Proyecto de Sistemas Informáticos
SISTEMA INTERACTIVO DE DOCENCIA

AUTORES

Fernando de la Parra Gimeno, Óscar Quintanilla Artero, Julio Tórtola del Moral

TUTOR

Luis Piñuel Moreno

LISTA DE PALABRAS CLAVE

docencia, aprendizaje, CRS, ISR, mando a distancia, profesor, alumno, aula, clase, preguntas, respuestas, ZigBee, radiofrecuencia, Microchip, PIC, 4620, 4550, ICD2, MPLAB, PICDEM Z

AUTORIZACIÓN

Los autores de este proyecto, Don Fernando de la Parra Gimeno, Don Julio Tórtola del Moral y Don Óscar Quintanilla Artero,

AUTORIZAN

A la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Firmado en Madrid, el día 4 de julio de 2007,

Don Fernando de la Parra Gimeno

Don Julio Tórtola del Moral

Don Óscar Quintanilla Artero

RESUMEN DEL PROYECTO EN CASTELLANO

Este proyecto trata de la construcción de un sistema de recogida de respuestas tipo test de mano de los alumnos de una clase ante una pregunta planteada por su profesor, para su almacenamiento y posterior análisis estadístico. Con esto se pretende animar la participación de los alumnos en clase, no excluyendo otros métodos de evaluación. Para ello, se ha empleado tecnología ZigBee, un novedoso estándar para redes WPAN de radiofrecuencia de bajo coste y consumo energético, así como el uso del framework .NET de cara a aprovechar la interoperabilidad que ofrece con la suite de oficina de Microsoft, la más usada por los profesores, usuarios finales del sistema.

RESUMEN DEL PROYECTO EN INGLÉS

This Project manages the interactive response collection system construction. These answers offered by students are collected by the system in order to be saved and perform a statistic analysis. The aim of the project is to encourage students' participation, allowing other evaluating methods. In order to get it, it's used ZigBee Technology, a young WPAN cheap standard with low power consumption, as well as .NET Framework because of its well-worth interoperability with Microsoft's office suite, the most used by teachers, the final system users.

AGRADECIMIENTOS

Queremos dar las gracias en primer lugar a Luis, nuestro tutor, por haber ideado este proyecto que hemos escogido como nuestro PFC, que si bien ha sido muy trabajoso y no se ha podido terminar por falta de tiempo, y pese también a estar ya muy cansados después de un año dedicado, se aceptó y trabajó con mucha ilusión por aprender cosas nuevas y hacer una herramienta útil para la docencia desde el principio, allá por julio de 2006.

Por otro lado, también nos gustaría agradecerle el apoyo prestado a todas aquellas personas que nos han ayudado en situaciones de estancamiento o con alguna buena idea. Dichas personas son Pablo, Carlos, José Antonio, Gonzalo, José Manuel y Zalo, espero no dejarme a nadie. De la misma forma que a los componentes de nuestro "grupo competidor", Esteban, José y Carlos, con quien también hemos compartido algunas ideas de cómo sacar esto adelante.

Yo, Fernando, por supuesto no me olvido de mi amigo Borditas, que día tras día me hacía la recogida del laboratorio, el camino al metro contando las penurias del curso, y el *"¿Cómo lo lleváis? ¿Habéis conseguido avanzar algo hoy?"*, y que además formó parte activa en la presentación al público de este proyecto junto con Sergio, portando materiales y participando de la misma en la demo en directo, y de las postreras cañejas de celebración. Personalmente, también me gustaría dar las gracias a mi madre y hermana que tanto me han tenido que aguantar con esto, saltándome algunas de las tareas domésticas ¡incluso alguna cena! por terminar alguna cosa pendiente, o simplemente descansar un poco… A mi padre ya no sólo le agradezco sino que le dedico este final de carrera: como puedes ver, ¡al fin lo he conseguido! También al resto de amigos por su comprensión cuando me volvía pronto para casa por cansancio de toda la semana o simplemente no podía salir de juerga algún día. (Se nota que en este apartado se pueden decir más ñoñeces que en ningún otro…)

Yo, Julio, me gustaría agradecer a nuestro profesor Luis por su apoyo durante la realización de este proyecto y su constante flujo de ideas que sirvió para mejorar este trabajo. También me gustaría agradecer a mis padres y a mi hermana por su gran ayuda moral (aunque hubiese broncas por el uso de internet) en los días de máximo agobio, a mi novia Silvia que cada día me animaba a seguir adelante en

este tarea y que sin su apoyo esto hubiese sido mas difícil, y por ultimo a mis grandes amigos Félix, y Nacho que como informáticos se interesaron en este proyecto e incluso aportando alguna idea. Y sobre todo me gustaría dedicar este proyecto a mi abuelo y abuela que este año nos dejaron, así que va por ellos.

ÍNDICE DE APARTADOS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 Motivación

La motivación de este proyecto no es otra que la implementación de un sistema de respuesta interactiva para uso académico o escolar, que hará posible al profesor o tutor de una clase obtener en tiempo real las respuestas de sus alumnos a preguntas planteadas, con el objetivo de fomentar la participación en clase y por tanto realizar una enseñanza más adaptada a cada alumno, así como poner en disposición del profesor un estudio estadístico acerca de la evolución de sus alumnos a lo largo del curso. El nombre que se da a uno de estos sistemas es el de CRS ("Classroom Response System"), aunque también podremos encontrarlo por IRS ("Interactive Response System") o ISR ("Interactive Student Response").

El número de dispositivos emisores necesarios para un aula variará según el tamaño y la configuración del espacio, así como el número de estudiantes que usan el sistema. El mando emisor es similar al de un televisor.

Cada sistema tiene una seria de botones que permitirán a los alumnos contestar a la pregunta de una forma intuitiva y rápida.

Fig. 1: Un sistema interactivo de respuesta

El caso uso principal del sistema sería (de forma general) el siguiente:

1. El profesor está dando su clase y en momento dado realiza una pregunta.

2. El profesor arranca la aplicación de respuesta interactiva, que habilita la comunicación entre los mandos de los alumnos y el receptor instalado en su ordenador portátil.

3. Los alumnos contestan a la pregunta, seleccionando en sus mandos la opción deseada.

4. Al cabo de unos segundos, el profesor da por finalizada el tiempo de respuestas y deshabilita la comunicación.

5. Se muestran por pantalla los resultados, así como se almacenan en algún tipo de soporte (Excel, XML) para su posterior estudio.

La realización del proyecto se dividirá en dos fases paralelas que a grandes rasgos son las siguientes:

- **Fase A.1:** Trabajo de investigación y programación sobre el kit de desarrollo ZigBee que se nos facilita (emulación de emisores y receptor), con el fin de conseguir que el dispositivo receptor pueda disponer de todas las respuestas dadas por los emisores, sin pérdida alguna en el tiempo que dure la pregunta.

- **Fase A.2:** Trabajo de diseño y análisis, así como de implementación de una aplicación software en C# .NET que sea capaz de trabajar con las respuestas "capturadas" por el dispositivo receptor, con el fin de poder mostrar por pantalla los resultados, realizar estadísticas, clasificaciones, etc.

Se pueden añadir un par de fases más, que continúan la línea de desarrollo de las anteriores, y una tercera línea de investigación que completa un poco más el proyecto:

- **Fase B.1:** Construcción del prototipo hardware con componentes compatibles con los usados en el kit de desarrollo y acabado final.

- **Fase B.2:** Mejora de las funcionalidades de la aplicación C# .NET menos básicas pero útiles de cara al producto final.

- **Fase B.3:** Investigación en una plataforma como un teléfono móvil, para la implementación de una mini-aplicación en J2ME que emule un mando a

distancia como los construidos, pero que su canal de votación hacia la aplicación del PC sea mediante el uso del protocolo TCP/IP a través de una petición HTTP.

1.2 Objetivos

El objetivo principal es el diseño e implementación del sistema con la funcionalidad solicitada, es decir que el sistema sea capaz de leer la respuestas que provienen de los mandos, que sea capaz de clasificarla, mostrarla por pantalla, almacenarla, etc.

Aparte de este objetivo principal, existen otros como son:

- Investigación y prueba de las características de la tecnología ZigBee dados los manuales y estándares.

- Realización de un proyecto de grandes dimensiones entre tres personas con el apoyo de un tutor, así como la obtención de una metodología de trabajo para proyectos de esta índole.

- Realización de un proyecto con vistas de una futura continuación, es decir, que pueda ser ampliado fácilmente tanto a nivel hardware como a nivel software por futuros integrantes de otros equipos de I+D+i en la asignatura de Sistemas Informáticos.

El objetivo más interesante entre todos es la investigación de la tecnología ZigBee, ya que es el pilar fundamental en que descansa el proyecto, y dado que es una tecnología relativamente nueva y con muchas posibilidades de uso en diversos proyectos de diferentes campos. Es interesante ver qué características principales, así como ventajas, hacen que el Zigbee sea el estándar elegido para su uso en el proyecto con respecto a otras tecnologías ya implantadas desde hace tiempo y muy rodadas, tecnologías que en algunos casos serían válidas para obtener el mismo resultado que obtendremos con ZigBee, pero que no resultan interesante por diversos motivos, principalmente por la gran idoneidad que ofrece de ZigBee según se concibió el sistema que se quería implementar, y la poca que ofrecen el resto de alternativas.

1.2.1 ¿Por qué ZigBee?

La tecnología ZigBee (basada en el IEEE 802.15.4) se está consolidando como un estándar de comunicaciones relativamente nuevo muy similar al Bluetooth (IEEE 802.15.1), ya consolidado, de cierta complejidad y muy aplicado para transmisiones de datos con cierto ancho de banda en PANs, con distancia muy pequeña entre los dispositivos que se comunican, como transferencia de ficheros o voz. ZigBee surgió de una alianza de empresas (ZigBee Alliance) sin ánimo de lucro a partir del estándar que el IEEE propone para PANs con poco coste y ahorro de energía, con el objetivo de conseguir el desarrollo final e implantación de una tecnología inalámbrica de bajo coste para PANs, común para todos los fabricantes.

Vemos que una alternativa al uso del Zigbee es el Bluetooth, ¿por qué usar entonces una tecnología tan nueva en vez de una que lleva más tiempo implantada?

Una razón importante es el coste, los dispositivos Bluetooth en comparación con otros ZigBee son más costosos de fabricar, y además consumen más energía. Esto no quiere decir que Bluetooth sea peor que ZigBee, sino simplemente que para el cometido a realizar es mucho mejor usar esta tecnología más barata y con múltiples posibilidades.

Otra característica muy interesante del ZigBee es la posibilidad de conectar en red una gran cantidad de pequeños sensores entre sí de manera sencilla, lo que es una ventaja importante respecto a Bluetooth. En nuestro caso, los nodos de la red los formarían los mandos de respuesta.

Las diferencias básicas que existen entre estos dos estándares para PANs son las siguientes:

- Una red ZigBee puede constar de un máximo de 255 nodos, frente a los 8 máximos de una red Bluetooth.

- Menor consumo eléctrico en sistemas ZigBee que en Bluetooth. Esto se debe a que el dispositivo ZigBee se queda la mayor parte del tiempo dormido, mientras que en una comunicación Bluetooth esto no se puede dar, y siempre se está transmitiendo y/o recibiendo.

- El estándar ZigBee tiene un ancho de banda de 250 kbps, mientras que el Bluetooth tiene 1 Mbps.

- ZigBee tiene un rango de operación de 10 a 75 m, respecto al de 1 a 100 m de Bluetooth según clase de dispositivo Bluetooth (hay clases 1, 2 y 3, cada cual de mayor consumo).

En el consumo y ancho de banda están las claves para entender el porqué del uso de ZigBee en vez de Bluetooth. Como vemos, tiene peor ancho de banda que el Bluetooth; esto hace que el Bluetooth sea más eficiente en aplicaciones Wireless USB o voz para teléfonos móviles, pero para aplicaciones como son controles remotos dependientes de la batería, donde el ancho de banda ofrecido por ZigBee es más que suficiente, unido al mínimo consumo y al bajo coste, hacen de ZigBee un estándar idóneo para la aplicación que vamos a darle en el proyecto.

Comentábamos además el bajo coste que se tiene al producir esta tecnología. Los módulos ZigBee son los dispositivos estándares inalámbricos más baratos jamás producidos de forma masiva. Con un coste estimado alrededor de los 2 € disponen de una antena integrada, control de frecuencia y una pequeña batería. En este aspecto, aún sabiendo que los dispositivos ZigBee son más baratos, en nuestro caso particular se torna al revés, pues todos los alumnos tienen teléfono móvil, y la gran mayoría de los nuevos terminales incorporan ya Bluetooth, por tanto, el coste de fabricación sería de 0 € por terminal.

Fig. 2: Bluetooth adaptado en teléfonos móviles

Otra alternativa que se barajó para la realización del sistema de respuesta fue usar rayos infrarrojos (luz), como los actuales mandos a distancia de TV, video/DVD, A/A, etc., o mismamente muchos de los sistemas comerciales que pretendíamos imitar. Esto fue rápidamente descartado debido a factores como el limitado alcance que hacía imposible su utilización en un aula relativamente grande, la cantidad muy limitada de usuarios que podría tener el sistema, así como la imposibilidad a traspasar objetos opacos (obstáculos en el haz de luz infrarroja), o que no pueden utilizarse en el exterior debido a que agentes naturales producen interferencias muy importantes que impiden una clara recepción del mensaje. Se trataba así mismo de mejorar ese tipo de sistemas, y la radiofrecuencia ofrecía una estupenda alternativa. De hecho, esta tecnología empieza a estar en desuso y otras como la de ZigBee están poco a poco suplantándola de manera progresiva.

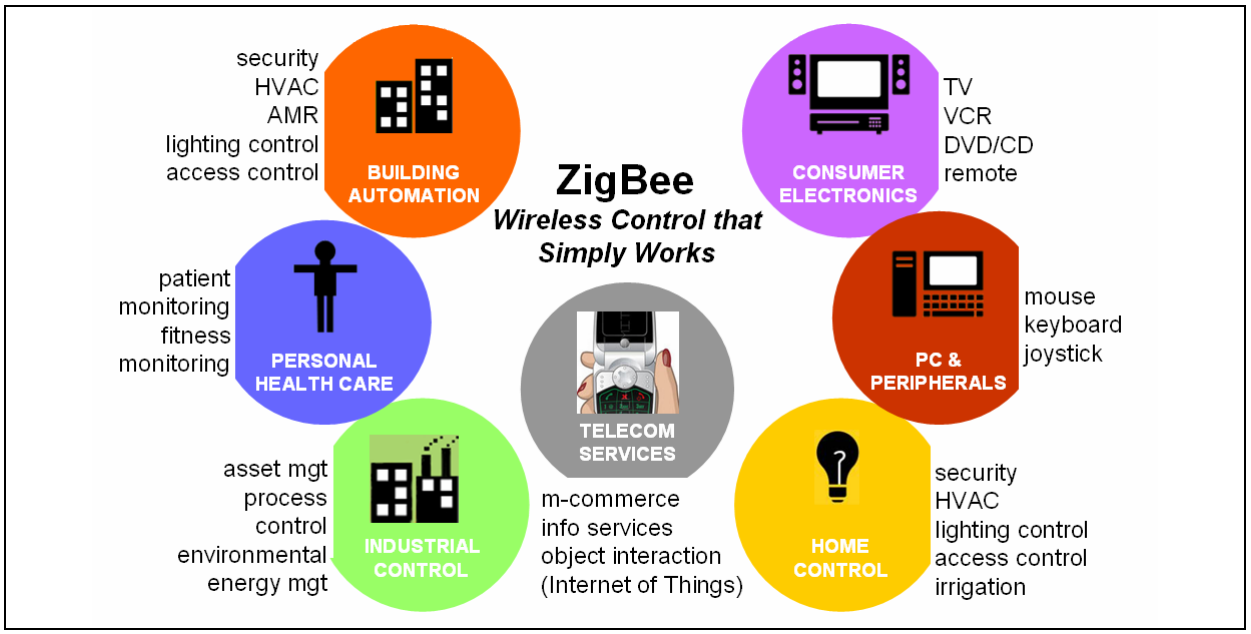


Fig. 3: Aplicaciones para ZigBee que propone la ZigBee Alliance

Por todas las razones expuestas arriba, se optó por el uso del ZigBee en lugar del Bluetooth o de los rayos infrarrojos puesto que era la tecnología que más se acercaba a las necesidades del proyecto, y si unimos todo lo ya mencionado a que ZigBee es un tecnología muy nueva, hace que sea un punto más a su favor en nuestro interés personal de descubrir algo novedoso.

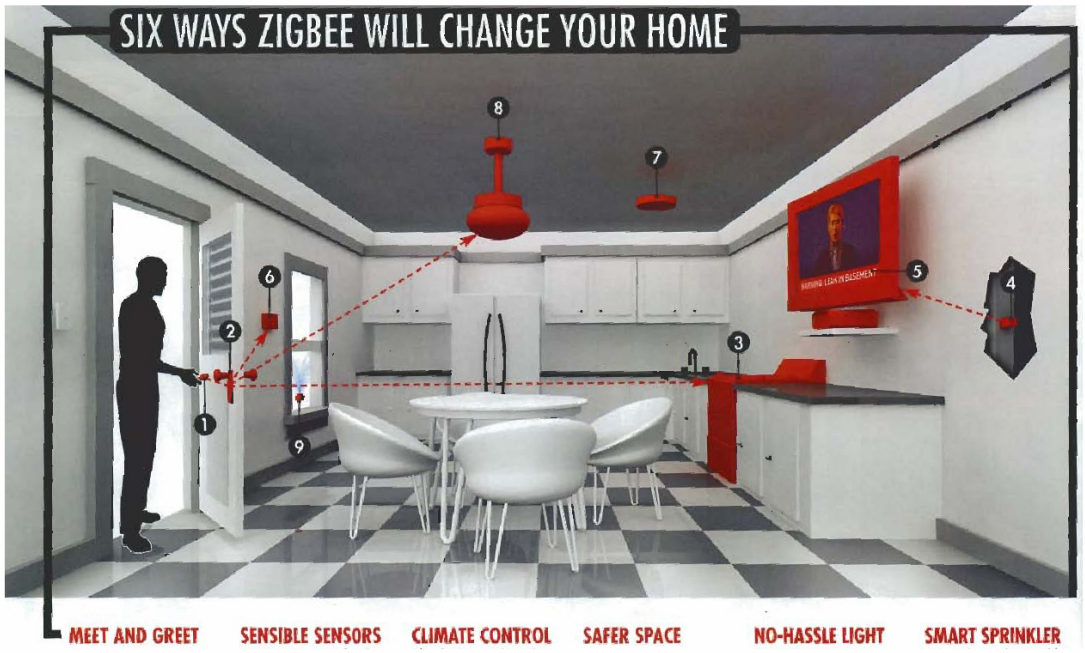


Fig. 4: ZigBee en domótica ofrecida por la "Popular Science Magazine"

1.3 Estructura de la memoria

Esta memoria se compone de varias partes bien diferenciadas. La primera de ellas, recogida en los puntos 1 y 2, se refiere a lo que ha llevado a la realización de este proyecto, es decir, tecnologías, sistemas actuales, etc.

La segunda parte incluida en el punto 3 versa acerca del entorno de desarrollo empleado para la consecución de los hitos marcados. En este apartado se desarrollan temas como las características del microcontrolador PIC empleado, la tecnología ZigBee, el kit de desarrollo (PICDEM Z), y demás herramientas empleadas.

La tercera parte localizada en el punto 4, trata del desarrollo del prototipo, es decir, de la implementación de los diferentes ámbitos de la aplicación como el software del PC, el firmware de los PICs, y el del teléfono móvil.

Una cuarta parte (punto 5) engloba los aspectos importantes de la solución OEM que, a pesar de no haberse alcanzado el hito deseado, se ha llevado a cabo una importante tarea de estudio e investigación.

Y una última parte (punto 6 y anexos) de conclusiones e información complementaria, más en detalle, que trata de matizar algunos aspectos relevantes de la aplicación.

2 CRSs (CLASSROOM RESPONSE SYSTEMS)

Los llamados "Sistemas de Respuesta (Interactiva) en Clase" llevan implantados desde hace 10 años, sobre todo en escuelas secundarias, pero no fue hasta principios del año 2000 cuando se vio la posibilidad de instaurar estos sistemas en centros docentes como universidades, empresas, etc. Esto llevó a un auge de empresas dedicadas a este tipo de productos lo que supuso la investigación y desarrollo de nuevas tecnologías como una parte de mejorar este sistema.

Los sistemas de respuesta interactiva son, por tanto, una solución tanto software como hardware, que facilita las actividades docentes debido a que:

- El profesor plantea una pregunta de selección múltiple a sus estudiantes mediante un ordenador, por ejemplo haciendo uso de una presentación PowerPoint para realizar la pregunta.

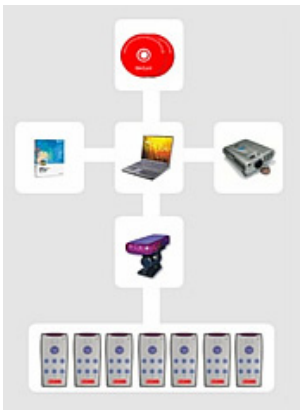
- Cada estudiante selecciona su respuesta con su mando que emite una señal de radiofrecuencia o infrarrojos que será recibida por el receptor del profesor que irá instalado en su ordenador.

- El software instalado en el ordenador del profesor permite listar todas las respuestas obtenidas, mostrar gráficos, estadísticas, etc.

Así pues, este tipo de sistemas se está implantando rápidamente por diversos centros educativos de distintas índoles. Una gran ventaja de los CRSs es que se puede adaptar su funcionalidad dependiendo de en qué centro se desee usar, por lo pueden instalarse en variedad de centros de enseñanza tales como colegios infantiles, de primaria, institutos, universidades, academias, empresas, e incluso se pueden usar para realizar encuestas. Por tanto, la variedad de uso de este sistema no sólo se ciñe a ámbitos educativos (aunque es su principal aplicación) sino que puede ser aplicado en diversas situaciones.

Fig. 5: El sistema de infrarrojos PRS de Interwrite Learning

2.1 Enseñanza con CRSs



La enseñanza con CRSs permite tomar diferentes opciones de uso. Los profesores pueden querer hacer preguntas relacionadas con el tema de la clase que están impartiendo, realizar preguntas eliminatorias, exámenes, o bien aplicar su propio estilo a la enseñanza con esta herramienta.

Unos posibles ejemplos de uso de CRSs son los siguientes:

- **Preguntar a la Audiencia:** A menudo descrito como el "Quién Quiere Ser Millonario", la actividad CRS más básica permite al instructor plantear una pregunta de selección múltiple y mostrar los resultados en tiempo real. Las preguntas pueden ser entremezcladas con el contenido de la conferencia o clase para permitir a los instructores calibrar el entendimiento del estudiante y ajustar sus conferencias en consecuencia.

- **Preguntas compartidas:** El profesor plantea una pregunta a sus alumnos. Los estudiantes consideran la pregunta silenciosamente y transmiten sus respuestas individuales con su mando. El profesor comprueba el histograma de respuestas de cada estudiante. Si hay un número significativo de estudiantes que escogen la respuesta incorrecta, el profesor indica a los estudiantes que comenten o discutan la pregunta con su compañero.

Después de unos minutos de discusión, los estudiantes responden otra vez. Con este tipo de uso se consigue (frecuentemente) que gracias a la discusión de la pregunta con sus compañeros los alumnos respondan correctamente a la pregunta.

- **Asistencia a clase:** Los transmisores son asignados a los estudiantes durante todo el curso, entonces el CRS puede ser usado para llevar constancia de los alumnos que asisten a clase.

- **Demostraciones Interactivas:** En clases de ciencias, se puede pedir a los estudiantes predecir el resultado de un experimento antes de ser mostrado. Esto da al profesor el sentido de las preconcepciones de los estudiantes y aumenta el valor de sorpresa del experimento cuando los estudiantes ven cuántos de sus compañeros de clase esperó resultados diferentes.

- **Recolección de datos:** El CRS puede ser usado para estudios demográficos, contrastar la opinión de los estudiantes, u otros datos de una clase. Esto es a menudo útil en las clases de ciencias sociales, que utilizan experimentos sociales o económicos en el aula.

2.2 Tipos de preguntas

Además de la variedad de posibles actividades con CRSs, éstos permiten al profesor preguntar de diferentes formas. Esto es importante, ya que en función de lo que se quiera preguntar puede ser apropiado proponer una pregunta de selección múltiple o una pregunta de "si/no", etc.

¿Qué tipos de pregunta pueden hacerse y admiten muchos de los CRSs que existen en el mercado? Aquí se proponen algunos:

- **Preguntas de "Verdad":** Estas preguntas podrían ser usadas para ver si los estudiantes han leído un texto propuesto por el profesor, para recordar los puntos importantes de clases previas, o han memorizado hechos claves, etc.

- **Preguntas Conceptuales:** Son preguntas de selección múltiple que se manifiestan para ver si realmente los estudiantes entienden los conceptos importantes y principios de la clase impartida por el profesor.

- **Preguntas "La respuesta mejor":** Estas preguntas incluyen múltiples opciones de respuesta de las que más que una podría ser argumentada como correcta. Se pide a los alumnos seleccionar la mejor respuesta de estas opciones (pueden ser la respuesta más completa, más exacta…)

- **Preguntas de Opinión:** Este tipo de preguntas no tiene una respuesta correcta, pero haciendo estas preguntas se puede observar la opinión de los estudiantes y provocar discusiones ricas, en particular en respuesta a publicaciones (cuestiones) éticas, legales, o morales.

- **Preguntas "Predictivas":** Son preguntas que se realizan como se ha comentado antes para, por ejemplo, experimentos de ciencias donde se le pide al alumno que prediga el resultado de dicho experimento.

- **Preguntas "Juego":** En clases de ciencias sociales, en particular la economía, un "juego" podría ser ilustrar puntos sobre el comportamiento humano. Un CRS puede facilitar la reunión de datos en tales juegos y ver la tendencia de cada alumno.

2.3 ¿Por qué usar un CRS?

Aunque en los anteriores párrafos se ha podido hacer una idea de por qué es interesante usar CRSs para la enseñanza, pasaremos a enumerar las causas por las que un profesor usaría un CRS o qué "mejoras" esperaría en su sistema docente por usar este sistema:

- Mantener la atención de los estudiantes durante una conferencia. Los estudios muestran que los lapsos de atención de la mayoría de la gente ocurren después de 10 minutos de audiencia pasiva. Insertando actividades de CRS durante una conferencia se puede ayudar a mantener la atención de los estudiantes.

- Promover el compromiso de los estudiantes durante una conferencia. El planteamiento de preguntas a los estudiantes así como el hecho de mostrar la respuesta correcta después de plantear dicha pregunta, hace que el alumno, en caso de no haber dado la respuesta correcta, se plantee el motivo

de no haberla dado, y por tanto, asimila más rápidamente los conocimientos que se le están presentando.

- Promover la discusión y la colaboración entre estudiantes durante la clase con los ejercicios de grupo que requieren que estudiantes dialoguen entre ellos y lleguen a un acuerdo general.

- Fomentar la participación de cada estudiante en una clase. Hacer una pregunta verbalmente hace que solo los alumnos que levanten la mano participen en clase, en cambio con un CRS todos los alumnos participarán en la clase.

- Comunicación: Crear un entorno en el cual las diferencias y opiniones del grupo pueden ser expuestas y discutidas al momento mientras cabe la posibilidad de que cada respuesta de cada alumno se mantenga en el anonimato.

- Crear un espacio a "salvo" para estudiantes tímidos e inseguros que les permita participar en la clase. Un CRS da a los estudiantes una posibilidad para responder a la pregunta de un profesor silenciosamente y en privado, permitiendo al estudiante más tímido o reservado dar su opinión o respuesta a una pregunta echa por el profesor, además de permitir responder de manera anónima a preguntas sensibles, éticas, legales, y morales.

- Comprobar para cada estudiante qué entiende durante la clase. Haciendo preguntas con un CRS los profesores pueden determinar si los estudiantes entienden los puntos importantes de la clase, y actuar en consecuencia cambiando su forma de dar clase, replanteando el tema de la clase, o ver personalmente con qué tema un alumno en concreto tiene problemas, para así poder realizar un seguimiento personal sobre ese alumno.

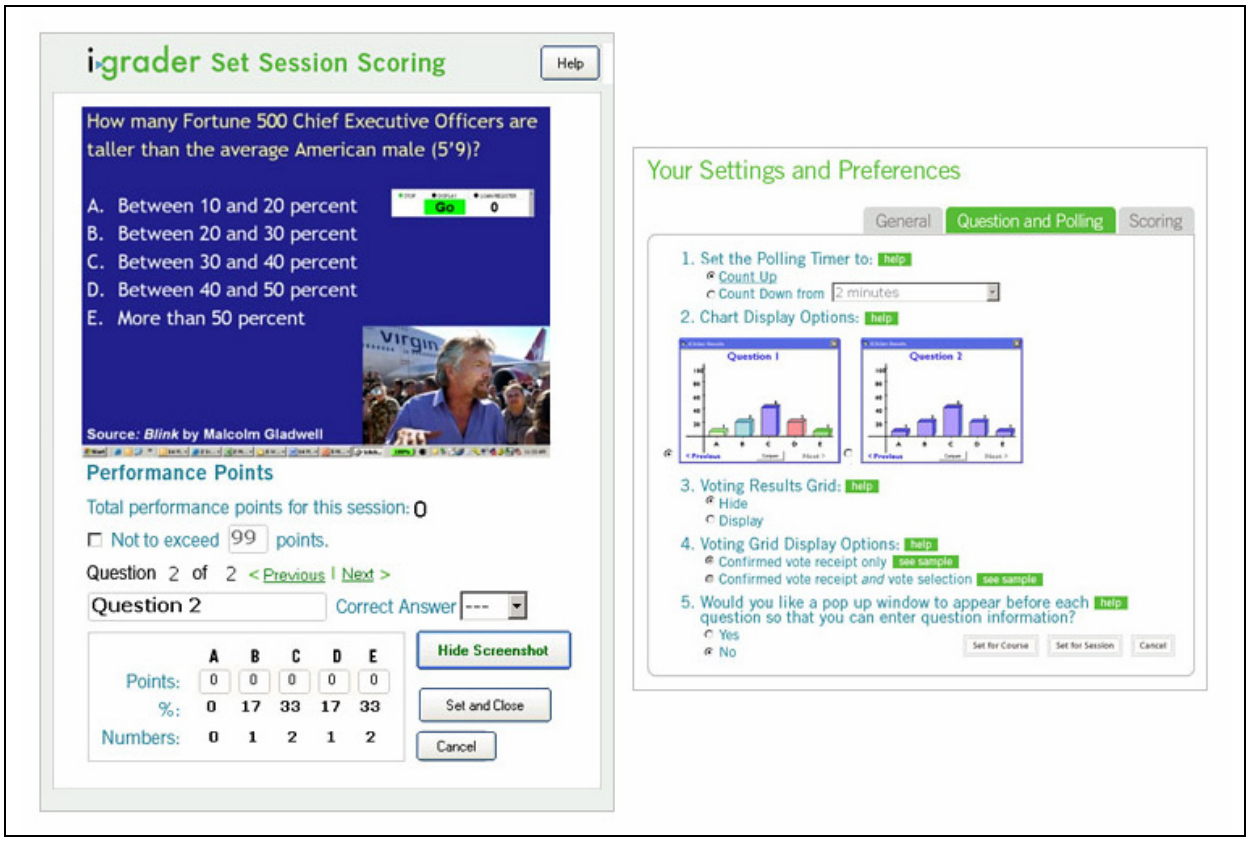


Fig. 6: Pregunta, respuestas y pantalla de configuración en i-clicker

- Dar clases en función de los conocimientos de los alumnos. Si un histograma de respuestas de un estudiante muestra que un número significativo de estudiantes escogió mal la respuesta a una pregunta, entonces el profesor puede volver de nuevo o clarificar los puntos sobre el tema de la pregunta que realizó. Si un histograma muestra que la mayor parte de estudiantes escogieron la respuesta correcta a una pregunta, entonces el profesor puede seguir avanzando con la clase.

- Ser capaz de llevar la asistencia a clase y rápidamente clasificar respuestas a un alumno concreto en tiempo de clase, para ello a cada alumno se le asigna un mando único para todo el curso.

Y por último, con un CRS se puede añadir un poco de "drama" a la clase, en el sentido de que dada una pregunta que todos los alumnos han contestado, la espera de la salida de los resultados puede suponer "un pequeño drama" a los alumnos, sobre todo si unimos esto a que, por ejemplo, las preguntas planteadas en clase

sumen nota para la calificación final de la asignatura. Ésta o alguna otra medida similar pueden hacer que los alumnos estén más concentrados en clase.

En estas dos fotografías se puede ver cómo estudiantes de una Escuela de Matemáticas y de una Escuela de Enfermería utilizan sus mandos para responder a las preguntas planteadas por el profesor.

2.4 Actividad del profesor

El profesor, aparte de dar su clase, ejerce una labor muy importante para el funcionamiento del sistema ya que:

- Explicar a los alumnos cómo funciona el sistema, qué repercusiones tendrá (si contará o no para la nota), explicarles que sus respuestas serán confidenciales, etc.

- Comprobar que los mandos emisores y los receptores funcionan perfectamente. Una buena solución es fijar una pregunta de prueba y pedir a los estudiantes que respondan para ver la funcionalidad del sistema. Este punto es importante para evitar posibles fallos o pérdidas en las respuestas de los alumnos.

- Indicar a los estudiantes cómo pueden saber si su respuesta ha sido recibida o no, así como cuál es la forma de cambiar su respuesta (en caso de que el sistema lo permita).

- Realizar preguntas con el CRS tras ciertos periodos de tiempo para hacer que los alumnos estén atentos a su clase y puedan contestar a las preguntas de manera correcta.

2.5 Hardware

En este apartado se describen qué componentes son necesarios para poder utilizar los sistemas CRS, que aunque ya se haya hablado de algunos en puntos anteriores, aquí se enumerarán todos los que se suponen.

2.5.1 Mandos Emisores

Unidades remotas de bidireccionales que son usadas por los estudiantes o miembros de audiencia para responder a preguntas planteadas por el profesor. Cuando una respuesta es correcta se le envía por medio del receptor situado en el portátil del profesor una confirmación que puede ser el encendido de una luz o el bloqueo del mando para evitar más respuestas, etc.

Suelen estar provistos de una batería en forma de pila de unos 9 voltios aunque en algunos sistemas es suficiente con un par de pilas AA, así como siempre tienen asignada una ID única (dependiendo del fabricante).

Fig. 7: Modelo de H-ITT, emisión por infrarrojos

2.5.2 Receptor o Coordinador

Estos receptores son utilizados para recoger las señales de las unidades emisoras y mandarlas al ordenador al que están conectados. Su interfaz con el ordenador suele ser por puerto de USB o puerto serie de comunicación, a través de un interfaz RS-232. En los sistemas basados en infrarrojos, cabe la posibilidad de utilizar más de un receptor por aula ya que de este modo se incrementaría el rendimiento del sistema, una medida recomendada es 1 unidad receptora por cada 50 estudiantes.

Debido a su conexión por USB no necesita alimentación extra ya que se la toma de la fuente del ordenador. En caso de necesitar alimentación externa sería necesario una pila o un alimentador conectado a la toma de la luz, para poder satisfacer la demanda energética.

Fig. 8: La unidad receptora de infrarrojos del sistema de H-ITT

2.5.3 Otros componentes necesarios

Aparte de los mandos emisores y la unidad receptora, es necesario un ordenador donde esté instalado el software necesario para el funcionamiento del sistema, así como de un proyector conectado al ordenador, para que sea posible la visualización de las preguntas así como sus resultados.

2.6 Software

Otro punto fundamental en el estudio de los CRSs es el tema de las características del software que se utiliza en dichos sistemas comerciales, y suelen ser las siguientes:

- Posibilidad de instalar varios programas o actualizaciones dependiendo del tipo de preguntas que se vayan a realizar o al tipo de personas a las que vayan dirigidas (escolares, universitarios, etc.)
- Instalación fácil e intuitiva del software.
- Multiplataforma: funcionan tanto en Windows, como en Linux o en Mac.
- Dos tipos de usos: uno para recoger respuestas y otro para clasificarlas.
- Totalmente automatizado: los ajustes y datos son salvados automáticamente.
- Interfaz "conectada" a la pizarra.

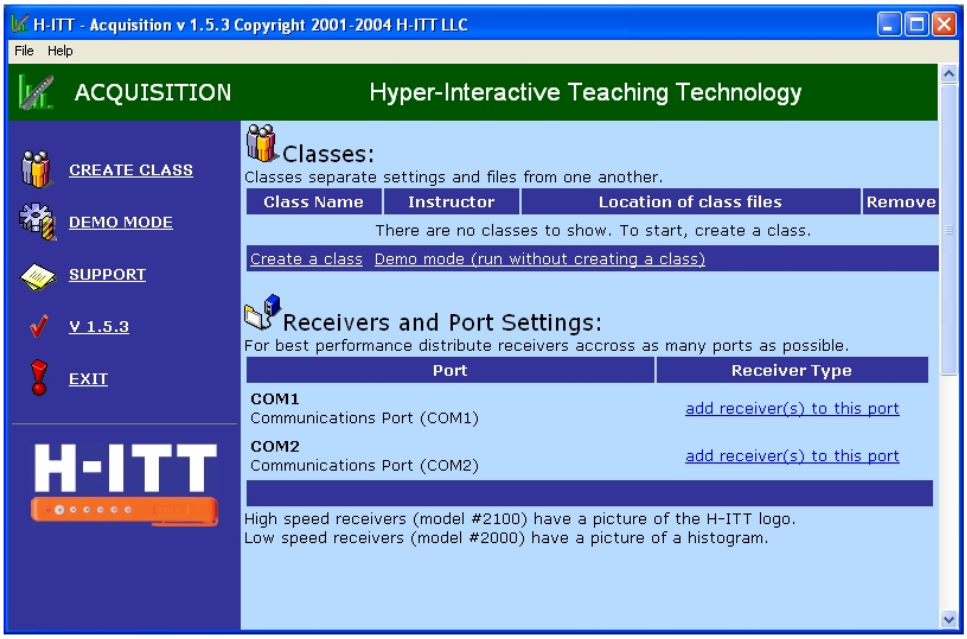


Fig. 9: Pantalla inicial del sistema H-ITT

- Construyen listas automáticamente con todos los alumnos en la clase en ese momento.

- Compatible con múltiples aplicaciones ya existentes (PowerPoint, Excel, etc.)
- Soporte para respuestas anónimas.
- Posibilidad de procesar varias respuestas de una misma persona en un instante determinado.
- Informes separados para diversas actividades como respuestas de revisión o resultados a preguntas.
- Las diapositivas creadas previamente soportan el "Cortar y Pegar".
- Facilidad en la modificación e inserción de diapositivas ya creadas.
- Diferentes formas de visualizar los resultados de las preguntas, desde gráficos de barras o circulares, o la posibilidad de ver con un gráfico (donde se ve la disposición de los alumnos en la clase) qué alumnos han contestado a qué preguntas, o incluso el tiempo que han tardado en contestar.

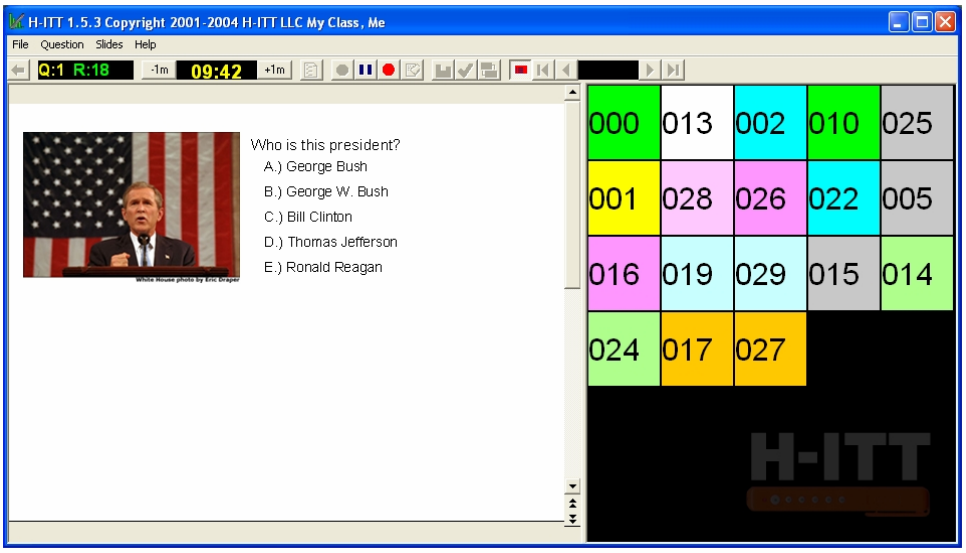


Fig. 10: Pregunta, tiempo restante y respuestas, sistema H-ITT

Además de estas características principales, existen una multitud de peculiaridades para los múltiples sistemas CRS que existen en el mercado, donde cada una intenta innovar de una u otra forma (a nivel de software), introduciendo novedades que más tarde con toda seguridad serán añadidas a los demás sistemas CRS.

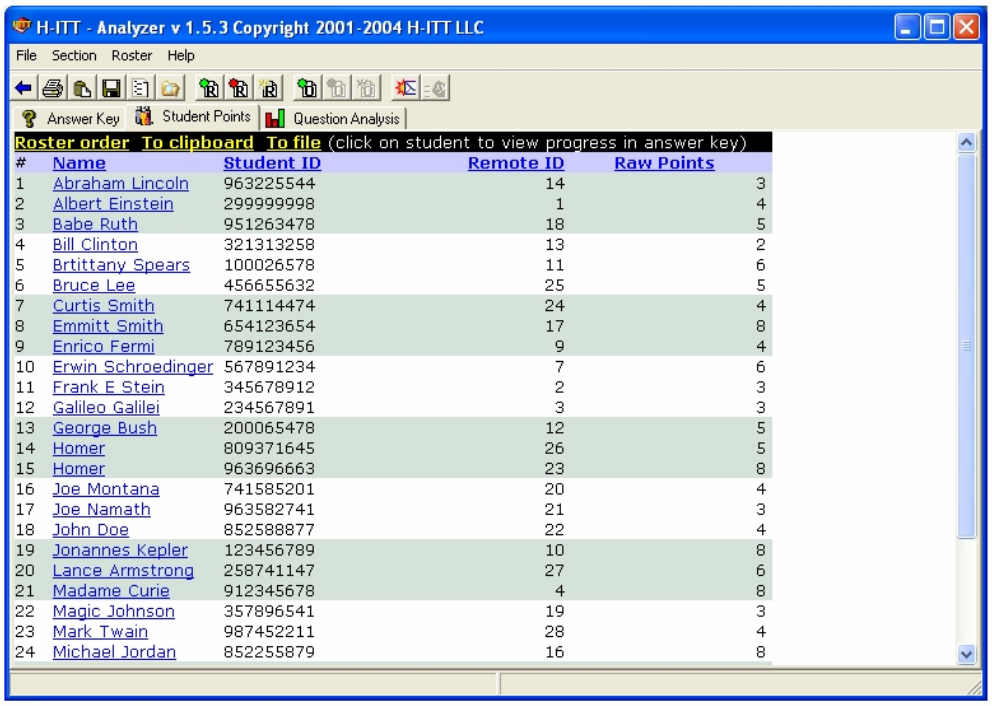


Fig. 11: Relación Alumno – ID mando – Puntos acumulados, H-ITT

2.7 Sistemas comerciales

Los sistemas CRS están siendo objeto de desarrollo y ventas por parte de muchas empresas, y cada una ofrece básicamente lo mismo, es decir, un sistema capaz de recoger las respuestas de los alumnos o participantes, procesar dichas respuestas, y mostrar los resultados. Aunque, como es lógico, cada empresa ofrece alguna funcionalidad nueva, sobre todo en su software que viene con los mandos y el receptor, como puede ser la posibilidad de cambiar la respuesta, o la posibilidad de responder a más de una pregunta a la vez. Pero también es cierto que, a medida que una empresa lanza una novedad, las demás tienden a incorporarlo a sus productos haciendo más dura la competencia en este campo.

A partir del año 2000, la cantidad de empresas que ofrecen estos productos se ha disparado. A continuación mostramos un listado de algunas empresas que ofrecen este producto:

- Wiley Higher Education.
- Interwrite learning.

- EduClick (quizá el sistema más famoso basado en CRSs).
- Turning Technologies.
- H-ITT (Hyper-Interactive Teaching Technology).
- Holtzbrinck Publishers (fabricante de i-clicker).
- …

En resumen, todas estas empresas con sus productos ofrecen la posibilidad de un nuevo tipo de enseñanza basada en las nuevas tecnologías. Hay incluso estudios pedagógicos[1] realizados que muestran la mejora que se obtiene en la educación con estos sistemas interactivos.

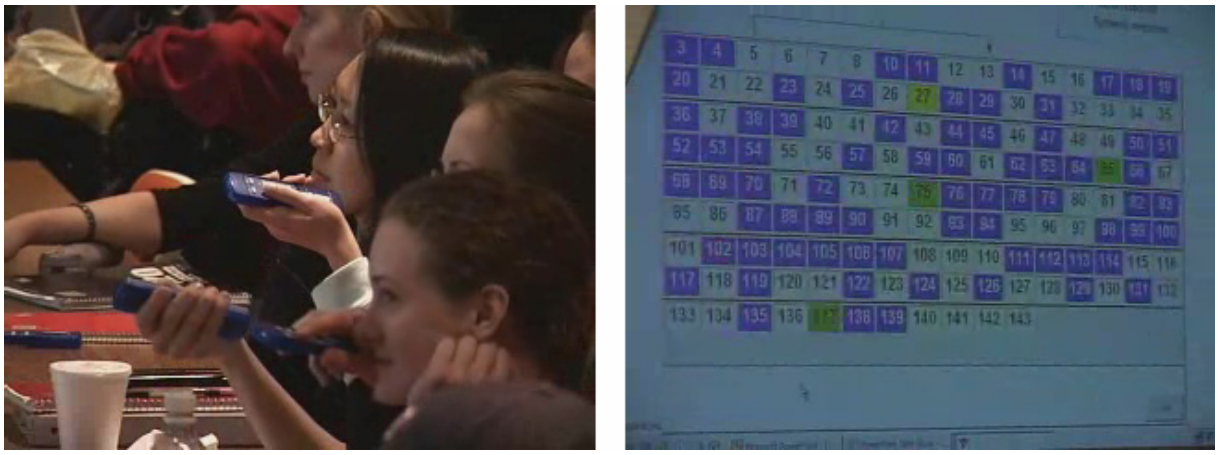


Fig. 12: Alumnos respondiendo y participación visible en directo (Classroom Performance System, University of Texas)

Como se puede observar en la gráfica de la figura posterior, en un estudio que se hizo con EduClick acerca del índice de atención en clase con y sin el sistema CRS, los resultados fueron que el índice de atención obtenido en sesiones con mandos interactivos aumenta desde el 47% hasta el 83%. En este ejemplo, el profesor ha

[1] Se pueden consultar en la siguiente dirección de Internet: http://sharepoint.cisat.jmu.edu/tsec/jim/CRS/pdf%20files/interactivity_in_classrooms.pdf

requerido respuestas de los alumnos en seis ocasiones, que corresponden a los picos en la gráfica. La atención decae algo después de cada pregunta, hasta que los alumnos tienen la oportunidad de responder nuevamente a la siguiente pregunta, en que vuelve a crecer.

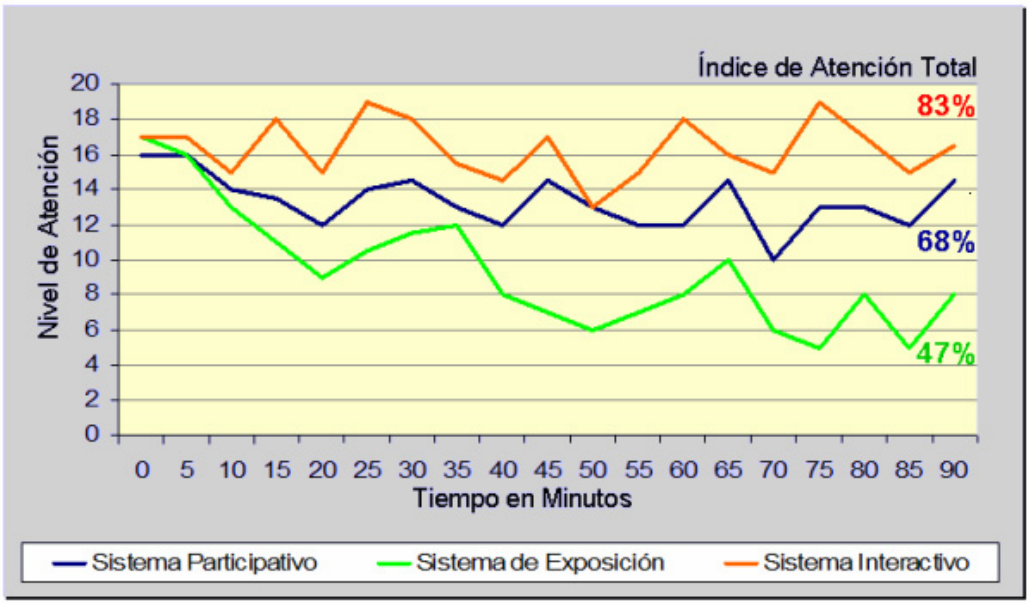


Fig. 13: Comparativa de niveles de atención según sistema docente

Gracias a esto se mejora en la participación del alumno, se permite comparar su opinión con la de los demás, se mejora la productividad de tiempo dedicado en clase…

Además la retención de conceptos mejora desde el 19% conseguida mediante el sistema participativo hasta el 27% al utilizar el sistema de mandos interactivos.

3 ENTORNO DE DESARROLLO

3.1 Microcontrolador PIC

3.1.1 Generalidades

Los microcontroladores con los que se ha trabajado son el PIC18LF4620 y el PIC18LF4550, de la popular familia PIC de la Microchip Technology Inc. tan extendida en el mundo de estos micros sencillos. Ambos microcontroladores en las versiones que hemos adquirido son casi 100% compatibles, habiéndose utilizado para el desarrollo del prototipo el 4620, con encapsulado DIP (Dual In-line Package), y para el desarrollo de la solución OEM ambos micros, con encapsulado SMD (Surface-Mount Device).

La elección del kit de desarrollo hardware *(PICDEM Z)* a utilizar del que posteriormente se hablará, se ha basado en cierta manera al micro que éste incluía, por diversas razones entre las que se pueden mencionar la gran expansión de esta familia de micros en la electrónica, y de ahí la buena documentación que se podía obtener a través de las "datasheets" u hojas de especificaciones del producto y foros de Internet, además de ser unos micros simples y de muy bajo consumo, característica muy importante en el objetivo para el que los queremos. Otra razón que ha tenido peso en la decisión de adquirir los PICDEM Z con el microcontrolador PIC ha sido la posibilidad de usar un compilador C totalmente gratuito y proporcionado por Microchip, con el que en gran parte se ahorra el esfuerzo de programación del propio micro, así como el precio relativamente barato que tiene el micro en sí.

El PIC, siendo un micro que entendemos sencillo en tanto en cuanto no tiene nada que ver con un micro de un PC, mucho más complejo y orientado a multitarea, alto rendimiento de proceso de datos y multimedia, es un procesador muy versátil para tareas sencillas que no requieran de grandes procesamientos de datos. Otras características de este micro son:

- Permite varios modos de ejecución (normal y varios de bajo consumo).

- Reloj programable, con varios osciladores que pueden servir de fuente de señal.
- Control de interrupciones (internas y externas), con posibilidad de cambiar sus prioridades.
- Módulos para comunicación serie con el exterior (MSSP y EUSART).
- Módulo USB v.2.0 para comunicación con el exterior (sólo el 4550)
- Conversor A/D.
- Comparador analógico.
- Memoria de programa tipo flash (64 KB el 4620, 32 KB el 4550, 100000 escrituras).
- Memoria de datos tipo EEPROM (1 KB el 4620, 256 bytes el 4550, 1000000 escrituras).
- Memoria de datos tipo SRAM (3.89 KB el 4620, 2 KB el 4550)
- Multiplicador 8x8 de un único ciclo.
- Conectividad para un depurador sobre el micro (Microchip ICD2).
- Watchdog Timer programable (WDT).
- …

La versión del PIC18LF4620 con la que hemos trabajado para el desarrollo del prototipo en el PICDEM Z es la de 40 pines, pero lo podemos encontrar también en versión de 44 pines. Éste es el esquema del patillaje y encapsulado del 4620, en formato PDIP (variante DIP):

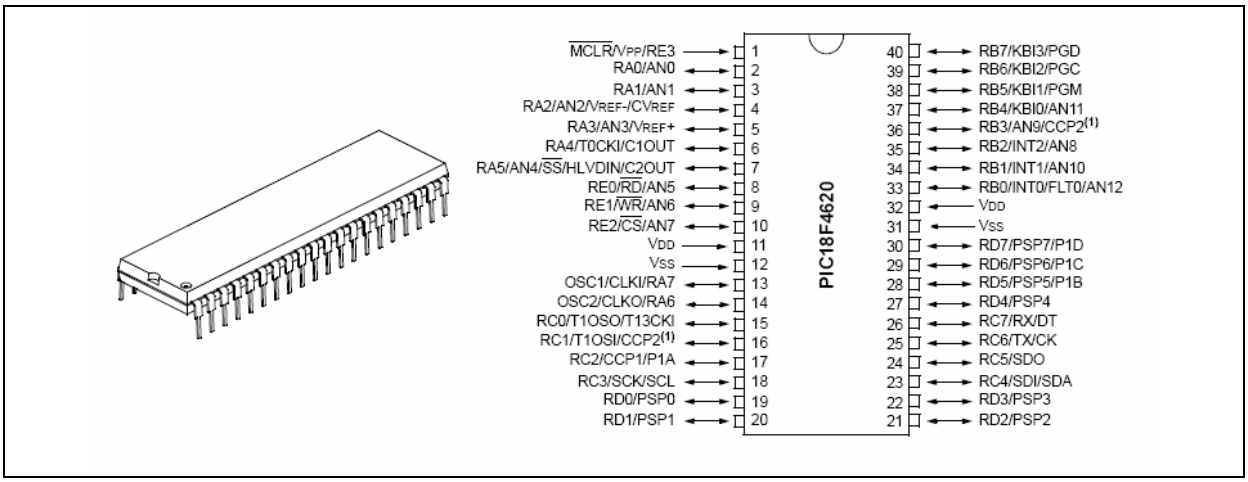


Fig. 14: Esquema del 4620, en formato PDIP (DIP)

El 4620 en formato QFN (variante SMD) presenta este otro aspecto, muchísimo más compacto que el PDIP:

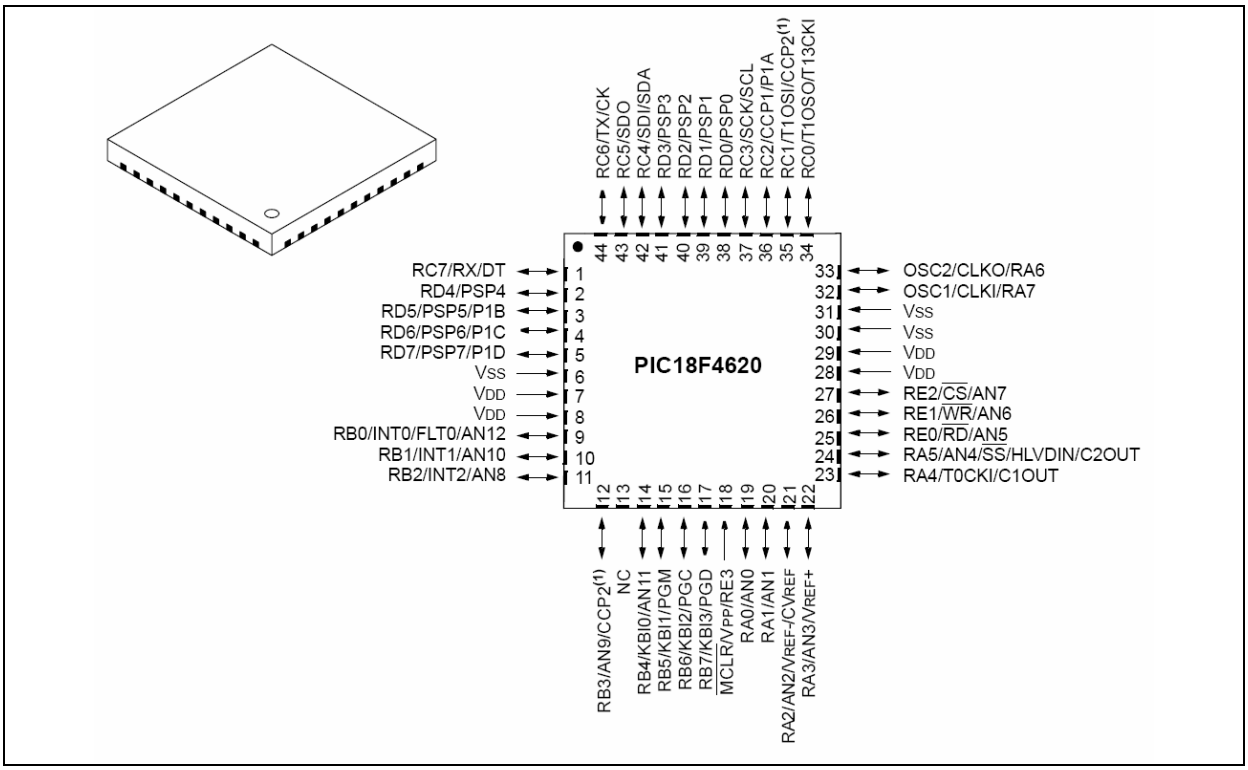


Fig. 15: Esquema del 4620 en formato QFN (SMD)

Parecida es la versión QFN de 44 pines del 4550 que hemos adquirido:

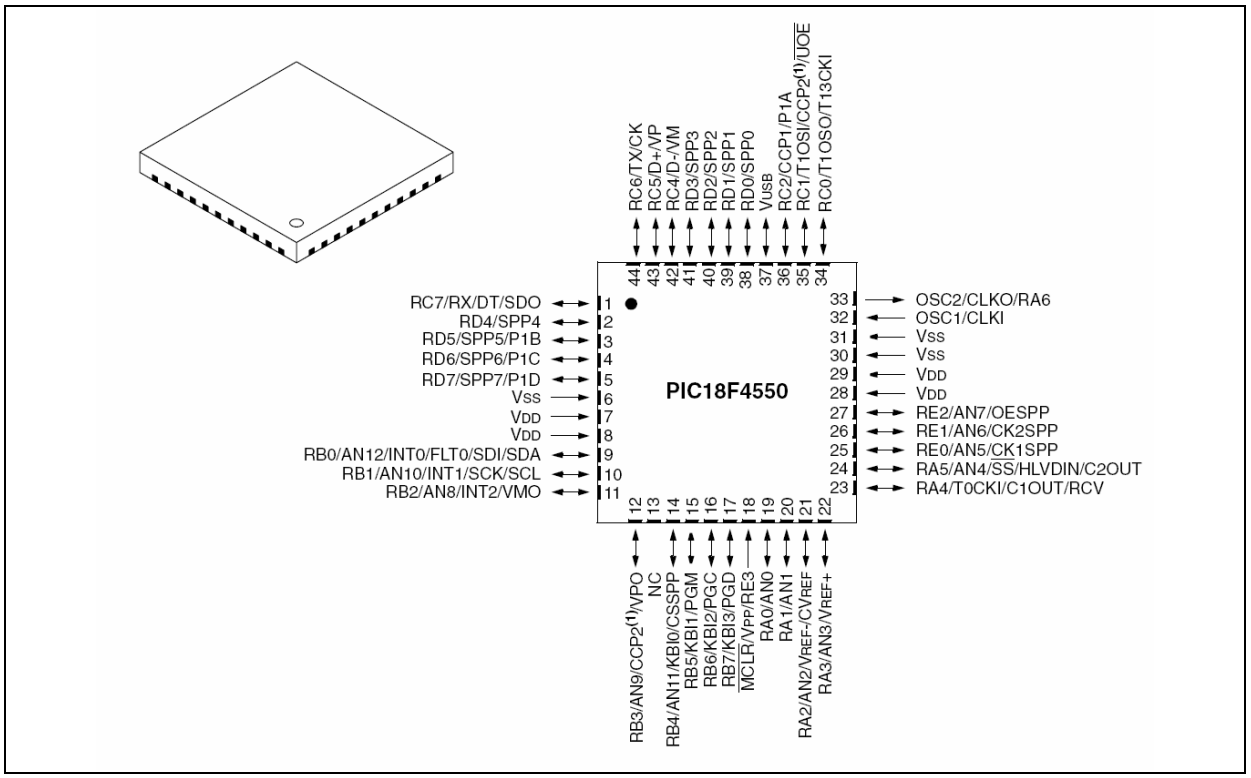


Fig. 16: Esquema del 4550 en formato QFN (SMD)

A modo ilustrativo se muestra en la siguiente figura el diagrama de bloques de ambos micros, si bien no hemos tenido que trabajar a tan bajo nivel, pero da una idea de los módulos que lo integran. No es el propósito de este apartado entrar a describir todos los pines de E/S, así como todos los registros de configuración, ni vamos a hablar de recursos que no hayamos utilizado del micro, porque para ello existen la "datasheet" de los chips que incluyen toda esa información muy bien explicada y detallada. Tan sólo comentaremos brevemente algunos aspectos de éste que hayan sido interesantes de cara a nuestros objetivos en el proyecto. Todos los subapartados que vienen a continuación versan sobre el modelo 4620, pero las explicaciones son igualmente válidas para el 4550, ya que es prácticamente de la misma construcción.

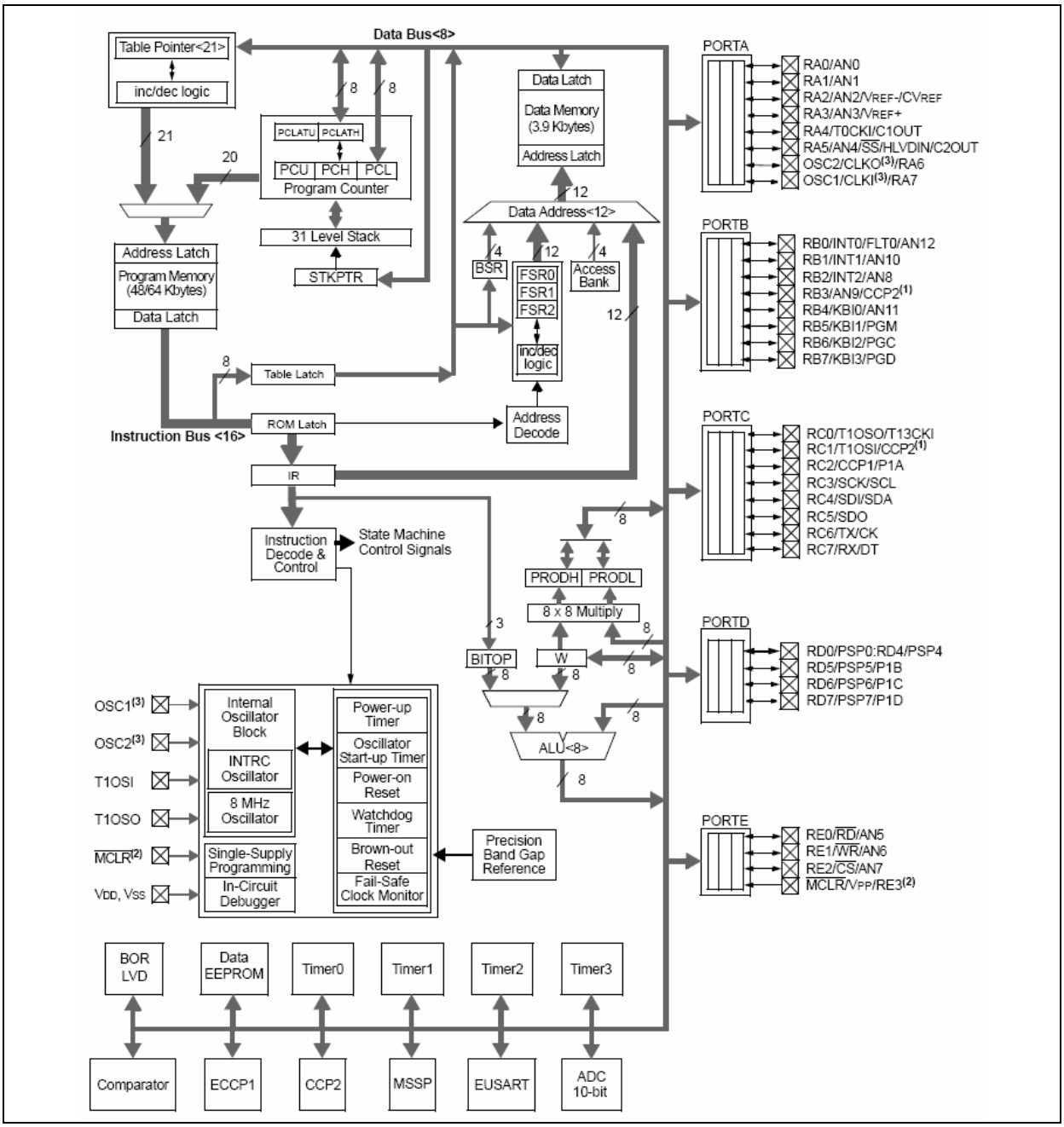


Fig. 17: Diagrama de bloques de un PIC18

3.1.2 Relojes y modos de control de energía

Un aspecto bastante interesante del PIC es su capacidad para el ahorro de energía, mediante el uso de un reloj más lento o la suspensión de ejecución parcial o total. Los modos de control de energía corresponden a esta última idea, que incluso llevan incluida la primera.

Como fuentes de reloj se tienen varios osciladores que podemos clasificar en:

- *Primarios:* uno principal e interno de 8 MHz (situado en el "bloque del oscilador interno"), además de varios externos conectados a los pines OSC.

- *Secundarios:* no conectados a los pines OSC, pueden funcionar aun teniendo el controlador en un modo de control de energía. Tanto el 4620 como el 4550 tienen como secundario el llamado Timer1 (32.702 KHz), usado como fuente de tiempo real además de permitir bajo consumo si se desea.

- *Bloque del oscilador interno:* también puede usarse en algunos de los modos de control de energía, y sirve de base para otros módulos tales como el Watchdog timer (WDT).

Una vez distinguidos los tipos de fuentes de reloj, los modos de control de energía pueden clasificarse en:

- Modos de ejecución *("run modes").*

- Modos de ocio *("idle modes").*

- Modo de dormir *("sleep mode").*

Hay tres modos de ejecución PRI_RUN, SEC_RUN y RC_RUN. En todos ellos, tanto CPU como periféricos están funcionando, la diferencia entre ellos es la fuente de reloj. El primero (PRI_RUN) es el modo normal y por defecto tras un reset, con derroche máximo de energía, y en el que la fuente de reloj es programable, mientras que en el segundo (SEC_RUN) hace uso del Timer1 para dar sincronía tanto a CPU como a periféricos. En el tercer modo (RC_RUN), tanto CPU como periféricos se sincronizan mediante el oscilador interno, y con la ventaja que es el modo de ejecución que menos energía consume.

Los llamados "modos de ocio" son aquellos para los que el procesador está apagado mientras los periféricos siguen funcionando. Tienen una nomenclatura similar a los modos de ejecución (PRI, SEC y RC), ya que existe una correspondencia unívoca entre éstos y ellos, dada por el cambio de modo, desde ejecución a ocio. Para volver a un modo de ejecución es necesaria una interrupción, un reset del chip o un time-out del WDT.

Para acabar con los modos de control de energía, el modo de dormir (sleep mode) desconecta los relojes tanto de CPU como de periféricos, entrando en suspensión

completa del chip. Sin duda es con el que se consigue un ahorro de energía máximo, a costa de no hacer nada y tener todo parado, con la salvedad que podemos despertarlo en cualquier instante por una interrupción antes programada o un time-out del WDT, lo cual es una ventaja frente a apagar todo el circuito y perder el puntero de pila, contador de programa, etc. La otra forma de salir de este modo es mediante la señal de reset, por la que volvería a ejecución.

La aplicación de este último modo en nuestro proyecto es muy importante, hemos hecho uso de la suspensión total de ejecución para no malgastar energía en el mando a distancia mientras no tenga que enviarse ninguna respuesta, despertando el chip con la pulsación de uno de los botones de respuesta, enviar ésta y volver a dormir.

3.1.3 Sistema de memoria

En este microcontrolador tenemos tres memorias diferentes: una tipo flash de programa, y dos de datos, de tipos EEPROM y SRAM. Es una arquitectura tipo Harvard en que se disponen de buses separados para instrucciones y datos, y además se puede considerar la memoria de datos EEPROM como un periférico cualquiera, pues su acceso se realiza mediante registros de control. Tan sólo indicamos el esquema de las memorias y algunas observaciones.

La memoria de programa que mostramos en la Fig. 18 presenta un esquema típico, en el que se permiten 31 niveles de anidamiento para subrutinas o interrupciones, y las instrucciones ocupan 2 ó 4 bytes. Se puede observar también que hay dos vectores de interrupción, y eso es porque se van a tener dos rutinas de tratamiento de interrupción en el programa, dependiendo si vienen por la línea de alta o baja prioridad. En el 4620, el tamaño de esta memoria es de 64 KB y como las instrucciones son de tamaño word, hacen un total de 32768 instrucciones como máximo incluyendo el espacio reservado. (Para el micro 4550 estos tamaños son la mitad.) La latencia de acceso para esta memoria de programa es de un ciclo salvo algunos saltos.

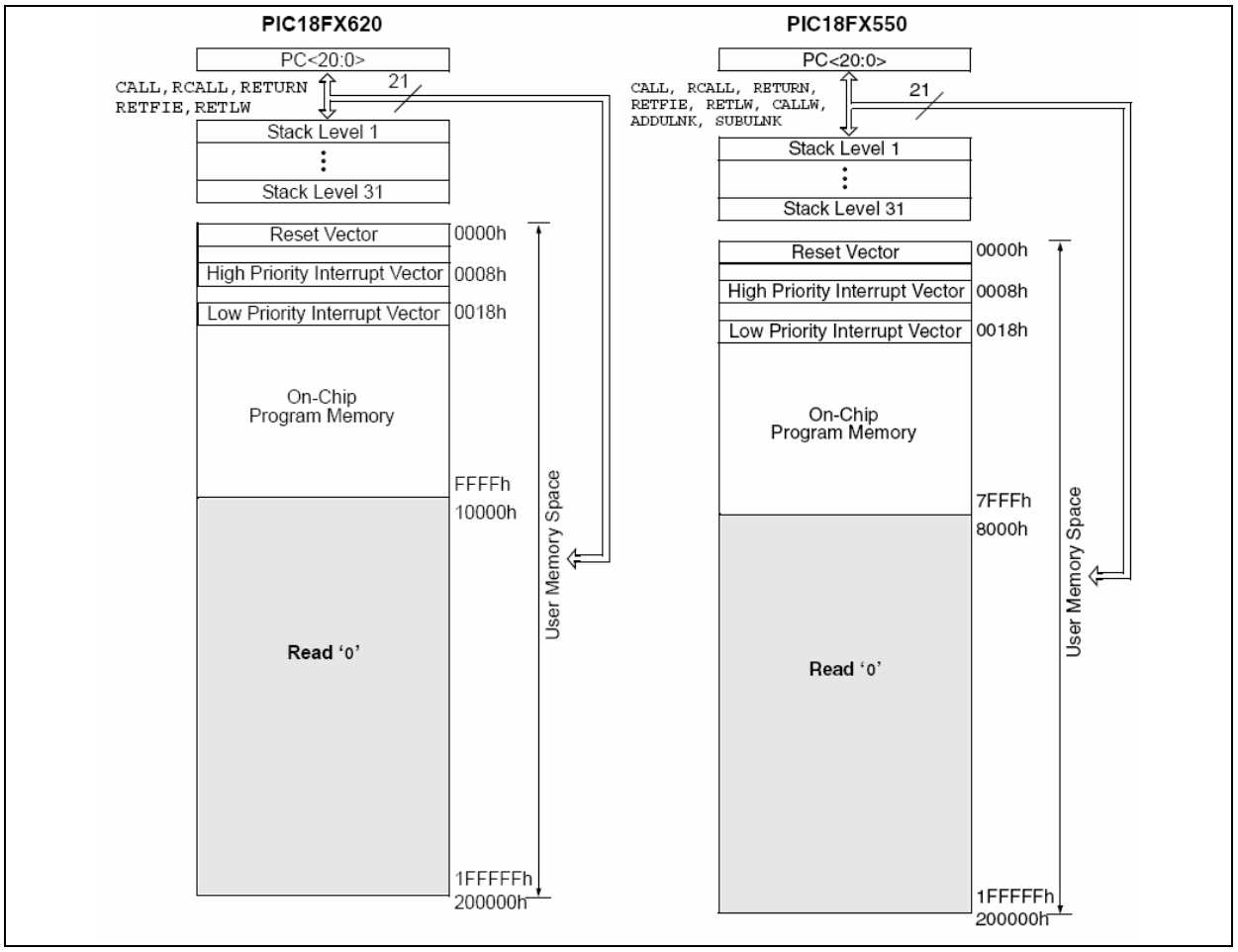


Fig. 18: Memoria de programa del 4620 (izq.) y 4550 (der.)

La memoria SRAM de datos tiene la peculiaridad de que mapea los registros de funciones especiales (SFRs), usados para estado y control de periféricos y los de propósito general (GPRs), para datos. La estructura de esta memoria de datos es de una organización en bancos, para conseguir un rápido acceso a datos, con los direccionamientos típicos conocidos. El esquema de esta memoria en el 4620 se compone de 16 bancos de 256 bytes cada uno, para un total de 4 KB, siendo las direcciones más altas las que mapean los registros especiales (128 bytes) y el resto quedando para propósito general. En el 4550 hay una diferencia, además de que son la mitad de bancos (8, para un total de 2 KB), y es que 4 de los bancos se usan como buffer de memoria de USB (USB RAM) si está activo el módulo USB; si no, se tiene igual que en el 4620 por direcciones de registros de propósito general. Además, los registros especiales se incrementan en 32 bytes más debido al control del módulo USB.

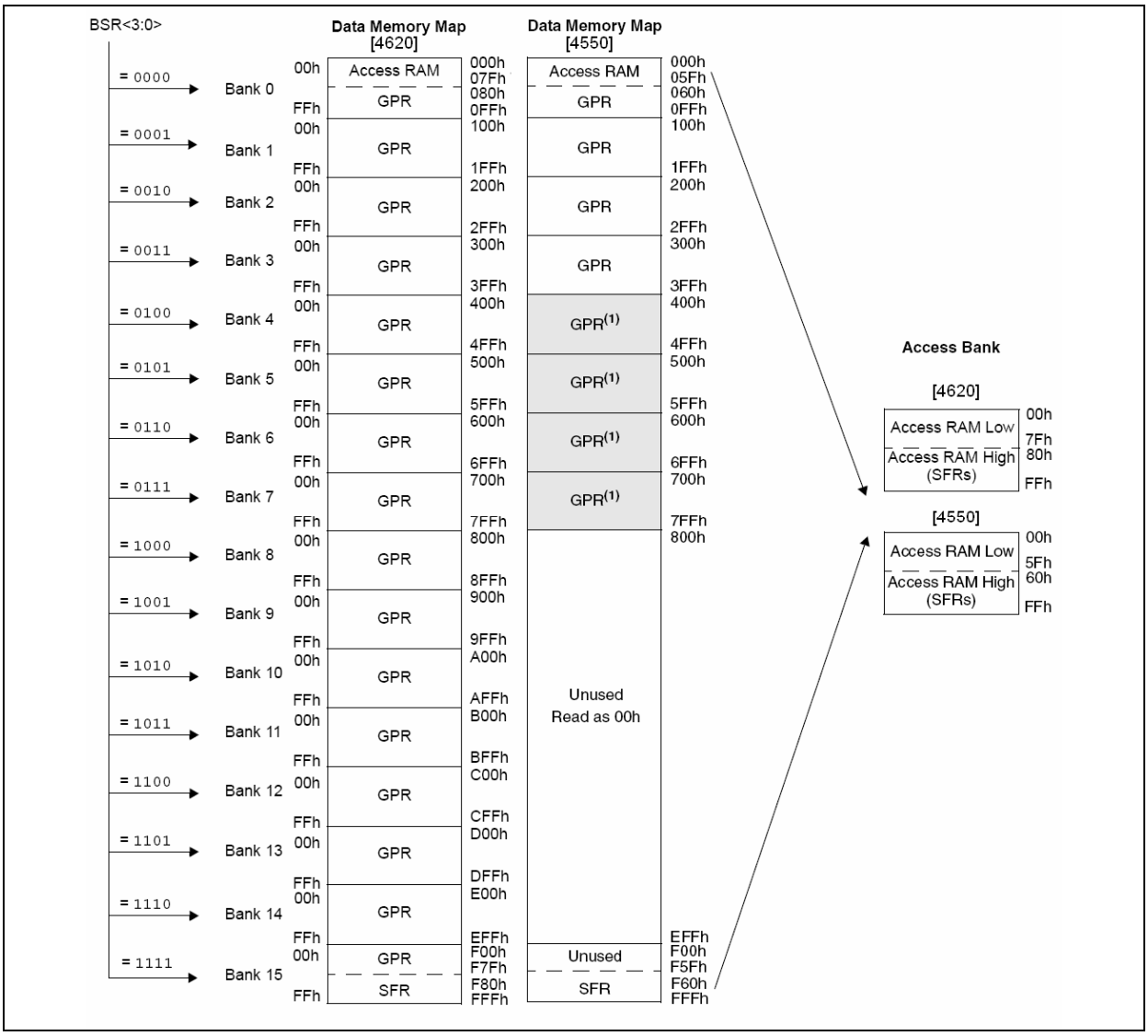


Fig. 19: Comparación de memoria de datos 4620 ↔ 4550

Por último, tenemos la memoria EEPROM de datos, pensada para almacenar datos de programa con larga duración. Se accede a ella indirectamente a través de registros especiales, y se permite lectura o escritura de 1 byte cada vez. Las escrituras se realizan tras un borrado previo (erase-before-write), y controladas por un timer, según las condiciones de temperatura y voltaje. Y ya simplemente mencionar el tamaño de esta memoria en ambos micros, siendo de 1 KB en el 4620 y de 256 bytes en el 4550.

3.1.4 Sistema de interrupciones

Estos micros tienen varias fuentes de interrupción, entre las que se pueden mencionar periféricos, un time-out de un timer, o un cambio en un pin de E/S común. Además de enmascaramientos de interrupciones se permite selectivamente dar "categoría de interrupción" a alguna de las fuentes, o dicho de otra forma, elegir para dicha fuente una alta o baja prioridad de interrupción. Así pues, por cada fuente posible de interrupción en general tendremos 3 bits: el de enable, el de prioridad y el flag de interrupción. Los dos vectores de interrupción antes mencionados en otro apartado vienen a corresponder a cada una de las líneas de preferencia. Las interrupciones se pueden capacitar o incapacitar por bloques de prioridad, pudiendo dejar sólo activas las de alta prioridad; además, una fuente de interrupción de mayor prioridad que otra segunda, puede interrumpirla. Podríamos comentar también el funcionamiento del micro al llegarle una activación de una línea de interrupción que está habilitada, pero es análogo al de cualquier otro microprocesador conocido.

Lo que sí vamos a indicar de forma general es el esquema completo de la lógica de interrupciones de un microcontrolador PIC18, que ya sí es particular de este grupo de micros. Podemos verlo en la Fig. 20.

En él se pueden observar los bits de los registros de control de interrupciones, además de las líneas de interrupción. Cada grupo de tres bits en una puerta AND con mismo nombre los tres salvo la última letra, corresponde a una misma fuente de interrupción. La última letra sirve para distinguir las tres categorías generales de bits que mencionamos al principio, siendo F para el bit de "Flag", E para el de "Enable", y P para la prioridad ("Priority"). Los que ocupan las puertas AND más generales en cada rama del supuesto árbol obviamente son los bits de capacitación generales. También cabe mencionar si acaso los registros relacionados con la capacitación y control de interrupciones, como son los tres INTCON ("Interrupt Control"), con bits de capacitación, flags y prioridad, los dos PIR ("Peripheral Interrupt Request (Flag)"), con flags relativos a los periféricos, los dos PIE ("Peripheral Interrupt Enable"), con las capacitaciones para las interrupciones de los periféricos, los dos IPR ("Peripheral Interrupt Priority"), con las prioridades de las interrupciones de periféricos y el RCON ("Reset Control Register"), con información relativa a las causas que provocaron un reset del dispositivo o un wake-up desde un estado de suspensión.

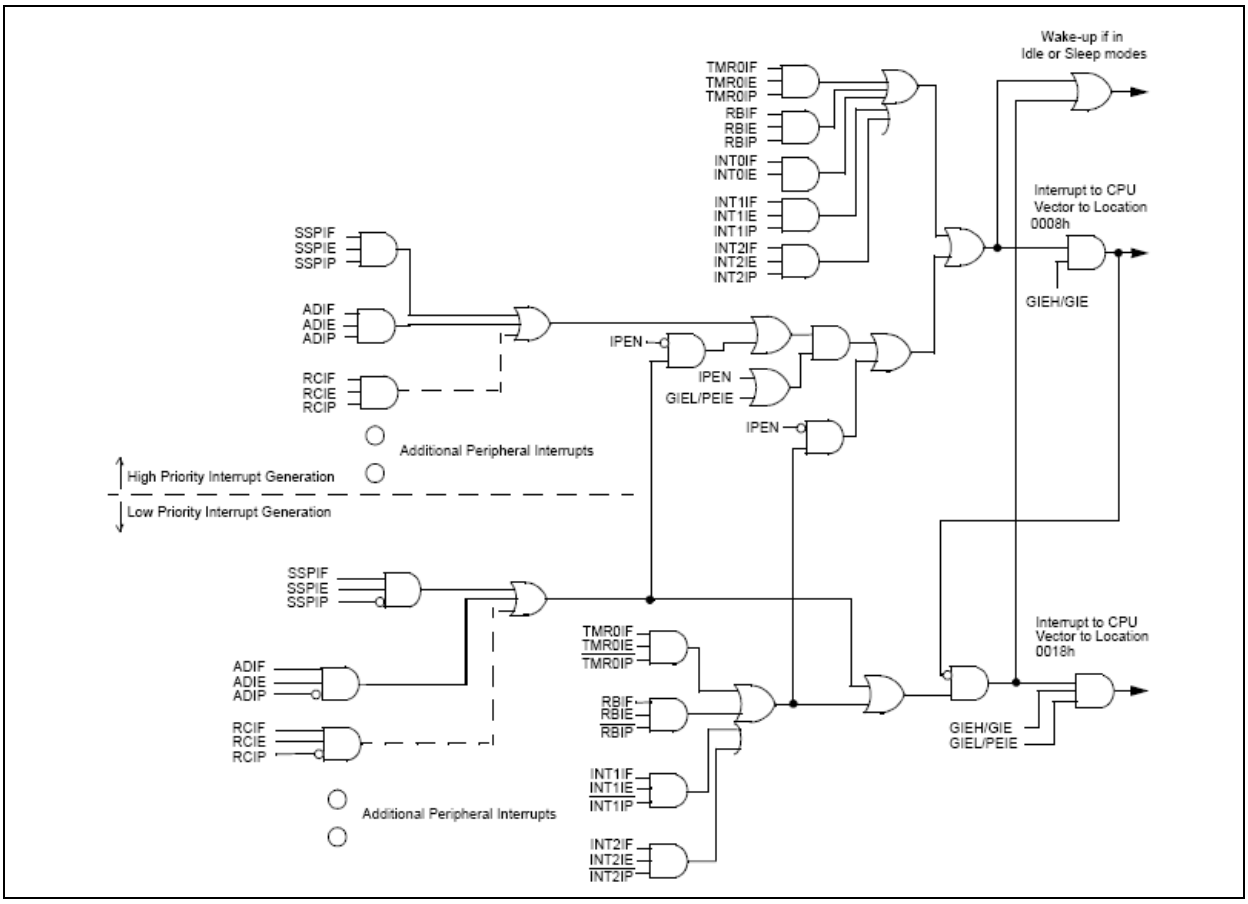


Fig. 20: Lógica de interrupciones de un PIC18

No vamos a entrar más en detalle que indicar un caso concreto que hemos tenido que tomar en nuestra aplicación a modo de ejemplo, ya que el propio esquema, aun sin explicar la descripción de cada bit que aparece, es bastante ilustrativo del modo de funcionamiento general del sistema de interrupciones de un PIC18. Por ejemplo, para nuestra aplicación en concreto hemos tenido que hacer uso de las interrupciones por los pines de E/S llamados puerto B y que a la postre corresponden a un registro de un byte llamado PORTB. El PORTB se compone de 8 bits que se denominan RBi, siendo $0 \leq i \leq 7$, de menor a mayor significatividad, y siendo sólo posible usar como fuentes de interrupción los 4 más significativos, es decir, de RB4 a RB7. En nuestro esquema electrónico dichos pines están conectados a los pulsadores de respuesta, así pues tenemos habilitadas interrupciones para detectar por este medio cuándo hemos pulsado un botón para responder. En el siguiente apartado se va a comentar algo más en detalle los registros y el funcionamiento de este puerto de E/S.

3.1.5 El PORTB

En este punto vamos a comentar brevemente los aspectos de interés sobre este registro y otros que están relacionados con él, como son el LATB y TRISB, comenzando por unas generalidades de los puertos de E/S.

Como se puede imaginar nada más haber enumerado estos tres registros relacionados entre sí (en nomenclatura por la letra 'B'), es cierto que tendremos otros puertos de E/S. En total son cinco, y sus distintivos son las letras de la 'A' a la 'E'. En general, tenemos que un puerto de E/S de éstos, requiere de tres registros de 8 bits para su funcionamiento: PORT, TRIS y LAT. El registro PORT guarda el estado de los niveles lógicos de los pines del micro, el TRIS sirve para indicar la dirección de los datos, desde o hacia el micro (E/S), y el LAT no es más que un latch de salida donde podemos leer un estado. Todos estos registros están mapeados en memoria.

El esquema general de un pin de uno de los puertos correspondería a algo así:

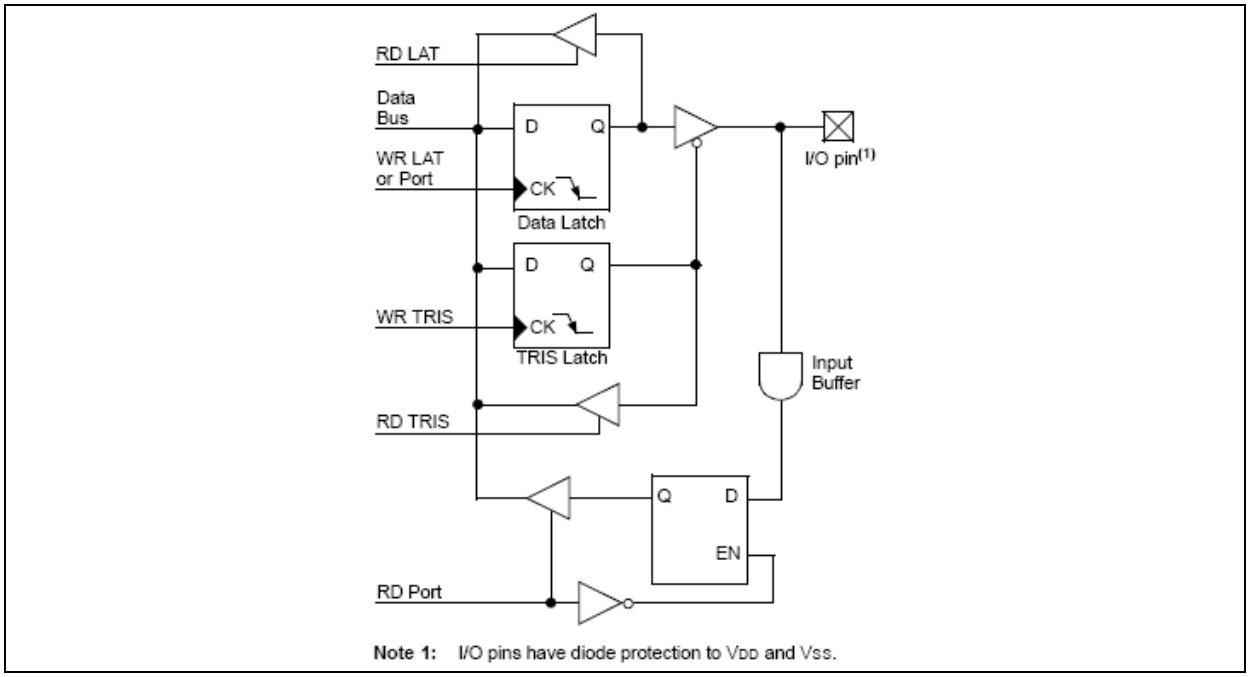


Fig. 21: Esquema de puertos de E/S

Se puede observar más o menos claramente el comportamiento que hemos descrito antes para cada uno de los registros que entran en juego al realizar una operación de entrada o de salida.

Vamos a particularizar ahora un poco hablando del puerto B y sus registros. El PORTB es un registro de 8 bits que se puede leer y escribir "directamente", sin mediar algún otro tipo de configuración extra. Su registro asociado de dirección del flujo de datos es el TRISB, y dependiendo del valor de cada uno de sus bits, indicaremos que los bits correspondientes del registro PORTB serán de entrada o de salida. Para que un bit del PORTB sea una entrada, el correspondiente bit de TRISB ha de estar a '1', siendo en caso contrario, una salida, copiándose su valor en el latch de salida correspondiente, es decir, el bit de LATB que sea en cada caso. Así pues, al hacer una operación de lectura sobre el estado del puerto, se está leyendo del registro LATB, y al hacer una operación de escritura, lo que hacemos es variar el valor en PORTB.

Podemos mencionar también en este punto que cada uno de los pines de PORTB tiene una pequeña resistencia de configuración interna ("pull-up resistor"), que puede habilitarse o no según conveniencia. El objeto de estas pequeñas resistencias es no dejar ningún pin al aire y como entrada, en cuyo caso los valores que podemos leer son prácticamente aleatorios y nada convenientes para el correcto funcionamiento de la aplicación que corra en el micro.

Otra peculiaridad del puerto B es la ya mencionada antes en el apartado de interrupciones, y es la posibilidad de que los 4 pines más significativos (RB4:RB7) puedan actuar como fuente de interrupción del microcontrolador, siempre que estos bits se indiquen como entradas (TRISB<i> = 1) y esté habilitada la línea de interrupción del puerto B por el bit RBIE (INTCON<3>). Para ello se compara su nuevo valor con el ya guardado en el latch de salida, provocando si así fuera, una interrupción por cambio de estado del puerto ("interrupt-on-change"). Una vez se ha provocado la interrupción, conviene limpiar el flag correspondiente RBIF para no tener problemas en la rutina de tratamiento de la interrupción correspondiente.

3.2 ZigBee

3.2.1 Introducción

En este apartado se van a comentar algunos aspectos que permitan conocer un poco a modo introductorio esta tecnología de comunicación.

3.2.1.1 ¿Qué es?

El origen de la palabra ZigBee viene de la forma de comunicarse de las abejas ("bee" en inglés), mediante vuelos en zig-zag. El ZigBee es un protocolo de comunicaciones inalámbrico, definido por la ZigBee Alliance para el estándar de comunicación 802.15.4, definido por el IEEE en 2003 para redes PAN inalámbricas mediante radiofrecuencia. La ZigBee Alliance surge como necesidad de completar el estándar del IEEE para disponer de una tecnología inalámbrica de bajo coste, y la forman más de un centenar de empresas relacionadas con este sector de las comunicaciones, y desde 2004 se cuenta con versiones completamente operativas.

La pila de red la forman básicamente cuatro capas, física, de acceso al medio, de red y de aplicación. El estándar 802.15.4 del IEEE define las capas más bajas de esta pila de red (la parte hardware), esto es, las capas física y de acceso al medio, y la alianza de empresas mantiene el protocolo en las capas superiores (parte software), la de red y la de aplicación, ya de usuario. Esto se detallará un poco más en otro apartado posterior.

La idea del surgimiento de este estándar es ocupar un poco el hueco dejado por debajo del Bluetooth en aplicaciones donde el uso de éste no es muy conveniente. Es parecido a Bluetooth, pero mucho más sencillo y barato, y de mucho menos consumo que pueden dotar al dispositivo que lo incorpora de una gran autonomía, además de una gran escalabilidad de red. Su principal uso es para comunicar sensores o controladores, y todas estas características antes mencionadas son muy destacables para ello. Como principal desventaja frente a Bluetooth se puede destacar el menor ancho de banda en la transmisión de datos, que es de 256 kbps como máximo, frente a los 1 Mbps del Bluetooth.

Poco a poco iremos viendo como este estándar de "reciente" aparición va incorporándose a nuestra vida, mediante control inteligente de nuestros electrodomésticos (domótica), mediciones biométricas, teléfono móvil como gateway a redes de dispositivos ZigBee, como puede ser el pago de una entrada de cine, y un largo etcétera.

3.2.1.2 Topologías de red

A la hora de hablar de una red, se antoja evidente pensar en diferentes funcionalidades en los dispositivos que conforman la propia red. Vamos pues a definir los tipos de dispositivos que pueden formar parte de una red ZigBee, y posteriormente indicaremos ya las posibilidades topológicas que ofrece este estándar.

En una red ZigBee se tienen dos posibles tipos de dispositivo:

- FFD, "Full Function Device" (Dispositivo de Función Completa): Puede actuar como Coordinador de la WPAN, como router o como elemento terminal, es decir, admiten todas las funcionalidades que puede tener un dispositivo de red ZigBee.

- RFD, "Reduced Function Device" (Dispositivo de Función Reducida): actúan como dispositivos terminales, y como su nombre indica, tan sólo tienen capacidad de envío/recepción de datos con un FFD concreto.

En toda red ZigBee se tiene un único Coordinador de red, que puede estar alimentado directamente de una toma de electricidad pues ha de estar funcionando todo el tiempo. Los demás nodos de la red generalmente irán alimentados por baterías, especialmente los nodos terminales.

Entre las topologías de red más comunes que se pueden implementar mediante ZigBee, se tienen estas tres:

- Red de Estrella. Se tiene una disposición de uno o más nodos terminales que hablan directamente con el coordinador. Inicialmente se arranca el coordinador, establece su red con identificador propio, y empieza a aceptar nodos en ella según éstos se lo solicitan.

- Red de Malla (punto a punto). La diferencia principal con el anterior tipo, es que cada nodo puede hablar directamente con otros que estén en su rango. La propia red es capaz de configurarse ella sola y repararse si cae un nodo.

- Red de Árbol. Es similar a la anterior pero teniendo en cuenta que la comunicación entre nodos es en forma de árbol, donde la raíz es el coordinador de red, y los RFDs sólo pueden ser hojas. El nodo coordinador

inicia la red lanzando una baliza que pueden recoger otros dispositivos solicitando su entrada en la red que forma el coordinador. Una vez el coordinador los admite, éstos pueden lanzar balizas a otros dispositivos para que se les unan a ellos. La principal ventaja es que se puede agrandar mucho la cobertura de la red, pero siempre a coste de la latencia de los mensajes.

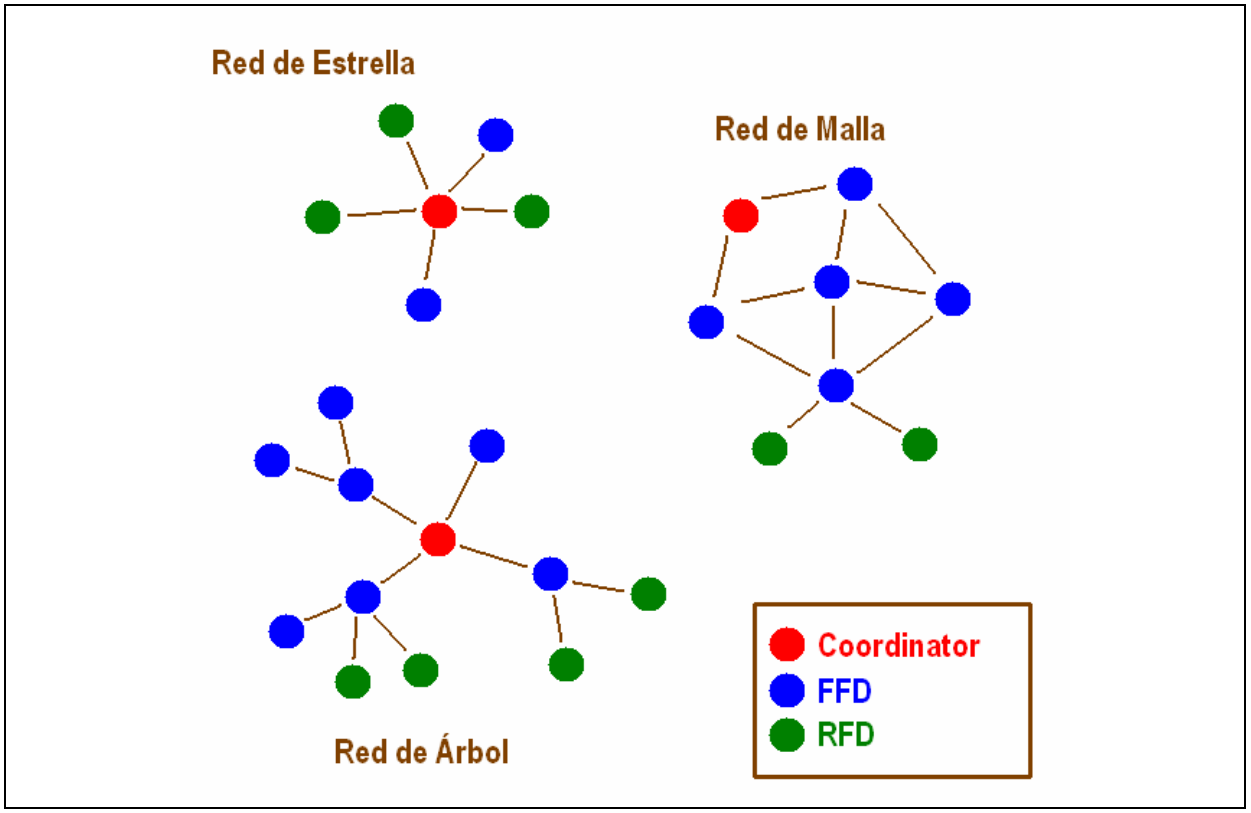


Fig. 22: Topologías de red ZigBee

3.2.1.3 Algunos conceptos

En este punto vamos a comentar algunos de los conceptos de la terminología ZigBee, referidos a la comunicación.

Llamamos *perfil* ("profile") a un acuerdo común en el conjunto de mensajes, formatos de mensaje y acciones de procesado que capacitan a diferentes dispositivos ZigBee para enviar comandos, solicitar datos y procesar comandos o peticiones para tener una aplicación distribuida. Es una "descripción" de la lógica de los dispositivos y sus interfaces. Los perfiles los desarrollan los fabricantes de dispositivos ZigBee para resolver de antemano problemas específicos, y su labor de

alguna forma es unificar esas soluciones con el estándar ZigBee aunando los esfuerzos de todos en un producto concreto.

Por otro lado, llamamos *atributo* ("attribute") a todo tipo de dato individual que podemos enviar de un dispositivo a otro. A cada atributo se le asigna un identificador único, quedando todos los de un mismo identificador agrupados bajo él en un *cluster*, y es a este nivel donde se especifican los interfaces. Un cluster se asocia directamente con un flujo de datos de entrada o salida del dispositivo, y a la hora de hacer las asociaciones de aplicación entre dispositivos se tienen en cuenta los identificadores de cluster, de forma que además pertenezcan a un mismo perfil.

Así pues, un perfil define los identificadores tanto de atributos como de clusters, así como el formato de los atributos, y el significado de sus valores. También sirve para indicar cuáles de los clusters son obligados y cuáles no, además de poder definir ciertos servicios ZigBee opcionales como obligatorios.

Teniendo en cuenta todas estas definiciones, un desarrollador puede ya crear código para su aplicación.

Cada bloque funcional de código que soporta uno o varios clusters se denomina *"endpoint"*, y es el punto por donde se comunican los diferentes dispositivos. Es decir, que dos dispositivos ZigBee pueden comunicarse si comparten un mismo perfil, mediante un mismo cluster y a través de sus respectivos endpoints, diferentes en cada extremo de la comunicación. Para saber un dispositivo el endpoint al que ha de enviar los datos pertinentes, los coordinadores de red disponen de una *Tabla de Asociación* ("Binding Table") donde se relacionan los cluster y endpoint de los dispositivos fuente y destino. El ejemplo típico que se muestra es el siguiente:

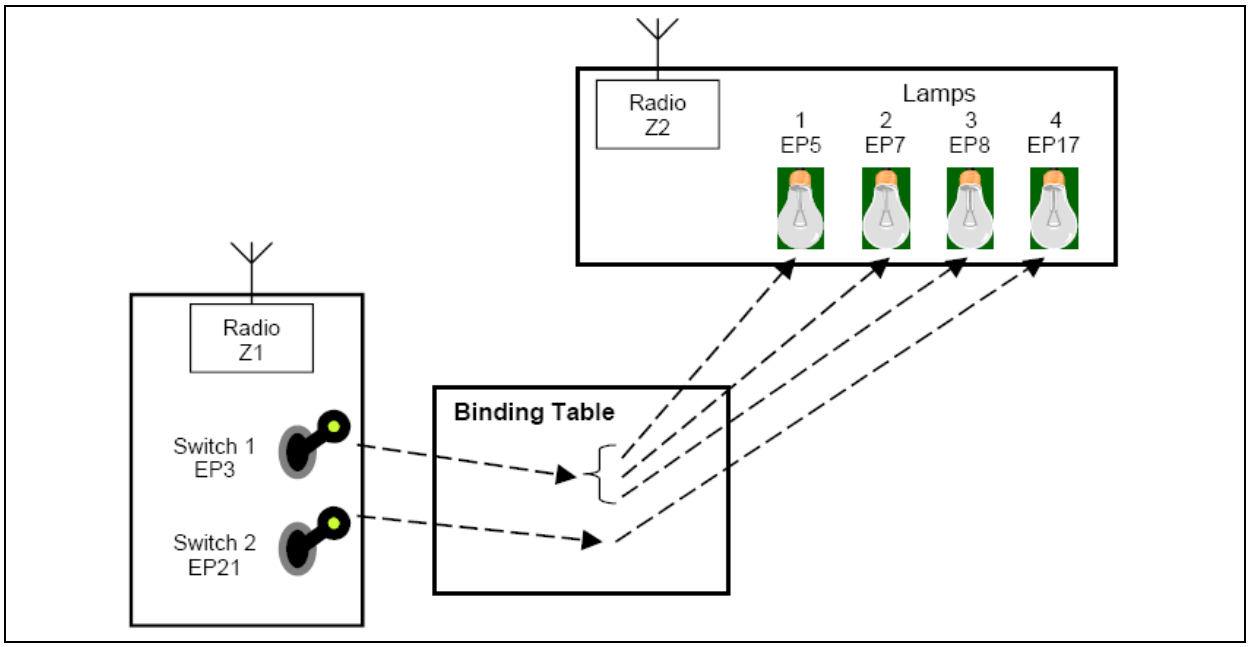


Fig. 23: Ejemplo de asociación (binding)

Se puede observar que en el dispositivo #1 (el controlador) se tienen un par de switches, cada uno asociado a un endpoint diferente, y en el dispositivo #2 (el controlado) tenemos cuatro bombillas, cada una asociada también a un endpoint diferente. El switch #1 controla un total de tres bombillas, las etiquetadas con los números 1..3, mientras que el switch #2 tan sólo controla la bombilla #4. A la hora de mandar una orden de encender o apagar una, mediante el cambio de posición de un switch, se consultará en la Tabla de Asociación ("Binding Table") el dispositivo al que está asociado el endpoint ordenante, y se formalizará el envío de la orden con la dirección y el endpoint de destino. Podría encenderse/apagarse una misma bombilla con los dos switches, siempre que exista una entrada en la Tabla de Asociación que lo permita. La razón de que esta tabla resida en el coordinador de red es que siempre ha de estar disponible para todos, y éste dispositivo principal suele estar alimentado por la red eléctrica directamente, sin necesidad de baterías, así que puede estar permanentemente funcionando.

El proceso de asociación (binding) tiene lugar antes de comenzar la comunicación en ejecución normal de la aplicación, y puede hacerse sólo con el coordinador de red introduciendo éste la entrada correspondiente, o a petición de un nodo cualquiera. De hecho, es un requisito para comunicarse, si no hay asociación, no hay comunicación.

Finalmente, podemos indicar mediante un esquema cómo quedan relacionados endpoints, clusters, atributos y la propia aplicación, basándonos en el mismo ejemplo anterior:

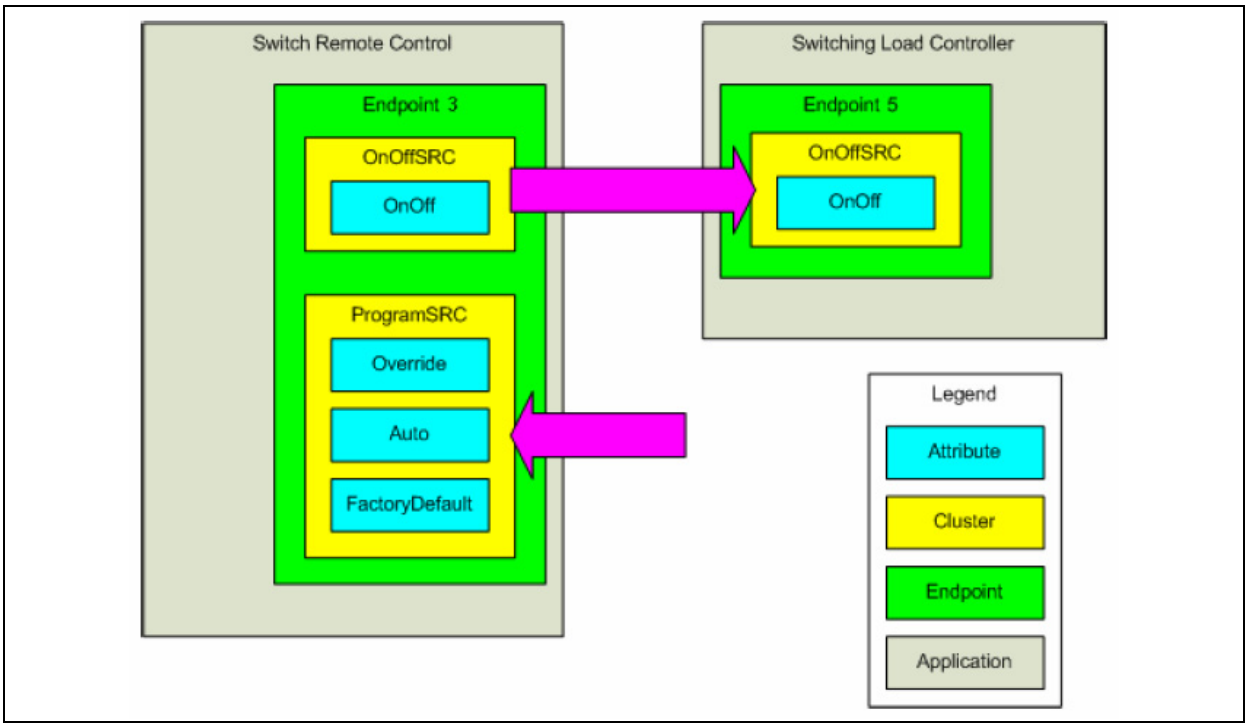


Fig. 24: Esquema aplicación-endpoint-cluster-atributo

Las aplicaciones "Switch Remote Control" y "Switching Load Controller" vendrían a ser las que corriesen respectivamente en los dispositivos #1 y #2, citados antes en el ejemplo. El atributo OnOff del dispositivo #2 (el que ejecuta "Switching Load Controller") sería el que guarda el estado de la bombilla #1, comunicado desde el control mediante el atributo del mismo nombre. En este esquema faltaría por indicar los endpoints restantes del ejemplo.

3.2.1.4 Formatos de trama y tipos de envío

Se definen dos tipos de formatos de trama, el formato KVP ("Key-Value Pair", o "par clave-valor") y el formato MSG ("Message"). Ambos formatos se asocian con un identificador de cluster, y la diferencia fundamental entre ellos es que mientras el formato MSG no impone restricciones a priori a la hora de hacer un envío, el formato KVP requiere de una estructura estricta ya que asocia la información a transmitir a un atributo concreto. El perfil de la aplicación se encargará de fijar

todas las restricciones y formatos pertinentes, teniendo en cuenta que un mismo cluster no admite ambos tipos a la vez.

Una trama KVP incluye la siguiente información en sus campos:

- Número de secuencia de la transacción.
- Tipo de comando y tipo de dato del atributo.
- Identificador del atributo.
- Código de error (opcional).
- Datos del atributo (de tamaño variable)

El tipo de comando indica lo que la aplicación supuestamente va a hacer con esa información. Por ejemplo, un comando tipo SET solicita que en el receptor se ponga el valor del atributo indicado en el campo de ID de atributo al que indica la trama en el campo de datos del atributo. Otra opción podría ser un comando tipo GET WITH ACKNOWLEDGE, que solicita al receptor que envíe el valor del atributo indicado por el campo ID de atributo.

Las tramas MSG simplemente incluyen estos campos:

- Número de secuencia de la transacción.
- Longitud de la transacción.
- Datos de la transacción.

A la hora de enviar una trama, además de tener que ser de uno de estos dos tipos, el envío requiere de un tipo de direccionamiento, y éste puede ser directo, indirecto o de difusión ("broadcast").

En el direccionamiento directo, se requiere de la dirección del destinatario además del endpoint al que va destinado el mensaje. Se asume que existe y funciona el dispositivo al que va destinado por la dirección de red y su endpoint, y toda esa información ha de conocerse de antemano.

En un direccionamiento indirecto, toda esa información del destinatario de la que se requería conocimiento previo en el caso anterior, se tiene almacenada en la Tabla

de Asociación del coordinador de red, y todo lo que tendría que hacer el emisor es hacer el envío vía-coordinador.

El direccionamiento de difusión o broadcast sirve para enviar mensajes a todos los endpoints de un determinado dispositivo destino. Esta forma de broadcast se llama de aplicación.

3.2.2 Arquitectura

La arquitectura ZigBee es una pila como cualquier otra arquitectura de red. Como tal está estructurada en varias capas, cada una de las cuales ofrece servicios de acceso a la capa superior. Estos servicios muestran interfaces de acceso a la capa superior a través de los llamados puntos de acceso de servicio o SAP ("Service Access Point"), los cuales soportan ciertas primitivas de servicio para llevar a cabo la funcionalidad requerida.

La arquitectura de pila ZigBee está basada en el modelo OSI ("Open Systems Interconnection"), el conocido estándar de arquitectura de red, pero a diferencia de éste, no define la totalidad de sus capas (son 7), sino que se especifican sólo aquellas que tendrían relevancia en el contexto de la funcionalidad que pretende ofrecer ZigBee. Recordemos que la definición de la arquitectura está realizada por dos entidades diferentes, por un lado el IEEE, que con su estándar 802.15.4-2006 provee de una especificación completa de lo que deberían ser las dos capas más bajas de la pila, y por otro lado la ZigBee Alliance, que completa la definición del resto de capas de la pila, exceptuando la última de ellas, que compete al usuario al desarrollar su aplicación.

A continuación indicamos un esquema de lo que es la arquitectura de pila ZigBee:

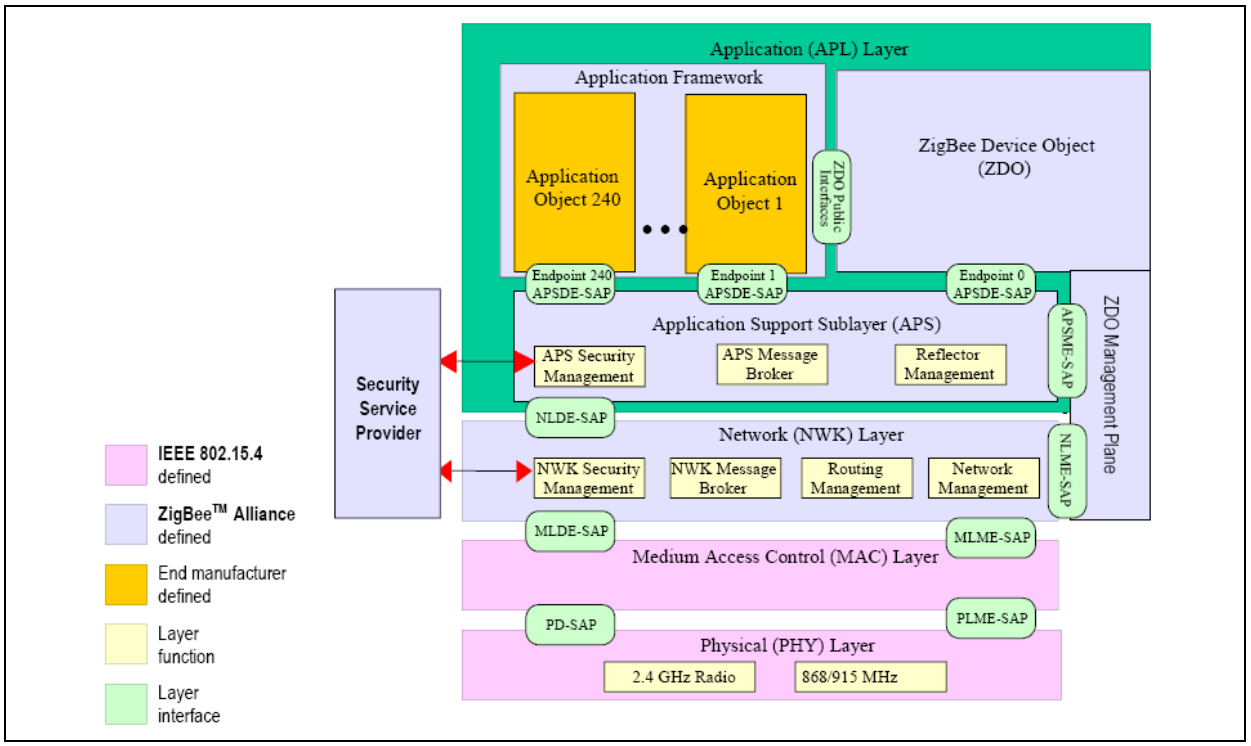


Fig. 25: Arquitectura en capas ZigBee

En dicho esquema se pueden distinguir tanto las capas que ha confeccionado cada organización, por colores. Dichas capas principales son PHY (capa física), MAC (capa de control de acceso al medio), NWK (capa de red) y APL (capa de aplicación).

Si se desea conocer de forma general las responsabilidades, servicios y formatos específicos que ofrece cada capa, en el Anexo A: se explican aspectos de cada una de ellas en particular.

3.2.3 Seguridad

No vamos a comentar mucho de esto en profundidad. Simplemente dejar constancia que la seguridad ha sido bien estudiada en la ZigBee Alliance, de cara a mandar datos cifrados por el medio aéreo que tenemos, y del que cualquiera con una radio en la frecuencia adecuada y el software apropiado podría hacer un esnifado de tráfico. Se ha dotado de capacidades de seguridad a las capas MAC, NWK y APS, y puede verse en la Fig. 26.

En el punto A.4.2 del Anexo se habla brevemente de un rol denominado Centro de Confianza (Trust Center), dentro del Gestor de Seguridad del ZDO, mediante el cual

se aseguran las obtenciones limpias de las claves de red, y otras auxiliares. Tan sólo mencionar que las claves son de 128 bits y hoy por hoy aseguran bastante confianza.

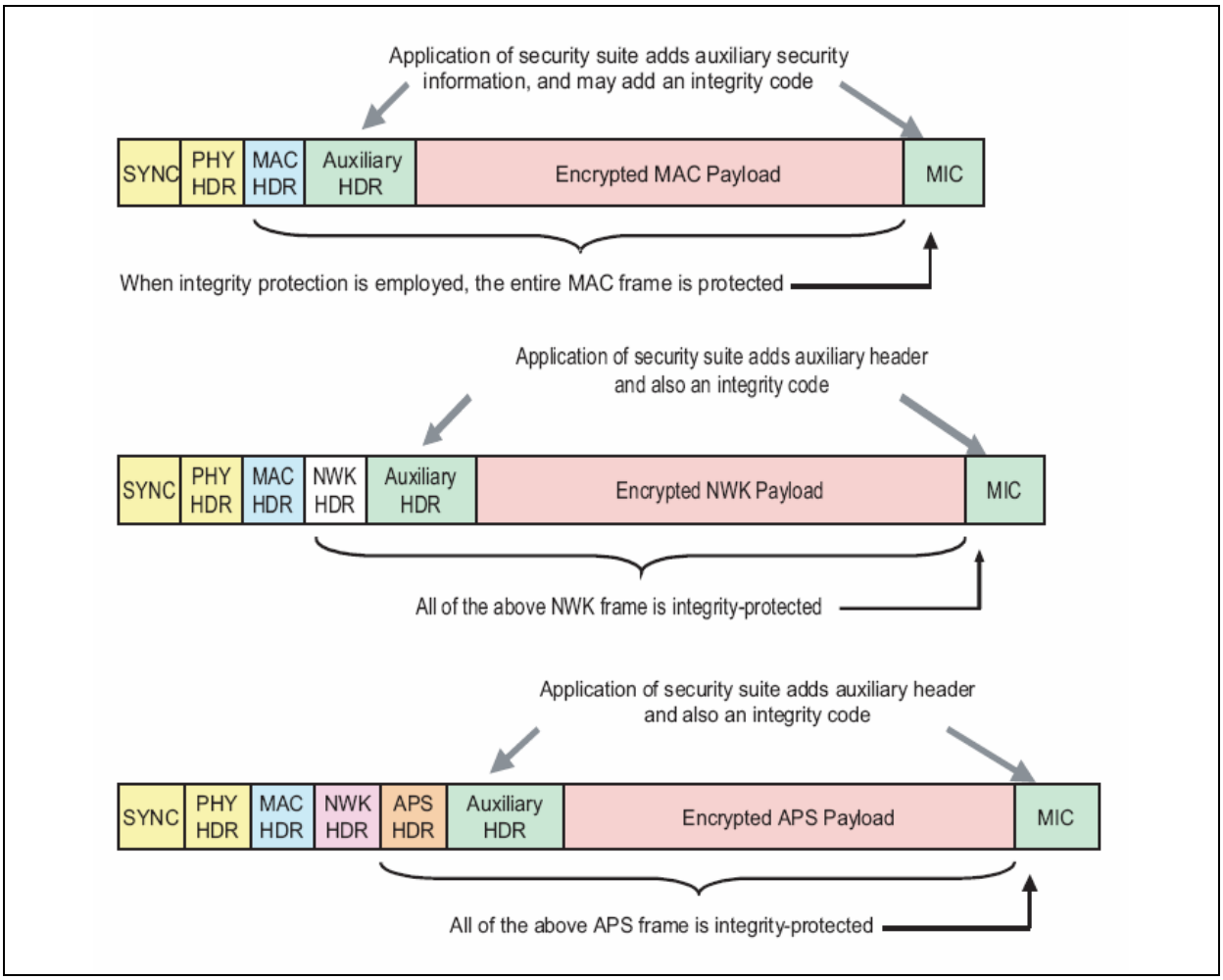


Fig. 26: Tramas MAC, NWK y APS dotadas de seguridad

No vamos a comentar nada más en materia de seguridad. Si se está interesado en profundizar en el tema, consúltese el punto 4 del documento [7] de la bibliografía.

3.2.4 Versión del stack utilizada (particularidades respecto al estándar)

La implementación del stack utilizada ha sido la versión 1.0-2.3 de Microchip, gratuita y disponible para su descarga en el website de microchip. Dicha versión data de abril de 2005 y aún no cumplía todos los aspectos que se mencionan tanto en el estándar IEEE 802.15.4 o en la Especificación ZigBee de la ZigBee Alliance (documentos [8] y [6] o [7] de la bibliografía).

La versión ofrece las siguientes características:

- Basada en la especificación 1.0 de ZigBee.
- Soporte automático para el compilador MPLAB C18.
- Portabilidad entre todas las familias de controladores PIC18.
- Pensada para la placa de desarrollo PICDEM Z.
- Soporte del transceptor Chipcon CC2420.
- Soporte de topología en estrella no ranurada.
- Implementa nodos Coordinador y RFD.
- Soporte por defecto para tablas de vecinos y de asociación no-volátiles.
- Soporte de hasta 254 dispositivos finales en una red,
- Soporte de hasta 254 entradas de asociación por red o nodo.
- Soporte de asociaciones de uno-a-muchos.
- Soporte para asociaciones de dispositivos finales.
- Provee de los objetos de descubrimiento de dispositivo y servicio obligatorios.
- Soporte de paquetes de broadcast.
- Soporte de endpoint de broadcast.

Limitaciones que se tienen respecto a la especificación:

- No hay soporte para redes cluster, P2P o de malla.
- No hay capacidades de seguridad o control de acceso.
- No hay soporte para el descriptor complejo y el de usuario (opcionales en la especificación).

3.3 PICDEM Z (kit de desarrollo)

El llamado PICDEM Z de Microchip ha sido el kit de desarrollo con el que hemos trabajado para llevar a cabo la emulación física y real en hardware del comportamiento de los mandos y el dispositivo receptor. Su utilidad genérica es la de poder desarrollar aplicaciones Microchip para el protocolo ZigBee.

La elección de este kit de desarrollo se ha basado fundamentalmente en el microcontrolador PIC que incluía, por sus válidas características en este proyecto y las posibilidades de portar el código a otros productos OEM con micros PIC, de fácil manera y poco coste económico.

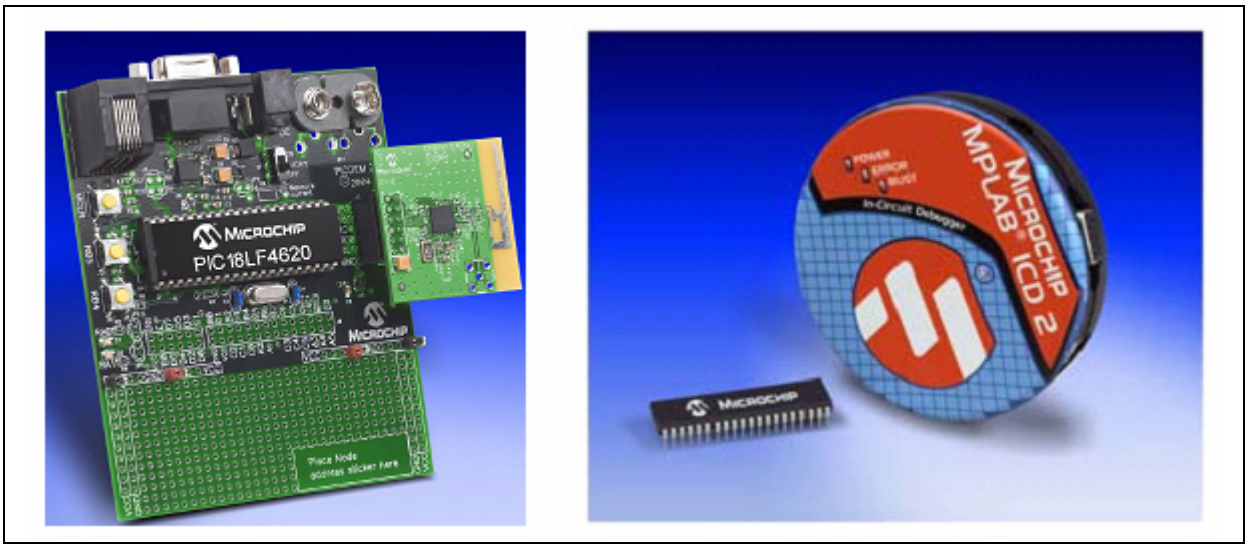


Fig. 27: Nodo ZigBee del PICDEM Z, y módulo ICD2

El kit lo forman un par de placas base que llevan el microcontrolador y los diferentes interfaces a los periféricos, y un par de placas de radiofrecuencia con el transceptor. Completan el kit: dos pilas de 9 V y un CDROM con manuales, ejemplos, y la pila de red ZigBee.

Aunque no forme parte del kit original, podemos completar éste con el software para PC MPLAB y el módulo MPLAB ICD2 de Microchip, que respectivamente son un IDE (Entorno Integrado de Desarrollo) para programación del firmware, y un depurador "on-circuit", mediante el que se pueden cargar y depurar los programas en el propio micro a través de un interfaz JTAG de acceso.

Vamos a comentar cómo están formadas las placas.

3.3.1 Placa base

La placa base es el elemento principal del kit, y está formada por un microcontrolador PIC18LF4620 y sus periféricos "on-board" e interfaces para dispositivos externos. El esquema de la placa a nivel físico de superficie superior sería el siguiente:

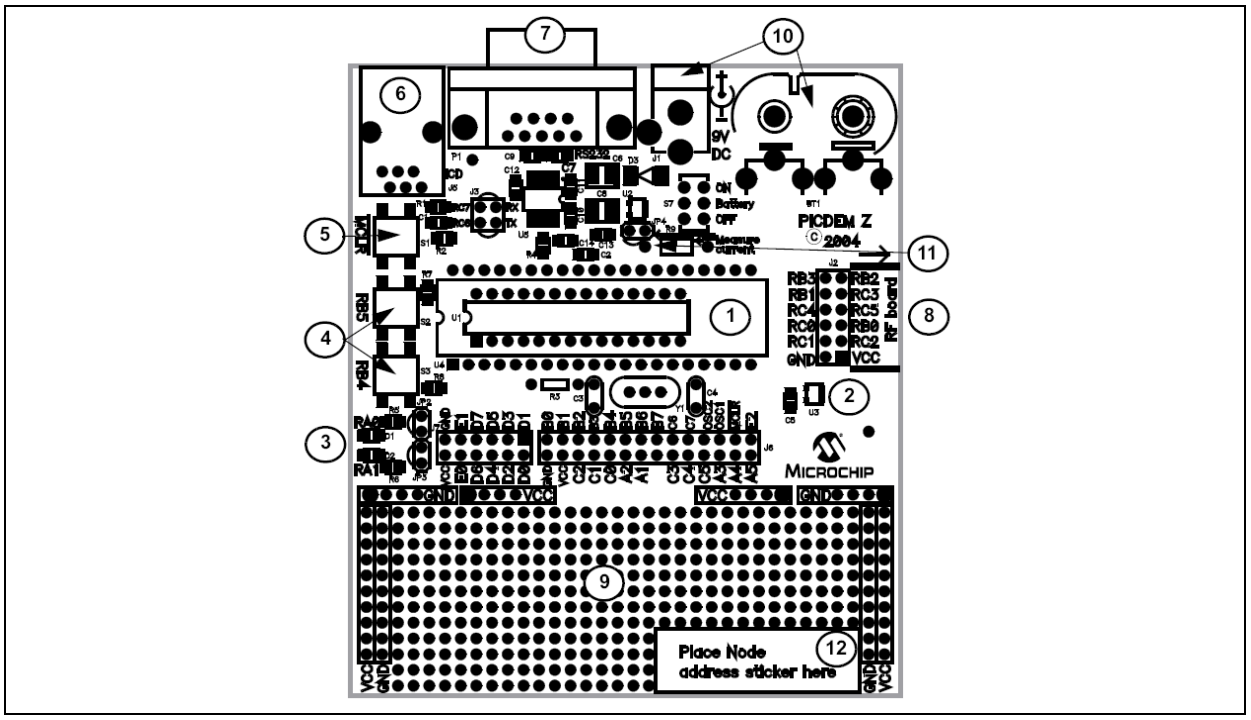


Fig. 28: Esquema placa base de PICDEM Z

Las partes destacables son las que indican los números de la figura:

1. Socket del microcontrolador.
2. Sensor de temperatura (TC77).
3. LEDs de usuario.
4. Botones de usuario.
5. Botón de reset, conectado al pin Master Clear (MCLR) del PIC, para resetear la placa.
6. Conector RJ-11 (6 hilos), para conectar el ICD2.

7. Conector RS-232 (DB9F), para conectar a otra placa o a un PC a través del puerto serie.

8. Conector de la tarjeta de radiofrecuencia.

9. Área de prototipado, para añadir circuitería extra para el desarrollo del prototipo.

10. Fuente de alimentación de la placa, tanto por pila de 9 V como de alimentación externa desde un alimentador conectado a la red eléctrica.

11. Jumper para medida de la corriente.

12. Pegatina con la ID del nodo.

13. Indicador de nivel de revisión del hardware (por detrás).

3.3.2 Placa de radiofrecuencia

Esta placa, que es la que lleva el transceptor de radio y la antena como principales elementos, puede ser variable según el PICDEM Z ya que admite diferentes tipos de transceptores. La placa que incluye nuestro PICDEM Z se llama "PICDEM Z 2.4 GHz RF Card" está fabricada por Microchip e incorpora el transceptor CC2420 de Chipcon, que opera en la banda de 2.4 GHz.

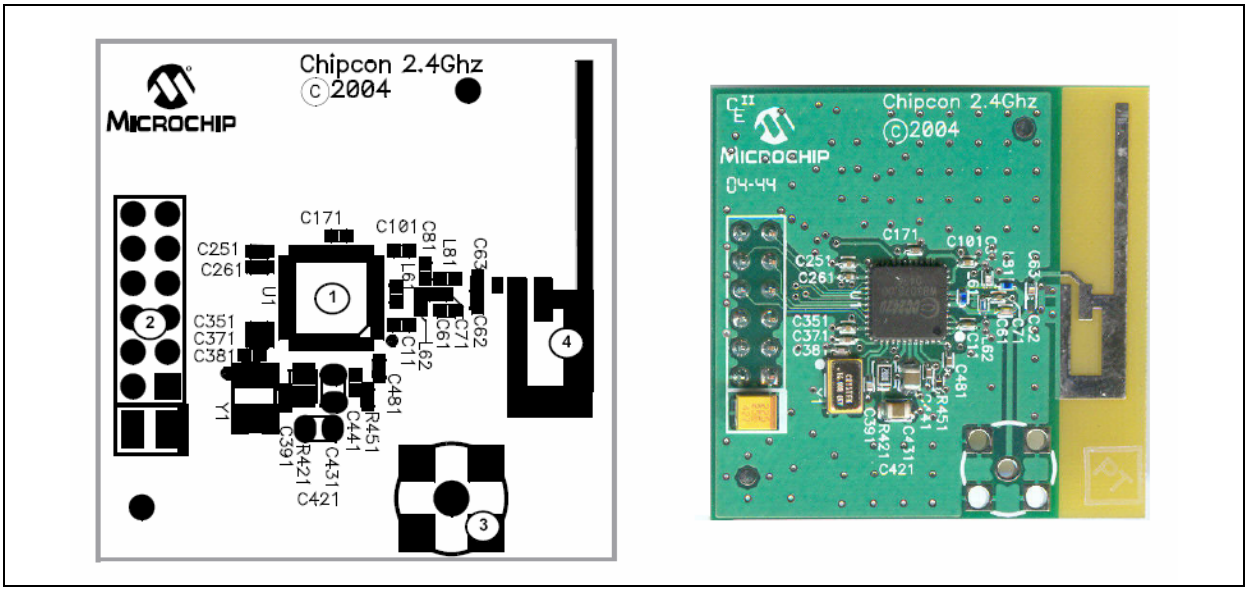


Fig. 29: Esquema y fotografía de la placa PICDEM Z 2.4 GHz RF Card

En la figura se pueden apreciar los cuatro elementos más importantes de esta tarjeta:

1. Transceptor de radiofrecuencia, que corresponde al modelo CC2420 de Chipcon en formato QLP de 48 pines, e implementa las capas PHY y parte de MAC del estándar 802.15.4 del IEEE (ZigBee).

2. Conector de la tarjeta, para enchufarla a la placa base.

3. Conector opcional SMA de cable coaxial, para usar una antena externa.

4. Antena PCB ("Printed Circuit Board"), impresa en la propia tarjeta.

5. Indicador de nivel de revisión del hardware (por detrás).

3.3.3 Microchip MPLAB IDE

Es el entorno de desarrollo integrado que hemos usado para escribir, compilar y depurar nuestro firmware, así como para programarlo en los microcontroladores a través del ICD2.

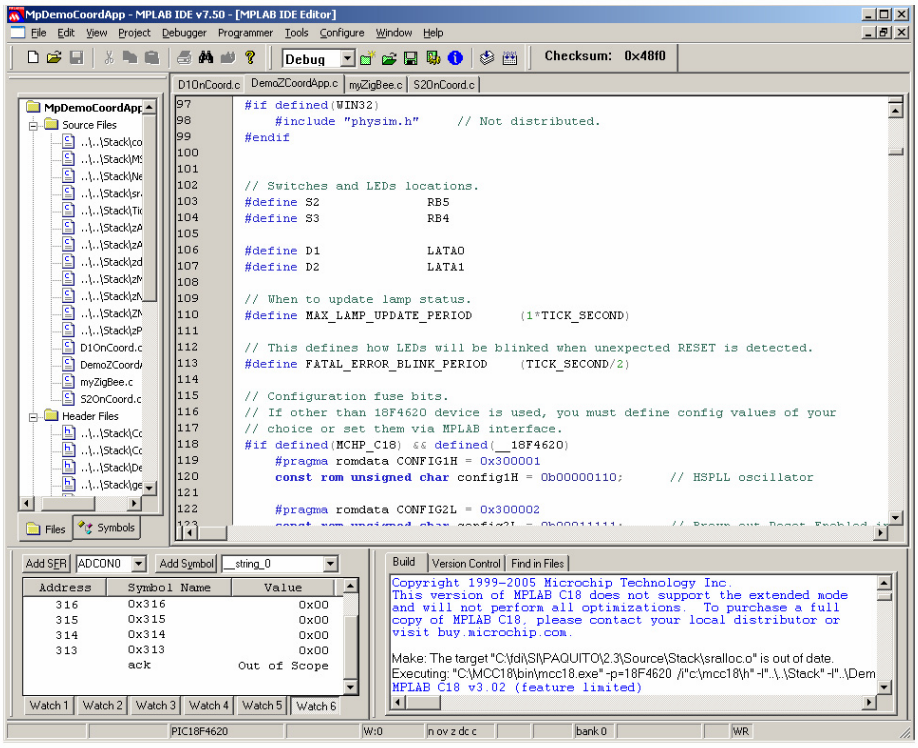


Fig. 30: Aspecto del MPLAB

El MPLAB puede seleccionar diferentes entornos de compilación, nosotros hemos optado por el compilador C18 de Microchip en su versión limitada de estudiante. Poco más se puede contar en este apartado, su uso es bastante intuitivo y en caso de querer profundizar en las funcionalidades que ofrece, se remite al lector a consultar la documentación [19] y [20] incluidas en la bibliografía de esta memoria.

3.3.4 Microchip MPLAB ICD2

El ICD2 es un programador/depurador de circuitos ya sobre el circuito físico real. El esquema de uso del ICD2 con un PC que hace de host y el propio circuito es el siguiente:

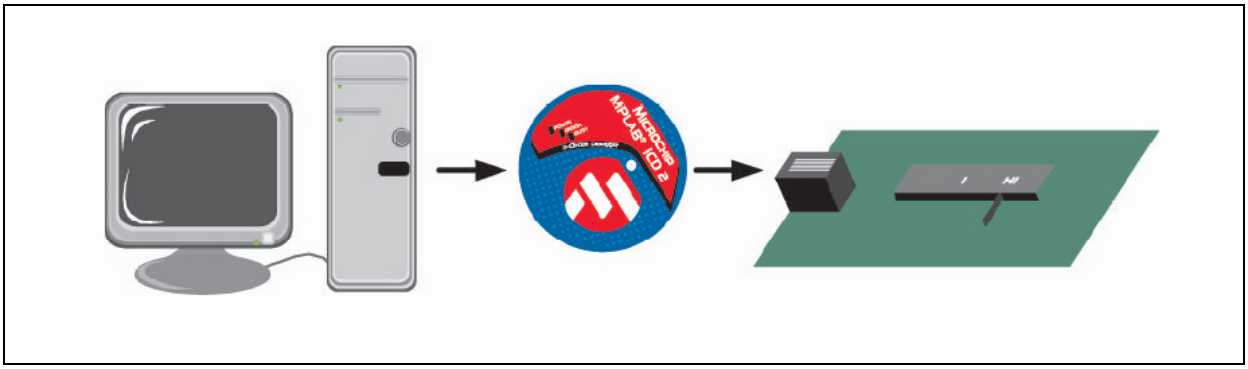


Fig. 31: Esquema de uso del ICD2

Así pues, el ICD2 se conecta al PC host mediante interfaz USB o RS-232, y al circuito con un interfaz serie de 6 pines RJ-12.

Entre las características que ofrece para depurar se pueden citar las siguientes añadidas a lo que ya se ha comentado:

- Ejecución de código en tiempo real y paso a paso,
- Uso de breakpoints, watches y modificación de variables y registros en tiempo de ejecución,
- Depuración sobre el circuito,
- Monitor del voltaje fuente del circuito, y
- LEDs de diagnóstico.

Al igual que en el punto anterior, remitimos al lector a la documentación [21] ofrecida en la bibliografía, si es que desea ampliar información sobre este producto.

Fig. 32: Vista del interior del módulo ICD2

4 DESARROLLO DEL PROTOTIPO

4.1 Software PC: TeleTest ZB

4.1.1 Motivación

Lo que se pretende obtener es una aplicación de sencillo manejo que permita gestionar los datos de las respuestas dadas por los alumnos de un modo "económico"; es decir, sin emplear grandes recursos. La idea consiste en lo siguiente: ejecutar una pequeña aplicación corriendo en background a modo de servidor que sea capaz de tomar el control e iniciar o parar la recepción de los mandos en cualquier momento, liberando en este caso los recursos del sistema y permaneciendo en background mientras la CPU se encarga de otras labores.

Para la gestión de dichos datos inicialmente se contemplaron varias posibilidades, tales como: emplear ficheros de texto en los que se incluyeran los resultados obtenidos, emplear algún gestor de base de datos, e incluso desarrollar un sistema autónomo de gestión.

Sin embargo, tras analizar el uso real que se le daría al sistema, se observó que lo más adecuado para continuar con la filosofía de "economizar" la aplicación sería optar por el uso, de modo subyacente, de una herramienta con potencial propio, y estandarizada en el contexto de utilización del CRS. Esta herramienta, sin ninguna duda, es Microsoft Excel. La familiaridad que tiene el usuario medio con una herramienta como Microsoft Excel facilita enormemente el uso de TeleTestZB ya que la simple adición y eliminación de datos o incluso recursos más "complejos", como conteos o elaboración de gráficas, se realizan como es costumbre en dicha herramienta.

A pesar de parecer ser todo ventajas, el uso de Microsoft Excel obligaba a que el archivo ofrecido por el usuario siguiera un determinado formato para que pudiera interactuar con la aplicación TeleTestZB. Esto se resolvió inicialmente con el uso de plantillas pero no parecía una solución lo suficientemente elaborada; por tanto, se decidió dotar a la aplicación de la capacidad de generar tales plantillas

automáticamente, de modo que el usuario sólo tuviera que rellenar tal plantilla con los datos de los usuarios (alumnos en este caso).

De este modo, el usuario dispone de varias posibilidades a la hora de generar un archivo válido para TeleTestZB.

En primer lugar tendrá la posibilidad de *crear* un archivo nuevo. Si el usuario elige esta opción, el sistema solicitará al usuario que introduzca el número de preguntas que quiere en el fichero (aunque más tarde se podrán añadir más), abrirá una hoja de Microsoft Excel en la que aparecerán las cabeceras de las columnas ya definidas y, sencillamente, tendrá que rellenar tales columnas con los datos correspondientes.

Debido a que este proceso podría resultar en ocasiones tedioso y con el objetivo de facilitar esta tarea al usuario, se optó por la incorporación de la posibilidad de *importar* archivos obtenidos como CSV's (Comma-Separated Values) del Campus Virtual, y ofreciendo de nuevo al usuario la posibilidad de determinar el número de preguntas que desea incluir en el archivo.

Además de estas opciones, el sistema también posee otras funcionalidades como *añadir* preguntas a un archivo ya creado, lo cual modifica el archivo indicado añadiéndole un número determinado de preguntas y de un modo totalmente transparente para el usuario, o *editar* el archivo, lo que conlleva abrir el archivo indicado en Microsoft Excel para su edición.

El uso de Microsoft Excel también facilita, en este caso al desarrollador, la elaboración de las gráficas de resultados.

4.1.2 Desarrollo

Para el desarrollo de este sistema se ha optado por emplear un lenguaje de programación potente, actual y nunca utilizado anteriormente por los autores. Dicho lenguaje es C#, incluido en el entorno de Microsoft Visual Studio .NET. La versión utilizada ha sido la correspondiente al año 2005.

Al tratarse de un entorno desconocido para los autores, los comienzos resultaron arduos y los avances se producían con bastante lentitud. Una vez tomada la medida con dicho entorno se descubrió el verdadero potencial del Framework de .NET y las

múltiples posibilidades que ofrece al desarrollador bajo el sistema operativo Microsoft Windows.

Una de las posibilidades que han resultado más beneficiosas de cara al desarrollo del sistema ha sido la interoperabilidad que ofrece el Framework .NET con las distintas aplicaciones de Microsoft Office.

4.1.2.1 Diagrama de clases

El diagrama de clases se corresponde con el siguiente:

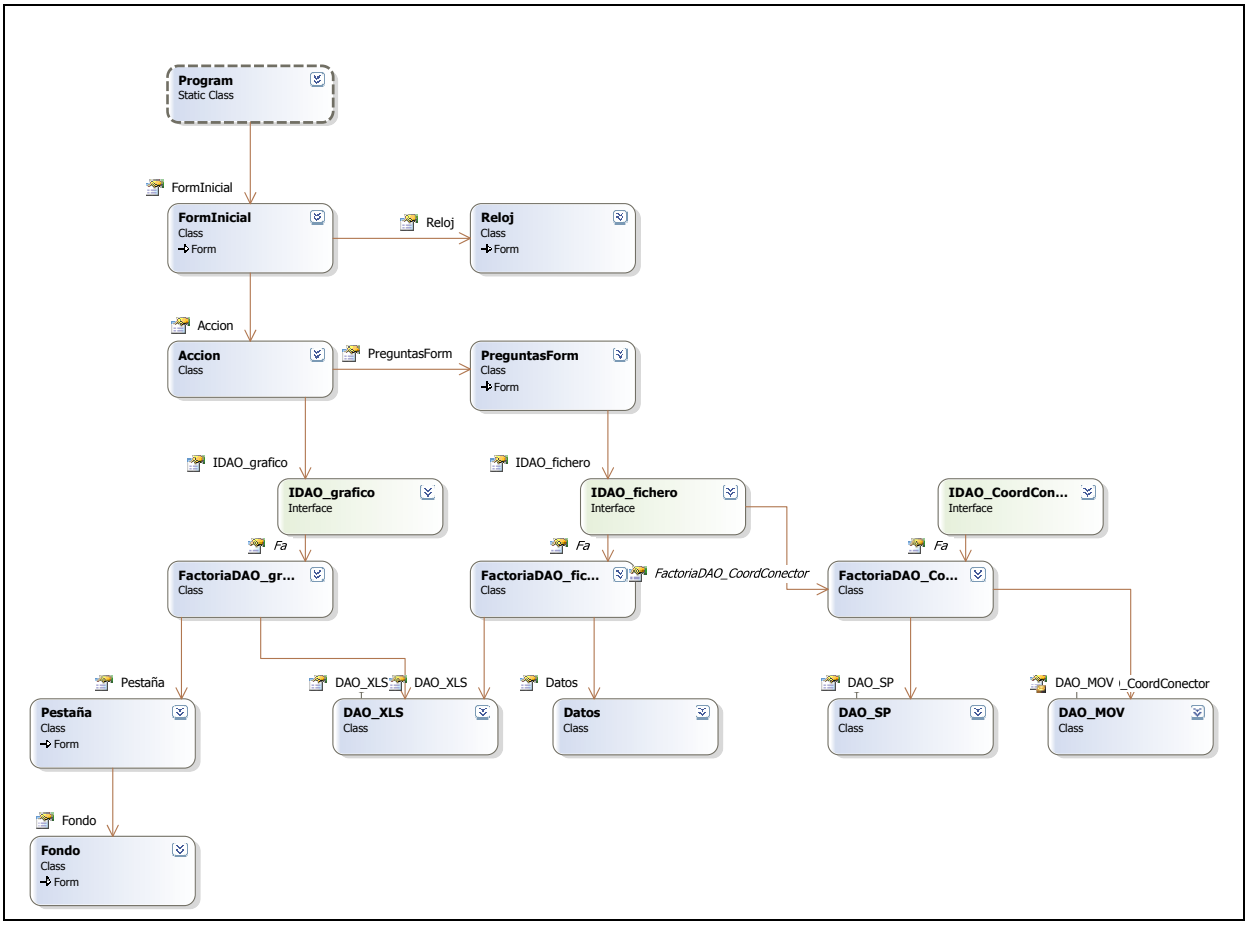
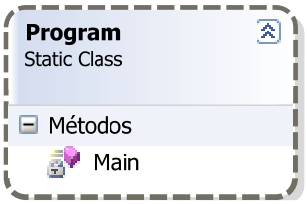


Fig. 33: Diagrama de clases TeleTestZB

Se ha pretendido obtener una estructura básica y bien delimitada, fácilmente sostenible y con una elevada capacidad de ampliación ante posibles requerimientos futuros mediante el uso de interfaces. Se le ha dado una gran importancia a la encapsulación y se han utilizado ciertos patrones de diseño como son el patrón factoría y el patrón de acceso a datos.

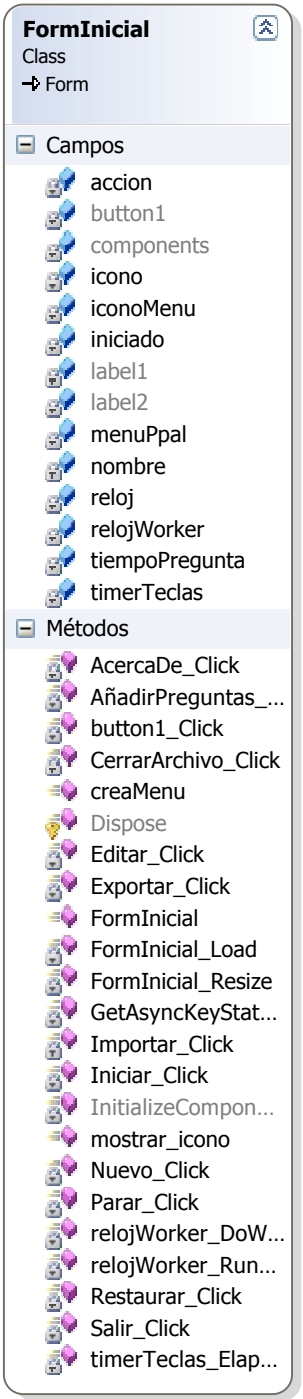
4.1.2.2 Detalle de clases

Clase *Program*



La clase estática *Program* es la clase principal de la aplicación y es autogenerada por Microsoft Visual Studio .NET. Esta clase llamará a la clase *FomInicial*. En P*rogram* se ha incluido un mutex para no permitir que haya en el sistema varias instancias de la aplicación TeleTestZB ejecutando a la vez. Esta clase no posee atributos.

Clase *FormInicial*

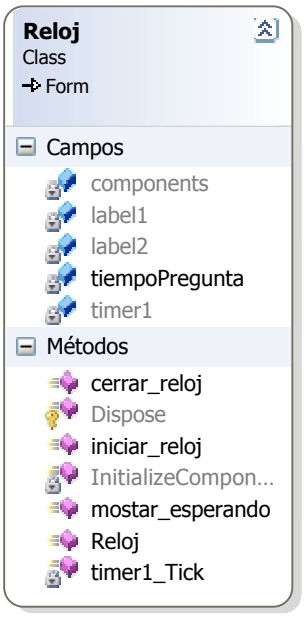


Esta clase es la que contiene el formulario inicial del proyecto. Esta clase se encarga de capturar los eventos lanzados por el usuario sobre el formulario.

En esta clase se define el menú principal de usuario. A dicho menú pertenecerán los métodos: *AcercaDe_Click*, *AñadirPreguntas_Click*, *CerrarArchivo_Click*, *Editar_Click*, *Exportar_Click*, *Importar_Click*, *Nuevo_Click*, etc.

Otra de las peculiaridades de esta clase es que en ocasiones lanza un nuevo hilo llamado, *relojWorker_DoWork*; que permite visualizar el reloj de tiempo para contestar, como un cuadro de diálogo transparente, sin producir merma en el sistema ni bloquearlo.

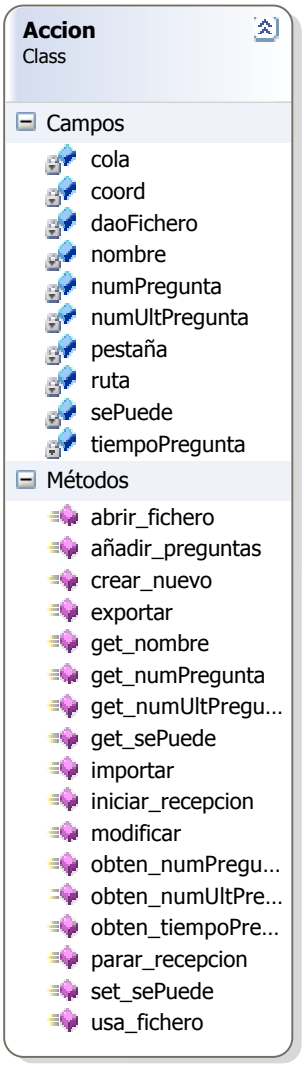
Esta clase también es la encargada de detectar cuándo se ha pulsado la combinación de teclas que inicia y para la recepción y de mantener la aplicación en modo *icono de notificación* en la barra de tareas de Microsoft Windows.



Clase *Reloj*

Esta clase se compone de un formulario transparente que mostrará el tiempo restante para contestar una pregunta determinada.

Esta clase es llamada desde *FormInicial* y se ejecuta en un hilo independiente.



Clase *Acción*

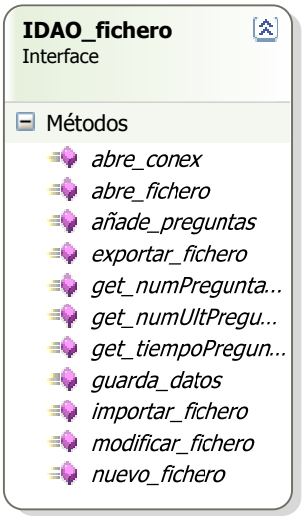
Esta clase es la que realiza la conexión entre la parte gráfica y el resto de la aplicación. Es la encargada de lanzar todo el trabajo una vez que se ha capturado el evento del usuario.

Los métodos de: *abrir_fichero*, *añadir_preguntas*, *crear_nuevo*, *exportar*, *obten_numPregunta*, *obten_numUltPregunta*, *obten_tiempoPregunta* e *importar* se enlazarán a través de la interfaz *IDAO_fichero* con la clase encargada de gestionar el fichero de resultados.

Iniciar_recepcion y *parar_recepcion* se enlazarán con la clase encargada de realizar la conexión con el puerto por medio de la interfaz *IDAO_CoordConector*.

Parar_recepcion se enlazará con la clase encargada de generar el gráfico a visualizar a través de la interfaz *IDAO_grafico*.

Interfaz *IDAO_fichero*



El uso de interfaces a este nivel del diseño otorga una elevada capacidad de ampliación al sistema ya que, en caso de requerir en el futuro que la aplicación funcione con otro tipo de ficheros de datos, sólo habrá que elaborar una clase que implemente estos métodos sobre dicho tipo de ficheros.

Esta interfaz será implementada por la clase *DAO_XLS*.

Interfaz *IDAO_CoordConector*

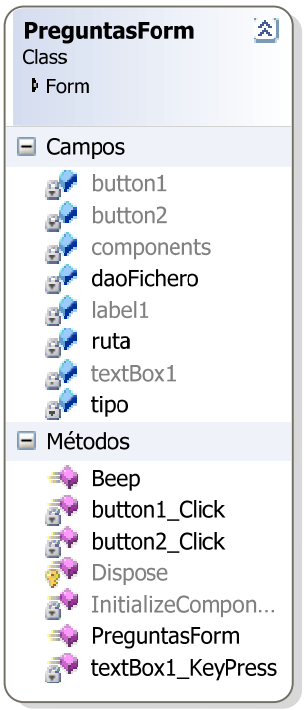
La interfaz *IDAO_CoordConector* es la encargada de definir los métodos necesarios para que se lleve a cabo correctamente la comunicación entre el puerto y la aplicación.

Esta interfaz será implementada por la clase *DAO_SP*.

Interfaz *IDAO_grafico*

La clase que implemente esta interfaz será la encargada de generar el gráfico que se mostrará al parar la recepción. Dicha clase será la clase *DAO_XLS*.

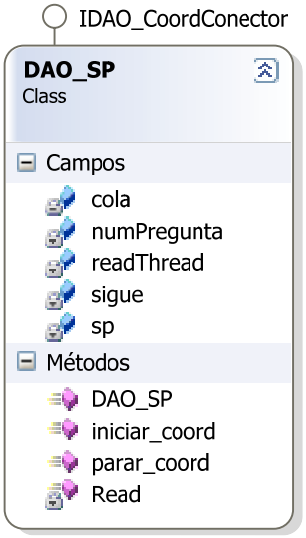
Clase *PreguntasForm*



Esta clase se compone de un formulario con el formato de un cuadro de diálogo que pregunta al usuario el número de preguntas que quiere añadir en un nuevo archivo, en un archivo importado o en un archivo ya existente.

Este formulario emite un *beep* si se pretende introducir letras en vez de números.

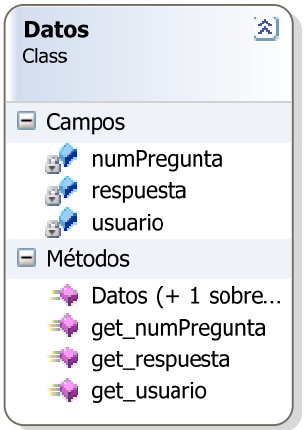
Clase *DAO_SP*



Esta clase implementa la interfaz *IDAO_CoordConector*.

Es la encargada de realizar la conexión con el puerto serie. En esta clase se implementa un nuevo hilo de ejecución para permanecer a la escucha del puerto sin merma en el rendimiento.

Clase *Datos*



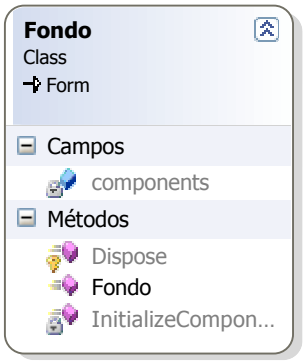
Esta clase define el objeto que se encola en la recepción para después ser volcados en el archivo.

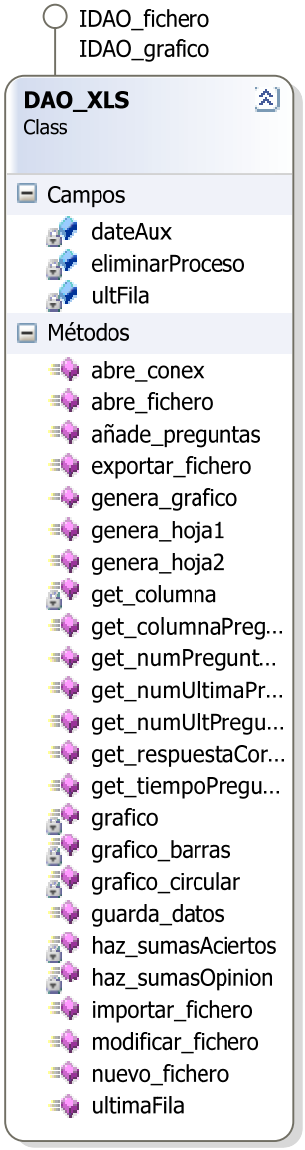
Clases *FactoriaDAO_xxx*

Las clases *FactoriaDAO_xxx* se encargan de generar una instancia de cada una de las interfaces explicadas anteriormente. *FactoriaDAO_grafico* y *FactoriaDAO_fichero* generarán sendas instancias de DAO_XLS mientras que *FactoriaDAO_CoordConector* generará una instancia de *DAO_SP*.

Clase *Fondo*

Esta clase se limita a mostrar un formulario oscurecido que se sitúa detrás de la gráfica mostrada.





Clase *DAO_XLS*

Ésta es la clase donde se encuentran todas las operaciones que interactúan con Microsoft Excel. Es la más extensa. Implementa las interfaces *IDAO_fichero* e *IDAO_grafico*.

Los métodos *genera_hoja1* y *genera_hoja2* son los encargados de generar las hojas de Microsoft Excel donde el usuario deberá introducir sus datos.

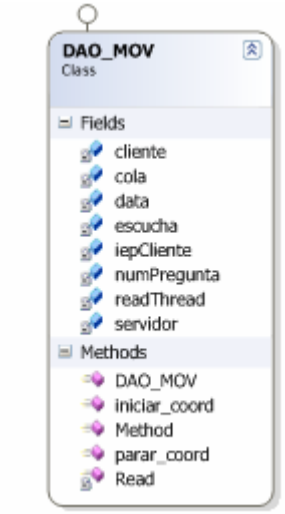
El método *añade_preguntas* añade un número determinado de preguntas que recibe por parámetro.

El método *genera_grafico* se encarga de elaborar la gráfica correspondiente y de almacenarla en un archivo en el directorio en que se encuentre la aplicación con el nombre *imagen.jpg*. Este archivo se irá sobrescribiendo cada vez que se genere una nueva gráfica. Para elaborar este gráfico el método llama a otro archivo, situado también en el directorio de la aplicación, llamado *fondo.bmp* que corresponde con el fondo que aparecerá en la gráfica.

Para generar las gráficas se requiere la implementación de algunos métodos auxiliares como: *haz_sumasAciertos* y *haz_sumasOpinion*.

En el método *guarda_datos* se establece una conexión mediante ADO .NET con Microsoft Excel para llevar a cabo el volcado de los datos anteriormente encolados, en nuestro archivo. Para llegar a tal fin, y una vez se ha establecido la conexión, se lanza una query que actualiza la respuesta del usuario correspondiente.

Clase *DAO_MOV*



Esta clase es la encargada de recibir todas las respuestas llegadas de los teléfonos móviles de los usuarios.

Al igual que *DAO_SP*, esta clase también implementa la interfaz *IDAO_CoordConector* y se compone de los métodos: *iniciar_coord*, *parar_coord*, *Read*.

El método *iniciar_coord* es el encargado de iniciar el hilo de lectura que leerá las respuestas de los móviles.

El método *parar_coord* detiene el hilo de lectura que posibilita que no se siga leyendo más respuestas.

El método *Read* es el encargado de leer todas las respuestas que le llegan de los móviles, así como de introducir en la estructura de datos "cola" dichas respuestas para su posterior volcado en el fichero de Excel.

4.2 Firmware PICs

En este apartado se pretende comentar qué aplicaciones se han facilitado para los PICs, así como la organización el código en cada caso. Así pues, se distinguirán dos subapartados fundamentales, correspondientes a las aplicaciones DemoZCoordApp y DemoZRFDApp, respectivos del dispositivo Coordinador de red (receptor de respuestas) y los dispositivos RFD (mandos emisores). En ambos casos se ha partido del código de ejemplo dado en el PICDEM Z y con disponibilidad de descargar desde el website de Microchip (www.microchip.com), ya que constituía un buen esqueleto de aplicación para los intereses que nos ocupaban.

4.2.1 DemoZCoordApp

Es el firmware que se carga en el dispositivo Coordinador. Básicamente es una máquina de tan sólo cuatro estados cuyo cometido es iniciar la red, y quedar esperando envíos de los mandos.

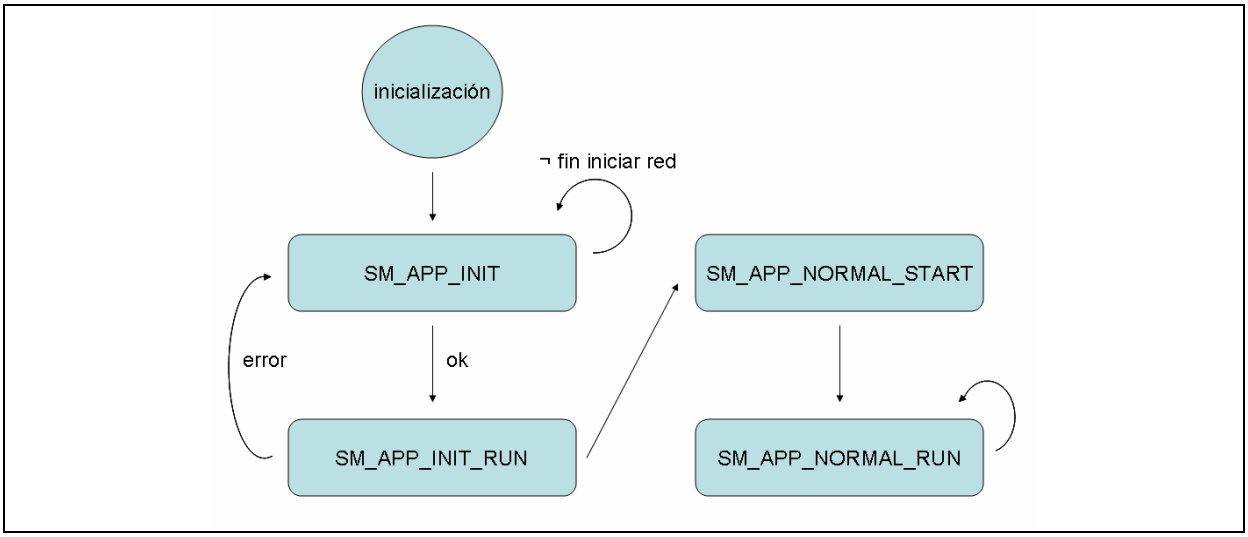


Fig. 34: Diagrama de estados DemoZCoordApp

Vamos a comentar de modo más o menos algorítmico las acciones que tienen lugar en cada etapa representada en el diagrama. Hay que comentar que las transiciones que aparecen sin texto son incondicionales.

La inicialización básicamente consiste en las siguientes acciones:

1. Habilitar el WDT (WatchDog Timer).

2. Limpiar los flags de control de la aplicación (control de flujo según botón pulsado), y los flags destinados a hacer binding personalizado.

3. Llamar a una función de inicialización de la placa *(InitializeBoard)*, cuyas tareas son:

    3.1. Inicializar la consola RS-232,

    3.2. Preparar las interrupciones por pulsación de botón (interrumpir al cambio),

    3.3. Habilitar resistencias “pull-up” (de polarización) internas en el PORTB,

    3.4. Preparar interrupciones por llegada de trama al transceptor de radio,

    3.5. Preparar sentidos del flujo de datos en PORTB como entradas,

    3.6. Indicar estado inicial del transceptor mediante seteo del latch de PORTC al que va conectado (deshabilitado y no seleccionado),

    3.7. Preparar sentidos del flujo de datos entre módulo SPI (MSSP, puerto serie) del micro y el exterior (el módulo RS-232 y el conector de la tarjeta del transceptor), a través de PORTC, así como setear su estado inicial,

    3.8. Poner PORTA como E/S digital,

    3.9. No seleccionar el sensor térmico TC77 deshabilitando su chip-select activo en baja (LATA[2] = 1),

    3.10. Habilitar como salidas los pines de PORTA conectados a los LEDs, y

    3.11. Habilitar globalmente las interrupciones.

4. Establecer el Timer0 como contador de ticks.

5. Encender los LEDs.

6. Si hay un error en este punto, deshabilitar WDT e interrupciones, y comenzar a hacer parpadear alternativamente los LEDs hasta resetear la placa, si no continuar.

7. Tanto si está pulsado el botón S3 como si no tengo dirección MAC asignada, poner el flag de de control para entrar al modo de configuración, a través de un menú por consola. En dicho menú se pueden encontrar opciones como "indicar la ID del dispositivo" (que completa los últimos 2 bytes de la dirección MAC), y otras propias de un Coordinador, como pueden ser "borrar la tabla de vecinos" o "borrar la tabla de asociaciones (bindings)".

```
*****************************************************************************
ZigBee Demo Coordinator Application v1.1 (Microchip Stack for ZigBee v1.1.0)
     Built on Jun 13 2007 (Non-debug Build)
*****************************************************************************
     1: Set node ID...
     2: Clear Neighbor Table.
     3: Clear Binding Table.
     4: Change to next channel.
     5: Transmit unmodulated signal.
     6: Transmit random modulated signal.
     0: Run application.

Enter a menu choice:
```

Fig. 35: Menú de configuración del Coordinador

8. Una vez configurado el dispositivo (al menos, dirección MAC asignada), se inicializa y habilita APL, comienza la inicialización de una red, se coloca el estado inicial del autómata y se apagan los LEDs, indicando que estamos ya ejecutando normalmente el bucle principal de la aplicación.

Una vez en este punto, entramos en el bucle principal de la aplicación. En cada iteración lo que hace es cambiar el estado de un LED (para que parpadee), deshabilitar el WDT y llamar a la función *APLTask* que es necesario llamar antes de realizar cualquier tarea, ya que llama separadamente y por orden a las funciones de todas las capas inferiores para ver si hay algún paquete entrante. A continuación, según el estado del autómata se discrimina por una rama u otra.

- Estado SM_APP_INIT: Simplemente muestra un mensaje de inicio de nueva red y pasa al siguiente estado (SM_APP_INIT_RUN).

- Estado SM_APP_INIT_RUN: Sigue en este estado mientras no termine de iniciar la red. Si hubiera algún error, lo reporta y vuelve al estado anterior (SM_APP_INIT), si no, establece nueva red con PAN ID propia y se pasa al siguiente estado (SM_APP_NORMAL_START).

- Estado SM_APP_NORMAL_START: Permite la asociación de nuevos dispositivos, abre los endpoints de S2 y D1, y pasa al siguiente estado (SM_APP_NORMAL_RUN).

- Estado SM_APP_NORMAL_RUN: Es ya el estado normal de funcionamiento. Simplemente queda en espera de recibir datos de los dispositivos RFD (mandos), de tal forma que al detectar una trama recibida, reenvía el mensaje por consola a través del puerto serie. En este estado queda ya para siempre en cada iteración del bucle principal.

El código de aplicación, escrito en C18, se reparte fundamentalmente en 3 ficheros de fuente sin contar sus ficheros de cabecera:

- DemoZCoordApp.c: Fichero principal, con la función *main* y las funciones de inicialización y configuración.

- S2OnCoord.c: Contiene la función *S2Task.*

- D1OnCoord.c: Contiene la función *D1Task.*

Además de ésos, se tiene un fichero de configuración (zigbee.def), dónde se definen varias de las constantes de precompilador relativas al ZigBee. Y aparte, todo el stack ZigBee y funciones de depuración y manejo de consola, así como las librerías que ofrece el C18 (manejo de memoria dinámica, strings,…)

4.2.2 DemoZRFDApp

Ésta es la aplicación firmware que corre en los dispositivos RFD. Es muy similar a la que corre en el Coordinador, pero cuenta con más estados en su autómata, que tiene esta pinta:

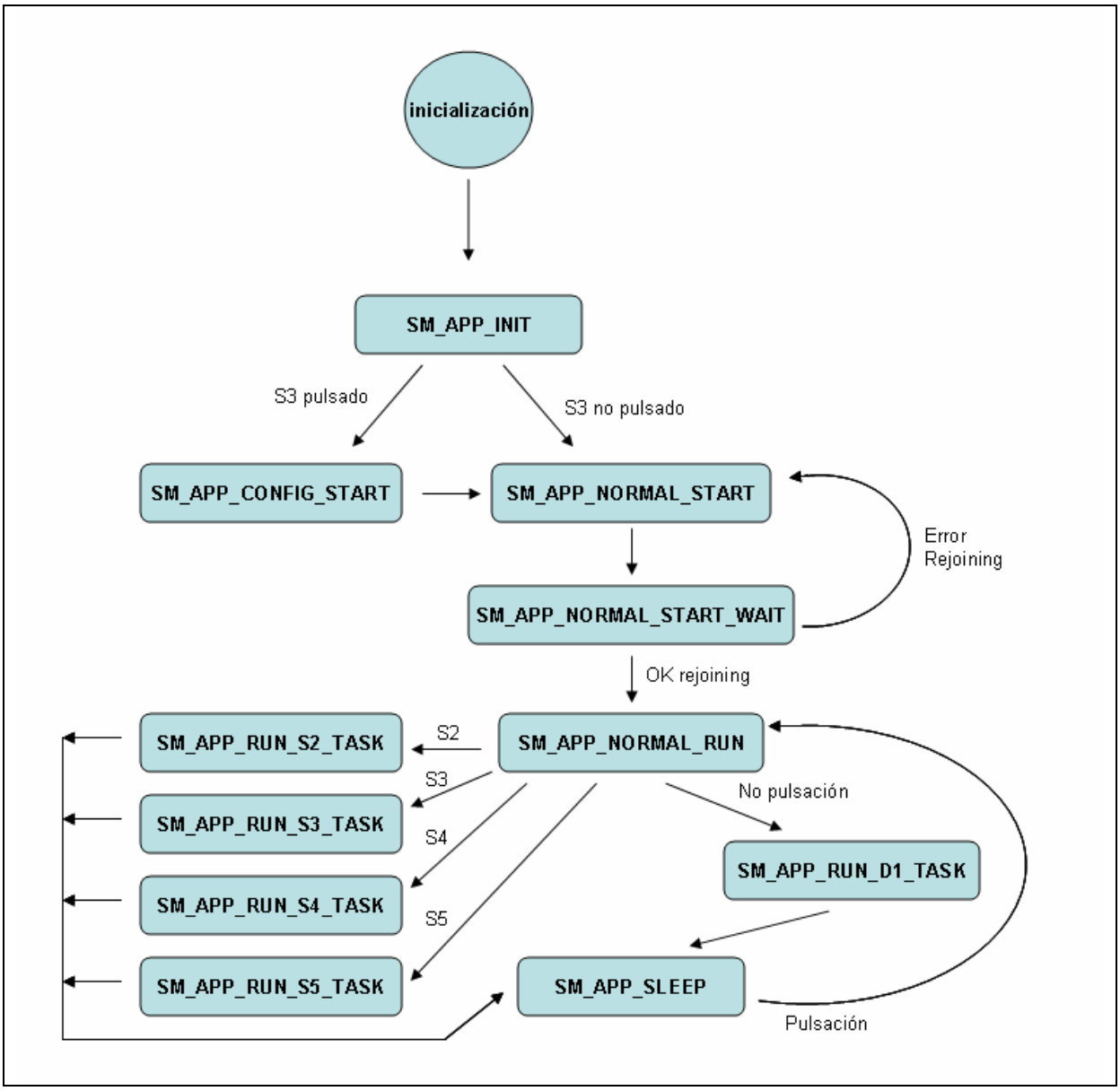


Fig. 36: Diagrama de estados DemoZRFDApp

La inicialización es prácticamente igual a la del Coordinador (salvo la configuración a través de terminal que se realiza dentro de la máquina de estados), así que no la repetimos. Lo que sí cambia un poco es el resto de estados, que comentamos brevemente a continuación:

- Estado SM_APP_INIT: Abre los endpoints correspondientes a S2 y D1, y según si estuvo pulsado de antemano S3, se pasará al estado de configuración (SM_APP_CONFIG_START), o bien si no lo estuvo, al siguiente estado de funcionamiento normal (SM_APP_NORMAL_START).

- Estado SM_APP_CONFIG_START: Aquí tiene lugar la configuración del dispositivo, mediante la llamada a la función *ConfigTask*. Se muestra un menú similar al que se mostraba al configurar por terminal el Coordinador, pero con las opciones propias de un RFD, tales como "unirse a una red", "quitarse de una red", "Hacer un binding de ejemplo", ... Desde este estado se pasa incondicionalmente al siguiente (SM_APP_NORMAL_START).

```
************************************************************************
ZigBee Demo RFD Application v1.1 (Microchip Stack for ZigBee v1.1.0)
     Built on Jun 13 2007 (Non-debug Build)
************************************************************************
     1: Set node ID...
     2: Join a network.
     3: Perform quick demo binding (Must perform #2 first).
     4: Bind statically known endpoints together (Must perform #2 first).
     5: Leave a previously joined network (Must perform #2 first).
     6: Change to next channel.
     7: Transmit unmodulated signal.
     8: Transmit random modulated signal.
     0: Save changes and exit.

Enter a menu choice:
```

Fig. 37: Menú de configuración de un RFD

- Estado SM_APP_NORMAL_START: En este punto se intenta hacer un rejoin al Coordinador (del que se ha de conocer previamente la ID). Paso incondicional al siguiente estado (SM_APP_NORMAL_START_WAIT).

- Estado SM_APP_NORMAL_START_WAIT: Se comprueba que el rejoin haya sido satisfactorio, en cuyo caso se muestra un mensaje de OK, y pasa al siguiente estado (SM_APP_NORMAL_RUN). En caso contrario, vuelve al estado anterior, mostrando el correspondiente mensaje de error por el terminal.

- Estado SM_APP_NORMAL_RUN: Aquí se comprueba si algún flag de pulsación de botón está activo, en cuyo caso se pasa al estado correspondiente a cada botón (SM_APP_RUN_Sn_TASK, donde n = 2, 3, 4 ó 5), mientras que si no ha habido pulsación previa, se pasa al estado SM_APP_RUN_D1_TASK. Es decir, este estado sólo sirve para discriminar la siguiente acción a realizar.

- Estados SM_APP_RUN_Sn_TASK (n = 2, 3, 4 ó 5): Se hace una llamada a la función *S2Task,* pasándole como argumento la letra de votación correspondiente a la pulsación previa que hubo. Así, se pasa una "A" si se pulsó S2, una "B" si fue S3, una "C" si fue S4, o una "D" si por el contrario fue S5. La función *S2Task* se encarga de mandar un mensaje de red al Coordinador con la ID del dispositivo y la respuesta concreta que se ha pulsado. El siguiente estado es incondicional y es SM_APP_SLEEP.

- Estado SM_APP_RUN_D1_TASK: Este estado comprueba mediante la función *D1Task* si se ha recibido algo del Coordinador para este dispositivo. El siguiente estado es incondicional y también es SM_APP_SLEEP. Los mensajes que espera actualmente son de confirmación del envío previo, pero tal cual está funcionando ahora la aplicación no tiene utilidad, ya que pasa a dormir tras enviar un mensaje, y de esta manera, no puede recibir nada. Tan sólo tiene sentido si se deshabilitase la suspensión del micro.

- Estado SM_APP_SLEEP: Se pasa a estado de bajo consumo o de suspensión, mediante una llamada a la instrucción ensamblador SLEEP, habiendo previamente habilitado las interrupciones por pulsación de botón.

En cuanto a los ficheros de fuente que tiene la aplicación, son análogos a los de la aplicación del Coordinador, es decir, se tienen tres ficheros de implementación .C, además de las cabeceras y ficheros correspondientes al stack ZigBee y librerías del C18. Dichos ficheros son:

- DemoZRFDApp.c: Fichero principal que incluye la función main y las de inicialización y configuración.

- S2OnEndDevice.c: Contiene la función *S2Task.*

- D1OnEndDevice.c: Cotinene la función *D1Task.*

4.2.3 Dificultades que han surgido en el desarrollo

Se van a enumerar algunas de las dificultades encontradas en cada paso del desarrollo propuesto por el tutor, en ambas aplicaciones:

- La primera gran dificultad que hubo fue hacerse con el entorno y herramientas de desarrollo (MPLAB + ICD2), ya que ninguno de los componentes del grupo había trabajado con ello antes, y en nada parecido. Pasaron varias semanas hasta que pudimos cargar satisfactoriamente el firmware de los proyectos de ejemplo que venían con el PICDEM Z.

- El siguiente paso fue conseguir una modificación sencilla sobre una de las aplicaciones, de manera que se viera cómo funcionaba la placa base. La modificación en sí consistía en hacer que el programa contase ascendentemente si se pulsaba un botón, y descendentemente si se pulsaba el otro (la placa base tiene dos botones de usuario), al tiempo que se mostraba la cuenta binaria en los dos LEDs que ofrece la placa. Este punto no ofreció gran dificultad.

- A continuación, el tutor propone hacer eso mismo, pero ya con las dos placas, de forma que al pulsar sobre una, se enviase un mensaje a través de la red WPAN ZigBee establecida y se mostrase el resultado de la cuenta por consola en el HyperTerminal de Windows. Este punto requería algo más de información para comprender el funcionamiento de las funciones que ofrece el stack ZigBee de Microchip con el que hemos trabajado.

- Una vez en ese punto, construir lo que sería prácticamente nuestro sistema final en su parte hardware ya era más bien trivial, pues se bastaba enviar mensajes ZigBee de aplicación en los que se mandara un identificador de mando y una respuesta. La dificultad fue un poco bucear entre las capas del stack para ver cómo se almacenaban variables del dispositivo y procesaban tramas, para comprender mejor cómo trabaja. Así se pudo sacar el ID del mando a partir de la dirección MAC de 16 bytes del dispositivo (últimos dos bytes).

- Se introdujeron un par de botones más en la placa de cara a ofertar un par de respuestas posibles más (A, B, C o D). La adaptación del código no fue más problemática.

- Por último comentar la otra gran dificultad que nos llevó de cabeza, que fue conseguir que el dispositivo que hacía las veces de mando a distancia "durmiese", es decir, permaneciera en estado de bajo consumo

despertándose únicamente para enviar una respuesta cuando se pulsara un botón. Esto llevó bastante tiempo dar con ello y hasta estuvo aparcado para seguir realizando otras cosas, pero al final se dio con el problema y la fácil solución. Al procesar una instrucción ensamblador SLEEP, el dispositivo pasaba a estado de "sleep" o suspensión, parando su reloj principal, y estando a la espera de una interrupción externa que lo despertase. El problema era que el dispositivo se despertaba nada más entrar en suspensión, efecto que podría darse si no teníamos bien configurado las resistencias pull-up (de polarización) internas del PORTB o bien no se limpiaban los flags de interrupción, o quizá una interrupción que no controlábamos estaba despertándolo. Tras comprobar que las dos primeras alternativas estaban correctamente configuradas, y tras un estudio de los registros implicados en las interrupciones del PIC, tanto de habilitación, prioridad o flags, y mediante la muestra por consola de sus estados y valores antes y después de ejecutar el SLEEP, se llegó a la conclusión que era el WDT el que provocaba la interrupción, así que bastó deshabilitarlo justo antes de entrar al modo "sleep", y volver a habilitarlo tras salir de él mediante la pulsación de un botón de la placa.

4.3 Software teléfono móvil

4.3.1 Motivación

Lo que se busca con esta ampliación es introducir la posibilidad de que no sólo se respondan a las preguntas planteadas con los mandos de radiofrecuencia, sino que sea posible además responder a tales preguntas con un teléfono móvil. Lógicamente no sirve cualquier teléfono móvil, debe de ser un móvil con tecnología GPRS o cualquier otra que permita acceso a internet, es decir, un móvil que permita la posibilidad de comunicarse usando el protocolo TCP/IP.

Debido al tiempo limitado del que se disponía, se implementó, usando J2ME con el entorno de desarrollo NetBeans de Sun, un sencillo emulador de teléfono móvil, que usando el protocolo TCP/IP, fuese capaz de enviar al receptor las respuestas a las preguntas planteadas pulsando un botón de su teclado. El hecho de elegir J2ME como entorno de ejecución fue debido a la alta probabilidad de que cualquier

terminal móvil de hoy en día lo tuviera incorporado. Dado que la versión actual no permite abrir directamente sockets y que simplemente mediante una petición HTTP podíamos llevar a cabo nuestro propósito, una aplicación en J2ME se convierte en la implementación más sencilla y con mayor probabilidad de aceptación entre los usuarios que otras como por ejemplo Symbian o Windows Mobile.

Por otro lado, se implementó un pequeño servidor en C# usando sockets que estuviese leyendo (mientras durase la comunicación) en los puertos donde los móviles emisores emitirían sus respuestas. Una vez leídas dichas respuestas, el servidor deberá recoger la información que le resulte útil (el ID del móvil y la respuesta seleccionada), e introducirlo en una estructura de datos para su posterior tratamiento y volcado al fichero Excel.

Hay que indicar que en este punto se tienen por tanto dos servidores ejecutándose a la vez (en dos hilos independientes), el primero corresponde al servidor encargado de leer del puerto serie las respuestas emitidas por los mandos, y el segundo es el encargado de leer las respuestas recibidas por el teléfono móvil.

4.3.2 Desarrollo

Para el desarrollo de esta ampliación se implementó un sencillo programa para el emulador de dispositivo móvil con NetBeans IDE. Éste es el aspecto final de la aplicación que se recoge en la pantalla del móvil:

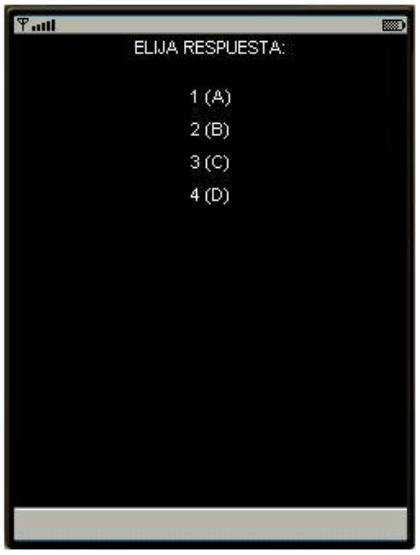


Fig. 38: Captura en el emulador del teléfono móvil

4.3.2.1 Detalle de clases

Clase *GameMIDlet*

La función principal de esta clase es la de definir el prototipo de móvil que será usado y lanzar un hilo que representa la ejecución de un prototipo de teléfono móvil. Poco más se puede decir de esta clase a la que podríamos calificar de clase de definición.

Clase *J2MEClientCanvas*

En esta clase se define la apariencia del móvil que tendremos: fundamentalmente el número de opciones: A, B, C y D, que se mostrarán por pantalla y que representan las respuestas a las preguntas. También se definen una serie de parámetros que representan características de apariencia de la aplicación móvil (color, forma, tamaño, etc.)

Lo más importante que se define en esta clase es la forma en que se enviarán las respuestas seleccionadas con el móvil al servidor que esta a la escucha.

Existe un método asociado al evento *"KeyPressed"* que selecciona la respuesta representada por la tecla pulsada, así como un ID del móvil en cuestión. Una vez que se tiene el ID del móvil y la respuesta deseada, se crea una instancia de la clase *"Sender"* que lanza otro hilo de ejecución que será el encargado, mediante una petición HTTP, de enviar la respuesta a la URL donde el servidor que está a la escucha seleccionará la parte "válida" de la información enviada y la introducirá en su estructura de datos.

La clase *Accion* de la aplicación del PC se vio modificada en una pequeña parte, ya que ahora en *iniciar_recepcion*, además de iniciar la recepción del servidor que lee las respuestas de los mandos *(DAO_SP)*, hay que iniciar el servidor que lee las repuestas de los móviles *(DAO_MOV)*.

5 DESARROLLO DE LA SOLUCIÓN OEM

Este punto se ha quedado incompleto por diversos problemas, entre los que podemos mencionar como principal la falta de tiempo o el hecho de no disponer de una comunicación directa con el técnico del laboratorio, que nos ha ayudado en la construcción de los mandos y problemas que iban surgiendo. Vamos a comentar los puntos que hemos ido siguiendo para este propósito.

Para la construcción de esta solución se han adquirido algunos productos ZigBee de Flexipanel Ltd. como son 17 unidades de Pixie SO y 2 unidades de UZBee. El objetivo es conseguir que cada uno de los Pixies sean parte de un mando a distancia, y una unidad de UZBee sirva de coordinador de red, que capture las respuestas que se emiten desde cada uno de los mandos.

Fig. 39: Unidad UZBee (izquierda) y placa Pixie SO (derecha)

Así pues, se van a tener dos partes diferenciadas, por un lado se va a tener que construir un mando a distancia y cargar el firmware correspondiente al RFD de nuestro prototipo de desarrollo del PICDEM Z, y por otro lado se va a tener que programar el firmware del Coordinador de red del prototipo inicial, adaptado para el nuevo dispositivo OEM. Vamos a comentar de qué elementos constan estos productos.

El UZBee a grandes rasgos es un terminal ZigBee que puede conectarse por interfaz USB a un PC, de forma que puede servir como punto de acceso a una red ZigBee, o bien formar parte activa de ella como Coordinador, por ejemplo. El microcontrolador que lo rige es un PIC18LF4550, de la misma familia que el 4620

del PICDEM Z, y totalmente compatible con éste. Al igual que el PICDEM Z, el transceptor de radio que posee es el CC2420 de Chipcon, así que en este aspecto no varía nada. Pese a ello, hay ciertas particularidades muy a tener en cuenta a la hora de adaptar el código firmware del prototipo del desarrollo al de esta solución OEM, el código del 4620 al del 4550, y que ya las hemos comentado en el 3.1 de esta memoria. Fundamentalmente se destaca la diferencia de memoria de programa, sabiendo que el 4620 dispone de 64 KB y el 4550 del UZBee tan sólo de 32, de los cuales tiene prácticamente la mitad dedicados al buffer del interfaz USB. Es una limitación importante, porque aunque nuestra aplicación sea relativamente sencilla, hay que conseguir un tamaño de firmware de 16 KB a lo máximo.

Por otro lado tenemos el Pixie SO, un producto que a priori es 100% compatible con nuestro firmware desarrollado en PICDEM Z. El Pixie SO consta de un microcontrolador PIC18LF4620, idéntico al que trae el PICDEM Z, salvo en el encapsulado, mucho más compacto. El transceptor de radio con el que cuenta es también el ya mencionado CC2420 de Chipcon, así que se ve claramente que los chips que se usan y programan son los mismos que los de la etapa de desarrollo.

Según lo que acabamos de exponer se entiende que no debería ser muy difícil el paso de la etapa de desarrollo con el PICDEM Z a la de fabricación final con UZBee y Pixie SO, pero no ha sido así, lejos de eso hemos tenido una serie de dificultades que nos han impedido llevar a buen puerto esta fase. Vamos a pasar a comentar los problemas que hemos tenido en este punto con cada uno de los nuevos productos adquiridos, comentando en primer lugar el tiempo; los materiales llegaron con tiempo suficiente para ir sacando todo prácticamente a la primera o con un pequeño margen de error, y la sucesión de problemas ha hecho que este factor haya sido determinante. Otra causa externa ha sido la falta de comunicación directa con el técnico del laboratorio, que ante cualquier problema con el material nos ha estado ayudando.

Ya como otros problemas que nos atañen más directamente podemos enumerarlos a continuación:

- Problemas al compilar el firmware del coordinador.
- Problemas al cargar el firmware del coordinador en el UZBee.
- Problemas a la hora de cargar el firmware del RFD en el Pixie.

- Problemas al ejecutar el firmware del Pixie.

El primero de los problemas enumerado se refiere a la compilación del coordinador de red para el 4550. El C18 reportaba continuamente errores en la fase del *linker* al definir las secciones de memoria que ocuparían determinadas estructuras como la Tabla de Vecinos por ejemplo, al reubicar mediante el *Linker Script* correspondiente a este microcontrolador. Se resolvió parcialmente reduciendo los tamaños de estas estructuras que dieron problemas a lo mínimo necesario para poder compilar, así como ubicándolos en segmentos de memoria existentes en este micro, ya que se usaban unos que se definían para el 4620 y no para el 4550. Así mismo, ha sido necesario disponer de toda la capacidad de optimización del C18, con la opción de abstracción procedural con valor 1, y se ha conseguido disponer de un fichero HEX compilado pero no correctamente programado. No se consiguió superar del todo este obstáculo por dejarlo aparcado de cara a solventar otros problemas.

Aun con un HEX ya disponible para la carga en el UZBee, no ha sido posible tampoco realizarla. El problema es la falta de herramientas para ese fin, ya que Flexipanel (el fabricante), tan sólo nos facilita carga de un firmware suyo a través del puerto USB, mediante un bootloader (cargador de firmware) ya preprogramado en el UZBee llamado "USBoot", que de cara al PC consigue que aparezca el dispositivo en modo de carga (USBoot) como un puerto serie en el panel de "Administración de dispositivos" y mediante una aplicación Windows se permite la carga de su firmware final, denominado "Star Lite USB". Se pueden obtener este software, así como las instrucciones para cargarlo ([15] de la bibliografía) de su página web (www.flexipanel.com). La solución a este problema pasa por conectar el ICD2 al propio UZBee, para así poder cargar el firmware deseado, lo que pasa es que para ello hay que abrir el envoltorio y encontrar los terminales correspondientes a los que soldar el conector de ICD2.

En cuanto al Pixie SO el primer problema ha sido la carga del programa a través del puerto serie. No ha habido forma hasta que se le soldó un conector de ICD2, momento a partir del cual las programaciones de firmware no han dado ni un solo problema más.

El último de los problemas que hemos enumerado se refiere a la no completa compatibilidad del mismo firmware que corría perfecto en el PICDEM Z. La primera dificultad en este punto ha sido la no posible visualización de mensajes por consola

a través de los pines habilitados para ello. La razón no hemos podido saber cuál era, pero sin duda ha sido un gran problema a la hora de saber qué era lo que realmente ocurría en el controlador al conectar la pila. Se llegó a saber por algunos momentos que funcionaba, dejando automática la configuración interactiva del dispositivo con su coordinador, viendo que enlazaba con éste y se permitía el envío de respuestas, con muchos rebotes y sin funcionamiento correcto del modo *sleep* (suponemos debido a los rebotes al pulsar), pero no se ha podido avanzar más en ese tema debido a la falta de tiempo y a los ya mencionados problemas de comunicación directa con el técnico del laboratorio.

6 CONCLUSIONES

En vista de la evolución del proyecto se puede concluir que el uso de esta joven tecnología ha supuesto un acierto a nivel de consumo, capacidad de uso, economía, etc. Estos hitos fueron posibles gracias a la facilidad que ofrece el entorno de desarrollo a los principiantes, la documentación aportada por los fabricantes y la proximidad de los estándares al objetivo perseguido.

Llegados a este punto, sólo queda por decir que el objetivo fundamental de este proyecto era ofrecer una alternativa eficaz y económica comparada con otros sistemas similares existentes en la actualidad.

La ilusión que promovió la realización de este proyecto se sustentó básicamente en el hecho de desear aproximar a toda la comunidad docente algo tan ansiado y desgraciadamente utópico como la evaluación continua.

Deseamos haber sido de ayuda en este propósito de todos.

Anexo A: CAPAS DE LA ARQUITECTURA ZIGBEE

A.1 Capa física (PHY)

La capa física es responsable de las siguientes tareas:

- Activación y desactivación del transceptor de radio,
- Detección de energía (ED, "Energy Detection") con el canal actual,
- Indicador de calidad de enlace (LQI, "Link Quality, Indicator") para los paquetes recibidos,
- Cálculo del canal libre (CCA, "Clear Channel Assessment") para CSMA-CA,
- Selección de frecuencia del canal, y
- Transmisión y recepción de datos.

A.1.1 Posibilidades de capa PHY

El estándar 2006 define un total de cuatro capas físicas, dos más que el de 2003. Son las que quedan recogidas en la siguiente tabla, según frecuencia y modulación:

PHY (MHz)	Frequency band (MHz)	Spreading parameters		Data parameters		
		Chip rate (kchip/s)	Modulation	Bit rate (kb/s)	Symbol rate (ksymbol/s)	Symbols
868/915	868–868.6	300	BPSK	20	20	Binary
	902–928	600	BPSK	40	40	Binary
868/915 (optional)	868–868.6	400	ASK	250	12.5	20-bit PSSS
	902–928	1600	ASK	250	50	5-bit PSSS
868/915 (optional)	868–868.6	400	O-QPSK	100	25	16-ary Orthogonal
	902–928	1000	O-QPSK	250	62.5	16-ary Orthogonal
2450	2400–2483.5	2000	O-QPSK	250	62.5	16-ary Orthogonal

Fig. 40: Tabla resumen PHY

Las que se marcan como opcionales son las nuevas recogidas en el estándar de 2006, y consiguen mejor tasa de transferencia de datos introduciendo más complejidad. La banda de 868 MHz es la usada en EE.UU., mientras que la de 915 MHz es la utilizada en Europa. La banda de 2.4 GHz se puede usar en todo el mundo. El Grupo 4 del IEEE 802.15 está trabajando ahora mismo en preparar capas PHY para Japón y China, de forma que se ajusten a sus regulaciones nacionales ya que las que recoge ahora mismo el estándar no las cumplen.

Según la banda por la que se opte, se tendrá un número diferente de canales por los que se pueda establecer la comunicación. Así, por la banda de 868 MHz tan sólo se tiene un canal, por la de 915 MHz, 10 canales, y por la de 2.4 GHz un total de 16 canales. El hecho de tener más de un canal de comunicación en una misma banda de frecuencia es útil si hay otra red ya establecida en un canal, para así poder establecer otra en otro canal, usando la misma banda. A continuación se muestran los canales que pueden establecerse según la banda de frecuencia que se esté usando:

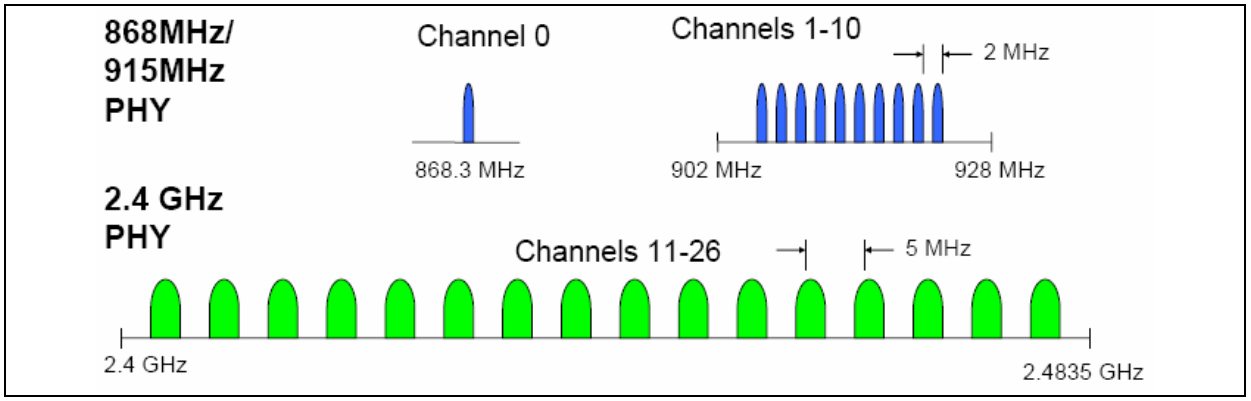


Fig. 41: Bandas de frecuencia y canales

A.1.2 Funciones PHY

El estándar permite selección dinámica del canal, una función que hace un escaneo por cada uno de los canales que soporta (almacenados en una lista), en busca de baliza, detección de la energía del receptor (ED), indicación de calidad del enlace (LQI), y cambio de canal.

Las funciones de ED y LQI son dos funciones de medida que indican el nivel de interferencia con un canal IEEE 802.15.4. La medida ED es una estimación de la potencia de la señal recibida con un canal del estándar, mientras que la LQI mide el

nivel de energía recogida y/o el SNR ("Signal to Noise Ratio", o relación señal/ruido) de cada paquete recibido; si se combina nivel de energía y SNR, esta medida puede indicar si un paquete corrupto es fruto de una baja señal o de una alta señal con interferencias.

El algoritmo de cálculo del canal libre (CCA) que hemos mencionado antes se realiza o bien por un método consistente en comprobar si la ED ha traspasado un cierto umbral, o bien detectando una señal con las características propias del estándar, o bien por combinación de ambos métodos. El uso de la primera de las opciones mejora la coexistencia por permitir transmisiones tras esperas aleatorias (backoff) si el canal lo ocupa otro dispositivo, sin importar el protocolo de comunicación usado.

A.1.3 PPDU: paquetes de PHY

Podemos también indicar dentro de este apartado de PHY, el formato de paquete de datos a este nivel. Tiene la siguiente pinta:

		Octets		
		1		variable
Preamble	SFD	Frame length (7 bits)	Reserved (1 bit)	PSDU
SHR		PHR		PHY payload

Fig. 42: PPDU (unidad de datos en PHY)

Un PPDU (así se llama a los paquetes de datos en este nivel), se compone básicamente de tres campos principales:

- Un encabezamiento de sincronización (SHR, "Synchronization Header"), que permite al dispositivo que lo recibe sincronizar el flujo de bits que va a recibir a continuación,
- Una cabecera de PHY que contiene información de la longitud de la trama.
- Una carga útil (payload) de longitud variable, que lleva el PDU de la capa MAC inmediatamente superpuesta a PHY.

No vamos a entrar ya más en la descripción y significado de cada uno de los campos, se puede ver en el estándar del IEEE con completo detalle.

A.1.4 Servicios de PHY

La capa PHY ofrece dos servicios a la capa MAC, la inmediatamente superior. Uno es el servicio de datos PHY (PHY Data o PD), y otro el de gestión de la capa (PHY Layer Management Entity o PLME), ambos con acceso MAC a través de los puntos de acceso PD-SAP y PD-PLME respectivamente, mediante primitivas de tipo "request", "confirm" e "indication". El servicio PD básicamente es el encargado del transporte de MPDUs entre PHY y MAC en ambos sentidos. Por otro lado, el PLME tiene el cometido de traer hasta PHY las órdenes o comandos de gestión de capa desde MAC, y devolver resultados de esas funciones hacia arriba en el stack. Los comandos que puede realizar son los ya comentados entre las funciones de la capa PHY:

- Ejecución del CCA,
- Toma de medida ED,
- Devolver o asignar dato relativo a las variables de PHY, recogidas en una pequeña base de datos llamada "PAN Information Base" o PIB, en el propio PLME,
- O fijar el estado de configuración del transceptor (transmisor + receptor) de radio, es decir, apagar transceptor, encender transmisor, o encender receptor.

Una vez comentados todos estos aspectos de PHY, ya podemos introducir la capa MAC.

A.2 Capa de control de acceso al medio (MAC)

La capa de acceso al medio, más comúnmente llamada MAC, tiene las siguientes responsabilidades dentro del stack:

- Generar balizas de red si el dispositivo es un coordinador,

- Sincronización mediante las balizas,
- Soporte de asociación y desasociación de la PAN,
- Soporte de seguridad del dispositivo,
- Uso de CSMA-CA para el acceso al canal,
- Manejo y mantenimiento del mecanismo de slots de tiempo garantizados ("Guaranteed Time Slots", GTS), y
- Provisión de un enlace fiable entre dos entidades MAC de diferentes dispositivos.

A.2.1 Estructura de supertramas

Para el acceso al medio, se dispondrá de dos opciones diferentes que son, usar un modelo de supertrama o no usarlo. El modelo de supertrama consiste en considerar el espacio de tiempo dividido en una estructura patrón ranurada que denominaremos "supertrama", y a la que todos los dispositivos que accedan al canal se deberán ceñir a la hora de hacer sus transmisiones de datos. El dispositivo que organizaría el formato de este modelo lógicamente sería el coordinador de red. En este modelo, cada supertrama se separa de la siguiente por medio de las balizas de red ("beacons"), y a su vez, cada una de ellas se divide en 16 pequeños slots (ranuras) de tiempo de igual tamaño. El primer slot de cada supertrama es por definición la baliza separadora.

En cada supertrama existe un periodo que llamamos de contención ("Contention Access Period", PAC), en el cual los dispositivos pueden competir por el uso del canal usando el algoritmo CSMA/CA, que explicaremos más adelante, y tras él, podemos tener opcionalmente otro periodo de acceso libre ("Contention Free Period", CFP), en el cual aseguramos que un dispositivo determinado puede acceder al medio sin necesidad de competir con nadie. Este CFP se divide virtualmente en otras unidades que agrupan uno o varios slots de tiempo, y que se llaman Slots de Tiempo Garantizado o GTS, pudiendo tener hasta un máximo de 7 de ellos, cada uno de los cuales puede aprovecharlo un único dispositivo. Naturalmente, se entiende que al final de cada GTS, el dispositivo que lo tiene ha terminado su transmisión, así como que al final de cada supertrama, todas las transmisiones que

han tenido lugar, tanto en el periodo de contención, como en el de libre acceso han concluido. Opcionalmente, se puede tener además un periodo de inactividad en la supertrama, en el cual no se transmite nada y el coordinador puede dormir. A continuación se indican unos esquemas resumen de cómo puede estar formada una supertrama:

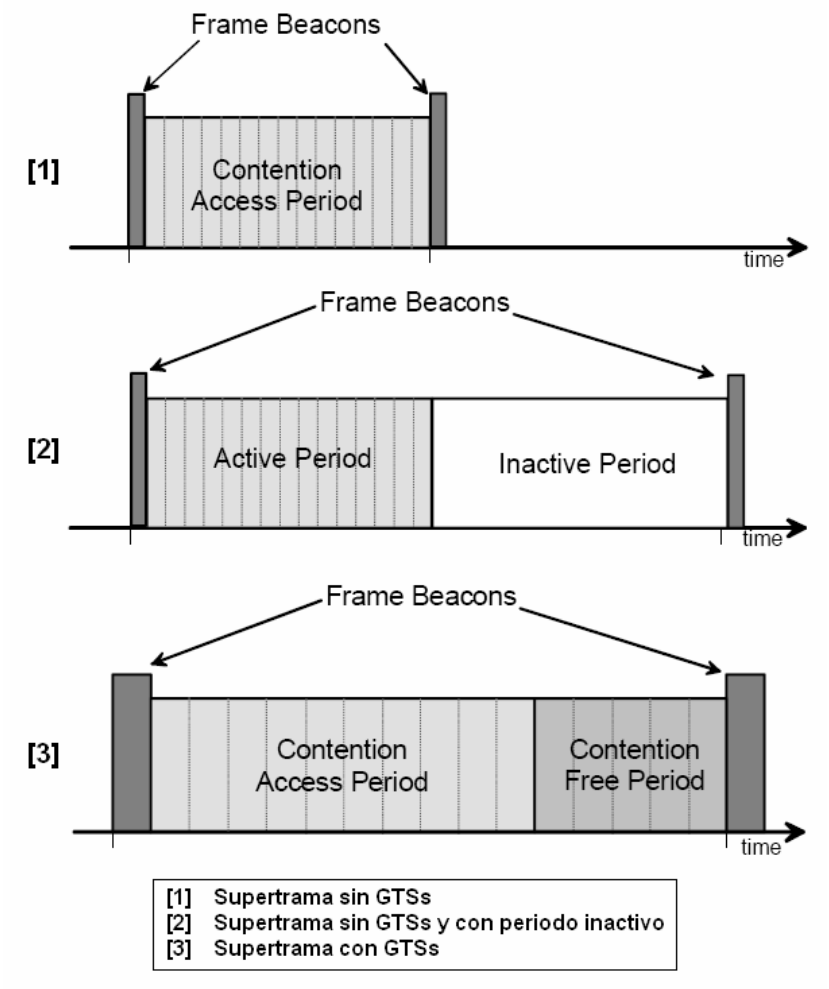


Fig. 43: Formatos de supertrama

Naturalmente, como hemos dicho al principio, el uso de las supertramas es completamente opcional, y se puede prescindir de ello, en cuyo caso, los dispositivos pueden competir en cualquier instante, ya que el espacio del tiempo no está ranurado ni distribuido. De cualquier forma, tanto con supertramas o sin ellas, el mecanismo de acceso al canal lo determina el algoritmo CSMA/CA, que vemos a continuación:

A.2.2 Algoritmo CSMA/CA

Las siglas CSMA/CA corresponden a "Carrier Sense Multiple Access / Collision Avoidance", o lo que es lo mismo, Acceso Múltiple con Detección de Portadora, en su variante con Evitación de Colisiones. Este algoritmo es una variante de tantas del conocido CSMA, en el que, como su propio nombre indica, se tienen muchos dispositivos que acceden a la par al canal, pueden reconocer la señal de transmisión de otro, y además, evitan transmitir mientras otro dispositivo lo está haciendo. Para este propósito, antes de invadir el canal, el propio dispositivo se jacta de ser el único que quiere transmitir, primero mirando que el canal no está ocupado por ningún otro, y a continuación, pidiendo paso para transmitir. Así, las únicas colisiones que pueden darse son entre señales de petición de toma del canal, por tanto, no se corrompen datos entre los dispositivos que forman la red.

El algoritmo tiene dos versiones, dependiendo de si tenemos una estructura de supertrama ranurada o no, aunque a la postre, la idea es exactamente la misma, y es la que acabamos de comentar. Y se usa en general antes de transmitir dato alguno, a excepción de los envíos de balizas, confirmaciones (ACKs) o datos que sean transmitidos en una supertrama mientras dura un periodo de transmisión libre, ya que no es necesario en ninguno de esos casos comprobar que el canal esté libre (que lo estará) ni pedir permiso (pues lo tenemos).

Para comenzar, podemos definir el periodo de backoff como la unidad de tiempo de espera en cualquiera de las versiones del algoritmo. Con esto, en una estructura ranurada, los inicios de los periodos de backoff coinciden con los límites de los slots, para así asegurar que se cumplen los intervalos definidos y las reglas de transmisión. En una estructura no ranurada, nada de esto se comprueba.

Vamos a poner el esquema del algoritmo y hacer una breve explicación sin detallar todo al completo:

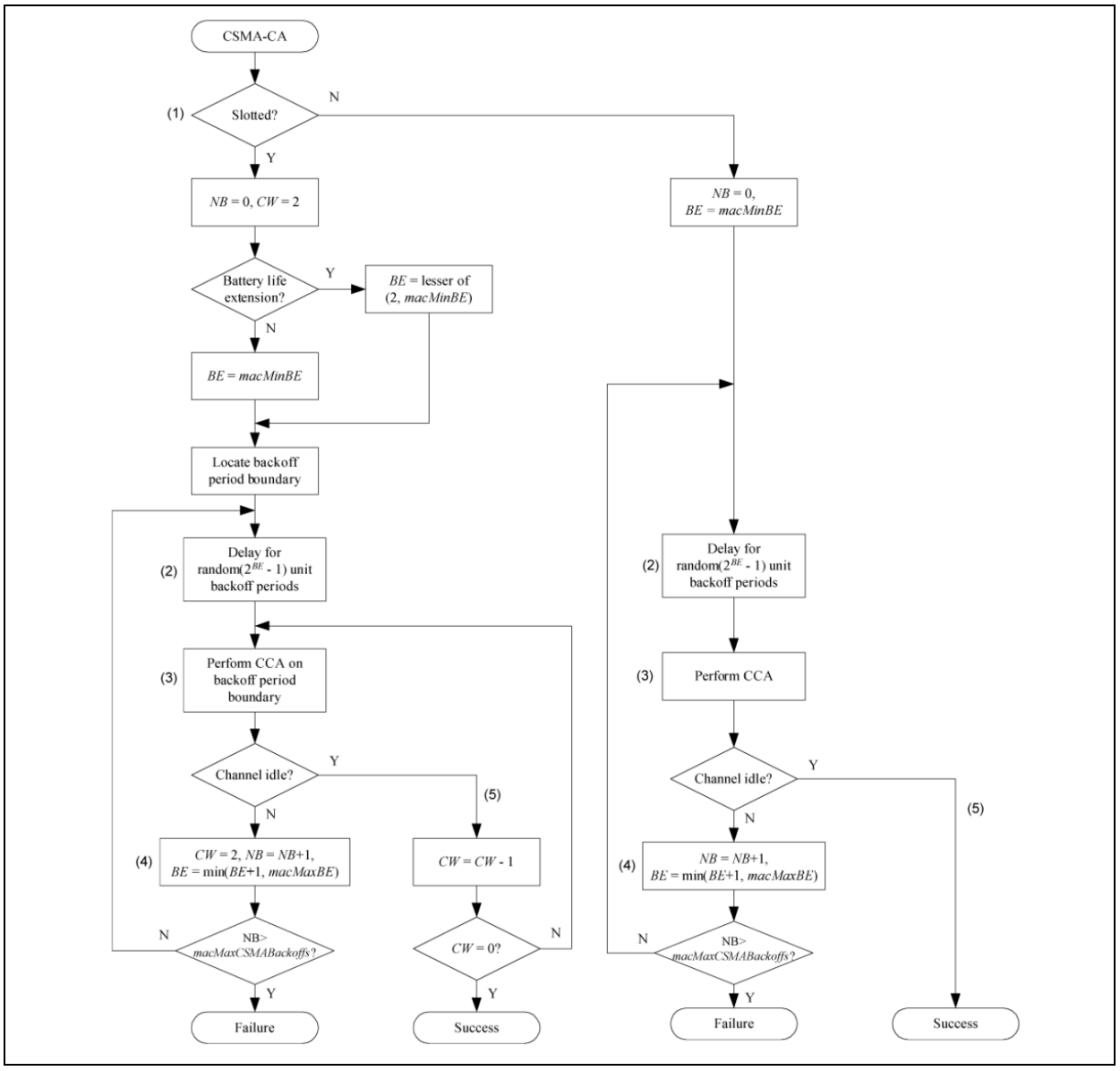


Fig. 44: Algoritmo CSMA/CA

Las variables NB, CW y BE se tienen por cada intento de transmisión. NB es el número de veces que el algoritmo hace que el dispositivo mantenga periodos de backoff para intentar la transmisión actual, se pone a cero al iniciar un intento de transmisión y servirá para ver si hemos superado un cierto umbral admisible de esperas, sumando uno cada reintento de una misma transmisión. CW es la anchura de la ventana de contención, y define el número de periodos de backoff que se necesitan para no tener ningún tipo de actividad en el canal antes de que la transmisión pueda comenzar; sólo se usa en la versión ranurada, tomando el valor 2 al inicio de un intento, y disminuyendo una unidad por cada comprobación de canal libre exitosa, de forma que cuando llega a cero, el dispositivo puede ocupar el

canal para transmitir. Finalmente, BE es el exponente de backoff, relacionado con cuántos periodos de backoff puede esperar un dispositivo antes de intentar hacer la detección de la portadora o cálculo del canal libre (CCA).

Se pueden observar así mismo las dos grandes ramas que ofrece el diagrama de flujo, correspondiendo cada una de ellas a las versiones ranurada y no ranurada del algoritmo.

A.2.3 MPDU: Paquetes de la capa MAC

El formato general que presentan los paquetes en esta capa es el siguiente:

<table>
<tr><td>Octets: 2</td><td>1</td><td>0/2</td><td>0/2/8</td><td>0/2</td><td>0/2/8</td><td>0/5/6/10/ 14</td><td>variable</td><td>2</td></tr>
<tr><td rowspan="2">Frame Control</td><td rowspan="2">Sequence Number</td><td>Destination PAN Identifier</td><td>Destination Address</td><td>Source PAN Identifier</td><td>Source Address</td><td rowspan="2">Auxiliary Security Header</td><td rowspan="2">Frame Payload</td><td rowspan="2">FCS</td></tr>
<tr><td colspan="4">Addressing fields</td></tr>
<tr><td colspan="7">MHR</td><td>MAC Payload</td><td>MFR</td></tr>
</table>

Fig. 45: MPDU (unidad de datos en MAC)

Este esquema es común para todos los tipos de paquetes que se pueden mover por la capa MAC. Como vemos, principalmente se componen de tres grandes campos:

- Encabezado de trama (MAC Header, MHR), que lleva información de control, número de secuencia, e informaciones de dirección y seguridad.
- La carga útil (payload), de longitud variable, con información específica del tipo de trama. Las tramas de tipo ACK no incluyen este campo.
- Un final de trama (MAC Footer, MFR), que contiene un campo de chequeo de integridad de la trama (Frame Check Sequence, FCS)

No se va a entrar en más detalles sobre los campos pues sus nombres son bastante descriptivos, tan sólo comentar que los campos marcados como “Addressing Fields” pueden variar según el tipo de trama que se tenga.

A.2.4 Modelo de transferencia de datos

Aquí cabe únicamente comentar los tres tipos de transacciones que se pueden realizar a este nivel, según las funciones de los dispositivos. Así, tenemos transacciones entre un coordinador y un dispositivo cualquiera, entre un dispositivo cualquiera y un coordinador, y una transacción P2P, entre dos dispositivos cualesquiera no coordinadores. Este último tipo no se da en topologías de estrella, donde todas las transacciones forzosamente pasan por el coordinador de la red.

La secuencia de pasos que se dan para completar una transacción determinada independientemente del tipo que sea de los ya mencionados, va en función de si tenemos habilitadas o no las balizas en nuestra red. Si están habilitadas, antes de cualquier transmisión de datos el dispositivo ha de "escuchar" el medio hasta encontrar una baliza que inicie la supertrama. A partir de ese momento, y en el instante que le corresponda, hará acceso al canal mediante el CSMA/CA en su versión ranurada. Así, para transacciones en sentido RFD-COORD, tendremos el siguiente esquema sencillo:

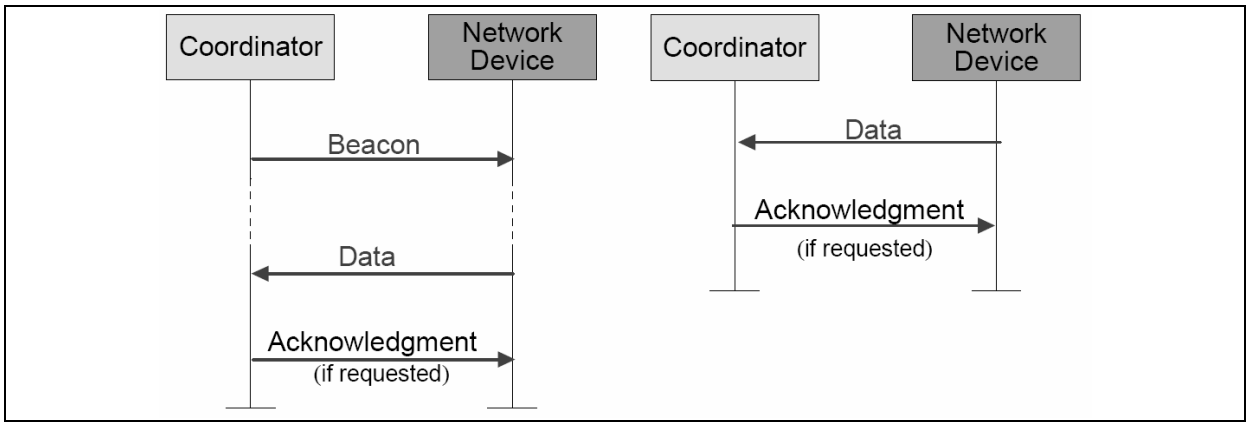


Fig. 46: Transacción RFD-COORD (con y sin balizas)

Simplemente se transmiten los datos en el momento que sea (si hay balizas, esperando la baliza primero y el instante de tiempo que corresponda según CSMA/CA ranurado después), y se espera un ACK si corresponde. Para el sentido de transacción contrario, la situación es la siguiente:

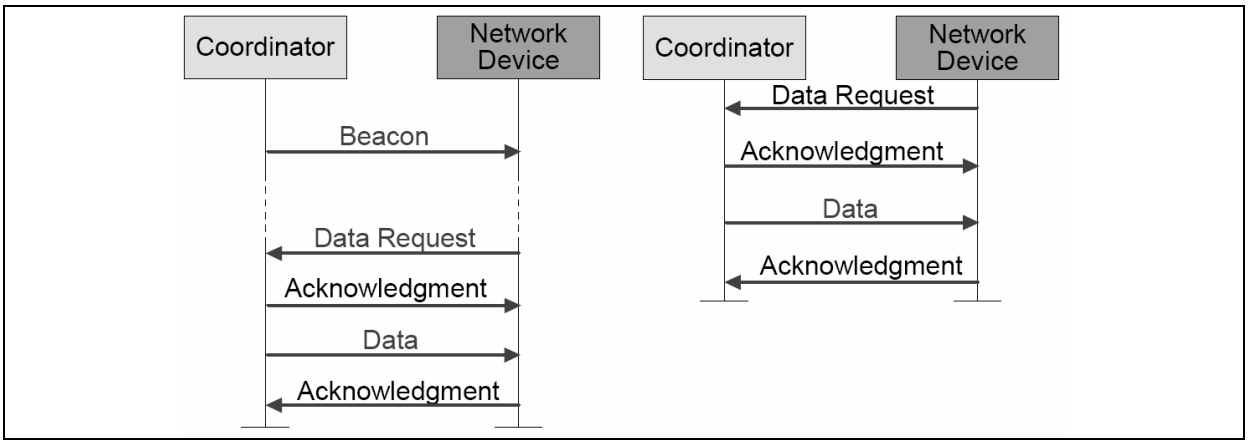


Fig. 47: Transacción COORD-RFF (con y sin balizas)

En este otro caso, el dispositivo coordinador primero tiene que avisar al RFD correspondiente que desea transmitirle datos. Ello ocurre de diferentes formas según estén o no habilitadas las balizas, pero lo que sí es común, es que en el momento que el coordinador quiere hablarle a un RFD se apunta esos datos como "pendientes" sin enviarlos directamente. Así, en un medio ranurado, el coordinador hace notar su deseo de transmitir dentro de una baliza, y ya en el momento que corresponda, el RFD le va a hacer la petición de esos datos pendientes. En el caso de tener una red sin balizas, es el dispositivo RFD el que tiene que preguntar si el coordinador tiene datos para él, y el coordinador se lo hace saber por medio de la trama ACK que responde a su petición de datos, y mandando a continuación los datos si así los tuviera pendientes.

En el último caso que contemplábamos, las transacciones P2P entre RFDs, quedan algo indefinidas en el estándar. Se tendrían dos opciones principales, o bien estar en modo de recibiendo constantemente, usando CSMA/CA no ranurado, o bien sincronizarse de alguna forma entre ellos.

A.2.5 Servicios de MAC

Al igual que PHY, MAC ofrece dos servicios análogos a su capa superior (NWK). Dichos servicios son: MCPS ("MAC Common Part Sublayer") y MLME ("MAC Layer Management Entity"), accesibles mediante sus respectivos SAP del mismo nombre.

Las funciones del MCPS son las de transportar unidades de datos de la capa de red (NPDU) hacia PHY y viceversa, a través de unidades de datos MPDU de este nivel,

incorporando o eliminando la información pertinente de las cabeceras y finales de datagrama según sea el sentido del flujo de datos. Como ejemplo, éste podría ser un esquema de paso de datos entre capas MAC de diferentes dispositivos, usando este servicio y sus primitivas:

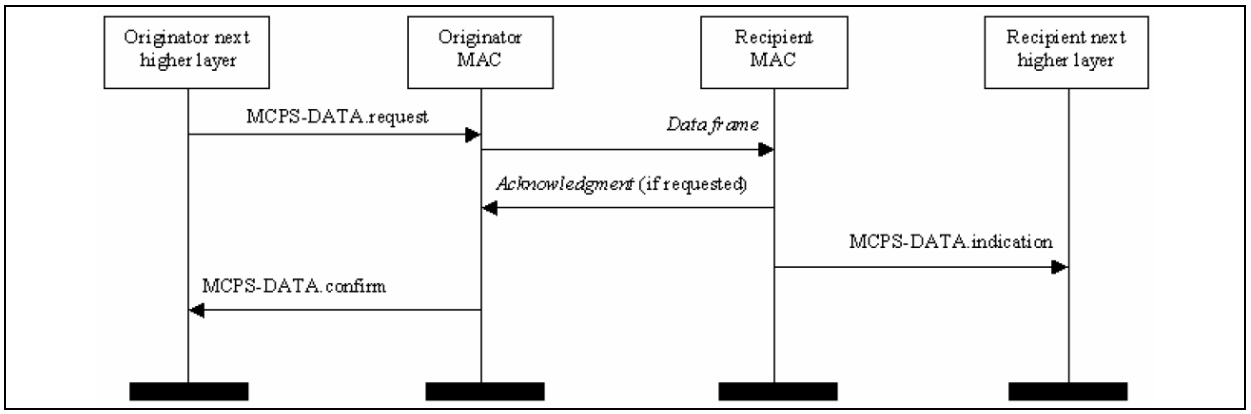


Fig. 48: Envío de datos a nivel MAC

El servicio MLME es más complejo, y realiza multitud de funciones asociadas a esta capa, a través de comandos MAC. Podemos encontrar algunas de las que ya hemos ido hablando previamente en esta lista de todos los posibles comandos o primitivas de MLME:

- Asociación y desasociación a una red.
- Notificación de datos de una baliza recibida a la capa superior, así como de una medida de calidad del enlace y hora a la que se recibió ésta.
- Lectura y escritura de valores del PIB (información de la PAN, contenida en PHY).
- Elaboración y mantenimiento de los GTSs (ranuras de tiempo de acceso garantizado).
- Notificación de presencia de dispositivos huérfanos. Los huérfanos son aquellos dispositivos que habían pertenecido a la red, y que se habían desasociado de ella.
- Reset a los valores por defecto de la capa MAC.
- Desactivar el receptor en un momento dado.

- Escaneo de canales, para detectar la presencia o ausencia de otras PANs en un canal o el uso de energía.
- Indicación del estado de una comunicación.
- Actualización de la configuración de supertrama.
- Sincronización con un coordinador, tanto el hecho de conseguirla como comunicar su pérdida a la capa NWK.
- Pedir datos al coordinador de red.

Con esto queda comentado todo lo que es relevante a la capa MAC, y podemos introducir ya la siguiente capa del stack, la capa de red.

A.3 Capa de red (NWK)

Empezamos comentando las responsabilidades generales de esta capa a modo introductorio:

- Unirse y dejar una red,
- Aplicar seguridad a las tramas,
- Guiar tramas a sus supuestos destinos,
- Descubrir y mantener rutas entre dispositivos,
- Descubrir vecinos 1-hop ("a un salto", que están en su área de cobertura), y
- Almacenar la información pertinente de los vecinos.

Un coordinador de red además de éstas, presenta otras como empezar una nueva red y asignar direcciones a los nuevos dispositivos asociados.

A.3.1 Establecimiento de una nueva red, unión y salida de dispositivos

Para establecer una nueva red, el procedimiento que se sigue empieza desde las capas superiores, llamando a la primitiva del servicio de NWK encargada de ello (NLME-NETWORK-FORMATION.request). En este instante, y ya desde esta capa, se

van haciendo solicitudes a servicios de MAC para que se vayan solventando todos los pasos del algoritmo de formación de red. Si no fuera posible establecer la red por alguna causa, se le notificará por medio de los valores oportunos a la capa superior.

Éste vendría a ser el proceso, con indicación de las primitivas que se llaman:

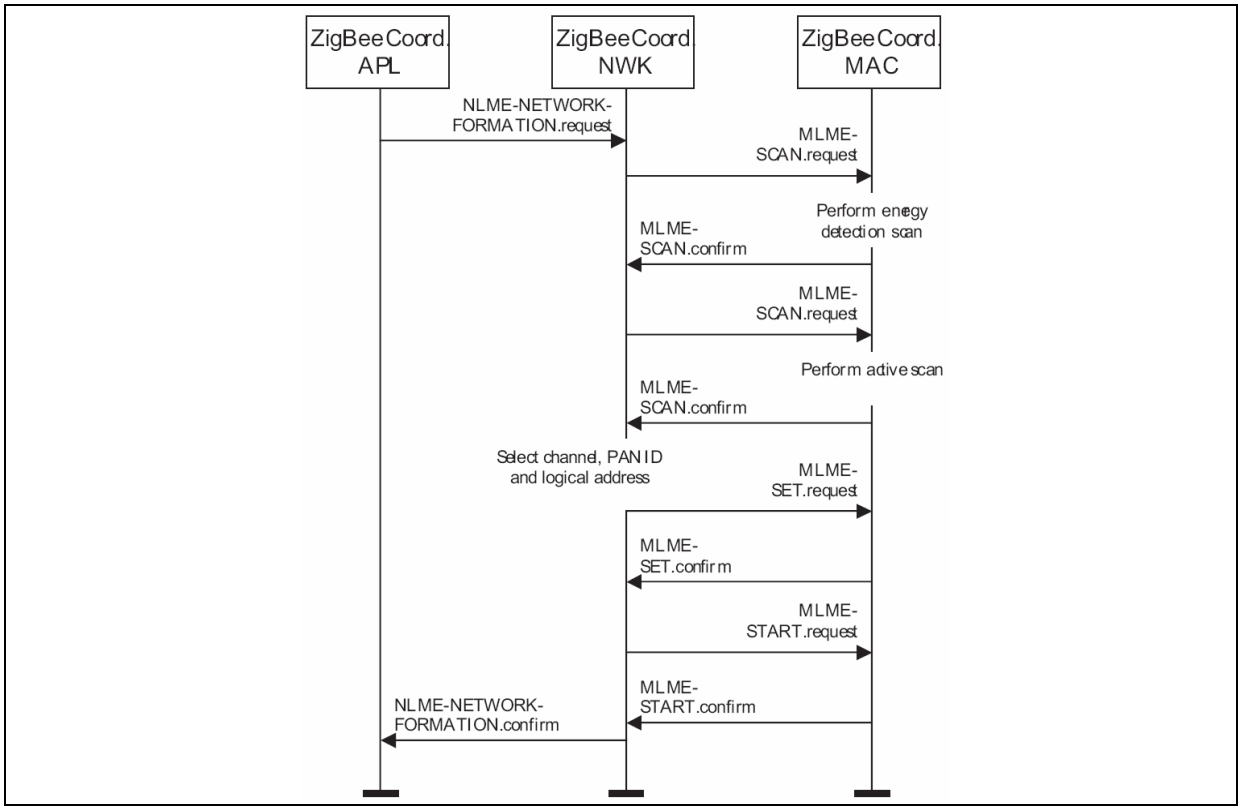


Fig. 49: Establecimiento de una nueva red

Como se puede ver en la Fig. 49, básicamente, consiste en:

1. Solicitar un escaneo del medio a MAC, para medir la energía en cada uno de los canales que se exploren. Elaborar una lista descendente de esos canales, en función de la medida de energía, descartando aquellos que estén por debajo de un determinado umbral.

2. Solicitar un nuevo escaneo (escaneo activo) sobre los canales preseleccionados, para esta vez, detectar otros dispositivos ZigBee que tengan establecidas ya redes. Para hallar el mejor canal en el que establecer la nueva red, se va a ir quedando con el canal con menor número de redes

existentes en él, favoreciendo aquél que no tenga ninguna. Si no se encuentra ninguna favorable, se termina el procedimiento.

3. Una vez tenemos un canal en el que establecer la nueva red, se escoge un identificador y dirección MAC de red, estableciendo esos valores en las variables destinadas al efecto, en el PIB de PHY.

4. Se inicia la nueva red, notificándolo a la capa superior.

El proceso para unir un dispositivo nuevo a la red es algo más complicado y no lo vamos a detallar tanto. Tan sólo comentaremos algunas ideas.

Básicamente, el dispositivo que permite que otro nuevo se anexe a la red tiene que ser o bien un coordinador de red, o bien un router. Éstos, tienen que recibir la petición superior que permite solicitar acceder a formar parte de la red, estableciendo un tiempo durante el cual va ser posible unirse a la red. A la hora de unirse un dispositivo se establecerá una relación parental, en la que el dispositivo que permite la entrada en la red se considerará "padre", y el que va a acceder a ella, "hijo". Para llevar a cabo este efecto, se tienen dos posibilidades, o bien se deja al procedimiento de asociación de la capa MAC, o bien se dispone de un procedimiento ya marcado mediante el cual el dispositivo hijo se añade automáticamente a la red a través de un padre prefijado. En ésta última posibilidad, la dirección del hijo se encuentra también prefijada, y se tiene que comprobar que no se está usando ya, es decir, comprobar que el dispositivo no exista previamente en la red. También cabe la posibilidad de hacer uniones de dispositivos que antes habían pertenecido a la red, por medio de funciones NWK en lugar de las MAC. La ventaja de ello es que se evitan hacer comprobaciones de direcciones y seguridad, y aun estando el padre en no disposición de admitir nuevos miembros, al ser uno conocido, tiene acceso.

En cuanto a las maneras de dejar una red, es decir, salirse de ella, podemos decir que básicamente son dos: o bien es el propio hijo el que solicita salirse, o bien el padre el que lo expulsa.

No vamos a entrar más en detalles de estas materias, si se tiene interés en esto bastará con ir a la Especificación de ZigBee que publica la ZigBee Alliance en su website.

A.3.2 Tabla de Vecinos

Una Tabla de Vecinos (Neighbor Table) es aquella de que disponen los dispositivos de la red para almacenar información relativa a otros dispositivos cercanos con los que se puede comunicar. Tiene dos usos fundamentales: uno es aquél del que se valen los dispositivos que quieren descubrir o volver a unirse a una red, teniendo apuntados los posibles padres que pueda tener, y otro es aquél que una vez en la red utiliza el dispositivo para almacenar relaciones y estado del enlace entre los vecinos. La tabla puede actualizarse con cada trama que se reciba del correspondiente vecino.

Algunos de los campos que tiene son:

- Dirección extendida del dispositivo,
- Dirección de red,
- Tipo de dispositivo,
- Si mantiene el receptor siempre encendido,
- La relación que tiene conmigo (padre-hijo, etc.),
- Si ha habido fallos de transmisión previos hacia ese dispositivo,
- La calidad del enlace (LQI),
- …

A.3.3 Encaminamiento (routing)

El encaminamiento o “routing” es una capacidad intrínseca de coordinadores de red y routers. Éstos, proveen de las siguientes funcionalidades a la red:

- Retransmitir tramas de datos para las capas superiores o para otros routers,
- Iniciar y participar en el descubrimiento de la ruta para establecer rutas para las consiguientes tramas, o rutas para dispositivos finales,

- Iniciar y participar en la reparación de una ruta de un extremo a otro, o en la de una ruta local,

- Emplear la métrica del coste del camino ZigBee, y

- Mantener tablas de enrutamiento para recordar las mejores rutas disponibles.

Como se ha mencionado en uno de los puntos, se hace uso de una métrica de coste de caminos, para poder comparar rutas durante su descubrimiento y mantenimiento. El coste recibe el nombre de coste del enlace, y está asociado con cada uno de los enlaces del camino, así como que estos costes son acumulables al hablar del camino completo, es decir, el coste de un camino es la suma de los costes de los enlaces que lo forman. Como curiosidad, el coste del enlace es una función que presenta esta pinta:

$$C\{l\} = \begin{cases} 7, \\ \min\left(7, round\left(\frac{1}{p_l^4}\right)\right) \end{cases}$$

Donde l es el enlace, y p_l es la probabilidad de envío de paquete para dicho enlace, y se deja a elección del desarrollador del stack que use el valor de 7 ó el resultado de la función *mínimo*, si quiere tener en cuenta estas probabilidades. La forma de estimar p_l es mediante las medidas de calidad del enlace LQI obtenidas en cada trama.

Para llevar a cabo los encaminamientos, coordinadores y routers pueden tener alojados en sus memorias un par de tablas, que reciben el nombre de Tablas de Encaminamiento y Descubrimiento de la Ruta (Routing Table y Route Discovery Table), y el hecho de tener la primera habilita a tener la segunda. Mediante estas tablas, el algoritmo de encaminamiento es capaz de obtener las rutas apropiadas para los paquetes que van a través de la red.

La Tabla de Encaminamiento presenta los siguientes campos:

- Dirección del destino: dirección de red o ID de la ruta; si el destino es un coordinador o router, se indica su dirección, pero si es un dispositivo terminal, se indica la dirección del padre.

- Estado de la ruta: activa/inactiva, descubrimiento en proceso, descubrimiento fallido, validación en proceso.
- Muchos-a-uno: indica si el destino es un concentrador que lanzó una petición de ruta de muchos-a-uno.
- Requerimiento de registro de ruta: indica que una trama comando de registro de ruta se ha de enviar antes de la siguiente de datos.
- Flag de GroupID: indica si la dirección destino es una GroupID.
- Dirección del próximo salto.

La de Descubrimiento de Ruta, estos otros:

- ID de petición de ruta: se incrementa con cada petición.
- Dirección fuente del que inició la petición.
- Dirección del remitente que ha reportado menor coste para esta ID de petición de ruta y dispositivo fuente.
- Coste de reenvío: el coste del camino acumulado desde la fuente hasta el actual.
- Coste residual: el coste del actual hasta el final.
- Tiempo de expiración: milisegundos restantes hasta el final del descubrimiento de la ruta.

Una vez comentados los campos que forman estas tablas, vamos a indicar el algoritmo genérico que se usa para el enrutamiento, mediante un diagrama de flujo explicativo, que se puede ver en la Fig. 50. No es tan detallado como los campos que hemos descrito de las tablas de las que hace uso, pero naturalmente en cada paso del algoritmo que se ha visto, se hace uso de las entradas correspondientes de las tablas en cada caso concreto de trama entrante en el router o coordinador. Es bastante gráfico así que no vamos a explicarlo con más detalle, tan sólo comentar la posibilidad del uso de broadcasting y multicasting.

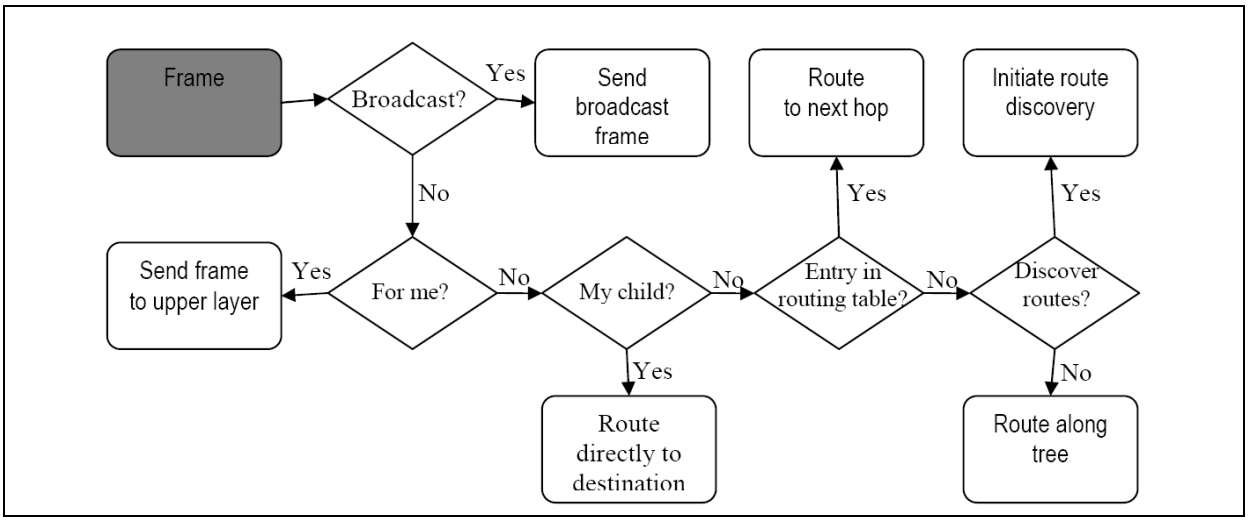


Fig. 50: Algoritmo de enrutamiento

A.3.4 NPDU: Paquetes en la capa de red

El formato general que presenta un paquete de NWK es el siguiente:

Octets: 2	**2**	**2**	**1**	**1**	**0/8**	**0/8**	**0/1**	**Variable**	**Variable**
Frame control	Destination address	Source address	Radius	Sequence number	Destination IEEE Address	Source IEEE Address	Multicast control	Source route subframe	Frame payload
NWK Header									Payload

Fig. 51: NPDU (Unidad de datos en NWK)

Como se ve en la figura, un NPDU consta básicamente de cabecera y carga útil (el PDU de la capa superior). Vamos a comentar por encima los campos que agrega la cabecera:

- Control de trama (Frame Control), de 16 bits, con información del tipo de trama, versión del protocolo, operaciones para el descubrimiento de ruta, y flags tales como de multicast, de seguridad, y de presencia en la cabecera de la ruta fuente y de las direcciones destino y fuente.

- Dirección Destino (Destination Address), con la dirección del dispositivo destino o broadcast si el flag de multicast está desactivado, y en caso contrario, el campo guarda el Group ID del grupo multicast.

- Dirección Fuente (Source Address).

- Radio (Radius), especifica el rango del radio de transmisión, y se decrementa una unidad por cada dispositivo que atraviesa en camino al destino.

- Número de Secuencia (Sequence Number), se incrementa en uno a cada nueva trama transmitida, y junto con la dirección fuente identifica a esta trama como única.

- Dirección IEEE Destino (Destination IEEE Address), con la dirección completa de 64 bits del destino eventual de la trama, siempre que el flag de control esté activo.

- Dirección IEEE Fuente (Source IEEE Address), con la direccio´n completa de 64 bits de la fuente de la trama, siempre que el flag de control esté activo.

- Control de Multicast (Multicast Control), se compone de 3 subcampos que se ocupan de permitir que dispositivos que no pertenezcan al grupo puedan o no enviar tramas a éste (Multicast Mode), el rango o número de intermediarios que no son miembros del grupo de destino (NonmemberRadius), y el valor máximo para este mismo campo (MaxNonmemberRadius).

- Ruta Fuente de la Subtrama (Source Route Subframe), formado por 3 subcampos y siempre que el flag en el campo de control esté activo, indica la ruta completa que debe seguir el paquete, mediante una lista de retransmisiones (Relay List) con las direcciones cortas (16 bits) de los nodos que retransmitirán la trama, un puntero al siguiente de la lista (Relay Index), y un contador de retransmisiones (Relay Count) que indica el total de elementos de la lista.

Aparte de este formato de trama general podemos tener un par de ellos particulares, como son el de una trama de datos y una trama de comando, los ilustramos en la Fig. 52. La idea de los campos es la misma que en una trama general como la que hemos visto antes, entendiendo que los campos de encaminamiento (Routing Fields) son los mismos campos que van tras el de control en la trama general, y el resto de los campos que vemos en la figura hablan por sí solos de su contenido.

[A]

Octets: 2	Variable	Variable
Frame control	Routing fields	Data payload
NWK header		NWK payload

[B]

Octets: 2	Variable	1	Variable
Frame control	Routing fields	NWK command identifier	NWK command payload
NWK header		NWK payload	

Fig. 52: NPDU de datos (A) y de comando (B)

A.3.5 Servicios de NWK

Al igual que las capas previas, se definen en esta capa dos servicios, análogos a los de éstas. Así pues, tenemos un servicio de transporte de datos entre capas (Network Layer Data Entity, NLDE) y otro de gestión de la capa (Network Layer Management Entity, NLME). Los puntos de acceso desde la capa superior se llaman también de la misma forma, NLDE-SAP y NLME-SAP. Además, existe un interfaz común interno entre ambos servicios que permite al NLME usar el NLDE.

El servicio de datos de NWK, a través de su SAP, permite el transporte de APDUs entre un par de entidades de aplicación, es decir, construye NPDUs para dichos APDUs, y accede a la capa MAC mediante el MCPS-SAP y pasando como parámetro el recién creado NPDU. A su vez, también es capaz de desmontar un PDU que le devuelve el servicio subyacente en MAC (MCPS) para obtener el APDU que reportará a la capa superior si todo es correcto.

El servicio de gestión de la capa es más complejo e incluye el resto de funcionalidades que hemos ido hablando en el apartado de esta capa del stack. También es accesible desde la capa superior, a la que ofrece las funciones que enumeramos a continuación:

- Descubrimiento de redes activas,
- Formación de red en caso de ser coordinador,
- Permiso para unirse a la red a otro dispositivo que lo solicita,

- Iniciar enrutamiento, en caso de ser un router que acaba de unirse a la red, así como poner en orden su configuración de supertrama,
- Peticiones para unirse un dispositivo a la red, tanto de forma normal mediante asociación, como unirse directamente, o volver a unirse (re-join) si estaba huérfano,
- Petición de un router o coordinador para que se una directamente a la red otro dispositivo,
- Dejar un dispositivo la red, u obligar a que un nodo hijo lo haga,
- Reset a los valores por defecto de NWK,
- Sincronización del dispositivo con un coordinador o router del que espera datos pendientes,
- Obtener y asignar datos a las variables de NIB (NWK Information Base),
- Reportar errores en el enrutado de una trama a la capa superior, e
- Iniciar descubrimiento de rutas unicast, multicast o muchos-a-uno.

Una vez expuestas las funciones de los servicios de NWK, podemos introducir la siguiente capa que hace uso de ellos.

A.4 Capa de aplicación (APL)

Esta capa es bastante más compleja que las anteriores, no en vano, se compone de tres subcapas, cada una con un buen abanico de funciones. Dichas subcapas son la de soporte de aplicación (APS), los objetos de dispositivo ZigBee (ZDO) y el framework de aplicación (AF). Vamos a comentarlas por separado.

A.4.1 Subcapa de soporte de aplicación (APS)

Según el orden que hemos ido dando en las anteriores capas, ésta sería la continuación natural en la explicación de la arquitectura ZigBee. Queda situada lógica y funcionalmente entre la capa de red NWK y el framework de aplicación, donde residen los objetos de la aplicación, así como del ZDO. Así pues, la subcapa

APS básicamente surte de servicios al framework de aplicación y al ZDO. Las responsabilidades que tiene se pueden resumir en estas:

- Mantener tablas para la asociación ("binding") o relación entre dos dispositivos basada en sus servicios y necesidades (Binding Table),
- Reenviar mensajes entre dispositivos asociados,
- Definición de dirección de grupo, eliminación y filtrado de mensajes dirigidos a un grupo,
- Mapeo de direcciones desde el formato de 64 bits del IEEE hasta el de 16 bits de NWK, o viceversa, y
- Fragmentación, reensamblado y transporte de datos fiable.

Explicaremos brevemente en qué consiste la asociación entre dos dispositivos, el direccionamiento de grupos y las posibilidades para transmitir mensajes, así como recibirlos:

A.4.1.1 Funciones de APS

La función más destacable de APS quizá pueda ser la asociación o "binding" de dispositivos, y es propia de coordinadores o dispositivos especiales de red. Dicha función es el camino de comunicación a nivel superior en el stack entre dos aplicaciones ZigBee de diferentes dispositivos, es decir, la relación el punto desde el que ha de salir el mensaje y el punto remoto que lo va a recibir. Lo podemos definir algo más formalmente como una relación entre un par de endpoints (puntos de acceso desde el framework de aplicación) correspondientes al dispositivo origen y al destino. Para almacenar todas estas relaciones, se hace uso de una tabla en memoria, llamada Tabla de Asociación (Binding Table), cuyas filas poseen la siguiente información:

$$(a_s, e_s, c_s) = \{ (a_{d1} [, e_{d1}]), (a_{d2} [, e_{d2}]) \dots (a_{dn} [, e_{dn}]) \}$$

Donde (a_s, e_s, c_s) representan respectivamente la dirección, endpoint y cluster del dispositivo fuente, y los pares $(a_{di} [, e_{di}])$ la dirección y endpoint (opcional) del destino, que están asociados al endpoint que localizamos mediante los índices de la parte izquierda de la igualdad.

El direccionamiento de grupo es otra forma de asociación no tan básica como el binding, pero muy funcional. La idea es relacionar mediante una Tabla de Grupo (Group Table) identificadores de grupo y endpoints de diferentes dispositivos, de forma que al dirigir mensajes a un grupo concreto, automáticamente se dirijan a todos aquellos endpoints que están suscritos a él.

Otra importante función es la transmisión propia de los datos. Desde APS se preparan ya las transmisiones, con todos los parámetros necesarios para el envío, como son destinatario, forma de transmisión y los datos en sí entre otros, pasando ya todo eso bien preparado a capas inferiores. Para hacer una transmisión, la primera condición que se requiere es que el dispositivo forme parte de alguna red, en cuyo caso contrario, APS descarta el mensaje y reporta el error correspondiente.

Existen varias formas de transmisión, según el direccionamiento. Así pues, tenemos transmisiones directas, indirectas o de direccionadas a un grupo. Por directas entendemos que son dirigidas directamente al destino, sin pasar por ningún dispositivo intermedio, mientras que las indirectas son aquellas que pasan por el coordinador de red como punto intermedio.

Para las directas se indican los endpoints fuente y destino, mientras que para las direccionadas a grupo se indica simplemente el endpoint fuente y la dirección del grupo, de la que se obtienen todos los endpoints destino.

Para las indirectas, tan sólo hay que indicar endpoint fuente o destino, debido a que el otro se consulta a partir del dado en la Tabla de Asociación. Si la Tabla de Asociación no reside en el dispositivo que está transmitiendo, éste ha de transmitir al coordinador de red, que consultando en la tabla sabe cuál es el destino de ese mensaje y lo redirige hacia él. En caso de tener el propio dispositivo una Tabla de Asociación, bastará hacer una transmisión directa consultando la entrada de la tabla correspondiente.

En caso de la recepción de mensajes, APS se encarga de filtrar hacia arriba todos aquellos mensajes que reporta satisfactoriamente NWK pero que no son interesantes en la capa superior. En caso de ser un mensaje válido, se estudia si viene con ambos endpoints, en cuyo caso se sabe que es una transmisión directa y puede darse paso a través del endpoint de destino correspondiente. Si además, el

endpoint indicado como destino fuese el de broadcast de aplicación, se ofrecerá el mensaje a todos los endpoints no reservados que estén activos.

En caso de recibirse un mensaje con únicamente el endpoint fuente, se asume que la transmisión era indirecta, y dependiendo si el receptor es coordinador o no, se procesará o descartará. Si no lo es, la trama se descarta, y si lo es, busca en la Tabla de Asociación la correspondiente entrada, para ver si puede hacer el reenvío del mensaje o descartarlo si no hallara entrada.

Además, se puede hacer uso en este nivel de mensajes de tipo "Acknowledgement" (ACK), que sirven para garantizar al emisor que el receptor ha recibido correctamente su mensaje, así como de retransmisiones de mensajes, esto es, si esperando un ACK ha pasado ya un tiempo máximo de espera, se vuelve a enviar el mensaje, y así un número determinado de veces hasta que se dé por perdido el receptor.

A.4.1.2 APDU: Paquetes de la capa APS

Un paquete de datos a nivel APS tiene la siguiente pinta en general:

<table>
<tr><td>Octets:
1</td><td>0/1</td><td>0/2</td><td>0/2</td><td>0/2</td><td>0/1</td><td>1</td><td>Variable</td></tr>
<tr><td rowspan="2">Frame control</td><td>Destination endpoint</td><td>Group address</td><td>Cluster identifier</td><td>Profile Identifier</td><td>Source endpoint</td><td rowspan="2">APS counter</td><td rowspan="2">Frame payload</td></tr>
<tr><td colspan="5">Addressing fields</td></tr>
<tr><td colspan="7">APS header</td><td>APS payload</td></tr>
</table>

Fig. 53: APDU (Unidad de Datos en APS)

Básicamente, podemos distinguir los siguientes campos en un APDU cualquiera: una cabecera, y la carga útil (payload), con la información específica del tipo de trama que sea. La cabecera incluye un par de campos de control como son el control de trama (información de tipo de trama y opciones de direccionamiento) y el contador APS (para evitar recibir duplicados), y a continuación campos de direcciones e identificadores, entre los que se pueden distinguir endpoints fuente y destino, ID de grupo, ID de cluster, e ID de perfil.

Ya en particular, los tipos de trama que tenemos son los de datos, comando y ACK. La trama de datos presenta el mismo formato que la general, y las de comando y ACK presentan este otro aspecto más reducido:

[A]

Octets: 1	0/2	1	1	Variable
Frame control	Group Address	APS counter	APS command identifier	APS command payload
APS header			APS payload	

[B]

Octets: 1	0/1	2	2	0/ 1	1
Frame control	Destination endpoint	Cluster Identifier	Profile identifier	Source endpoint	APS counter
APS header					

Fig. 54: APDU de comando (A) y de ACK (B)

No vamos a comentar todos los campos, pues básicamente son los mismos, tan sólo destacar la no presencia de payload en la trama de ACK (no hace falta para nada), y la composición de un comando, que es un ID y carga útil asociada a él. Para conocer los valores específicos de cada campo y su significado, remitimos al lector una vez más a la hoja de especificaciones de la ZigBee Alliance.

A.4.1.3 Servicios de APS

Al igual que las capas inferiores, APS incorpora un par de servicios totalmente análogos a los de ellas. Por un lado, un servicio de datos (APSDE, APS Data Entity), y por otro lado, un gestor de la capa (APSME, APS Management Entity). Los puntos de acceso a APSDE son un total de 241, técnicamente llamados APSDE-SAP, más comúnmente conocidos como endpoints, numerados desde el 0 hasta el 240, y siendo el 0 de acceso para la subcapa ZDO, y el resto se reparten uno cada uno para los 240 posibles objetos del framework de aplicación. El servicio de gestión APSME sólo ofrece un punto de acceso, el APSME-SAP, dirigido también al ZDO.

La función principal del servicio de datos es la que se le supone al llegar a este punto de lectura, esto es, la de empaquetar o desempaquetar las tramas que vayan desde los objetos de aplicación hacia NWK o vengan desde NWK y se reportan libres de cabeceras APS al framework. También se ocupa del filtrado de direcciones de

grupo, de rechazar duplicados, de mantener un transporte fiable gracias empleo de reintentos end-to-end (comunicación entre las capas APS de los diferentes dispositivos), así como el ya nombrado binding o asociación de dispositivos, para transferir un mensaje remitido por un determinado dispositivo y endpoint al endpoint correspondiente de su APS o viceversa.

El servicio de gestión es algo más complicado, debido a que aglutina más funciones. Permite principalmente a una aplicación comunicarse con el stack subyacente de red, pero incorpora otras importantes funciones como la gestión del binding, la de la Base de Información de APS (AIB), de grupos (declarar direcciones únicas para ser compartida por varios dispositivos, así como añadir o quitar éstos del grupo), y uso de claves de seguridad para el cifrado de los mensajes. Tan sólo quedaría apuntar que lógicamente, es en el servicio de gestión de APS donde reside la ya mencionada Tabla de Asociación (Binding Table), ya que sobre ella se realizan las funciones de asociación y desasociación de endpoints y dispositivos, así como la Tabla de Grupo.

Ahora podemos ya pasar a ver la siguiente subcapa dentro de APL, en nuestro camino hacia arriba en la arquitectura.

A.4.2 ZigBee Device Objects (ZDO)

Esta subcapa está situada justo encima de APS, a la que accede mediante el endpoint #0 (APSDE-SAP #0 especial) y el APSME-SAP, así como de NWK, con la que se comunica a través del NLME-SAP, así como con el framework de aplicación situado también encima de APS, a través del llamado interfaz público del ZDO que luego veremos. Así pues, es una subcapa que presenta varias relaciones con otras subentidades de la arquitectura.

Las responsabilidades que tiene entre otras, son:

- Definir el rol que juega el dispositivo dentro de la red (coordinador, dispositivo final, …),
- Descubrir dispositivos en la red y determinar qué servicios de aplicación ofrecen,
- Inicializar y/o responder peticiones de binding, y

- Establecer una relación segura entre dos dispositivos de la red.

Además, el ZDO inicializa las capas APS, NWK, el SSP (Security Service Provider), y cualquier otra capa ZigBee a excepción de las aplicaciones finales que residen tras los endpoints 1-240.

El ZDO está formado por cinco objetos:

- Descubrimiento de Dispositivos y Servicios (Device and Service Discovery),
- Gestor de Red (Network Manager),
- Gestor de Asociación (Binding Manager),
- Gestor de Seguridad (Security Manager), y
- Gestor de Nodo (Node Manager).

Describiremos las características relevantes o descripción de cada uno de ellos siguiendo el orden con el que los hemos enumerado. Pero antes de comenzar con el primero de ellos, definiremos lo que se conoce como "Caché Primaria de Descubrimiento".

En toda red ZigBee hay uno o más dispositivos (coordinadores o routers) que actúan como "Caché Primaria", una máquina de estados servidora que ofrece a sus clientes (otros dispositivos) la posibilidad de registrarse en una tabla con información suya relativa a posibles peticiones de descubrimiento de otros. La "Caché Primaria" es importante de cara a los dispositivos finales que pueden estar en estado de suspensión (sleeping end devices), ya que pueden registrar su información para otros que puedan estar interesados, mientras que ellos mismos no pueden proporcionarla directamente. En la Fig. 55 se muestra el funcionamiento de la mencionada "Caché Primaria".

El objeto de Descubrimiento de Dispositivos y Servicios varía un poco en sus funciones dependiendo si un dispositivo es además una "Caché Primaria" o no, pero básicamente se ocupa de lo que indica su nombre, es decir, ofrecer servicios relativos a hallar otros dispositivos y servicios que estos ofrezcan dentro de la WPAN. Si actúa como "Caché Primaria" podrán proporcionar datos de otros dispositivos registrados, si no, únicamente los suyos. Las peticiones que se le hacen

a este objeto son para obtener sus direcciones de red o sus dispositivos asociados, en caso de usar los servicios de "descubrir dispositivos", o de devolver endpoints y descriptores asociados en caso de usar los de "descubrir servicios".

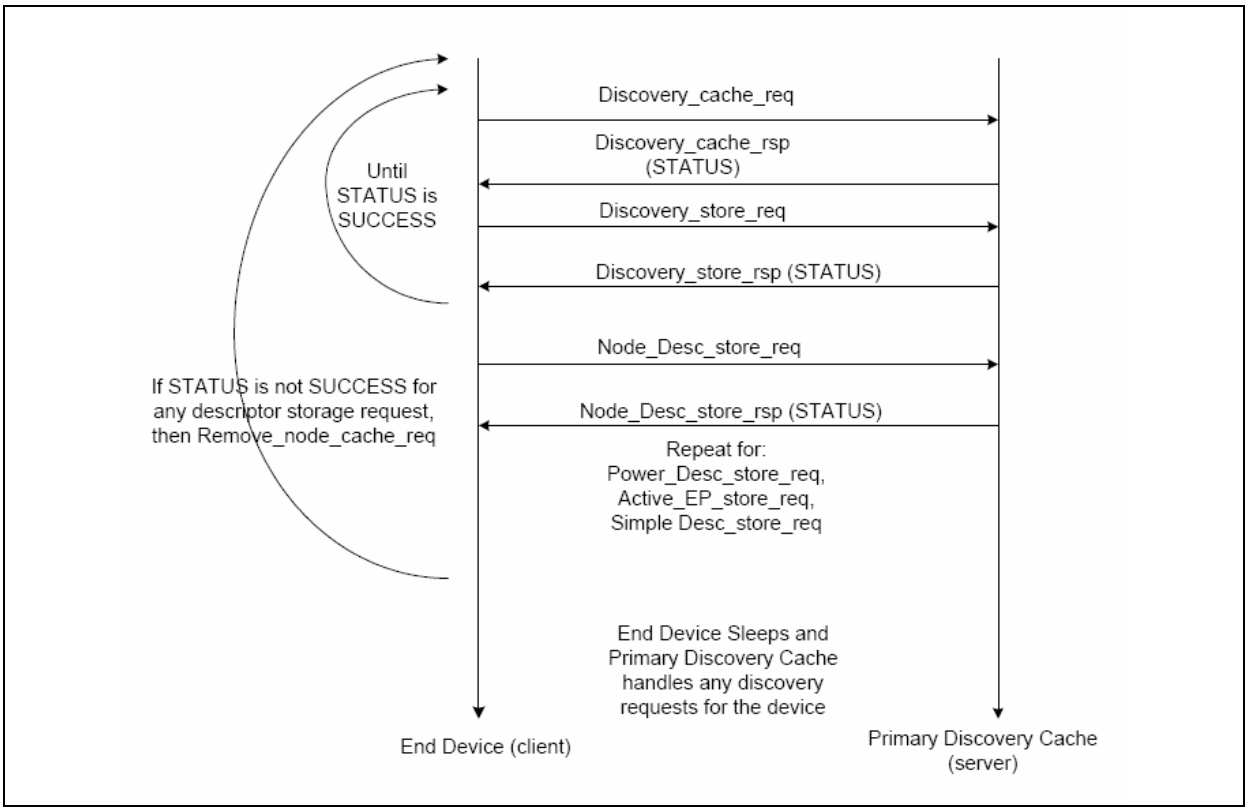


Fig. 55: Máquina de estados de la Caché Primaria de Descubrimiento

El segundo de los objetos, el Gestor de Red, es el encargado de definir el rol en la red del dispositivo. Es decir, si va a ser un coordinador, router, o un dispositivo final. Dicho rol, está parcialmente predefinido de antemano, aunque en el caso de un supuesto router que inicie ejecución y vea que no hay ninguna red formada en su radio de acción, puede asumir el papel de coordinador. Las funciones que ofrece este objeto son claramente diferentes en cada caso, así un coordinador es capaz de hacer un escaneo de canales o mandar iniciar una red en un canal, mientras que un dispositivo final tiene capacidades de solicitar unirse a una red, o reunirse (rejoin) si previamente formó parte de ella.

El Gestor de Asociación en primer lugar define el tamaño de los recursos para la Tabla de Asociación de la subcapa APS, pero además recibe las peticiones de asociación y desasociación de otros dispositivos (que darán lugar a cambios en la

tabla), así como ofrecer una asociación manual para un dispositivo final en caso de ser un coordinador, mediante la pulsación de un botón o similar.

El cuarto de los objetos es el Gestor de Seguridad, y éste es quien se ocupa de las labores de seguridad en el protocolo, empezando por tenerla activa o desactiva. Si está activa, realiza las siguientes funciones: establecer la clave, transportarla, pedirla, actualizar un dispositivo, quitar un dispositivo, y cambiar la clave. El Gestor de Seguridad organiza los Servicios de Seguridad que se encuentran en cada capa que los ofrezca, se pone en contacto con el llamado Centro de Confianza (Trust Center) alojado en el coordinador, para obtener la clave de red (NWK Key) mediante primitivas de establecimiento, transporte, etc., y haciendo uso de otras claves intermedias (Master Key y Link Key).

El último de los objetos del ZDO es el Gestor de Nodo, y está destinado a coordinadores de red y routers. Sus funciones son permitir comandos de gestión remota relativos al descubrimiento de una red, así como proveer otros para recuperar la tabla de enrutamiento, la de asociación (binding), o el LQI (indicador de calidad del enlace) de los vecinos de un dispositivo remoto. También se ocupa de permitir que haya dispositivos que tengan su propia tabla de asociación o de realizar operaciones de backup sobre determinadas entradas de dicha tabla.

A.4.3 Application Framework

El Application Framework se podría definir como un entorno donde están ubicados los objetos de aplicación del dispositivo ZigBee. Es lo que podríamos llamar un conjunto de objetos de aplicación definidos por el fabricante, e implementados por el usuario. Además de los objetos que implementa el usuario, por un lado tenemos perfiles, por otro, descriptores. De los perfiles de aplicación ya hemos hablado en el apartado 3.2.1.3 de esta memoria, dónde también se mencionaban elementos de éstos como son los clusters y atributos, accedidos por endpoints. Así que vamos a comentar brevemente en este apartado qué son los descriptores y para qué sirven.

Los descriptores son estructuras de datos que almacenan información del dispositivo ZigBee que sirven para describirse a sí mismo frente a otros. Hay un total de cinco descriptores: nodo, energía del nodo, sencillo, complejo, y usuario.

El descriptor de nodo contiene información de las capacidades del nodo. Es único y obligatorio. Entre esa información se encuentra:

- Tipo lógico: tipo de dispositivo del nodo ZigBee (coordinador, router, dispositivo final).
- Descriptor complejo disponible: si está o no disponible.
- Flags APS: capacidades de APS.
- Banda de Frecuencia: la banda elegida para transmitir (explicadas en PHY, apartado A.1.1).
- Flags de capacidades de MAC: si puede ser un coordinador, si el receptor ha de estar encendido en modo suspendido, si se usan las directivas de seguridad, …
- Código del fabricante: proporcionado por la ZigBee Alliance.
- Tamaño máximo del buffer: tamaño del APDU para el nodo.
- Máscara de servidor: para obtener información relativa a capacidades de servidor en el nodo (Centro de Confianza, Tabla de Asociación, o Caché de Descubrimiento)

El descriptor de energía del nodo también es único y obligatorio por nodo, y ofrece una indicación del estado de energía del nodo. Más en detalle, ofrece:

- El modo de energía actual, es decir, el tipo de modo de ahorro de energía empleado por el nodo.
- Las fuentes de energía disponibles: red eléctrica o bien batería recargable o desechable.
- La fuente de energía actual: cualquiera de las anteriores.
- El nivel de energía actual: crítico, 33%, 66%, 100%.

El descriptor sencillo es también obligatorio, pero no único, hay uno por cada endpoint usado del nodo, del que contiene información específica (su número, perfil, dispositivo, versión, cluster, …)

El descriptor complejo ya es opcional en la implementación del stack, y lleva información extendida de cada descripción del dispositivo contenida en el nodo, haciendo uso de etiquetas XML comprimidas. Entre esas informaciones, encontramos el juego de caracteres empleado, nombre del fabricante, modelo, número de serie, URL, icono, …

Finalmente, el descriptor de usuario, también opcional, contiene información que permite al usuario identificar el dispositivo mediante una cadena de caracteres familiar a él, como por ejemplo “Luces de la escalera” o “Encendido de la tele”.

Con los descriptores se completa toda la información del Framework donde desarrolla el fabricante, y de la que ya habíamos comentado parte en el apartado 3.2.1.3 de esta memoria.

Anexo B: MANUAL DE CARGA DE FIRMWARE EN LOS PICS

En este anexo se pretende explicar qué herramientas y qué pasos hay que seguir para cargar el firmware en los microcontroladores. Se entiende que en este punto se dispone ya de una versión compilada del proyecto MPLAB para un microcontrolador en concreto, en un único fichero HEX.

En primer lugar se va a hacer una enumeración de las herramientas usadas para llevar a cabo estos propósitos:

- Microchip MPLAB® IDE v.7.50
- Microchip MPLAB® ICD2
- Microchip MPLAB® C18 C Compiler, Student Edition v.3.2

A continuación se distingue entre carga en los microcontroladores del PICDEM Z o carga en los de los equipos OEM.

B.1 Carga en PICDEM Z

La carga en el 4620 del PICDEM Z, haciendo uso del ICD2 es más bien sencilla. Suponiendo que tenemos las fuentes completas, vamos a indicar las opciones de configuración de MPLAB relativas a la programación del PIC.

En el menú *Configure → Settings → Debugger,* además de tener ciertas opciones interesantes que cada cual pondrá a su gusto, se indica mediante una advertencia que hay que chequear una casilla en la ventana de configuración del ICD2 (Fig. 56). Dicha ventana se puede encontrar por igual en los menús *Debugger/Programmer → Settings,* una vez seleccionado ICD2 como dispositivo depurador o programador.

Para configurar el ICD2, se puede ejecutar un asistente paso a paso desde el menú *Debugger /Programmer → MPLAB ICD2 Setup Wizard,* una vez seleccionado el ICD2 como dispositivo depurador o programador. En dicho asistente se indica el puerto de conexión del ICD2 al PC (COM o USB), si el circuito se abastece de potencial desde el ICD2 o tiene su propia fuente, si se conecta automáticamente al abrir un proyecto o si se descarga automáticamente un sistema operativo para el ICD2 según el dispositivo que se va a programar (ver Fig. 57).

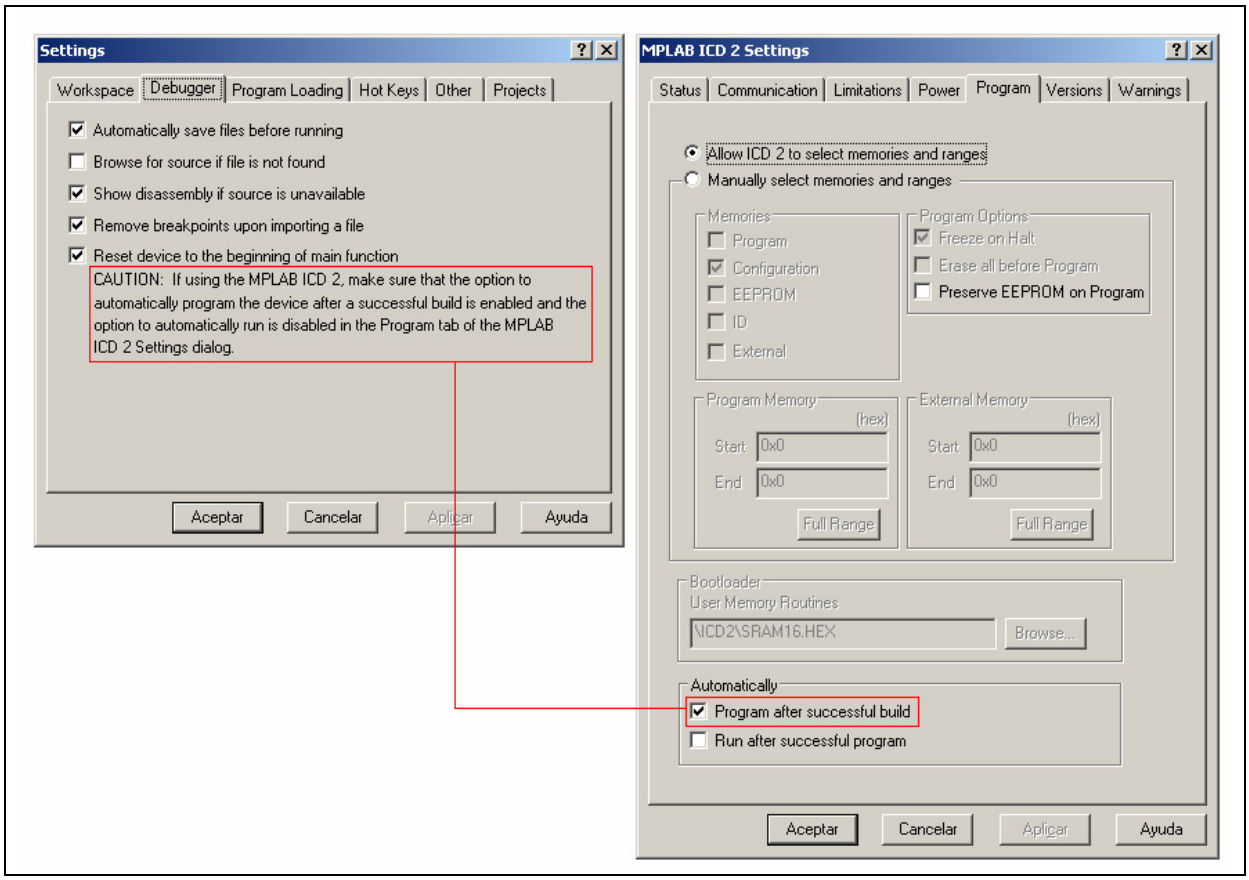


Fig. 56: Advertencia en ventana sobre ICD2

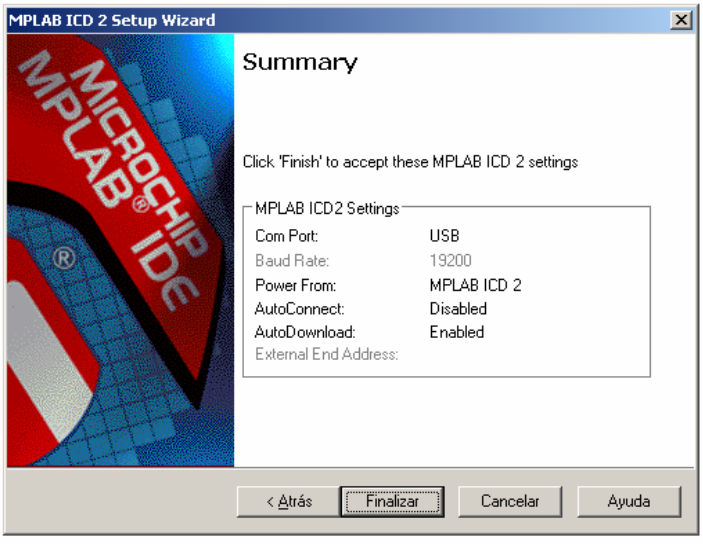


Fig. 57: Configuración típica del ICD2 tras ejecutar el asistente

La configuración del ICD2 se puede completar en su ventana de configuración, anteriormente mencionada. Una vez ya lo tenemos configurado, podemos proceder a la carga del firmware. Hay que ver varias cosas en este punto:

- Que la conexión entre el PC y el ICD2 sea la que se especificó en sus opciones de configuración (normalmente lo haremos por USB), así como que el ICD2 está conectado a la placa base del PICDEM Z, a través del interfaz RJ-12,

- Que la corriente que se abastece al circuito provenga del sitio convenido en las opciones de configuración del ICD2 (que suele ser desde el propio ICD2),

- Que no haya otra aplicación en uso del ICD2 (normalmente otra instancia de MPLAB),

- Si es la primera vez que se conecta al PICDEM Z, preguntará mediante un diálogo si se desea actualizar el sistema operativo del ICD2 para su uso con el PIC18LF4620.

Una vez que todo esté correcto al conectar el ICD2, ya se puede programar el microcontrolador. Para ello deberíamos ejecutar un *MAKE* (posiblemente con un *CLEAN* previo) o *REBUILD* del proyecto, que provoca el recompilado completo de todo el proyecto, generación de un fichero .HEX y su programación en el micro. Esto es válido, tanto en modo *Debugger* como en modo *Programmer*. Si no se realiza el borrado previo de ficheros intermedios con un *CLEAN* o *REBUILD*, y hemos hecho otras programaciones previas en esa sesión, el compilador no se "queja", el motivo de hacerlo es si las programaciones previas fueron en otras sesiones, momento en el que sí devuelve un error al compilar.

Sólo queda comentar, tras una correcta carga del firmware, que si el modo ha sido el de programación *(Programmer)* hay que desconectar el ICD2 de la placa base del PICDEM Z para poder ejecutar. En el modo de depuración *(Debugger)* es obvio que esto no hay que hacerlo puesto que es a través del ICD2 desde donde se dirige la depuración del microcontrolador.

B.2 Carga en dispositivos OEM

Este punto queda incompleto en esta versión de la memoria, debido a que no se han realizado satisfactoriamente las cargas del firmware en todos los dispositivos OEM. Indicar que mediante el ICD2 sí que se ha programado correctamente el microcontrolador PIC del módulo Pixie SO, y todo parece indicar que mediante esta vía también se podrá programar correctamente el UZBee, como ya se indicaba en el Desarrollo de la Solución OEM de la memoria.

Anexo C: MANUAL DE USO DE LA APLICACIÓN TELETEST ZB

En este manual se pretende dar una visión rápida y certera de las características que tiene esta aplicación y su funcionamiento.

El proceso de instalación de la aplicación es realmente simple e intuitivo, puesto que se realiza mediante un asistente-instalador. Tan sólo hay que indicar el directorio final de la aplicación en nuestro disco duro. Los requisitos de instalación para un correcto funcionamiento son:

- Sistema operativo: Windows 2000/XP.
- Procesador Pentium 4 o superior.
- Espacio de disco duro: 2 MB.
- 512 MB de memoria RAM.
- Framework .NET en su versión 2.0.
- Windows Installer.
- Microsoft Office 2003.

Al iniciar la aplicación, se observa la siguiente ventana principal:

Fig. 58: Ventana principal de TeleTest ZB

En ella se dispone de de un menú Archivo con las opciones fundamentales y un botón central de inicio. Se comentarán por orden de aparición dichas opciones.

Fig. 59: Menú Archivo

MENÚ ARCHIVO → NUEVO

Mediante esta opción la aplicación solicita al usuario el número de preguntas que quiere incluir en el nuevo archivo. Tras dar un valor comprendido entre 1 y 250, se abre una nueva instancia de Excel con un nuevo libro que lleva el formato requerido por la aplicación TeleTest ZB.

MENÚ ARCHIVO → MODIFICAR → AÑADIR PREGUNTAS

Añade preguntas a un fichero Excel ya creado.

MENÚ ARCHIVO → MODIFICAR → EDITAR

Abre una nueva instancia de Excel con un libro ya creado, y deja la posibilidad de modificar los datos existentes tales como Apellidos, Nombre, ID de usuario, Nota, Timeout, Texto y Respuesta Correcta.

MENÚ ARCHIVO → IMPORTAR

La aplicación solicita al usuario que seleccione el archivo de texto con formato CSV del Campus Virtual (WebCT) donde se tienen los datos de los alumnos, e indicando a continuación el número de preguntas que querrá incluir y un nuevo fichero XLS, importa todos esos datos al libro XLS con el que trabaja TeleTest ZB.

MENÚ ARCHIVO → EXPORTAR

Realiza la acción contraria a importar. Además, ofrece la posibilidad al usuario de abrir con el bloc de notas el fichero de texto en formato CSV generado.

MENÚ ARCHIVO → CERRAR ARCHIVO

Cierra el fichero XLS en uso y activa el botón iniciar.

MENÚ ARCHIVO → SALIR

Cierra la aplicación.

BOTÓN INICIAR

Muestra un diálogo de apertura de fichero para que el usuario seleccione el archivo con el que desea trabajar. Una vez realizada esta acción, el botón queda desactivado hasta el cierre del archivo. Al abrir un fichero, la ventana principal pasará al estado de icono de notificación en la bandeja del sistema de Windows.

INICIAR/PARAR DE RECEPCIÓN DE RESPUESTAS

El usuario podrá llevar a cabo estas acciones mediante dos alternativas:

1. Botón derecho del ratón sobre el icono de notificación, y seleccionar la opción *iniciar/parar*.

2. Pulsar la siguiente combinación de teclas: CTRL + F11 + F12.

Una vez parada la recepción de respuestas, se genera automáticamente la gráfica con los resultados obtenidos, y clicando con el botón derecho dentro de la gráfica mostrada, se ofrecen varias alternativas de visualización en un menú emergente. Para abandonar la gráfica basta con pulsar la opción *Salir* de dicho menú o pulsar la tecla F10.

BIBLIOGRAFÍA

[1] Microchip PIC18F2525/2620/4525/4620 Data Sheet (DS-39626b)

[2] Microchip PIC18F2455/2550/4455/4550 Data Sheet (DS-39632c)

[3] Microchip Stack for the ZigBee™ Protocol (AN965)

[4] Microchip Stack for ZigBee™ V1.0-2.0 April 29, 2005

[5] Microchip PICDEM™ Z Demonstration Kit User's Guide (DS-51524a)

[6] ZigBee Specification by ZigBee Alliance v.1.0 (December 14th, 2004)

[7] ZigBee Specification by ZigBee Alliance v.1.0 r13(December 1, 2006)

[8] IEEE Std 802.15.4™-2006 (Revision of IEEE Std 802.15.4-2003)

[9] Presentación PowerPoint: Wireless Sensors and Control Networks: Enabling New Opportunities with ZigBee, by Bob Heile (Zigbee Alliance)

[10] ZigBee/IEEE 802.15.4 Summary, by Sinem Coleri Ergen

[11] Presentación PowerPoint: ZigBee IEEE 802.15.4, by Yuan Yuxiang, Kuroda Lab. Department of Electrical Engineering - Keio University

[12] Flexipanel UZBee™ 2.4GHz IEEE 802.15.4 RF transceiver with PIC microcontroller USB interface (DS-492)

[13] Flexipanel Pixie™ PIC microcontroller with 2.4GHz IEEE 802.15.4 transceiver and ZigBee stack (DS-481)

[14] Flexipanel EasyBee™ 2.4GHz ZigBee ready IEEE 802.15.4 RF transceiver (DS-480)

[15] Flexipanel StarLite USB™ USB to IEEE 802.15.4 Transceiver for Star, Broadcast & Addressed Communications (DS-506)

[16] Microchip MPLAB® C18 C Compiler Libraries (DS-51297f)

[17] Microchip Dynamic Memory Allocation for the MPLAB® C18 C Compiler (DS-00914a)

[18] Microchip MPLAB® C18 C Compiler User's Guide (DS-51288j)

[19] Microchip MPLAB IDE Quick Start Guide (DS-51281f)

[20] Microchip MPLAB IDE User's Guide (DS-51519b)

[21] Microchip MPLAB ICD2 User's Guide (DS-51331b)

ENLACES

[1] www.msdn.es

[2] http://www.vanderbilt.edu/cft/resources/teaching_resources/technology/crs.htm

[3] http://he-cda.wiley.com/WileyCDA/Section/id-103701.html

[4] http://sharepoint.cisat.jmu.edu/tsec/jim/CRS/default.htm

[5] http://www.h-itt.com/

[6] http://www.iclicker.com/

[7] http://www.educlick.es/