es la correcta, la transferencia de datos de un componente al otro y posteriormente el envío de éstos será satisfactorio.

En el momento de la transferencia de datos del sensor a la tarjeta encontramos tres obstáculos:

1. Tipo de conector del sensor (RS232).
2. Diferencia en el nivel de energía de los componentes.
3. Paso de los datos al dispositivo ZigBee.

En la Figura 21 podemos ver representados los principales bloques que posteriormente deberán ser implementados en una placa.

Figura 21. Bloques implementados en la placa adaptadora entre el sensor y el dispositivo

Nuestro oxímetro utiliza un conector del tipo RS232, por lo tanto debemos suministrar a la placa un conector que sea compatible con el conector del sensor. Para

[21] Oxímetro: Aparato destinado a la medición del nivel de oxígeno en sangre. A su vez, también es posible la medición del ritmo cardíaco.

ello se ha soldado a la placa un terminal RS232 “macho” para poder conectar el sensor y hacer la adquisición de los datos capturados por el lector de éste. En la Figura 22 podemos observar la implementación del conector a la placa.

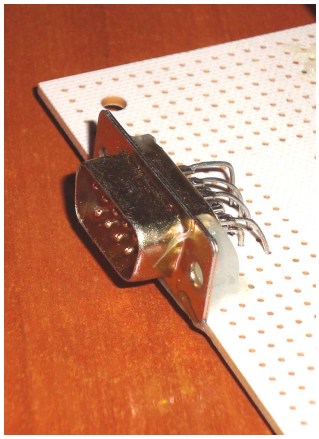

Figura 22. Conector RS232

El segundo de los problemas ocurre a causa de la diferencia de potenciales que se les deben suministrar a los dos componentes. Por un lado, al oxímetro se le debe suministrar una energía de entre unos 2 a 6V, en cambio a nuestro dispositivo ZigBee se le proporciona un suministro de voltaje máximo de unos 3.6V. Esta diferencia de voltajes es un inconveniente a la hora de transferir los datos. Lo que se ha hecho ha sido emplear un conversor para hacer una bajada en tensión. A este tipo de conversor se le denomina MAX3232. En la Figura 23 podemos ver representado el conversor que se ha utilizado.

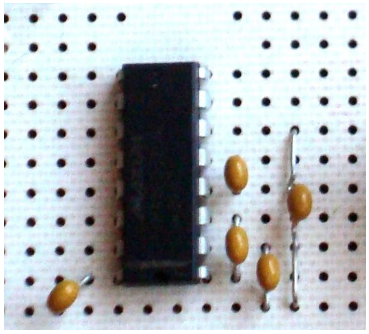

Figura 23. Conversor MAX3232

A lo que refiere a la entrada de datos al dispositivo ZigBee, se ha tenido que buscar algún método para hacer tal tarea. Lo que se ha hecho ha sido soldarle a la base de la batería tres pins. Con estos tres pins de entrada estamos consiguiendo conectar el dispositivo a tierra (pin 1 - GND), al suministro de voltaje (pin 2 – VCC (3.6V)) y a la entrada de datos por parte del sensor (pin 3 – P3.5 / UCA 0RXD / UCA 0SOMI). En la Figura 24 se muestra un esquemático de los pins utilizados del soporte de la batería, y en la Figura 25 una imagen de los pins soldados al soporte.

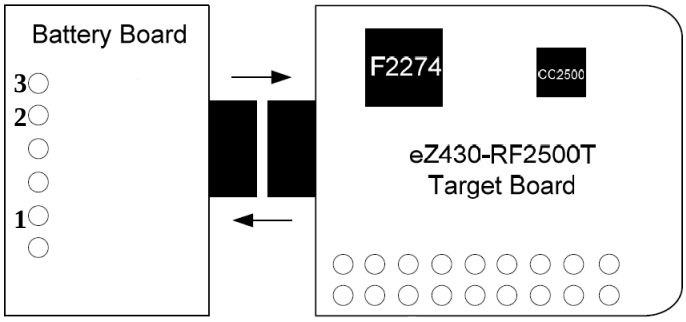

Figura 24. Esquemático de los pins utilizados del soporte de la batería [14]

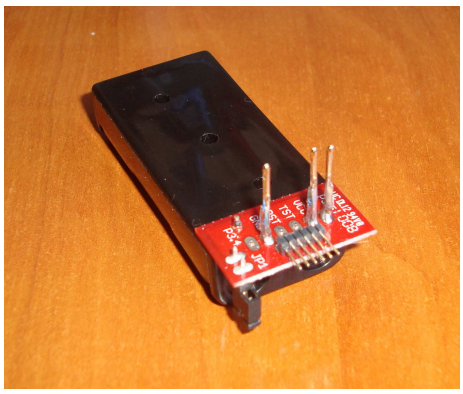

Figura 25. *Input* pins del dispositivo

En el momento de creación de la placa se ha hecho uso, como guía, del esquemático de un circuito que también utiliza el conversor MAX3232. No solo queda representada la utilización de dicho conversor, sino que también representa la utilización de un conector RS232 y la entrada de pins (*input*). En la Figura 26 se observa el esquemático utilizado como guía y en la Figura 27 una imagen final de la placa realizada donde se pueden observar los distintos elementos comentados anteriormente.

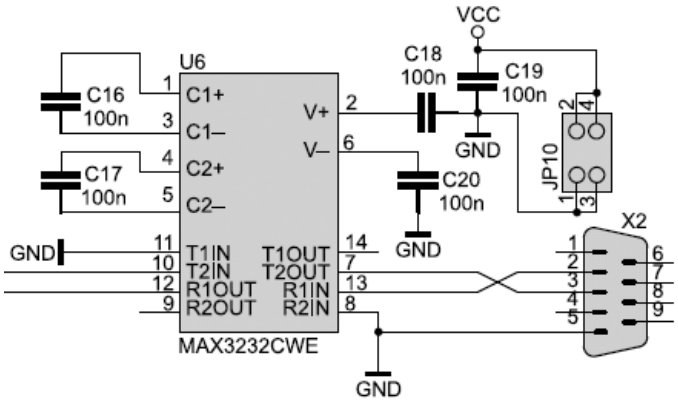

Figura 26. Esquemático del circuito (RS232 + MAX3232) [19]

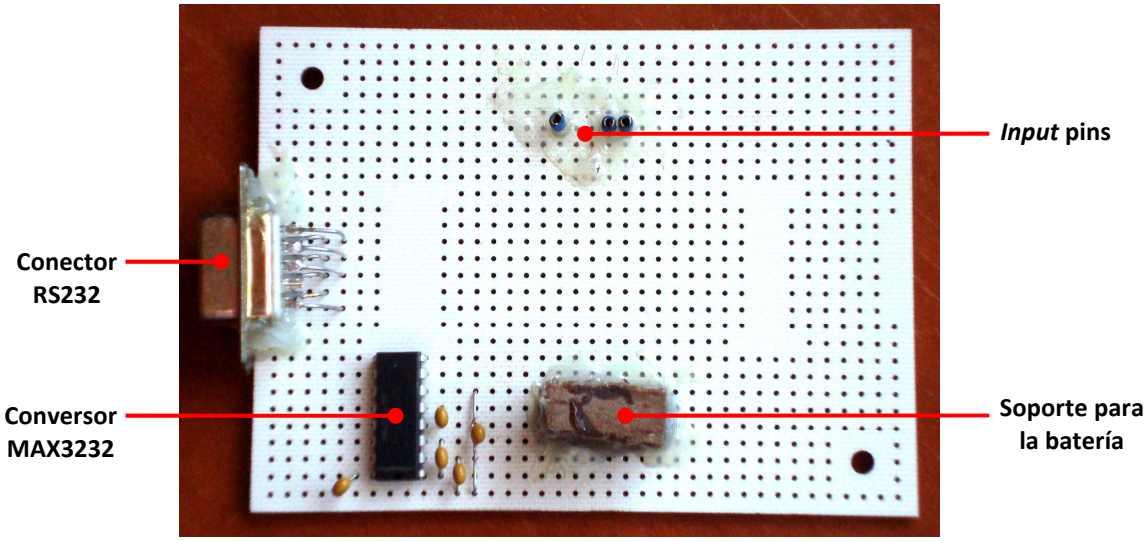

Figura 27. Placa adaptadora final

En el anexo, apartado "Placa adaptadora", se han adjuntado diferentes imágenes en las cuales puede verse representada la placa desde diferentes ángulos.

3.4 Monitorización de los datos

Una vez finalizada la programación de los dispositivos y la implementación de la placa, el siguiente paso, como es lógico, es comprobar si funciona y si se reciben los datos capturados por el lector del sensor.

3.4.1 Monitorización mediante HyperTerminal

En un primer momento, para visualizar si los datos enviados se reciben de forma correcta, se utilizó el aplicativo "HyperTerminal". Gracias a este aplicativo somos capaces de ver si se recibe algo por el puerto serie del ordenador y a su vez, si esto no fuese así, poder modificar el código del programa para realizar pruebas con los dispositivos.

Una vez instalado el aplicativo, lo siguiente que hay que hacer es crear una conexión entre el propio aplicativo y el puerto serie del ordenador. Para ello debemos indicar el puerto de conexión que se va a utilizar, en nuestro caso el COM4. Seguidamente, al igual que vimos en el apartado 3.1, se debe configurar el puerto con los mismos parámetros utilizados anteriormente. Finalmente ya solo queda aceptar y de este modo ya se habrá creado un vínculo entre puerto serie y aplicativo.

Creada la conexión se comprobó si el envío de datos entre *End Device* y *Access Point* se realizaba de forma correcta. En la Figura 28 podemos visualizar los datos que se reciben en nuestro puerto serie por parte de nuestro dispositivo final.

Figura 28. Monitorización de los datos con HyperTerminal

Con la primera monitorización de datos podemos llegar a dos conclusiones:

- La primera conclusión es positiva ya que comprobamos como sí se reciben datos mandados por parte del sensor. A partir de esta observación se puede confirmar como la transferencia de datos medidos por el lector del sensor hacia la tarjeta programada como *End Device* se produce de forma correcta y que posteriormente el envío y recepción de los datos también es satisfactorio.
- La segunda de las conclusiones no es tan positiva puesto que los datos recibidos no son interpretables en un primer momento. Con ello debemos plantearnos la descodificación de los datos recibidos para poder obtener valores razonables y entendibles por parte del usuario.

Visto que el objetivo de la transmisión de datos desde un emisor hasta un receptor se realiza de forma correcta, nos aparece una nueva tarea que consiste en la descodificación e interpretación de la información recibida. Para ello se deberá conocer el formato de envío de datos por parte del sensor y posteriormente generar un algoritmo que nos permita descifrar los caracteres recibidos.

Para poder trabajar de forma más cómoda con los datos recibidos, el aplicativo de HyperTerminal nos ofrece la posibilidad de guardar los datos recibidos en un fichero .txt. De este modo se podrán utilizar estos ficheros para generar las funciones y procedimientos que nos posibiliten la descodificación.

3.4.2 Descodificación de los datos recibidos

Para el proceso de descodificación de los datos se ha utilizado el documento "Xpod Module Specification and Technical Information" del oxímetro de Nonin[20].

El documento nos informa de lo siguiente: el sensor envía 3 paquetes cada segundo, donde cada paquete consiste en 25 tramas, y cada una de estas tramas está formada por 5 bytes. El paquete informativo generado por el sensor lo podemos observar en la Figura 29.

Frame

Packet

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5
1	01	STATUS	PLETH	HR MSB	CHK
2	01	STATUS	PLETH	HR LSB	CHK
3	01	STATUS	PLETH	SpO_2	CHK
4	01	STATUS	PLETH	REV	CHK
5	01	STATUS	PLETH	reserved	CHK
6	01	STATUS	PLETH	reserved	CHK
7	01	STATUS	PLETH	reserved	CHK
8	01	STATUS	PLETH	reserved	CHK
9	01	STATUS	PLETH	SpO2-D	CHK
10	01	STATUS	PLETH	SpO_2 Fast	CHK
11	01	STATUS	PLETH	SpO_2 B-B	CHK
12	01	STATUS	PLETH	reserved	CHK
13	01	STATUS	PLETH	reserved	CHK
14	01	STATUS	PLETH	E-HR MSB	CHK
15	01	STATUS	PLETH	E-HR LSB	CHK
16	01	STATUS	PLETH	E-SpO_2	CHK
17	01	STATUS	PLETH	E-SpO_2-D	CHK
18	01	STATUS	PLETH	reserved	CHK
19	01	STATUS	PLETH	reserved	CHK
20	01	STATUS	PLETH	HR-D MSB	CHK
21	01	STATUS	PLETH	HR-D LSB	CHK
22	01	STATUS	PLETH	E-HR-D MSB	CHK
23	01	STATUS	PLETH	E-HR-D LSB	CHK
24	01	STATUS	PLETH	reserved	CHK
25	01	STATUS	PLETH	reserved	CHK

Figura 29. Representación del paquete informativo generado por el sensor [20]

Tal y como podemos observar en la Figura 29, el byte que nos interesa descifrar es el cuarto byte. En éste podemos ver como se representa tanto el valor para el pulso cardíaco (HR) como el nivel de oxígeno en sangre (SpO2). El resto de bytes son útiles para informarnos del inicio de una nueva trama, del estado del sensor o de si el envío de las tramas se realiza de forma correcta.

Visto que el byte que nos interesa es el cuarto, debemos saber cuál es el formato para generar los valores del HR y del SpO2. En las Figuras 30 y 31 podemos ver sus formatos genéricos, respectivamente.

	7	6	5	4	3	2	1	0
HR MSB	X	X	X	X	X	X	HR8	HR7

	7	6	5	4	3	2	1	0
HR LSB	X	HR6	HR5	HR4	HR3	HR2	HR1	HR0

Figura 30. Representación del formato genérico del HR

	7	6	5	4	3	2	1	0
SPO_2	X	SP6	SP5	SP4	SP3	SP2	SP1	SP0

Figura 31. Representación del formato genérico del SpO2

De la Figura 30 podemos observar como el valor del pulso cardíaco está formado por 9 bits, donde los dos primeros bits del HR MSB representan los bits más significativos (MSB – *Most Significant Bit*) y los siete bits del HR LSB representan los bits menos significativos (LSB – *Less Significant Bit*). De este modo habrá que tener presente el juntar las dos tramas para así obtener el valor completo del *Heart Rate*.

En el caso del nivel de oxígeno en sangre, la Figura 31 nos informa que el valor lo podemos obtener de los siete primeros bits del SPO2.

Esta información será importante saberla a la hora de visualizar los datos ya que de este modo sabremos cuales son los bytes a los que deberemos acudir para conocer las medidas capturadas.

El siguiente paso es el de descodificar los datos recibidos. Es por ello que se han generado un seguido de funciones y procedimientos que nos permiten analizar los datos recibidos para que finalmente puedan ser mostrados por pantalla de forma correcta. Todo este trabajo de conversión se ha realizado mediante funciones y procedimientos implementados en Visual Basic[22].

Al igual que en los códigos de programa realizados para los microprocesadores de los dispositivos, se comentarán los principales procedimientos generados para la descodificación de los datos recibidos.

El primer paso que debe realizarse es el de obtención de los datos. Para ello debemos abrir el fichero donde se encuentran los datos recibidos por el *Access Point* y seguidamente almacenarlos en una matriz. Una vez ejecutada la tarea deberemos manipular la matriz generada para adecuarla a nuestros intereses.

[22] Para más información, véase el Anexo: “Código de programación del aplicativo de Visual Basic”

Los comandos siguientes son los utilizados para la recopilación de los datos en una matriz (en nuestro caso se llamará "bytes"):

```
Open "Fichero" For Binary As #1 // Abrimos fichero
Do While cont <= UBound(bytes)
   Get #1, , bytes(cont) // Almacenamos los datos del fichero en nuestra matriz
   cont = cont + 1
   DoEvents
Loop
Close #1 // Cerramos fichero
```

A continuación se realizarán un seguido de tareas para la adaptación de los datos recibidos:

- Búsqueda del inicio del paquete: Este proceso es fundamental para poder obtener los datos que nos interesan del paquete. Puede darse el caso de que a nuestro AP le llegue un paquete ya iniciado y que, por lo tanto, esté a medias. Este hecho puede ocasionar que no se sepa donde se encuentran los datos útiles (recordamos que es en el byte 4). Para realizar este proceso nos hemos ayudado del primer y segundo byte de las tramas. Tal y como nos indican en la Figura 29, el byte 1 siempre estará formado por un 1 (01). El byte 2, tal y como nos informan en el documento del sensor, nos indica el estado del sensor. Este byte está formado por 8 bits donde el último siempre será un 1, por lo tanto su valor siempre será mayor o igual a 128 (2 elevado a la 7). A su vez nos informan de que la primera trama de un paquete, en el byte 2 su primer bit (SYNC) será un 1, mientras que en las 24 tramas restantes será 0. Por lo tanto, sabiendo que el byte 1 siempre es 1 y que el byte 2 es siempre mayor que 128, y en concreto la primera trama del paquete es impar, cuando encontremos un 1 seguido de un 128+1 habremos encontrado el inicio de un paquete. Una vez encontrado el inicio, todos los bytes que le preceden deberán ser eliminados de nuestra matriz. En las siguientes instrucciones se realiza la tarea anteriormente descrita.

```
// Comprobamos donde está el inicio de un paquete (1 + 128-impar)
// En offset almacenamos el nº de caracteres que no son el inicio
Do While (True)
   If c(offset) = 1 Then
      b = toBits(c(offset + 1))
      If b(0) = 1 And b(7) = 1 Then
         Exit Do
      End If
   End If
      offset = offset + 1
Loop
```

- Detección de caracteres repetidos: Otra situación posible que debemos plantearnos es la de recibir paquetes de información en el cual hayan caracteres repetidos de forma consecutiva. Esta situación puede darse por motivos tales como una posible mala transferencia entre emisor y receptor. La primera prueba que se realiza es comprobar si cada 5 bytes encontramos un 1. Si no es así significará que hay algún byte de más y que deberá ser eliminado. En el momento de la eliminación del byte repetido iremos comparando los bytes que se encuentran entre dos 1, y en el momento en que se encuentre dos bytes seguidos iguales se procederá a su eliminación. Pensar que si se elimina un byte de la trama el tamaño de la matriz se verá modificado (en este caso reducido), por lo tanto mediante la función "shift_esq" conseguimos desplazar toda la matriz 'x' posiciones a la izquierda, donde 'x' es el número de caracteres repetidos en una trama. En las siguientes instrucciones se realiza la tarea anteriormente descrita.

```
// Detectamos si hay caracteres repetidos y los borramos
n = offset
For i = 0 To UBound(packets) - 1
   For j = 1 To 25
      If c(n) = 1 And c(n + 5) = 1 Then
         n = n + 5
      Else
         For k = 1 To 5
            If c(n) = c(n + 1) Then
               c = shift_esq(c, n): k = k - 1 // Desplazamos el array a la izquierda
            Else
               n = n + 1
            End If
         Next k
      End If
   Next j
Next i
```

- Creación de un fichero binario: Como último proceso se crea un fichero "n.txt" que contendrá los diferentes paquetes del fichero obtenido en un principio, paquetes y tramas correctamente diferenciados, y todo ello representado de forma binaria. Con la obtención de este fichero seremos capaces de comprobar si los datos recibidos son correctos y de este modo asegurarnos de que todo está funcionado de forma adecuada. En las siguientes instrucciones se realiza la tarea anteriormente descrita.

```
// Creamos el fichero n.txt que contiene nuestros paquetes en binario
n = offset
Open "n.txt" For Output As #1
For i = 0 To UBound(packets) - 1
   For j = 1 To 25
      s = ""
      For k = 1 To 5
         s = s & toBitsString(c(n)) & " "
         n = n + 1
      Next k
      Print #1, s
   Next j
   Print #1, ""
Next i
Close #1
```

A parte de los tres procesos descritos anteriormente, se han generado un seguido de otras funciones que nos permiten la descodificación de los datos recibidos. Todo este seguido de funciones y procedimientos han sido implementados en Visual Basic para que, finalmente, los datos recibidos puedan ser monitorizados y visualizados de una manera más cómoda a través de un aplicativo realizado por nosotros.

3.4.3 Monitorización mediante Visual Basic

Visto que la recepción de datos es correcta pero que se requiere de una conversión se decidió crear un aplicativo propio de visualización de datos que, aparte de monitorizar los datos recibidos por el puerto serie y de realizar la pertinente conversión, pueda hacer un seguido de operaciones distintas. Para ello se empleó el entorno de programación "Visual Basic".

Nuestro aplicativo VB nos permite realizar dos modos de lectura de datos diferentes: ya sea a partir de un fichero ya guardado por nosotros, o bien a partir de la recepción instantánea de datos mediante ZigBee y posterior lectura de éstos una vez almacenados.

Antes de describir las diferentes opciones que nos proporciona el aplicativo realizado con Visual Basic, en la Figura 32 podemos observar la máscara principal de nuestro aplicativo. Como se puede observar en la imagen podemos seleccionar dos orígenes de lectura (desde fichero o desde ZigBee), especificar el fichero que nos interesa tanto para leer como para guardar, y dos pantallas que nos mostrarán los datos recibidos mediante ZigBee (pantalla superior) o los datos una vez descodificados (pantalla inferior).

Figura 32. Máscara principal del aplicativo para la monitorización de los datos con Visual Basic

3.4.3.1 Lectura desde fichero

Una posibilidad que nos proporciona el aplicativo es la de realizar una lectura desde un fichero guardado con anterioridad y que contienen nuestras medidas corporales.

Esta opción es muy útil para la comparación de medidas realizadas en diferentes momentos y en diferentes estados del paciente. Recordando que uno de los principales intereses de este proyecto era el de poder comparar diferentes medidas corporales para poder establecer unos límites de estrés, esta opción es óptima para poder leer y comparar medidas de diferentes archivos generados en diferentes situaciones del paciente (relajado/excitado).

Figura 33. Monitorización de los datos con Visual Basic (desde fichero)

Como se puede observar en la Figura 33, el funcionamiento del aplicativo es muy sencillo. Seleccionamos el origen de lectura, en este caso desde fichero, examinamos la ubicación del fichero y finalmente presionamos sobre el botón "Leer Fichero". Como podemos comprobar en la pantalla inferior se monitorizan los datos almacenados en el fichero seleccionado, donde aparecen los valores para el ritmo cardíaco (HR) y para el nivel de oxígeno en sangre (SPO2).

3.4.3.2 Lectura desde ZigBee

En este caso la lectura escogida es desde ZigBee. Al querer monitorizar los datos que nos llegan *in situ* mediante ZigBee, previamente deberá seleccionarse el fichero que se utilizará para almacenar la información.

Hacer constancia de la diferencia de función de este fichero con el anterior. Mientras que el fichero seleccionado en el caso anterior nos hacía de base de datos para la monitorización, en este caso nuestro fichero nos hará de base de datos para el almacenamiento.

Si se pretende la monitorización desde ZigBee, deberá existir una comunicación entre nuestro aplicativo y el puerto serie del ordenador. En Visual Basic encontramos una herramienta que nos es muy útil para este tipo de proceso que queremos realizar puesto que este control nos permite la comunicación de una aplicación VB con el puerto serie. Al control se le conoce como "Control Personalizado Microsoft COMM" (MSComm) y en la Figura 34 podemos ver representado su icono.

Figura 34. Icono del Control Personalizado Microsoft COMM (MSComm)

Este control, al contactar directamente con nuestro puerto serie, se le debe programar un seguido de propiedades. Una de estas propiedades tiene que ver con las mencionadas en el apartado 3.1 y se basa en la configuración de la velocidad de transmisión, control de paridad, bits de información y bits de parada. En el código que se muestra a continuación se ven reflejadas las propiedades más importantes que se han tenido que programar.

```
MSComm1.CommPort = 4
MSComm1.PortOpen = True
MSComm1.Settings = "9600,N,8,1"
MSComm1.Handshaking = comNone
MSComm1.InputLen = 0
MSComm1.RThreshold = 1
```

Del código de programa anterior se pueden extraer un seguido de comentarios:

- Las primeras dos líneas del código hacen referencia al puerto serie de nuestro ordenador. La primera de éstas indica el número de puerto que utilizaremos, en nuestro caso será el 4. Con la segunda línea lo que indicamos es que el puerto indicado anteriormente esté abierto y por lo tanto disponible para su uso.
- La configuración de los ajustes de transferencia de datos es muy importante y debe coincidir con la establecida en la configuración de los dispositivos (3.1).
- El *Handshaking* especifica el método de control sobre el flujo de información. En una comunicación serie se necesita conocer si el puerto puede enviar información y necesita indicarle al módem que él está preparado para recibir. En nuestro caso no existe control de flujo.
- El parámetro "InputLen" hace referencia al buffer de recepción. Si indicamos que éste valga 0 hacemos que se lea el buffer por completo.
- Según el valor de la propiedad "RThreshold" produce el evento "OnComm". En nuestro caso vale 1, esto significa que con la recepción de 1 carácter se activará el evento mencionado anteriormente.

Una vez configurada nuestra herramienta de comunicación con nuestro puerto serie, ya sólo queda indicar el tipo de lectura y el fichero de almacenamiento. En la Figura 35 podemos observar la recepción de datos mediante ZigBee una vez seleccionado el estado de recepción "Inicio".

Figura 35. Monitorización de los datos con Visual Basic (desde ZigBee) - I

Observamos como los datos recibidos desde ZigBee están codificados y lo que se recibe son caracteres. Para poder obtener estos datos de forma descifrada debemos presionar sobre el botón “Leer Fichero” para que se ejecuten nuestras funciones de descodificación. En la Figura 36 podemos comprobar cómo se monitorizan los datos codificados pero esta vez de forma entendible.

Figura 36. Monitorización de los datos con Visual Basic (desde ZigBee) - II

Para finalizar con la monitorización de los datos hacer una última muestra del correcto funcionamiento de nuestro diseño. En la Figura 37 podemos observar cómo queda representado el HR=511 y el SPO2=127. Estos dos valores son valores de error [20]. Éstos se muestran por pantalla siempre que nuestro sensor detecte que no hay ningún dedo encima de su lector.

Figura 37. Monitorización de los datos con Visual Basic (desde ZigBee / sin dedo)

3.4.4 Monitorización mediante aplicativo Nonin

Para finalizar, hacer constancia de que nuestro diseño implementado con ZigBee puede hacerse servir con otros aplicativos de monitorización de datos.

Con el oxímetro de Nonin se nos proporciona un CD en el cual se encuentra un aplicativo que nos posibilita la visualización de nuestro ritmo cardíaco y de nuestro nivel en sangre.

En un principio este aplicativo podía utilizarse únicamente si nuestro ordenador venía equipado con una entrada RS232, ya que esta entrada es necesaria para poder conectar el oxímetro al PC. Una vez finalizado el proyecto podemos garantizar que puede hacerse uso del aplicativo de Nonin sin tener una entrada de este tipo.

Puesto que nuestros dispositivos ZigBee van equipados con una interface USB, gracias al proyecto realizado podemos utilizar el aplicativo Nonin para visualizar los datos medidos por el sensor a través del USB.

A parte de posibilitar la opción de poder conectar el oxímetro en otro tipo de conector (en este caso USB, más común que el RS232), también podemos realizar las medidas desde una cierta distancia al ordenador gracias al *wireless* de ZigBee.

En la Figura 38 queda representada una monitorización del ritmo cardíaco y del nivel de oxígeno en sangre haciendo uso del aplicativo de Nonin y mediante ZigBee (y no de cableado).

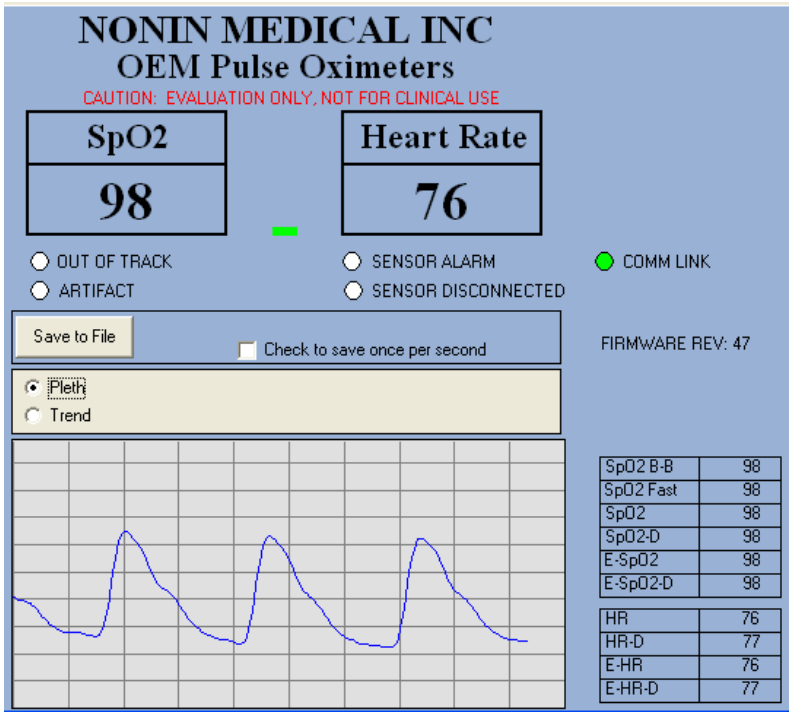

Figura 38. Monitorización de los datos con el aplicativo de Nonin (mediante ZigBee)

Conclusiones

Una vez finalizado el proyecto podemos llegar a un seguido de conclusiones valorando el trabajo y los resultados obtenidos.

Del estudio realizado al medio de transmisión se ha extraído la conclusión de que ZigBee es la mejor opción para el tipo de proyecto que se pretende llevar a cabo. Vistas que las principales propiedades de ZigBee son el bajo coste y el bajo consumo, esto nos asegura que para un proyecto al que la durabilidad de sus dispositivos es el principal requisito ZigBee es el medio que estamos buscando.

Seguidamente se realizó el análisis de los diferentes dispositivos compatibles con nuestro medio que hay en el mercado. De dicho análisis se obtuvo que los módulos de Texas Instrument son los más adecuados para nuestro proyecto y para nuestras peticiones. El soporte de batería y la interface USB que facilita la transmisión de datos a un PC fueron los dos principales motivos por los cuales se escogió el kit eZ430-RF2500. Mencionar que la implementación del proyecto es posible con módulos de otros fabricantes, por lo tanto se propone la posibilidad de realizar el mismo trabajo con otros dispositivos y de este modo extraer nuevas conclusiones y resultados.

Iniciada la implementación del proyecto, se empezaron a obtener los primeros resultados prácticos. De esta parte más práctica decir que los resultados obtenidos son satisfactorios puesto que con ellos cumplimos el objetivo principal establecido en un principio, la transmisión y recepción de datos a través de un medio inalámbrico. A parte de la correcta transmisión de los datos sin cable, otro proceso que se ha conseguido desarrollar con éxito ha sido la descodificación de los datos. Éste fue uno de los principales problemas que nos encontramos a la hora de implementar el proyecto puesto que los datos que recibíamos no eran entendibles para nosotros. Finalmente, y tras una larga labor de estudio de su codificación y generación del algoritmo para descodificar, se obtuvo una más que aceptable interpretación y monitorización de los resultados.

Para finalizar, y dejando de lado la satisfacción de haber obtenido resultados positivos, hacer mención que con el trabajo desempeñado en este proyecto se ha podido hacer uso de diferentes conocimientos adquiridos a lo largo de la carrera (conocimientos acerca de protocolos de transmisión de datos, electrónica aplicada a la circuitería de la placa adaptadora, utilización de lenguajes de programación, etc.).

- **Líneas futuras:**

Una vez valorado todo el trabajo realizado y los resultados obtenidos, sería acertado proponer un seguido de líneas futuras que podrían seguirse una vez finalizado el proyecto.

Del trabajo generado podrían retocarse y/o modificarse un seguido de puntos:

- Retocar el código de programación de los dispositivos: Con ello se revisaría todo el código realizado, tanto para el *Access Point* como para el *End Device*, y de este modo podrían detectarse posibles erratas. A su vez podrían programarse nuevas instrucciones que realizasen el mismo proceso en menor tiempo.
- Reducir el tamaño de la placa adaptadora: Uno de los principales requisitos del diseño del proyecto era el tamaño reducido que debería tener la adaptación entre sensor y dispositivo. Éste sería uno de los puntos a modificar, pero puesto que se trata de un prototipo de momento se podría pasar por alto.
- Perfeccionar la circuitería de la placa adaptadora: La conversión y transmisión de datos del sensor al dispositivo podría ser no muy óptima puesto que la circuitería utilizada es muy básica. Sumándole que la placa ha sido manipulada continuamente, todo ello puede provocar que su estado no sea el mejor.

A partir del trabajo y resultado obtenido podría seguirse trabajando en los siguientes puntos:

- Utilización de módulos de otros fabricantes: Obtenidos unos resultados a partir del uso de dispositivos de Texas Instrument, sería bueno el utilizar otros dispositivos para poder comparar resultados a partir de nuevos materiales.
- Adaptación de los dispositivos a nuevos sensores biomédicos: Uno de los propósitos del proyecto era el de poder utilizar una amplia gama de sensores biomédicos. En nuestro caso se hizo uso de un oxímetro, por lo tanto el siguiente paso sería ampliar la gama de sensores compatibles con nuestros módulos ZigBee.

Anexo

Ficha técnica XBee ZNet 2.5 RF Module

- **Funcionamiento:**
  - Potencia de salida:
    - En modo de iniciativa: 2mW (+3dBm)
    - En modo normal: 1.25mW (+1dBm)
  - Distancia en interior: 40 metros
  - Distancia en exterior: 120 metros
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
  - Sensibilidad en recepción:
    - En modo de iniciativa: -98dBm
    - En modo normal: -97dBm
- **Interconexión:**
  - Espectro Ensanchado por Secuencia Directa: DSSS (*Direct Sequence Spread Spectrum*)
  - Topología de redes que soporta: Mesh, point-to-point y point-to-multipoint
  - Manejo de errores: Reintentos y reconocimiento (*Acknowledgements*)
  - Opciones de filtrado: PAN ID, Channel y direcciones de 64-bit
  - Capacidad de canales: 16 canales
  - Direcciones: 65000 direcciones de red para cada canal
- **Consumo:**
  - Suministración de voltaje: 2.1 - 3.6V
  - Consumo Tx:
    - En modo de iniciativa: 40mA
    - En modo normal: 35mA
  - Consumo Rx:
    - En modo de iniciativa: 40mA
    - En modo normal: 38mA
  - Potencia consumida en modo dormido: <1µA a 25°C
- **General:**
  - Banda de frecuencia: 2400 - 2483MHz

- Opciones de interface: 3V CMOS UART, (4) 10-bit ADC inputs, (10) remote-settable Digital I/O

- **Propiedades físicas:**
  - Tamaño: 2.438cm x 2.761cm
  - Peso: 3gramos
  - Opciones de antena: U.FL RF connector, chip antenna, whip antenna ó conector RPSMA RF
  - Temperatura: -40°C a 85°C

Ficha técnica XBee-PRO ZNet 2.5 RF Module

- **Funcionamiento:**
  - Potencia de salida:
    - En modo de iniciativa: 50mW (+17dBm)
    - En modo normal: 10mW (+10dBm)
  - Distancia en interior: 120 metros
  - Distancia en exterior: 1600 metros
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
  - Sensibilidad en recepción: -102dBm
- **Interconexión:**
  - Espectro Ensanchado por Secuencia Directa: DSSS (*Direct Sequence Spread Spectrum*)
  - Topología de redes que soporta: Mesh, point-to-point y point-to-multipoint
  - Manejo de errores: Reintentos y reconocimiento (*Acknowledgements*)
  - Opciones de filtrado: PAN ID, Channel y direcciones de 64-bit
  - Capacidad de canales: 16 canales
  - Direcciones: 65000 direcciones de red para cada canal
- **Consumo:**
  - Suministración de voltaje: 3.0 - 3.4V
  - Consumo Tx: 295mA
  - Consumo Rx: 45mA
  - Potencia consumida en modo dormido: <1µA a 25°C

- **General:**
  - Banda de frecuencia: 2400 - 2483MHz
  - Opciones de interface: 3V CMOS UART, (4) 10-bit ADC inputs, (10) remote-settable Digital I/O
- **Propiedades físicas:**
  - Tamaño: 2.438cm x 3.294cm
  - Peso: 3gramos
  - Opciones de antena: U.FL RF connector, chip antenna, whip antenna ó conector RPSMA RF
  - Temperatura: -40°C a 85°C

Ficha técnica EasyBee ZigBee Transceiver Module

- **Funcionamiento:**
  - Potencia de salida: 1mW (+0dBm)
  - Distancia: 120 metros
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
- **Interconexión:**
  - Espectro Ensanchado por Secuencia Directa: DSSS (*Direct Sequence Spread Spectrum*)
  - Modulación por cuadratura en offset: OQPSK (*Offset Quadrature Phase Shift Keying*)
  - Topología de redes que soporta: Mesh, point-to-point y point-to-multipoint
  - Based on ChipCon CC2420 RF controller:
    - 4-wire SPI interface
    - Separate 128-byte Rx & Tx FIFO buffers
  - Integrates with controller of developer's choice, e.g. PIC running Microchip stack or ATMEL running
  - Capacidad de canales: 16 canales
  - Direcciones: 65000 direcciones de red para cada canal
- **Consumo:**
  - Suministración de voltaje: 2.1 - 3.6V
  - Consumo Tx: 18mA

- Consumo Rx: 20mA

- **General:**
  - Banda de frecuencia: 2400 - 2483MHz
  - Cumple con FCC / CE
- **Propiedades físicas:**
  - Tamaño: 2.6cm x 2cm
  - Temperatura: -40°C a 85°C

Ficha técnica ZB-21 ZigBee OEM Module

- **Funcionamiento:**
  - Potencia de salida: 1mW (+0dBm)
  - Distancia en interior: 30 metros
  - Distancia en exterior: 100 metros
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
  - Sensibilidad en recepción: -90dBm
- **Interconexión:**
  - OKI ARM7 microprocessor up to 33MHz
  - 128k o 64kbytes of flash memory
  - 16kbytes of SRAM memory
  - 2kbytes of EEPROM memory
  - Topología de redes que soporta: Mesh, point-to-point y point-to-multipoint
  - 128-bit encryption security
- **Consumo:**
  - Suministración de voltaje: 3.3V
  - Consumo Tx: 50mA
  - Consumo Rx: 25µA
  - Potencia consumida en modo dormido: <25µA a 25°C
- **General:**
  - Banda de frecuencia: 2405 – 2480MHz
  - 16550 UART
  - SPI interface, I2C interface
  - 4 A/D inputs, 12 general purpose I/O (3 are 20mA capable), AT command set

- Apto para estándar ZigBee
- Módulo completo de RF

- **Propiedades físicas:**
  - Tamaño: 1.5cm x 2.7cm
  - Antena integrada
  - Temperatura: -20°C a 70°C

Ficha técnica MICAz Module

- **Funcionamiento:**
  - Potencia de salida: 1mW (+0dBm)
  - Distancia en interior: 20-30 metros
  - Distancia en exterior: 75-100 metros
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
  - Sensibilidad en recepción: -94dBm
  - Adjacent channel rejection: ±5MHz
- **Interconexión:**
  - Program flash memory: 128kbytes
  - Measurement flash: 512kbytes
  - Configuration EEPROM: 4kbytes
- **Consumo:**
  - Suministración de voltaje: 2.7V - 3.3V
  - Consumo Tx: 17.4mA
  - Consumo Rx: 19.7mA
  - Potencia consumida microcontrolador: Activo: 8mA / Dormido: <15µA
  - Potencia consumida transceiver: Activo: 20µA / Dormido: 1µA
- **General:**
  - Banda de frecuencia: 2400MHz a 2483.5MHz
  - Serial communications: UART
  - Interfaces: Digital I/O, I2C, SPI
  - Conversor Analógico/Digital: 10bit
  - Apto para estándar ZigBee
  - Módulo completo de RF

- Uso de 2xAA baterías
- 3 LEDs (rojo, verde y amarillo)
- Expansion connector: 51-pin

- **Propiedades físicas:**
  - Tamaño: 3.2cm x 5.8cm x 0.7cm (incluyendo batería)
  - Peso: 18gramos (incluyendo batería)
  - Antena integrada

Ficha técnica ATMEL Module

AT86RF230:

- **Funcionamiento:**
  - Potencia de salida: 4mW (+3dBm)
  - Tasa de transferencia: 250kbps
  - Frecuencia operativa: 2.4GHz
  - Sensibilidad en recepción: -101dBm
- **Características de transmisión:**
  - 128 bytes SRAM for Data Buffering
  - Programmable clock output to clock the host microcontroller or as timer reference
  - Integrated TX/RX switch
  - Fully integrated PLL with on-chip loop filter
  - Fast PLL settling time
  - Battery monitor
  - Fast power-up time < 1ms
  - Configuration EEPROM: 4kbytes
- **Consumo:**
  - Suministración de voltaje: 1.8V - 3.6V
  - Consumo Tx: 16.5mA
  - Consumo Rx: 15.5mA
  - Consumo modo dormido: 20nA
- **General:**
  - Registers and Frame Buffer Accessible through Fast SPI

- Only two microcontroller GPIO lines necessary
- One interrupt pin from radio transceiver
- Clock output with prescaler from radio transceiver
- 32-pin low-profile QFN
- RoHS/Fully green

- **Propiedades físicas:**
  - Temperatura: -40ºC a 85ºC
  - Algunos componentes situados de forma externa: cristal, capacidades y antena.
  - Robustez Excelente ESD
- **Hardware de soporte especial para IEEE 802.15.4:**
  - FCS computation
  - Clear cannel assessment
  - Energy detection / RSSI computation
  - Automatic CSMA-CA
  - Automatic Frame Retransmission
  - Automatic Frame Acnowledgement
  - Automatic Frame Filtering

ATmega1281:

- **Arquitectura avanzada RISC:**
  - 135 powerful instructions
  - Most single clock cycle execution
  - 32 x 8 general purpose working registers
  - Fully static operation
  - Up to 16 MIPS throughput at 16MHz
  - On-chip 2-cycle multiplier
- **Segmentos de memoria y su duración:**
  - 64K/128K/256K bytes of in-system self-programmable flash
  - 4K bytes EEPROM
  - 8K bytes internal SRAM
  - Wirte/erase cycles: 10.000 flash / 100.000 EEPROM
  - Data retention: 20 years at 85ºC / 100 years at 25ºC
  - Optional boot code section with independent lock bits

- **Consumo:**
  - Modo activo: 500µA
  - Modo pasivo: 0.1µA a 1.8µA
- **Características especiales del microcontrolador:**
  - Power-on reset and programmable brown-out detection
  - Internal calibrated osillator
  - External and internal interrupt sources
  - Six sleep modes: Idle, ADC noise reduction, power-save, power-down, standby and extended standby
  - Temperatura: -40ºC a 85ºC
- **Características periféricos:**
  - Two 8-bit timer/counters with separate prescaler and compare mode
  - Four 16-bit timer/counter with separate prescaler, compare-and capture mode
  - Real time counter with separate oscillator
  - Four 8-bit PWM channels
  - Six/twelve PWM channels with programmable resolution from 2 to 16 bits
  - Output compare modulator
  - 8/16-channel, 10-bit ADC
  - Two/four programmable serial USART
  - Master/slave SPI serial interface
  - Byte oriented 2-wire serial interface
  - Programmable watchdog timer with separate on-chip oscillator
  - On-chip analog comparator
  - Interrupt and wake-up on pin change

Compra del kit eZ430-RF2500

2008/06/13

MI CESTA

Nº de línea	Código Farnell	Cantidad	Nombre y ref. fabricante	Descripción producto	RoHS	Disponibilidad	Precio unitario	Precio por línea
1	1172234	1	TEXAS INSTRUMENTS EZ430-F2013	TOOL KIT, DEVELOPMENT, USB STICK		128	18,30 €	**18,30 €**
2	1382267	1	TEXAS INSTRUMENTS EZ430-RF2500	BOARD KIT, FOR MSP430	No	277	41,70 €	**41,70 €**
3	1359259	1	TEXAS INSTRUMENTS EZ430-T2012	BOARD KIT, FOR MSP430 USB STICK		124	9,20 €	**9,20 €**

Subtotal:	**69,20 €**
Portes:	A confirmar
IVA:	**11,07 €**
Total:	**80,27 €**

Código de programación del Access Point

```
//*****************************************************************************
//  PROGRAMA PARA EL ACCESS POINT -> IVÁN BARNEDA FAUDOT
//*****************************************************************************

// Declaramos las librerías
#include "bsp.h"
#include "mrfi.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "nwk_frame.h"
#include "nwk.h"
#include "msp430x22x4.h"
#include "vlo_rand.h"

// Declaramos las funciones y/o procedimientos
void createRandomAddress(); // Creación de las direcciones
void MCU_Init(void); // Inicialización de la MCU
void transmitByte(char pre, unsigned char val); // Preparación del envío
void TXString( char* string, int length ); // Transmisión del mensaje (mensaje+longitud)

__no_init volatile char Flash_Addr[4] @ 0x10F0;      // Reserva de memoria para almacenar las direcciones
                                                     // Es un vector de 4 posiciones, puesto que las direcciones son de 4 bytes

static linkID_t sLID[NUM_CONNECTIONS]; // Reserva de memoria para las nuevas conexiones
static uint8_t  sNumCurrentPeers; // Nº de conexiones
static uint8_t sCB(linkID_t); // Callback handler
static uint8_t sPeerFrameSem; // Recibimos el paquete de un dispositivo de la red
static uint8_t sJoinSem; // Posible conexión con otro dispositivo

void main (void)
{
  addr_t lAddr; // Variable creada para almacenar las direcciones
  bspIState_t intState; // Variable creada para indicar el estado del sistema

  WDTCTL = WDTPW + WDTHOLD; // Stop WDT

  // Bucle de retardo para asegurar un buen inicio
    volatile int i;
    for(i = 0; i < 0xFFFF; i++){}

  // Calibración de los dispositivos
  if( CALBC1_8MHZ == 0xFF )
  {
    volatile int i;
    P1DIR |= 0x03;
    BSP_TURN_ON_LED1();
```

```
  BSP_TURN_OFF_LED2();
  while(1)
  {
    for(i = 0; i < 0x5FFF; i++){}
    BSP_TOGGLE_LED2();
    BSP_TOGGLE_LED1();
  }
}

// Inicialización del hardware
BSP_Init();

if( Flash_Addr[0] == 0xFF &&
    Flash_Addr[1] == 0xFF &&
    Flash_Addr[2] == 0xFF &&
    Flash_Addr[3] == 0xFF )
{
  createRandomAddress(); // Llamamos a la función para obtener las direcciones
}

// Almacenamos las direcciones
lAddr.addr[0]=Flash_Addr[0];
lAddr.addr[1]=Flash_Addr[1];
lAddr.addr[2]=Flash_Addr[2];
lAddr.addr[3]=Flash_Addr[3];
SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr); //Configuración en tiempo de ejecución

// Inicializamos la MCU
MCU_Init();

// Inicialización de la red
TXString("\r\nInicializando Red...",22);

SMPL_Init(sCB);

// Red inicializada
TXString(" FINALIZADO!!\r\n",15);

// Bucle principal que se realizará siempre
while (1)
{
  // Condición que se cumplirá siempre que haya un ED que quiera conectarse (variable JoinSem activa)
  // y que el nº de dispositivos en la red sea menor que el máximo establecido
  if (sJoinSem && (sNumCurrentPeers < NUM_CONNECTIONS))
  {
    SMPL_LinkListen(&sLID[sNumCurrentPeers]);
    sNumCurrentPeers++;
    BSP_ENTER_CRITICAL_SECTION(intState);
    if (sJoinSem)
    {
      sJoinSem--;
    }
    BSP_EXIT_CRITICAL_SECTION(intState);
```

```
  // Condición que se cumplirá siempre que haya un paquete en espera (variable PeerFrameSem activa)
  if (sPeerFrameSem)
  {
    uint8_t  msg[MAX_APP_PAYLOAD], len, i;

    for (i=0; i<sNumCurrentPeers; ++i)
    {
      if (SMPL_Receive(sLID[i], msg, &len) == SMPL_SUCCESS)
      {
        ioctlRadioSiginfo_t sigInfo;
        sigInfo.lid = sLID[i];
        SMPL_Ioctl(IOCTL_OBJ_RADIO, IOCTL_ACT_RADIO_SIGINFO, (void *)&sigInfo);
        // Configuración en tiempo de ejecución

        if (len > 0)
        {
          transmitByte('A', len);
          uint8_t j;
          for (j=0; j<len; j++)
          {
            transmitByte(j%10 + 48, msg[j]);
          }
        }

        BSP_TOGGLE_LED2(); // Encendido del LED2 (verde) con la recepción de paquetes
       BSP_ENTER_CRITICAL_SECTION(intState);
       sPeerFrameSem--;
       BSP_EXIT_CRITICAL_SECTION(intState);
      }
    }
   }
  }
}

/*-------------------------------------------------------------------------
   ..CREACIÓN DE LAS DIRECCIONES..
-------------------------------------------------------------------------*/
void createRandomAddress()
{
  unsigned int rand, rand2;
  do
  {
    rand = TI_getRandomIntegerFromVLO(); // El primer byte no puede ser 0x00 o 0xFF
  }
  while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
  rand2 = TI_getRandomIntegerFromVLO();

  BCSCTL1 = CALBC1_1MHZ;            // Set DCO to 1MHz
  DCOCTL = CALDCO_1MHZ;
  FCTL2 = FWKEY + FSSEL0 + FN1;  // MCLK/3 for Flash Timing Generator
  FCTL3 = FWKEY + LOCKA;            // Clear LOCK & LOCKA bits
  FCTL1 = FWKEY + WRT;              // Set WRT bit for write operation
  Flash_Addr[0]=(rand>>8) & 0xFF;
  Flash_Addr[1]=rand & 0xFF;
```

```
  Flash_Addr[2]=(rand2>>8) & 0xFF;
  Flash_Addr[3]=rand2 & 0xFF;
  FCTL1 = FWKEY;     // Clear WRT bit
  FCTL3 = FWKEY + LOCKA + LOCK;          // Set LOCK & LOCKA bit
}

/*-------------------------------------------------------------------------
    ..INICIALIZACIÓN DE LA MCU..
-------------------------------------------------------------------------*/
void MCU_Init()
{
  BCSCTL1 = CALBC1_8MHZ;          // Set DCO
  DCOCTL = CALDCO_8MHZ;
  BCSCTL3 |= LFXT1S_2;            // LFXT1 = VLO
  TACCTL0 = CCIE;                 // TACCR0 interrupt enabled
  TACCR0 = 12000;                 // ~1 second
  TACTL = TASSEL_1 + MC_1;        // ACLK, upmode
  P3SEL |= 0x30;                  // P3.4,5 = USCI_A0 TXD/RXD
  UCA0CTL1 = UCSSEL_2;            // SMCLK
  UCA0BR0 = 0x41;                 // 9600 from 8Mhz
  UCA0BR1 = 0x3;
  UCA0MCTL = UCBRS_2;
  UCA0CTL1 &= ~UCSWRST;           // **Initialize USCI state machine**
  IE2 |= UCA0RXIE;                // Enable USCI_A0 RX interrupt
  __enable_interrupt();
}

/*-------------------------------------------------------------------------
    ..PREPARAMOS EL ENVÍO..
-------------------------------------------------------------------------*/
// Creación del mensaje que se pretende enviar
void transmitByte(char pre, unsigned char val)
{
  char msg[10];
  msg[0] = ' ';
  msg[1] = pre;
  msg[2] = ':';
  msg[3] = (val/100)%10 +48;
  msg[4] = (val/10)%10 +48;
  msg[5] = (val)%10 +48;
  msg[6] = '\r';
  msg[7] = '\n';
  msg[8] = 0;
  TXString(msg, 9);
}

/*-------------------------------------------------------------------------
    ..TRANSMITIMOS EL MENSAJE (MENSAJE+LONGITUD)..
-------------------------------------------------------------------------*/
void TXString( char* string, int length )
{
  int pointer;
  for( pointer = 0; pointer < length; pointer++)
  {
```

```
    volatile int i;
    UCA0TXBUF = string[pointer];
    while (!(IFG2&UCA0TXIFG)); // Mientras el buffer esté listo (ocupado)
  }
}

/*-------------------------------------------------------------------------------------------------
    ..CUANDO SE ACTIVE ESTA VARIABLE PUEDE DARSE EL CASO DE QUE
         SE RECIBA UN PAQUETE O QUE SE REALICE UNA CONEXIÓN..
  -------------------------------------------------------------------------------------------------*/
static uint8_t sCB(linkID_t lid)
{
  if (lid)
  {
    sPeerFrameSem++;
  }
  else
  {
    sJoinSem++;
  }
  return 0;
}

/*-------------------------------------------------------------------------
 * ADC10 interrupt service routine
 *-------------------------------------------------------------------------*/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF); // Clear CPUOFF bit from 0(SR)
}

/*-------------------------------------------------------------------------
 * Timer A0 interrupt service routine
 *-------------------------------------------------------------------------*/
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{

}

/*-------------------------------------------------------------------------
 * USCIA interrupt service routine
 *-------------------------------------------------------------------------*/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  char rx = UCA0RXBUF;
}
```

Código de programación del End Device

```
//*****************************************************************************
// PROGRAMA PARA EL END DEVICE -> IVÁN BARNEDA FAUDOT
//*****************************************************************************

// Declaramos las librerías
#include "bsp.h"
#include "mrfi.h"
#include "nwk_types.h"
#include "nwk_api.h"
#include "bsp_leds.h"
#include "bsp_buttons.h"
#include "vlo_rand.h"
#define MAXRXBUFSIZE 10

// Declaramos las funciones y/o procedimientos
void createRandomAddress(); // Creación de las direcciones
void MCU_Init(void); // Inicialización de la MCU
void linkTo(void); // Creación del enlace
void TXString( char* string, int length ); // Transmisión del mensaje (mensaje+longitud)

__no_init volatile char Flash_Addr[4] @ 0x10F0; // Reserva de memoria para almacenar las direcciones
                                                // Es un vector de 4 posiciones, puesto que las
                                                direcciones son de 4 bytes

// Variables declaradas relacionadas con el buffer de UART y los datos recibidos del RS232
volatile static uint8_t RXUartSize=0,RXUartSize2=0;
static uint8_t RXUartBuf[MAXRXBUFSIZE],RXUartBuf2[MAXRXBUFSIZE];
volatile int qui_guarda=1,pot_enviar=0;

void main (void)
{
 addr_t lAddr; // Variable creada para almacenar las direcciones

 WDTCTL = WDTPW + WDTHOLD; // Stop WDT
 // Bucle de retardo para asegurar un buen inicio
  volatile int i;
  for(i = 0; i < 0xFFFF; i++){}

 // Calibración de los dispositivos
 if( CALBC1_8MHZ == 0xFF )
 {
  volatile int i;
  P1DIR |= 0x03;
  BSP_TURN_ON_LED1();
  BSP_TURN_OFF_LED2();
  while(1)
  {
   for(i = 0; i < 0x5FFF; i++){}
   BSP_TOGGLE_LED2();
   BSP_TOGGLE_LED1();
  }
```

```
  }
  // SimpliciTI va a cambiar la configuración de los pins del modo siguiente
  P1DIR = 0xFF; // Dirección del Puerto 1
  P1OUT = 0x00; // Dirección del Puerto 1 Out
  P2DIR = 0x27;
  P2OUT = 0x00;
  P3DIR = 0xC0;
  P3OUT = 0x00;
  P4DIR = 0xFF;
  P4OUT = 0x00;

  // Inicialización del hardware
  BSP_Init();

  if( Flash_Addr[0] == 0xFF &&
      Flash_Addr[1] == 0xFF &&
      Flash_Addr[2] == 0xFF &&
      Flash_Addr[3] == 0xFF )
  {
    createRandomAddress(); // Llamamos a la función para obtener las direcciones
  }

  // Almacenamos las direcciones
  lAddr.addr[0]=Flash_Addr[0];
  lAddr.addr[1]=Flash_Addr[1];
  lAddr.addr[2]=Flash_Addr[2];
  lAddr.addr[3]=Flash_Addr[3];
  SMPL_Ioctl(IOCTL_OBJ_ADDR, IOCTL_ACT_SET, &lAddr); //Configuración en tiempo de ejecución

  // Inicialización de la MCU
  BCSCTL1 = CALBC1_8MHZ; // Set DCO after random function
  DCOCTL = CALDCO_8MHZ;
  BCSCTL3 |= LFXT1S_2; // LFXT1 = VLO
  TACCTL0 = CCIE; // TACCR0 interrupt enabled
  TACCR0 = 12000; // ~ 1 sec
  TACTL = TASSEL_1 + MC_1; // ACLK, upmode
  P3SEL |= 0x30; // P3.4,5 = USCI_A0 TXD/RXD
  UCA0CTL1 = UCSSEL_2; // SMCLK
  UCA0BR0 = 0x41; // 9600 from 8Mhz
  UCA0BR1 = 0x3;
  UCA0MCTL = UCBRS_2;
  UCA0CTL1 &= ~UCSWRST; // **Initialize USCI state machine**
  IE2 |= UCA0RXIE; // Enable USCI_A0 RX interrupt

  // Mientras el ED trata de contactar con el AP, los LEDs permanecen parpadeando
  while (SMPL_NO_JOIN == SMPL_Init((uint8_t (*)(linkID_t))0))
  {
    BSP_TOGGLE_LED1();
    BSP_TOGGLE_LED2();;
    __bis_SR_register(LPM3_bits + GIE);
  }
  // Llamada a la función linkTo al realizar una conexión con AP con éxito
linkTo();
}
```

```
/*-------------------------------------------------------------------------
    ..CREACIÓN DE LAS DIRECCIONES..
-------------------------------------------------------------------------*/
void createRandomAddress()
{
  unsigned int rand, rand2;
  do
  {
    rand = TI_getRandomIntegerFromVLO(); // El primer byte no puede ser 0x00 o 0xFF
  }
  while( (rand & 0xFF00)==0xFF00 || (rand & 0xFF00)==0x0000 );
  rand2 = TI_getRandomIntegerFromVLO();

  BCSCTL1 = CALBC1_1MHZ;           // Set DCO to 1MHz
  DCOCTL = CALDCO_1MHZ;
  FCTL2 = FWKEY + FSSEL0 + FN1;  // MCLK/3 for Flash Timing Generator
  FCTL3 = FWKEY + LOCKA;           // Clear LOCK & LOCKA bits
  FCTL1 = FWKEY + WRT;             // Set WRT bit for write operation

  Flash_Addr[0]=(rand>>8) & 0xFF;
  Flash_Addr[1]=rand & 0xFF;
  Flash_Addr[2]=(rand2>>8) & 0xFF;
  Flash_Addr[3]=rand2 & 0xFF;

  FCTL1 = FWKEY;     // Clear WRT bit
  FCTL3 = FWKEY + LOCKA + LOCK;          // Set LOCK & LOCKA bit
}

/*-------------------------------------------------------------------------
    ..CREACIÓN DEL ENLACE..
-------------------------------------------------------------------------*/
void linkTo()
{
  linkID_t linkID1;

  // Enlace que se realizará siempre que la conexión sea satisfactoria
  while (SMPL_SUCCESS != SMPL_Link(&linkID1))
  {
    __bis_SR_register(LPM3_bits + GIE);
    BSP_TOGGLE_LED1();
    BSP_TOGGLE_LED2();
  }

  // Apagado de todos los LEDs
  if (BSP_LED1_IS_ON())
  {
    BSP_TOGGLE_LED1();
  }
  if (BSP_LED2_IS_ON())
  {
    BSP_TOGGLE_LED2();
  }
```

```
for(int i=0;i<5;i++)
   Buf2[i]=(uint8_t)(65+i);
   Size2=5;

while (1)
{
   if(pot_enviar==1) // if(RXUartSize==MAXRXBUFSIZE)
   {
   // Enviamos, vía ZigBee, los datos al buffer RXUartBuf
   if (SMPL_SUCCESS == SMPL_Send(linkID1, RXUartBuf, RXUartSize))
   {
     RXUartSize = 0; // Si los datos han sido enviados con éxito inicializamos el buffer RX a 0
     pot_enviar = 0;
     BSP_TURN_OFF_LED2();// Durante la transmisión de datos el LED verde está apagado
     BSP_TOGGLE_LED1();// Durante la transmisión de datos el LED rojo parpadea
   }
   }
   if(pot_enviar==2) // if(RXUartSize2==MAXRXBUFSIZE)
   {
   // Enviamos, vía ZigBee, los datos al buffer RXUartBuf2
   if (SMPL_SUCCESS == SMPL_Send(linkID1, RXUartBuf2, RXUartSize2))
   {
     RXUartSize2 = 0; // Si los datos han sido enviados con éxito inicializamos el buffer RX a 0
     pot_enviar = 0;
     BSP_TURN_OFF_LED2();// Durante la transmisión de datos el LED verde está apagado
     BSP_TOGGLE_LED1();// Durante la transmisión de datos el LED rojo parpadea
   }
   }
}

/*-------------------------------------------------------------------------
   ..TRANSMITIMOS EL MENSAJE (MENSAJE+LONGITUD)..
-------------------------------------------------------------------------*/
void TXString( char* string, int length )
{
  int pointer;
  for( pointer = 0; pointer < length; pointer++)
  {
    volatile int i;
    UCA0TXBUF = string[pointer];
    while (!(IFG2&UCA0TXIFG)); // Mientras el buffer esté listo (ocupado)
  }
}

/*-------------------------------------------------------------------------
 * ADC10 interrupt service routine
 *-------------------------------------------------------------------------*/
#pragma vector=ADC10_VECTOR
__interrupt void ADC10_ISR(void)
{
  __bic_SR_register_on_exit(CPUOFF); // Clear CPUOFF bit from 0(SR)
}

/*-------------------------------------------------------------------------
```

```
 * Timer A0 interrupt service routine
 *-------------------------------------------------------------------------*/
#pragma vector=TIMERA0_VECTOR
__interrupt void Timer_A (void)
{
  __bic_SR_register_on_exit(LPM3_bits); // Clear LPM3 bit from 0(SR)
}

/*-------------------------------------------------------------------------------------------------------------
   ..PREPARAMOS LA TRAMA QUE VAMOS A ENVIAR. UNA VEZ ENVIADA SE BORRA..
 *-------------------------------------------------------------------------------------------------------------*/
#pragma vector=USCIAB0RX_VECTOR
__interrupt void USCI0RX_ISR(void)
{
  uint8_t rx = UCA0RXBUF;

  if (qui_guarda==1)
  {
    if (RXUartSize < MAXRXBUFSIZE)
    {
      BSP_TURN_ON_LED2();  // Durante la recepción de datos se enciende el LED verde
      RXUartBuf[RXUartSize++] = rx;  // RXUartSize++ , RXUartSize=1
    }
    else
    {
      pot_enviar=1;
      qui_guarda=2;
      RXUartSize2=0;
    }
   }
  if (qui_guarda==2)
  {
    if (RXUartSize2 < MAXRXBUFSIZE)
    {
      BSP_TURN_ON_LED2();  // Durante la recepción de datos se enciende el LED verde
      RXUartBuf2[RXUartSize2++] = rx;  // RXUartSize2++ , RXUartSize2=1
    }
    else
    {
      pot_enviar=2;
      qui_guarda=1;
      RXUartSize=0;
      RXUartBuf[RXUartSize++] = rx;
    }
  }
}
```

Placa adaptadora

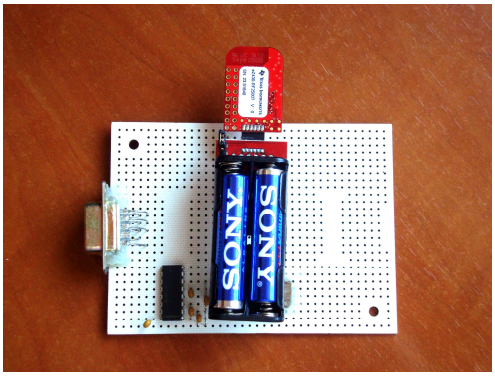

Figura X. Placa adaptadora (vista superior)

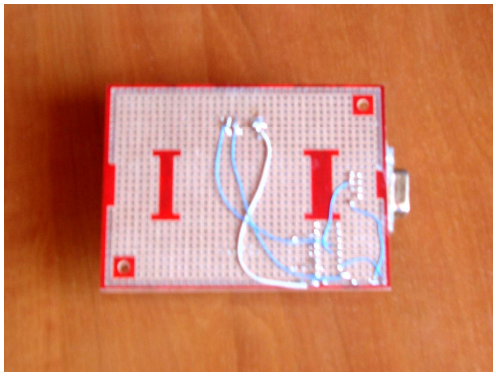

Figura X. Placa adaptadora (vista inferior)

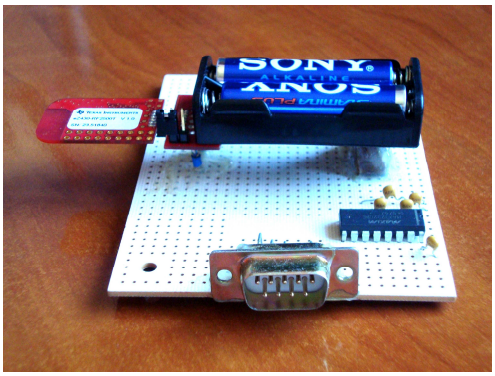

Figura X. Placa adaptadora (vista frontal)

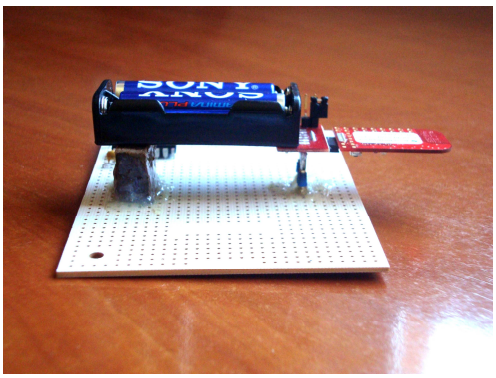

Figura X. Placa adaptadora (vista trasera)

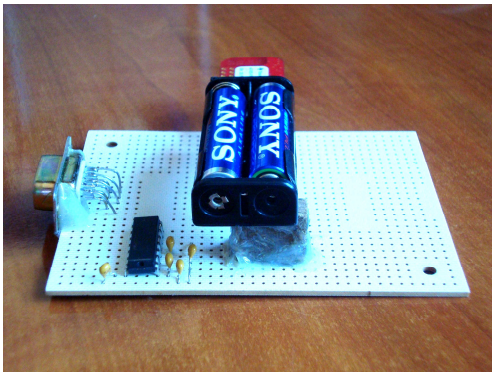

Figura X. Placa adaptadora (vista lateral 1)

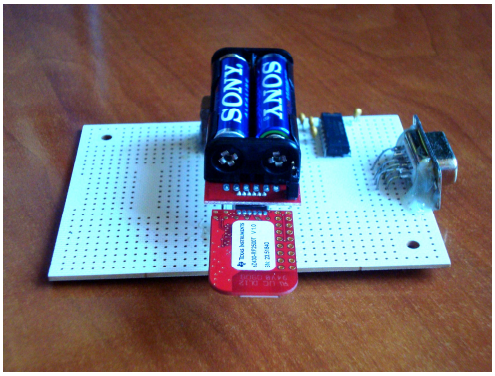

Figura X. Placa adaptadora (vista lateral 2)

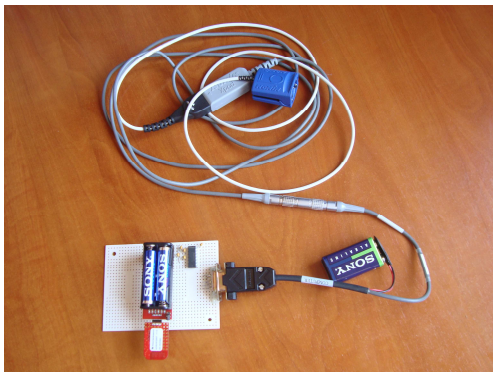

Figura X. Placa adaptadora + oxímetro Nonin

Código de programación del aplicativo de Visual Basic

```
Function toBitsString(b As Byte) As String // Extraemos un vector con todas las cadenas de bits

Dim bits() As Byte
Dim s As String

bits = toBits(b)
s = ""
For i = 0 To UBound(bits) - 1
   s = s & Trim(Str$(bits(UBound(bits) - i - 1)))  // Trim = borrar espacios laterales
                                                   // Str$ = lo pasa a string
Next i
toBitsString = s

End Function
-------------------------------------------------------------------------------------------------------------
Function subArray2(v2() As Byte, ind1 As Integer, ind2 As Integer) As Byte()

Dim v1() As Byte
ReDim v1(ind2 - ind1 + 1)

For i = ind1 To ind2
   v1(i - ind1) = v2(i)
Next i
subArray2 = v1

End Function
-------------------------------------------------------------------------------------------------------------
Function shift_esq(v2() As Byte, posToDel) As Byte() // Función de mover el array a la izquierda

Dim v1() As Byte
ReDim v1(UBound(v2))   '-1

For i = 0 To posToDel - 1
   v1(i) = v2(i)
Next i
For i = posToDel To UBound(v1) - 1
   v1(i) = v2(i + 1)
Next i
shift_esq = v1

End Function
-------------------------------------------------------------------------------------------------------------
Function toBits(b As Byte) As Byte() // Extraemos la cadena de bits

Dim bits(8) As Byte

For i = 0 To 8 - 1
   x = b And 2 ^ i
```

```
    If x <> 0 Then
        bits(i) = 1
    Else
        bits(i) = 0
    End If
Next i
toBits = bits

End Function
-----------------------------------------------------------------------------------------------------------------
Function toInt(bits() As Byte) As Integer // Extraemos el número decimal de la cadena de bits

result = 0
For i = 0 To UBound(bits) - 1
    result = result + (2 ^ i) * bits(i)
Next i
toInt = result

End Function
-----------------------------------------------------------------------------------------------------------------
Function getPackets(c1() As Byte) As Nonin()

Dim aux() As Byte
Dim c() As Byte
Dim s As String
Dim packets() As Nonin

c = c1 // A c le asignamos el array que contiene los datos del fichero
offset = 0

// Comprobamos donde está el inicio de un paquete (1 + 128-impar)
// En offset almacenamos el nº de caracteres que no son el inicio
Do While (True)
    If c(offset) = 1 Then
        b = toBits(c(offset + 1))
        If b(0) = 1 And b(7) = 1 Then
            Exit Do
        End If
    End If
        offset = offset + 1
Loop

// Calculamos el nº de paquetes
num_paq = CInt((UBound(c) - offset) / 125) - 1
ReDim packets(num_paq)

// Detectamos si hay caracteres repetidos y los borramos
n = offset

For i = 0 To UBound(packets) - 1
    For j = 1 To 25
        If c(n) = 1 And c(n + 5) = 1 Then
            n = n + 5
        Else
```

```
            For k = 1 To 5
               If c(n) = c(n + 1) Then
                  c = shift_esq(c, n): k = k - 1 // Desplazamos el array a la izquierda
               Else
                  n = n + 1
               End If
            Next k
         End If
      Next j
   Next i

   // Creamos el fichero n.txt que contiene nuestros paquetes en binario
   n = offset

   Open "n.txt" For Output As #1
   For i = 0 To UBound(packets) - 1
      For j = 1 To 25
         s = ""
         For k = 1 To 5
            s = s & toBitsString(c(n)) & " "
            n = n + 1
         Next k
         Print #1, s
      Next j
      Print #1, ""
   Next i
   Close #1

   For i = 0 To UBound(packets) - 1
      Set packets(i) = New Nonin
      aux = subArray2(c, offset + 125 * i, offset + 125 * (i + 1))
      Call packets(i).init(aux)
   Next i

   getPackets = packets

   End Function
   ----------------------------------------------------------------------------------------------------------------
   Private Sub Timer1_Timer()

   Label1.Caption = Time

   End Sub
   ----------------------------------------------------------------------------------------------------------------
   Private Sub Form_Activate()

   Text1.Text = "Esperando.."

   End Sub
   ----------------------------------------------------------------------------------------------------------------
   Private Sub BAbrir_Click()

   Dim bytes() As Byte
   Dim cont As Integer
```

```
Dim packets() As Nonin

cont = 0
ReDim bytes(3000)

If Fichero = True Then
Open Text2.Text For Binary As #1
TBTexto.Text = ""

Do While cont <= UBound(bytes)
   Get #1, , bytes(cont)
   cont = cont + 1
   DoEvents
Loop
Close #1

packets = getPackets(bytes)
For i = 0 To UBound(packets) - 1
   TBTexto.Text = TBTexto.Text & packets(i).toString
Next i
End If

If ZigBee = True Then
Open Text3.Text For Binary As #2
TBTexto.Text = ""

Do While cont <= UBound(bytes)
   Get #2, , bytes(cont)
   cont = cont + 1
   DoEvents
Loop
Close #2

packets = getPackets(bytes)
For i = 0 To UBound(packets) - 1
   TBTexto.Text = TBTexto.Text & packets(i).toString
Next i
End If

End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Examinar1_Click()

If Fichero = True Then
   Menu.ShowOpen
   Text2.Text = Menu.FileName
End If

End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Examinar2_Click()

If ZigBee = True Then
   Menu.ShowSave
```

```
   Text3.Text = Menu.FileName
End If

End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub ZigBee_Click()

If ZigBee = True Then
   If Examinar2 = True Then
      Call Examinar2_Click
      Open Text3.Text For Binary As #2
   End If
End If
Close #2

End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Start_Click()

Dim dato As String

MSComm1.CommPort = 4
MSComm1.Settings = "9600,N,8,1"
MSComm1.Handshaking = comNone
MSComm1.InputLen = 0
MSComm1.RThreshold = 1
MSComm1.PortOpen = True

Open Text3.Text For Binary As #2
Text1.Text = ""
Do While Start = True
   dato = MSComm1.Input
   Text1.Text = Text1.Text & dato
   Put #2, , dato
   DoEvents
Loop
Close #2

End Sub
-----------------------------------------------------------------------------------------------------------
Private Sub Stop_Click()

MSComm1.PortOpen = False
Close #2
Text1.Text = "Stop"

End Sub
```

- ***Módulos de clase: Nonin.cls***

```
Dim hr, spo2, spo2_d, spo2_fast, spo2_b_b, e_hr, e_spo2, e_spo2_d, hr_d, e_hr_d As Integer
-----------------------------------------------------------------------------------------------------------
```

```
Function toString() As String // Mostramos los valores de HR y SPO2

toString = "HR=" & hr & " SPO2=" & spo2 & Chr(13) & Chr(10)

End Function
-----------------------------------------------------------------------------------------------------------------------
// Obtenemos del paquete de 25 tramas los valores que nos interesan
Sub init(b() As Byte)
Dim i As Integer
Dim j As Integer

For j = 1 To 25
i = j - 1
Select Case j
   Case 1
     hr = getHR(b(i * 5 + 3), b((i + 1) * 5 + 3))

   Case 3
     spo2 = getSPO2(b(i * 5 + 3))
   Case 9
     spo2_d = getSPO2(b(i * 5 + 3))
   Case 10
     spo2_fast = getSPO2(b(i * 5 + 3))
   Case 11
     spo2_b_b = getSPO2(b(i * 5 + 3))
   Case 14
     e_hr = getHR(b(i * 5 + 3), b((i + 1) * 5 + 3))
   Case 16
     e_spo2 = getSPO2(b(i * 5 + 3))
   Case 17
     e_spo2_d = getSPO2(b(i * 5 + 3))
   Case 20
     hr_d = getHR(b(i * 5 + 3), b((i + 1) * 5 + 3))
   Case 22
     e_hr_d = getHR(b(i * 5 + 3), b((i + 1) * 5 + 3))
End Select
Next j

End Sub
-----------------------------------------------------------------------------------------------------------------------
Function toBits(b As Byte) As Byte() // Extraemos la cadena de bits

Dim bits(8) As Byte

For i = 0 To 8 - 1
   x = b And 2 ^ i
   If x <> 0 Then
      bits(i) = 1
   Else
      bits(i) = 0
   End If
Next i
toBits = bits
End Function
```

```
-----------------------------------------------------------------------------------------------------------
Function toInt(bits() As Byte) As Integer // Extraemos el número decimal de la cadena de bits

result = 0
For i = 0 To UBound(bits) - 1
   result = result + (2 ^ i) * bits(i)
Next i
toInt = result

End Function
-----------------------------------------------------------------------------------------------------------
Function getHR(msb As Byte, lsb As Byte) As Integer // Extraemos el valor del Heart Rate

Dim m() As Byte, l() As Byte
hr = 0

m = toBits(msb)
l = toBits(lsb)
Dim r(9) As Byte
r(8) = m(1): r(7) = m(0)
For i = 0 To 7 - 1
   r(i) = l(i)
Next i
hr = toInt(r)
getHR = hr

End Function
-----------------------------------------------------------------------------------------------------------
Function getSPO2(b As Byte) As Integer // Extraemos el valor del SPO2

Dim b2() As Byte
spo2 = 0

b2() = toBits(b)
Dim r(7) As Byte
For i = 0 To 7 - 1
   r(i) = b2(i)
Next i
spo2 = toInt(r)
getSPO2 = spo2

End Function
```

Bibliografía

- ***Información acerca de ZigBee:***
  1. SG Software Guru – ZigBee, Comunicación para Dispositivos: *http://www.sg.com.mx/content/view/392*
  2. ZigBee Alliance: *http://www.zigbee.org/en/index.asp*
  3. RF DesignLine: Overcome challenges in wireless sensor networks: *http://www.rfdesignline.com/howto/lowpowerrf/197007510*
  4. Dexma. Wireless Sensor Networks Intelligence: *http://www.dexmasensors.com*
  5. Using ZigBee Wireless Networking to Develop Commercial Products *http://www.rtcmagazine.com/home/article.php?id=100656*
  6. Home networking with ZigBee: *http://www.embedded.com/shared/printableArticle.jhtml?articleID=52600868*
  7. Bluetooth and ZigBee: Compare and Contrast: *http://www.techworld.com/mobility/features/index.cfm?featureid=1261&Page=1&pagePos=12*
- ***Búsqueda de módulos y dispositivos ZigBee:***
  8. XBee ZNet 2.5 RF Module: *http://www.digi.com/products/wireless/zigbee-mesh/xbee-series2-module.jsp*
  9. XBee-PRO ZNet 2.5 RF Module: *http://www.digi.com/products/wireless/zigbee-mesh/xbee-pro-series2-module.jsp*
  10. EasyBee ZigBee Transceiver Module: *http://www.rfsolutions.co.uk/acatalog/EasyBee_Zigbee_Module.html*
  11. ZB-21 ZigBee OEM Module: *http://ampedrf.com.global-nameservers.com/document/ZB21_ Datasheet_ 080303.pdf*
  12. MICAz Module: *http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf*
  13. ATMEL Module: *http://www.atmel.com/dyn/resources/prod_documents/S-RZ200SDK.pdf* *http://www.atmel.com/dyn/resources/prod_documents/doc5131.pdf* *http://www.atmel.com/dyn/resources/prod_documents/2549S.pdf*

- ***Estudio del kit MSP430 Wireless Development Tool (eZ430-RF2500):***

14. User's Guide eZ430-RF2500 Development Tool:
    http://focus.ti.com/lit/ug/slau227c/slau227c.pdf
15. Low-Cost Low-Power 2.4GHz RF Transceiver CC2500:
    http://focus.ti.com/lit/ds/swrs040b/swrs040b.pdf
16. The MSP430 Family of ultralow-power Microcontrollers:
    http://focus.ti.com/lit/ds/slas504b/slas504b.pdf
17. Application Report Wireless Sensor Monitor Using the eZ430-RF2500:
    http://focus.ti.com/lit/an/slaa378a/slaa378a.pdf
18. Introduction to SimpliciTI:
    http://focus.ti.com/lit/ml/swru130a/swru130a.pdf

- ***Diseño e implementación de la placa adaptadora:***

19. Practical Electronics:
    http://www.ep.com.pl/common/obrazek_duzy.php?sect=msp430_mbarowski&box=rys1a&imgnum=

- ***Descodificación de los datos medidos por el sensor:***

20. Oximenter Nonin model 700P. Xpod Module Specification and Technical Information:
    http://www.nonin.com/documents/Xpod%20Specifications.pdf

- ***Programación del aplicativo realizado con Visual Basic 6.0:***

21. Aprender Visual Basic 6.0 como si estuviera en primero:
    http://mat21.etsii.upm.es/ayudainf/aprendainf/VisualBasic6/vbasic60.pdf
22. Control Timer – Temporizador:
    http://www.recursosvisualbasic.com.ar/htm/tutoriales/control_timer.htm
23. Intercambio Dinámico de Datos (DDE) – Control Personalizado Microsoft COMM:
    http://tec.upc.es/ie/practi/manual_VB/ceubas06.doc
24. Ficheros en Visual Basic:
http://www.scribd.com/doc/481094/Capitulo-6-Visual-Basic

Resumen del proyecto

Castellano

Pensado en un ámbito de tecnología médica, el propósito principal es el de realizar un seguido de medidas en diferentes partes del cuerpo para establecer unos valores máximos que nos permitan detectar cuando un paciente empieza a padecer estrés. Para ello se necesita un proceso de medición y otro de transmisión de los datos. Es aquí donde aparece el trabajo realizado en el proyecto. "ZigBee aplicado a la transmisión de datos de sensores biomédicos" está pensado para realizar la tarea de transmisión de los datos desde que el sensor realiza la medida hasta que los datos son monitorizados y almacenados. En la memoria del proyecto podremos encontrar el estudio realizado al medio de transmisión inalámbrico utilizado (ZigBee), el análisis del kit eZ430-RF2500 compatible con el medio, y finalmente la implementación del proyecto. Todo este trabajo finalizará con la recepción satisfactoria de los datos medidos por nuestro sensor biomédico (oxímetro) en el aplicativo personal programado con Visual Basic.

Català

Pensat en un àmbit de tecnologia mèdica, el propòsit principal es el de realitzar un seguit de mesures en diferents parts del cos per establir uns valors màxims que ens permetin detectar quan un pacient comença a patir estrés. Per a això és necessari un procés de mesura i un altre de transmissió de les dades. És aquí on apareix el treball realitzat en el projecte. "ZigBee aplicado a la transmisión de datos de sensores biomédicos" està pensat per a realitzar la tasca de transmissió de les dades des de que el sensor realitza la mesura fins que les dades son monitoritzades i emmagatzemades. En la memòria del projecte podrem trobar l'estudi realitzat al medi de transmissió inalàmbric utilitzat (ZigBee), l'anàlisi del kit eZ430-RF2500 compatible amb el medi, i finalment la implementació del projecte. Tot aquest treball finalitzarà amb la recepció satisfactòria de les dades mesurades pel nostre sensor biomèdic (oxímetre) en l'aplicatiu personal programat amb Visual Basic.

English

Thought about an area of medical technology, the principal intention is of fulfilling the consecutive one by measures in different parts of the body to establish a few maximum values that they allow us to detect when a patient starts suffering stress. For it is needed a process of measurement and other one of transmission of the information. It is here where the work realized in the project appears. "ZigBee aplicado a la transmisión de datos de sensores biomédicos" is thought to realize the task of transmission of the information since the sensor realizes the measure until the information is monitored and stored. In the memory of the project we will be able to find the study realized to the wireless used conduit (ZigBee), the analysis of the kit eZ430-RF2500 compatibly with the way, and finally the implementation of the project. All this work will finish with the satisfactory receipt of the information measured by our biomedical sensor (oximeter) in the personal display programmed with Visual Basic.