DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE UN PROTOCOLO DE ENCAMINAMIENTO GEOGRÁFICO EN UNA RED DE SENSORES INALÁMBRICOS

Autor: Miriam López Alvarez

Tutor: Manuel Urueña Pascual

Director: Ángel Cuevas Rumín

Leganés, febrero de 2012

Agradecimientos

Agradezco a mis tutores Manuel Urueña Pascual y Ángel Cuevas Rumín, por acompañarme a lo largo de mi aprendizaje y prestarme su ayuda siempre que la he necesitado. Sin ellos, no hubiera sido posible llegar hasta aquí. Gracias.

También me gustaría dar las gracias a todos aquellos amigos, compañeros de laboratorio, compañeros de trabajo y familiares que me han dado su apoyo y sus consejos, sobre todo en la recta final que es cuando más los he necesitado. Gracias a todos.

Resumen

En este proyecto fin de carrera se desarrolla una pila de protocolos basada en encaminamiento geográfico para una red de sensores inalámbricos IEEE 802.15.4.

Las redes de sensores inalámbricos están limitadas en recursos y consumo de energía. Para mitigar estos inconvenientes, debe optimizarse el envío de mensajes entre los distintos nodos que forman la red. Para ello, se utilizan protocolos de encaminamiento geográfico específicos.

El objetivo de este proyecto es implementar un protocolo de encaminamiento geográfico, *Greedy Forwarding*, que optimice el envío de mensajes entre los nodos, disminuyendo la utilización de recursos de memoria y ancho de banda y el consumo de energía.

El encaminamiento geográfico se realiza a nivel de red. Para el nivel de aplicación, se ha implementado una aplicación de ejemplo para evaluar el correcto funcionamiento del encaminamiento geográfico de forma sencilla. Se utilizan las capas física y de enlace definidas en el estándar IEEE 802.15.4, que permiten la comunicación entre los dispositivos de una red inalámbrica.

Por último, se ha desplegado una red de sensores inalámbricos real para comprobar el correcto funcionamiento del protocolo de encaminamiento geográfico implementado.

Palabras clave: red de sensores inalámbricos, encaminamiento geográfico, *Greedy Forwarding*, IEEE 802.15.4.

Abstract

In this final project a protocol stack based on geographic routing is developed for an IEEE 802.15.4 wireless sensor network.

Wireless sensor networks are limited in resources and power consumption. To mitigate these drawbacks, messages sending between different nodes in the network must be optimized. In order to do this, specific geographic routing protocols are used.

The aim of this project is to implement a geographic routing protocol, Greedy Forwarding, to optimize messages sending between nodes, reducing use of memory and bandwidth resources and power consumption.

Geographic routing is done at network layer. For application layer, an example application has been implemented to evaluate the correct operation of the geographic routing easily. Physical and link layers defined in the IEEE 802.15.4 standard are used to allow communication between the wireless network devices.

Finally, a real wireless sensor network has been deployed to check the correct operation of the implemented geographic routing protocol.

Keywords: wireless sensor network, geographic routing, Greedy Forwarding, IEEE 802.15.4.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción y objetivos

1.1 Introducción

En este capítulo se presentan las motivaciones de este proyecto fin de carrera, se explican los objetivos que se pretenden conseguir y se describe la estructura de la memoria.

1.2 Motivaciones

Las redes de sensores inalámbricos han experimentado un importante desarrollo durante los últimos años. Se utilizan en multitud de aplicaciones de diferentes ámbitos tales como la agricultura, la medicina, los procesos industriales, la detección de incendios, los estudios medioambientales y entornos militares, entre otros. Son una de las tecnologías más valoradas por los analistas dentro del ámbito de las tecnologías inalámbricas.

Este tipo de redes se utilizan porque permiten realizar un despliegue masivo de dispositivos y tienen una serie de ventajas como su pequeño tamaño, su bajo coste y su capacidad de operar sin necesidad de mantenimiento durante largos períodos de tiempo. Están especialmente indicadas para despliegues en zonas hostiles donde resulta difícil el acceso para recargar o sustituir las baterías y donde el mantenimiento es una tarea complicada. También son adecuadas para aplicaciones que requieren reducción de costes y de cableado. Sin embargo, cuando se trabaja con redes de sensores inalámbricos también hay que considerar sus

limitaciones. Los puntos más importantes a tener en cuenta son la optimización del consumo de energía y los recursos limitados de memoria, ancho de banda y cobertura.

Debido a que la comunicación supone uno de los mayores gastos de energía en la operación de los sensores, una forma de hacer frente a estos inconvenientes es la optimización del envío de mensajes entre los distintos nodos de la red. Para ello, se pueden utilizar protocolos de encaminamiento geográfico. Las ventajas de utilizar este tipo de protocolos de encaminamiento son que se necesita almacenar muy poca información en los nodos para realizar el encaminamiento de mensajes (lo que optimiza la utilización de los recursos de memoria), se reduce el número de mensajes enviados (que debido a las limitaciones de ancho de banda podrían llegar a saturar la red) y cada nodo tiene que hacer un esfuerzo menor para encaminar el paquete hacia el destino final porque se realizan comunicaciones multisalto, lo que supone un ahorro de energía.

Teniendo en cuenta lo expuesto, en este proyecto se desarrollará una pila de protocolos basada en encaminamiento geográfico con objeto de optimizar el consumo de recursos en una red de sensores inalámbricos real, mediante la optimización del envío de mensajes, para que cualquier aplicación pueda utilizar estas redes de una manera más eficiente, beneficiándose de todas sus ventajas e intentando disminuir los efectos de sus inconvenientes.

1.3 Objetivos

El objetivo principal de este proyecto es implementar una pila de protocolos basada en encaminamiento geográfico que optimice el envío de mensajes entre los nodos de una red de sensores inalámbricos. Para ello, se han de llevar a cabo una serie de pasos, que se especifican a continuación.

- Realizar un estado del arte sobre redes de sensores inalámbricos para entender sus características, así como sus ventajas y sus limitaciones.
- Analizar las características del protocolo de comunicaciones inalámbricas IEEE 802.15.4, que se utiliza como nivel físico y de enlace en redes de sensores inalámbricos.
- Realizar un estudio sobre los protocolos de encaminamiento geográfico, con el objeto de averiguar qué protocolo es el más adecuado para conseguir el objetivo buscado.
- Elegir los dispositivos adecuados para evaluar el funcionamiento del encaminamiento geográfico y familiarizarse con su utilización.
- Diseñar una pila de protocolos que permita ejecutar y evaluar el funcionamiento del encaminamiento geográfico y desarrollar las funciones necesarias para ello.
- Desplegar una red de sensores inalámbricos que permita evaluar el funcionamiento del encaminamiento geográfico en entornos reales.

1.4 Estructura de la memoria

Esta memoria está formada por los capítulos que se describen brevemente a continuación:

- Capítulo 2: en este capítulo se realiza un estado del arte sobre redes de sensores inalámbricos y protocolos de encaminamiento geográfico. También se describen las tecnologías utilizadas en este proyecto, como son el protocolo IEEE 802.15.4 y los dispositivos JN5139 proporcionados por el fabricante Jennic.
- Capítulo 3: en este capítulo se detalla el diseño de la pila de protocolos que se va a desarrollar. Se define el esqueleto de las funciones que se van a implementar en cada capa de la pila y la manera en que van a interactuar unas con otras.
- Capítulo 4: este capítulo corresponde a la implementación y la evaluación de la pila de protocolos. En él se explica en detalle cómo se implementan las funciones descritas en el capítulo anterior, las decisiones tomadas a la hora de implementar el código y cómo se solucionan las dificultades encontradas a lo largo del desarrollo. También se describen las pruebas realizadas para la comprobación del correcto funcionamiento del encaminamiento geográfico.
- Capítulo 5: en este capítulo se explican las distintas fases de desarrollo del proyecto y se detalla el presupuesto, formado por los costes materiales, los costes de personal y los costes indirectos.
- Capítulo 6: en este capítulo se exponen las conclusiones extraídas y las líneas de trabajo futuras que se pueden desarrollar a partir de la realización de este proyecto.
- Glosario: está constituido por las definiciones de algunos términos y acrónimos utilizados en esta memoria.
- Anexo: en el apéndice se explica cómo se instala y cómo se utiliza el entorno de desarrollo proporcionado por el fabricante Jennic, así como datos técnicos adicionales.

Capítulo 2

Estado del arte

2.1 Introducción

En este capítulo se realiza un estudio sobre las tecnologías utilizadas en este proyecto.

Primero, se introducen las redes de sensores inalámbricos explicando en qué consisten y su importancia en el mundo actual. A continuación se describen sus características, los elementos que las componen y sus ventajas e inconvenientes. Por último, se describe la amplia gama de aplicaciones que se benefician de la utilización de este tipo de redes.

En segundo lugar, se realiza una panorámica sobre los protocolos de encaminamiento geográfico que se pueden utilizar en este tipo de redes, incluyendo *Greedy Forwarding*, que es el que se ha desarrollado en este proyecto.

Por último, se describen las tecnologías empleadas directamente en este proyecto, como son el protocolo de comunicaciones inalámbricas IEEE 802.15.4 y los sensores del fabricante Jennic, modelo JN5139. Por una parte, se realiza una amplia descripción del estándar IEEE 802.15.4. Primero se explican las razones por las que se creó dicho estándar. En segundo lugar, se exponen las principales aplicaciones que se benefician del uso de este estándar. Después se especifican las bandas de frecuencia y tasas de datos utilizadas. Por último, se detallan características técnicas como la descripción de su arquitectura, el direccionamiento de los dispositivos involucrados y los modos de seguridad del estándar.

En cuanto a los sensores utilizados durante la realización del proyecto, primero se citan las características de los microcontroladores JN5139, a continuación se describen los elementos que forman el kit de desarrollo de software proporcionado por el fabricante Jennic y por último, se detallan las características hardware de las placas.

2.2 Redes de sensores inalámbricos

Una red de sensores inalámbricos (WSN, *Wireless Sensor Network)* es una red de comunicaciones inalámbricas desplegada en un área geográfica determinada y que está formada por numerosos dispositivos que utilizan sensores para controlar determinadas condiciones en distintos puntos, tales como la temperatura, la humedad, la presión, el sonido, la vibración o el movimiento. Estos dispositivos son nodos autónomos constituidos por un microcontrolador, una fuente de energía (que normalmente es una batería), un radiotransceptor y un sensor (1). Estos nodos también se denominan motas, en inglés "mote". El nombre de "mote" fue asignado por la Universidad de Berkeley porque se piensa que en un futuro su tamaño será insignificante, como el de una mota de polvo (2).

Las tecnologías de redes inalámbricas han experimentado un importante desarrollo durante los últimos años. El desarrollo más interesante es el de las redes de sensores inalámbricos debido al gran número de aplicaciones de diferentes sectores en los que estas redes tienen cabida. Las WSN son una de las opciones de futuro más valoradas por los analistas tecnológicos, dentro del ámbito de las tecnologías inalámbricas. Algunos fabricantes como Microsoft, Intel, IBM, Motorola y Texas Instruments han lanzado líneas de investigación en esta tecnología.

Este tipo de redes se engloban dentro de la llamada "inteligencia ambiental", que consiste en la creación de objetos de uso cotidiano con capacidades de adquisición de información, procesamiento y comunicación, de forma que puedan comunicarse entre ellos y ofrecer nuevos servicios a los usuarios (3).

Como indicadores de la importancia de las WSN se pueden citar los siguientes hechos:

- El MIT identificó en febrero de 2003 a las WSN como la primera de las diez tecnologías emergentes que cambiarán el mundo.
- El número de empresas que fabrican sensores en un país se considera un indicador tecnológico (4).
- Esta tecnología se ha integrado en diferentes campos como la agricultura, la biología, la medicina, etc.
- Se han hecho posibles acciones que antes eran impensables, como la interacción humana con el medio, gracias a la computación ubicua y a la inteligencia ambiental.
- Científicos e investigadores de gran renombre se han subido al tren de las WSN (5).

2.2.1 Características

Las principales características de este tipo de redes son las siguientes:

- Facilidad de despliegue.
- Se utilizan tecnologías inalámbricas de corto alcance, el encaminamiento entre dos nodos sin visión directa se realiza mediante comunicaciones multisalto.
- Topología dinámica: nodos autoconfigurables, tolerancia a fallos.
- Utilización de envío de mensajes en *broadcast.*
- Consumo extremadamente bajo: estas redes funcionan con pilas y tienen una larga autonomía de funcionamiento. Pueden operar sin mantenimiento durante varios meses o años.
- Coste muy bajo.
- Pequeño tamaño (5).
- Integración con otras tecnologías como agricultura, biología, medicina, etc.
- Interacción de los seres humanos con el medio: redes vehiculares, etc.
- Menor uso de recursos computacionales y de memoria (6).

2.2.2 Elementos

En la Figura 2.1 se muestran los principales elementos que forman una WSN y se describen a continuación:

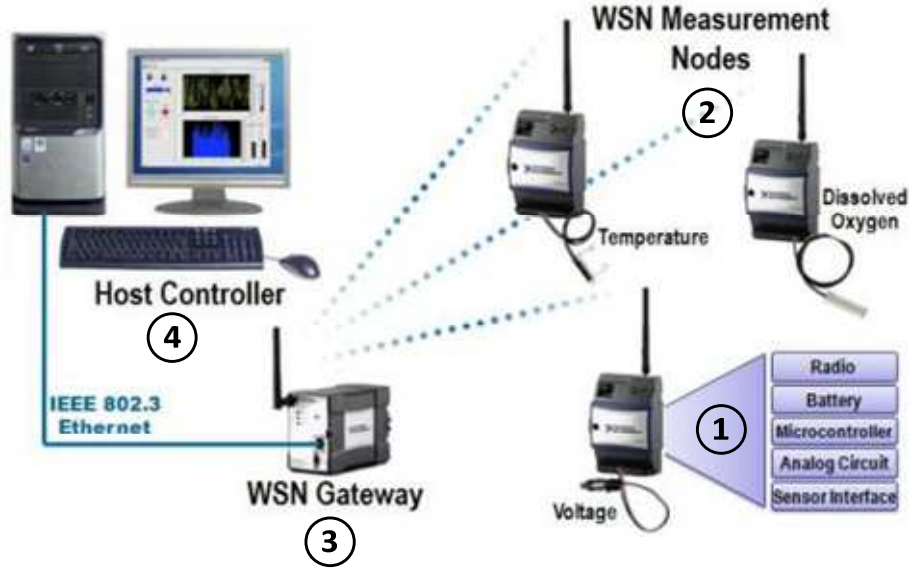

Figura 2.1. Arquitectura común de una WSN (7).

1) Sensores: miden características del medio y convierten esta información en señales eléctricas.
2) Nodos o motas: transmiten la información obtenida por el sensor a la estación base.
3) Puerta de enlace (*gateway*): interconecta la red de sensores con una red de datos (TCP/IP).
4) Estación base: es un ordenador común o sistema embebido que se encarga de recolectar los datos (5).

En el apartado 2.5 se describen en detalle los sensores y nodos utilizados.

2.2.3 Ventajas e inconvenientes

La utilización de este tipo de redes supone ciertas ventajas aunque también tiene algunos inconvenientes.

El despliegue de redes de sensores inalámbricos ofrece las siguientes ventajas (2):

- Redes auto-organizativas.
- Gran robustez.
- Diversidad.
- Escalabilidad.
- Mayor densidad espacial de medida.

Los problemas de este tipo de redes se citan a continuación:

- Optimización del consumo de energía. Para conseguir reducir el consumo de energía y, por tanto, alargar el tiempo de vida de la red, se pueden utilizar ciertos recursos como el modo "dormir" de los nodos cuando no están transmitiendo, la economización de la distancia de las comunicaciones (4) y una programación eficiente.
- Ancho de banda y cobertura de la red limitados.
- Recursos de computación (memoria y CPU) limitados.
- Soluciones ad-hoc para redes ad-hoc.
- Topología muy dinámica de la red: elementos móviles, nodos con alta probabilidad de fallo, nodos que entran y salen del sistema, se necesitan muchos nodos para mejorar el rendimiento (5); todo esto aumenta la complejidad de la red.

Esta problemática está siendo tratada por varios centros de investigación, entre los que destacan los siguientes (2):

- Centro de investigación de Intel (*Intel research laboratory*).
- Centro de investigación de Palo Alto.
- Universidad de Berkeley, California.
- Centro para Mallas de Sensores Empotrados, Universidad de California.
- Universidad de Ohio.
- Universidad de Virginia.
- *Swiss Federal Institute of Technology*.

2.2.4 Aplicaciones

Las WSN se utilizan para aplicaciones en diversos ámbitos. Algunos de los escenarios de aplicación de las WSN se explican a continuación (5).

➢ Monitorización del entorno

Consiste en la lectura de variables durante un período de tiempo en entornos inaccesibles y hostiles con el objeto de detectar cambios o tendencias. Las características necesarias para este tipo de aplicaciones son las siguientes:

- Gran número de nodos sincronizados, midiendo y transmitiendo periódicamente.
- Redes con tiempo de vida alto.
- Redes con precisión en la sincronización.
- Topología física relativamente estable.
- Requerimientos de latencia no estrictos, los datos recogidos se utilizan en análisis futuros, no en tiempo real.
- Infrecuente reconfiguración de la red.

Algunos ejemplos de aplicaciones en este campo son la agricultura de precisión y los estudios medioambientales.

➢ Monitorización de seguridad

Se utilizan las redes para la detección de anomalías o ataques mediante la monitorización por sensores. Los requisitos de las redes para este tipo de aplicaciones son los siguientes:

- Los nodos no transmiten continuamente, sólo informan cuando detectan algo anómalo.
- Se requiere un menor consumo de energía.
- Importancia del estado de un nodo.
- Importancia de la latencia de las comunicaciones, aplicaciones en tiempo real.

Aplicaciones de este ámbito son la domótica y la inmótica, la detección de incendios y las aplicaciones militares.

➢ *Tracking*

Se controlan objetos que están etiquetados con nodos sensores en una región determinada.

A diferencia de los casos anteriores, en este escenario la red es muy dinámica, debido al movimiento continuo de los sensores. La WSN debe ser capaz de descubrir nuevos nodos y formar nuevas topologías.

Estas aplicaciones se utilizan para logística, control de recursos, monitorización industrial y monitorización de fauna.

Un caso de monitorización de fauna es el llamado “Proyecto ZebraNet” (8). Un grupo de científicos de la Universidad de Princeton colocó dispositivos GPS en los collares de decenas de cebras Grevy (una especie en peligro de extinción) para recopilar información acerca de su paradero, sus velocidades y su capacidad de movimiento. Este método ha permitido a los expertos estudiar el comportamiento de estas cebras.

Figura 2.2. Proyecto ZebraNet (4).

➢ Redes híbridas

Se denomina así a los escenarios que tienen características de las tres categorías anteriores, como por ejemplo, la monitorización hospitalaria.

Como se dijo anteriormente, las WSN tienen aplicaciones en multitud de ámbitos diferentes (agricultura, salud, ciencia, domótica, control industrial, etc.). Algunas de las aplicaciones más importantes se explican a continuación.

➢ Agricultura proactiva o de precisión

Mediante la utilización de redes de sensores y la medición de determinados parámetros se llevan a cabo las siguientes acciones:

- Control de la cantidad de agua, fertilizante o pesticida que las plantas necesitan.
- Decisión del momento óptimo para realizar la cosecha.
- Optimización de la producción y la calidad de una cosecha.
- Gestión de alarmas por intrusión de animales o daños provocados por heladas.

➢ Incendios forestales

Las WSN son muy útiles para la estimación del riesgo, la prevención y la detección temprana de incendios forestales. Mediante el despliegue de una red de sensores en la zona que se desea proteger, se puede llevar a cabo la monitorización de variables relacionadas con el riesgo de incendios forestales como la temperatura, la humedad, el nivel de precipitaciones y la velocidad del viento. El conocimiento preciso y continuo de estas variables permite estimar el riesgo de incendio en cada punto de la zona de interés para prevenir el incendio, detectar de forma temprana un foco de incendio y examinar las causas del incendio, a posteriori.

➢ Logística y control de recursos

En el ámbito industrial, se pueden facilitar las tareas de logística y control de recursos mediante la utilización de sensores. Para ello, se asignan nodos inteligentes a los contenedores de productos o se sustituyen los códigos de barras por etiquetas

inteligentes, de manera que los productos están controlados durante todo el recorrido de la cadena logística.

➢ Control de procesos

Para utilizar redes de sensores en el control de procesos hay que tener en cuenta que para este tipo de aplicaciones todos los sensores son vitales y el tiempo es un factor crítico.

Algunas de las aplicaciones en este ámbito son la telemetría, las medidas de calidad, el diagnóstico de maquinaria y la monitorización.

Los principales beneficios que se consiguen mediante la utilización de WSN son la reducción de costes y de cableado.

➢ Estudios medioambientales

Otro campo importante de aplicación de las WSN son los estudios medioambientales. Mediante la medida de variables como luz, velocidad del viento, temperatura, humedad, precipitación, presión barométrica, actividad sísmica, etc. los científicos esperan obtener más conocimiento sobre contaminantes de la tierra fértil, cambios geológicos, flujo del agua, especies invasoras, ciclos de océanos, lugares en los que se almacena el carbono atmosférico, erupción de volcanes y movimiento de virus y fragmentos de genes por el medio ambiente.

La ventaja de utilizar este tipo de dispositivos es que se pueden tener miles de sensores repartidos por grandes áreas de forma permanente y desatendidos.

Un ejemplo de este tipo de estudios medioambientales es el estudio del microclima de las secuoyas en Sonoma (California). Los investigadores colocaron 150 motas en las secuoyas con objeto de conocer el microclima monitorizando desde 70 kilómetros de distancia la temperatura, la luz, la presión y la humedad. Algunas de estas motas disponen de un módulo GPS para facilitar la geolocalización.

Figura 2.3. Estudio del microclima de las secuoyas en Sonoma (California) (5).

Otro ejemplo de estudios de microclima con redes de sensores es el de la isla Great Duck, en Estados Unidos. Existe una red de 150 sensores inalámbricos que monitoriza el microclima en los refugios donde anidan las aves marinas y en los alrededores. El propósito es que los investigadores puedan supervisar especies en peligro de extinción y sus hábitats de manera no intrusiva.

Figura 2.4. Estudio de microclima en Great Duck Island (5).

➢ Entornos inteligentes

Dentro de la domótica y la inmótica destacan los entornos inteligentes, en los que se desarrollan las siguientes aplicaciones:

- Control de iluminación y climatización.
- Detección de presencia para optimizar la energía.
- Control de acceso y detección de intrusos.
- Monitorización de personas mayores.

➢ Entornos interactivos

Mediante la utilización de WSN se pueden desarrollar las siguientes aplicaciones:

- Comunicaciones entre vehículos.
- Señales de emergencia desde ambulancias, policía, bomberos, etc. al resto de vehículos del entorno inmediato.
- Comunicación y avisos en cruces entre semáforos y vehículos en direcciones perpendiculares.
- Gestión de aparcamientos.

Un ejemplo de localización es la aplicación *Cricket* (9) desarrollada por el MIT. *Cricket* es un sistema de localización en interiores para entornos basados en sensores. Algunas de las aplicaciones que se pueden desarrollar usando *Cricket* incluyen la localización de recursos, la navegación humano-robot, los juegos de ordenador físico-virtuales, las aplicaciones médicas como seguimiento de pacientes y material, etc. *Cricket* está pensado para utilizarse en interiores o en áreas urbanas donde otros sistemas como GPS no funcionan bien. Tiene una precisión de entre 1 y 3 centímetros, por tanto, es útil para aplicaciones que requieran una mejor precisión que la que se obtiene con GPS. Está diseñado para un bajo consumo de energía y puede ser utilizado por redes de sensores. Proporciona información de localización a los dispositivos mediante una combinación de tecnologías de radiofrecuencia y ultrasonidos. En la Figura 2.5 se puede ver el despliegue de *Cricket* en un laboratorio:

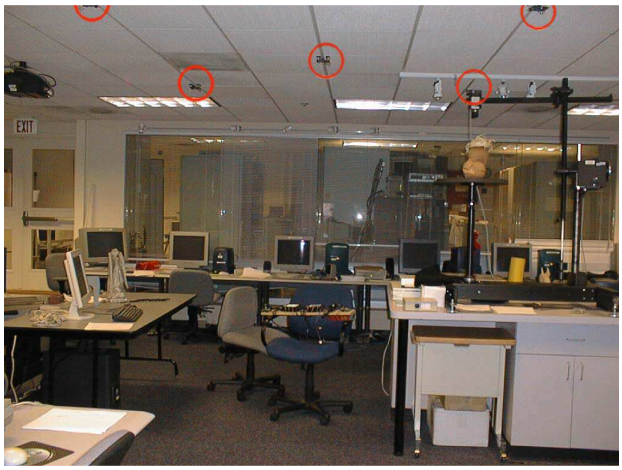

Figura 2.5. Despliegue de sensores de la aplicación *Cricket* (9).

También se puede destacar el proyecto FIRE (10) (*Fire Information and Rescue Equipment*) desarrollado por la Universidad de Berkeley en colaboración con el cuerpo de bomberos de Chicago. La extinción de incendios puede convertirse en un entorno extremadamente caótico, teniendo en cuenta que hay que tomar decisiones rápidas con poca información y prestar atención a diferentes eventos, lo que dificulta las tareas de búsqueda y rescate. El proyecto FIRE trata de solucionar estos problemas mediante la utilización de tecnologías como WSN. El objetivo es crear herramientas de información y mejorar la comunicación de los bomberos para mejorar la seguridad, eficiencia y eficacia de las respuestas de emergencia. El proyecto está enfocado a grandes incidentes urbanos y edificios comerciales e industriales.

Mediante la utilización de WSN se realiza un seguimiento de los bomberos en el edificio y se distribuye información sobre su localización, su estado de salud y el estado del fuego. Cada bombero dispone de un pequeño ordenador o de un dispositivo integrado en el casco con los planos de las plantas del edificio e información sobre su localización y la de sus compañeros. También se muestra la localización del fuego. El plano sólo debe mostrar información básica para que los bomberos puedan obtener la información de un solo vistazo. En la Figura 2.6 se muestra una imagen sobre cómo serían estos planos:

Figura 2.6. Plano de información del proyecto FIRE (10).

La WSN utilizada en el proyecto FIRE se denomina *SmokeNet*. Está formada por sensores de humo y temperatura y nodos de señalización radio que transmiten información crítica a los bomberos. Esta red debe ser robusta y redundante ya que es la columna vertebral del sistema FIRE. Permitirá a los bomberos saber rápidamente dónde se ha originado el fuego, cómo se está extendiendo y qué rutas de evacuación son seguras. Este sistema proporcionará información importante para una rápida toma de decisiones a los ocupantes del edificio y a los bomberos.

➢ Entornos militares y de alta seguridad

En este ámbito las WSN son útiles para la prevención de ataques terroristas, por ejemplo, en centrales nucleares, aeropuertos y edificios gubernamentales.

Algunos ejemplos de aplicaciones son:

- *Portable Intrusion Detection System* (PIDS) (11): equipamiento portátil para detección de intrusos, alimentado con batería, con comunicación remota directa al teléfono móvil o al centro de monitorización, apropiado para todas las aplicaciones de seguridad.
- Control del "fuego amigo".
- Plano instantáneo del campo de batalla.

➢ Aplicaciones médicas

Existen numerosas aplicaciones médicas en las que se aplican redes de sensores inalámbricos, entre las que destacan las siguientes:

- Monitorización continua de pacientes en tiempo real: permiten localización y movilidad, otorgando libertad y movimiento para el paciente en traslados de habitación, en ambulancias, etc., posibilitan monitorización pre-hospital, en el propio hospital y ambulatoria, permiten realizar medidas de constantes vitales (pulsaciones, presión, etc.), sustituyen sistemas de telemetría vía cable que son caros e incómodos.
- Localización de personal sanitario.
- Supervisión de pacientes crónicos y ancianos en su casa: las redes se encargan de recoger datos periódicamente para analizar tendencias y de enviarlos al médico. Así se reduce el tiempo de permanencia en el hospital.
- Bases de datos con recopilación de datos clínicos a largo plazo: se relacionan los datos obtenidos por los sensores con otra información de los pacientes. Permiten realizar análisis de datos de poblaciones y estudios sobre los efectos de intervenciones.

Dentro de estas aplicaciones cabe destacar un proyecto desarrollado por la Universidad de Harvard llamado *CodeBlue: Wireless Sensors for Medical Care* (12). Se están investigando las aplicaciones de la tecnología de redes de sensores inalámbricos para una amplia gama de aplicaciones médicas incluyendo las emergencias pre-hospitalarias y hospitalarias, la respuesta a desastres y la rehabilitación de pacientes que han sufrido infarto cerebral. Las redes de sensores permiten la recolección automática de datos, de tal forma que se monitorizan signos vitales y se integran en el registro de atención al paciente para utilizarse en clasificación en tiempo real, correlación con los registros del hospital y observación a largo plazo. Se han desarrollado diversos sensores médicos entre los que se encuentran un medidor inalámbrico de la saturación de oxígeno en sangre y un electrocardiograma basado en dos electrodos. Estos dos dispositivos se muestran en las Figuras 2.7 y 2.8:

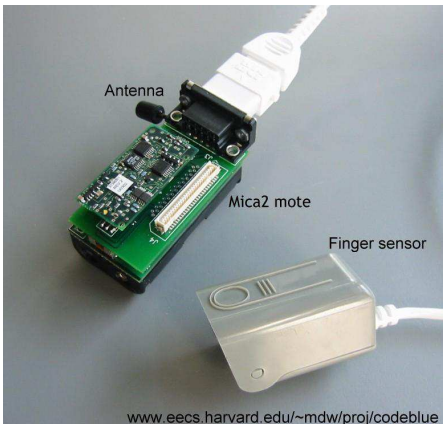

Figura 2.7. Sensor inalámbrico para la medida de saturación de oxígeno (*CodeBlue*) (12).

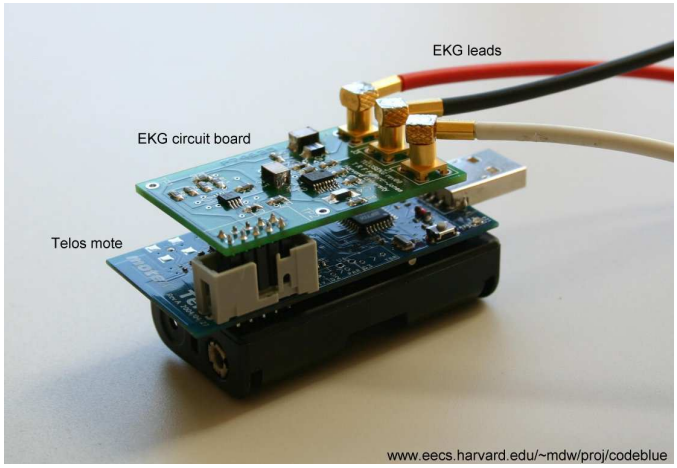

Figura 2.8. Electrocardiograma inalámbrico de dos electrodos (*CodeBlue*) (12).

Estos dispositivos miden la frecuencia cardíaca, la saturación de oxígeno y los datos del electrocardiograma y los transmiten en una red inalámbrica de corto alcance (100 metros) a cualquier número de dispositivos de recepción, incluyendo PDAs, portátiles o terminales en ambulancias. Los datos se pueden visualizar en tiempo real e integrarse en el registro de atención pre-hospitalaria del paciente. Los dispositivos se pueden programar para procesar los datos de signos vitales y, por ejemplo, crear una alerta cuando éstos caigan por debajo de los valores normales. Cualquier cambio en el estado de un paciente puede ser comunicado al personal sanitario más cercano.

En la Figura 2.9 se representa la arquitectura de la plataforma de *CodeBlue* para respuesta a emergencias:

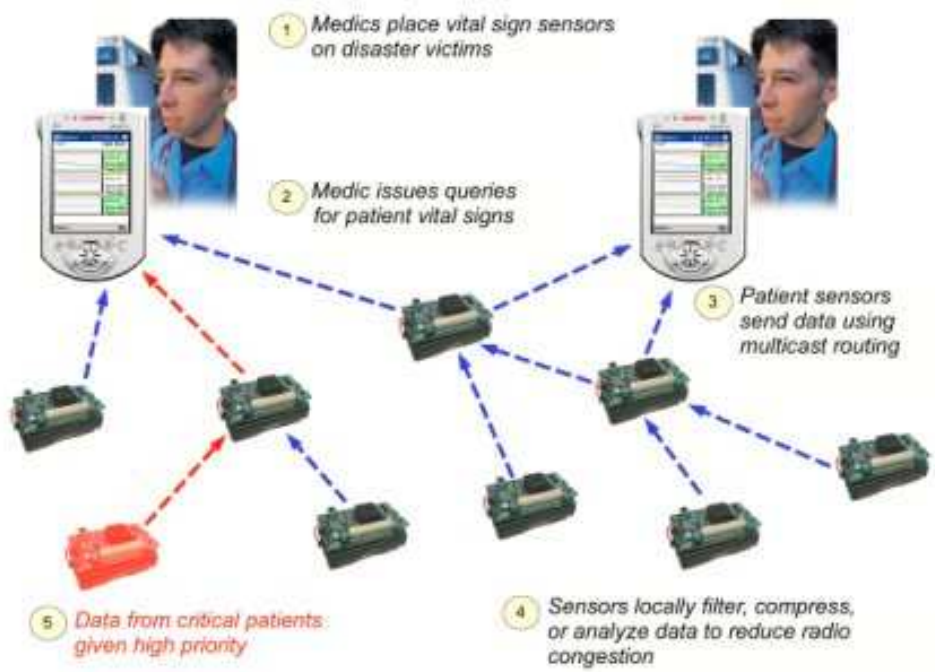

Figura 2.9. Arquitectura para respuesta a emergencias de *CodeBlue* (12).

Además del hardware, se está desarrollando una infraestructura de software escalable para dispositivos médicos inalámbricos, es lo que se denomina *CodeBlue*. *CodeBlue* está diseñado para proporcionar encaminamiento, nombrado, descubrimiento de dispositivos y seguridad para sensores médicos inalámbricos, PDAs, PCs y otros dispositivos que se utilicen para monitorizar y tratar a los pacientes.

Parte del sistema de *CodeBlue* incluye *MoteTrack*, que es un sistema para el seguimiento de la ubicación de los distintos dispositivos de los pacientes, utilizando la información de las señales de radio. En *MoteTrack* se equipa el hospital (o el área que corresponda) con nodos fijos de señalización por radio que se utilizan para calcular la posición 3D de los sensores inalámbricos que llevan los pacientes o el equipo médico. El error de medida de *MoteTrack* es de alrededor de 2 metros, que es una precisión suficiente para este tipo de aplicaciones.

Las investigaciones de este equipo de la Universidad de Harvard se focalizan en las siguientes áreas:

- Integración de sensores médicos con redes inalámbricas de bajo consumo.
- Protocolos de encaminamiento para redes inalámbricas ad-hoc para cuidados intensivos. Los requisitos necesarios son seguridad, robustez y priorización.
- Arquitecturas hardware para sensado, computación y comunicación, de consumo ultra bajo.
- Interoperabilidad con los sistemas de información hospitalaria (problemas de privacidad y fiabilidad).
- Localización 3D utilizando la información de señales radio.
- Gestión adaptativa de recursos, control de congestión y asignación de ancho de banda en redes inalámbricas.

2.3 Protocolos de encaminamiento geográfico

Las características, ventajas e inconvenientes de las WSN descritos en los apartados 2.2.1 y 2.2.3 implican que el diseño de protocolos de encaminamiento para este tipo de redes deba tener en cuenta las siguientes características (13):

- Limitación en el consumo de energía: los dispositivos están alimentados con baterías y por tanto, tienen una capacidad de almacenamiento de energía limitada. Es imprescindible disminuir el consumo de energía porque estos dispositivos pueden trabajar en entornos hostiles en los que es muy difícil o imposible acceder a los nodos y recargar sus baterías. También es importante porque al descender el nivel de energía por debajo de un cierto umbral los sensores empiezan a fallar y afecta al correcto funcionamiento de la red. El diseño de protocolos de encaminamiento debe tener en cuenta este factor para prolongar el tiempo de vida de las baterías, alargando el tiempo de vida de la red y garantizando su eficacia.
- Localización de los sensores: la mayoría de los protocolos de encaminamiento asumen que los dispositivos están equipados con GPS o algún otro mecanismo de localización que permita conocer su ubicación. Si no disponen de ningún mecanismo, hay que gestionar la localización de los sensores.
- Despliegue de nodos aleatorio y masivo: el despliegue de nodos en una WSN puede ser aleatorio y éstos pueden repartirse de forma heterogénea dentro de un entorno hostil o inaccesible. Si la distribución de los nodos no es uniforme, es necesario agrupar los nodos para permitir la comunicación entre ellos de manera energéticamente eficiente.
- Red dinámica: las WSN son muy dinámicas debido a la adición o sustracción de nodos, a fallos en los enlaces y a la disminución de la energía disponible. El medio inalámbrico es, en sí mismo, un medio ruidoso, propenso a fallos y variante en el tiempo. El dinamismo y la movilidad en las redes de sensores son factores a tener en cuenta a la hora de diseñar protocolos de encaminamiento en cuanto a requisitos de cobertura y conectividad.
- Agrupación de datos: los paquetes similares de múltiples nodos pueden agruparse para reducir las transmisiones y así optimizar el consumo de energía.
- Requisitos dependientes de la aplicación: dependiendo de la aplicación que se trate habrá requisitos determinados. Ningún protocolo de encaminamiento puede satisfacer las necesidades de todas las aplicaciones de las redes de sensores.
- Escalabilidad: el protocolo de encaminamiento debe funcionar independientemente del tamaño de la red.
- Heterogeneidad: dependiendo de la aplicación puede haber nodos y enlaces con características diferentes, como capacidad de computación, comunicación y batería.
- Calidad del servicio: en algunas aplicaciones es importante la latencia, sin embargo, en otras aplicaciones es más importante el ahorro de energía, por tanto, se puede reducir el consumo reduciendo la calidad del servicio (14).

Los protocolos de encaminamiento para redes de sensores inalámbricos se pueden clasificar en las siguientes categorías (13):

- Protocolos basados en localización.
- Protocolos centrados en datos.
- Protocolos jerárquicos.

- Protocolos basados en movilidad.
- Protocolos basados en multicamino.
- Protocolos basados en heterogeneidad.
- Protocolos basados en calidad de servicio.

En esta sección sólo se trata sobre los protocolos pertenecientes a la primera categoría, que se corresponden con el tema que nos ocupa.

Los protocolos basados en localización utilizan la posición de los nodos para calcular la distancia entre ellos. Las coordenadas relativas a los nodos vecinos se pueden obtener mediante el intercambio de información entre nodos. Si los nodos están equipados con un sistema GPS, la ubicación de los nodos también se puede conocer directamente. Como medida de ahorro de energía, algunos sistemas basados en localización mantienen los nodos dormidos si no hay actividad, aunque esto aumenta la complejidad de las redes (14).

A continuación se explican algunos de los protocolos de encaminamiento geográfico basados en localización más importantes.

2.3.1 *Greedy Perimeter Stateless Routing* (GPSR)

Greedy Perimeter Stateless Routing (GPSR) (15) consiste en la combinación de dos métodos para el reenvío de paquetes: *Greedy Forwarding*, que se utiliza siempre que sea posible, y *Perimeter Forwarding*, que se utiliza en las regiones en las que no se puede utilizar *Greedy Forwarding*.

GPSR es un protocolo de encaminamiento para redes de paquetes inalámbricas que utiliza la posición de los nodos y el destino del paquete para tomar decisiones de reenvío. GPSR toma decisiones de reenvío *greedy* (avariciosas, impacientes) utilizando únicamente información sobre los nodos vecinos en la topología de red. Cuando un paquete alcanza una región donde es imposible el reenvío *greedy*, el algoritmo se recupera mediante el encaminamiento alrededor del perímetro de la región.

El encaminamiento geográfico sólo requiere que cada nodo conozca las posiciones de sus vecinos. La posición del destino del paquete y las posiciones de los candidatos a siguiente salto constituyen la información necesaria para tomar decisiones de reenvío correctas. Se requiere que todos los nodos conozcan su posición, mediante GPS u otras técnicas.

A continuación se explican los dos métodos que forman GPSR.

- ***Greedy Forwarding***

  Este mecanismo es el que se ha implementado en este proyecto.

  En el origen se define la localización del destino y esta información se incluye en el paquete para que los nodos puedan elegir el siguiente salto. Como los nodos conocen las posiciones de sus vecinos, la elección del siguiente salto consiste en elegir al vecino que esté geográficamente más cerca del destino del paquete. Progresivamente, el siguiente nodo vuelve a elegir al vecino que se encuentre más cerca del destino como siguiente salto y reenvía el paquete, hasta que se alcanza el destino. En la Figura 2.10 se muestra un ejemplo de la elección del siguiente salto:

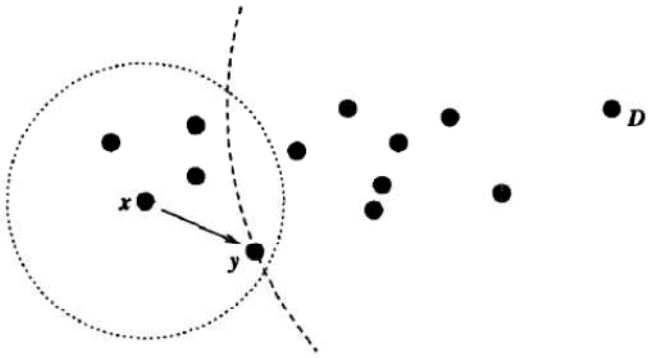

Figura 2.10. Ejemplo de *Greedy Forwarding* (15).

En el ejemplo de la figura el nodo "x" recibe un paquete cuyo destino es el nodo "D". El rango de alcance de "x" viene definido por la circunferencia trazada alrededor de él y el arco indica la distancia entre el nodo "D" y el nodo "y". Como se puede observar, la distancia entre "y" y "D" es menor que la distancia entre "D" y los demás vecinos de "x", por tanto, el nodo "x" reenvía el paquete al nodo "y". Este proceso se repite hasta que el paquete alcanza el nodo "D".

Para que todos los nodos conozcan la posición de sus vecinos se utiliza un algoritmo de señalización. Cada nodo envía periódicamente un mensaje en *broadcast* que contiene su propio identificador (por ejemplo, su dirección IP) y su posición. La posición consiste en dos coordenadas: x e y. Si no se reciben mensajes de un vecino durante un intervalo de tiempo predefinido, el nodo asume que el vecino ha dejado de funcionar o se ha salido del rango y lo elimina de su tabla de vecinos. La elección de este intervalo para que las tablas de vecinos se mantengan actualizadas correctamente depende de la movilidad de la red y del radio de cobertura de los nodos.

La gran ventaja de *Greedy Forwarding* es que sólo se necesita conocer a los vecinos inmediatos para el reenvío. En redes multisalto el número de vecinos dentro del rango es sustancialmente menor que el número total de nodos en la red. Esta ventaja también supone un inconveniente: hay topologías de red en las que la única ruta para alcanzar el destino requiere que el paquete se mueva temporalmente a un nodo más alejado del destino geográficamente. En la Figura 2.11 se muestra un ejemplo que ilustra esta situación:

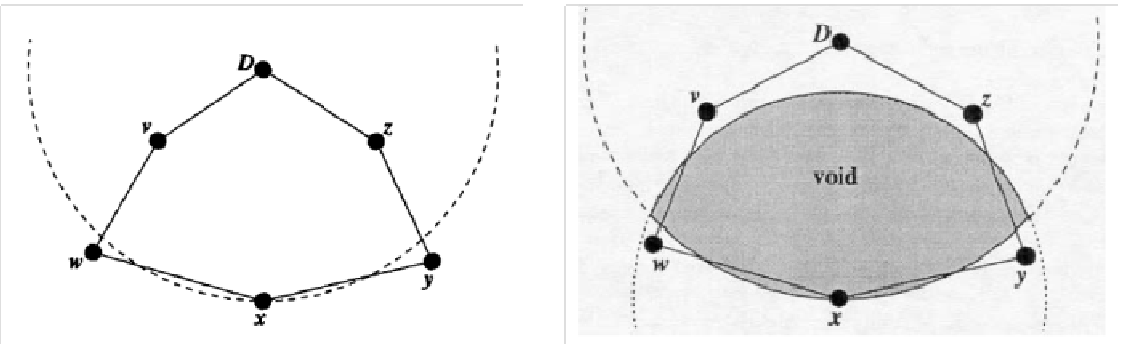

Figura 2.11. Ejemplo de fallo de *Greedy Forwarding* (15).

En el caso de la Figura 2.11, el nodo “x” recibe un paquete cuyo destino es el nodo “D”. El arco representa la distancia entre el nodo “x” y el nodo “D”. Aunque existen dos caminos posibles para alcanzar el destino (“x-y-z-D” y “x-w-v-D”), el algoritmo no reenviará el paquete porque la distancia entre el destino y los nodos “y” y “w” es mayor que la distancia entre el destino y el nodo “x”, es decir, el nodo “x” es un mínimo local debido a su proximidad al nodo “D”. En estas situaciones, se debe emplear otro mecanismo para el reenvío de paquetes, como el que se explica a continuación.

- **La regla de la mano derecha: *Perimeter Forwarding***

Como se observa en la Figura 2.11, hay una zona en la que el nodo “x” no tiene vecinos. Esta zona, que se muestra sombreada en la figura, se denomina hueco. El nodo “x” busca reenviar el paquete alrededor del hueco, ya que si existe un camino hacia “D” no incluirá nodos que estén dentro del hueco porque, en ese caso, se hubiese reenviado con *Greedy Forwarding*.

La regla de la mano derecha se explica mediante la Figura 2.12:

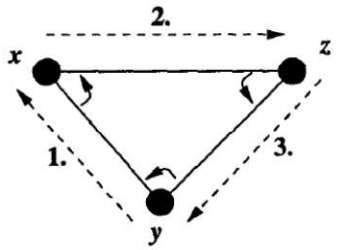

Figura 2.12. Regla de la mano derecha (15).

Cuando se llega al nodo “x” desde el nodo “y”, el siguiente borde que se recorre es el siguiente en sentido contrario a las agujas del reloj desde la línea que une “x” e “y”. La secuencia de bordes recorrida por la regla de la mano derecha se denomina perímetro.

Recorriendo el perímetro de un hueco se mejoran los resultados. Una vez que mediante *Perimeter Forwarding* se llega a un nodo más cercano que aquel donde falla *Greedy Forwarding*, se vuelve a utilizar *Greedy* hasta que el paquete alcanza el destino, sin peligro de volver a caer en el mínimo local anterior.

2.3.2 *Bounded Voronoi Greedy Forwarding* (BVGF)

Este algoritmo es una variante de *Greedy Forwarding* utilizando diagramas de Voronoi. Un diagrama de Voronoi (16) consiste en la división del plano en tantas regiones de Voronoi como nodos haya en la red, de tal forma que un punto del plano cae dentro de la región de un nodo si, y sólo si, ese nodo es el más cercano al punto. En la Figura 2.13 se muestra un diagrama de Voronoi:

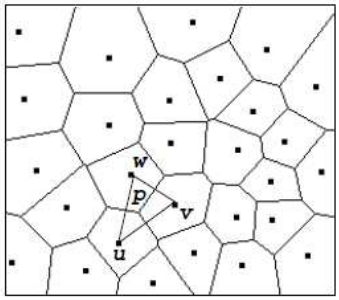

Figura 2.13. Diagrama de Voronoi de una red de sensores (16).

BVGF es un algoritmo localizado que toma decisiones de encaminamiento *greedy* basadas en las posiciones de los vecinos que se encuentran a un solo salto. Un nodo puede ser elegido como siguiente salto sólo si la línea que une el origen con el destino atraviesa su correspondiente región de Voronoi o coincide con uno de los bordes de su región de Voronoi. BVGF elige como siguiente salto aquel nodo que tenga la menor distancia euclídea hasta el destino dentro del conjunto de nodos que pueden ser elegidos. Cuando hay varios nodos elegibles que se encuentran a la misma distancia del destino, el algoritmo elige uno de ellos aleatoriamente como siguiente salto.

En la Figura 2.14 se muestra un ejemplo de encaminamiento utilizando BVGF:

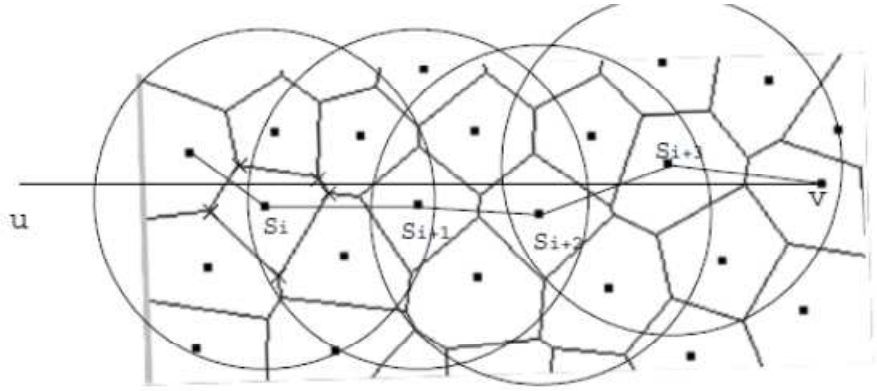

Figura 2.14. Ejemplo de encaminamiento con BVGF (16).

El segmento entre “u” y “v” es la línea que une el origen con el destino. Las circunferencias indican el radio de cobertura de los nodos. El encaminamiento se realiza a través de los nodos s_i, s_{i+1}, s_{i+2}, s_{i+3}. Como se puede observar, el siguiente salto de un nodo no tiene por qué estar en una región de Voronoi adyacente.

Para mantener actualizada la tabla de vecinos, cada nodo envía periódicamente un mensaje de señalización en *broadcast* incluyendo su posición y las posiciones de los vértices de su región de Voronoi.

2.3.3 *Greedy Other Adaptive Face Routing* (GOAFR)

Greedy Other Adaptive Face Routing (GOAFR) (17), pronunciado “gopher”, es una combinación de G*reedy* y OAFR (*Other Adaptive Face Routing*). En principio, siempre que sea posible se utiliza G*reedy* (explicado en la sección 2.3.1). Para escapar de los mínimos locales se utiliza OAFR.

Para entender este algoritmo, se necesitan algunas nociones sobre AFR (*Adaptive Face Routing*). La base de este algoritmo es *Face Routing*, que se basa en la regla de la mano derecha (explicada en la sección 2.3.1). Mientras se recorre un perímetro o cara (*face*) este algoritmo toma nota de los puntos que se cruzan con la línea que une el origen y el destino. Cuando se termina de rodear la cara el algoritmo vuelve a aquella de las intersecciones que se encuentre más cerca del destino y explora la siguiente cara a la que pertenece este punto, que estará, por tanto, más cerca del destino. El problema es que se necesita explorar completamente el borde de las áreas. *Face Routing* puede ser mejorado utilizando *Bounded Face Routing* (BFR) en el que se restringe el área de búsqueda a una elipse cuyo tamaño se calcula a partir de una estimación de la longitud del camino óptimo. La mejora que incluye *Adaptive Face Routing* (AFR) es que si no se consigue alcanzar el destino, se dobla el tamaño de la elipse de búsqueda.

También es necesario conocer el algoritmo OAFR. Se basa en *Other Face Routing* (OFR), que se diferencia de *Face Routing* en que cuando se termina de recorrer el borde de la cara se vuelve al punto del borde más cercano al destino. La diferencia entre *Face Routing* y *Other Face routing* se ilustra en la Figura 2.15:

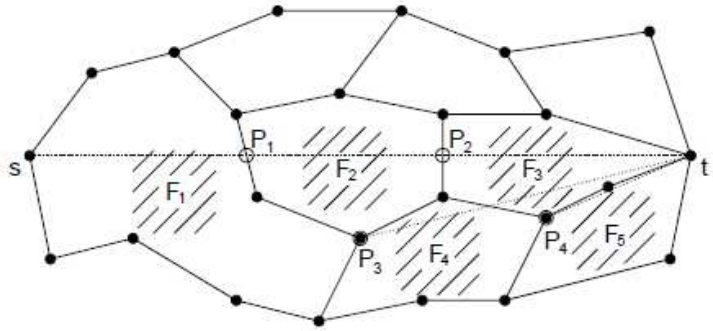

Figura 2.15. Diferencia entre *Face Routing* y OFR (17).

Si se considera el nodo "s" como el origen y el nodo "t" como el destino, el mecanismo de *Face Routing* consiste en explorar la cara "F_1" y dado que el punto "P_1" se cruza con la línea que une origen y destino, después recorrerá "F_2" que es la otra cara a la que pertenece este punto. Lo mismo ocurre con el punto "P_2" y la cara "F_3", hasta que finalmente alcanza el destino. Por el contrario, si se ejecuta OFR, cuando se termina de recorrer el borde la cara "F_1" se selecciona el punto "P_3" como punto del borde más cercano al destino y la siguiente cara a recorrer será la "F_4". Lo mismo ocurre con el punto "P_4" y la cara "F_5" hasta que se alcanza el destino.

Para mejorar el algoritmo OFR, surge OBFR (*Other Bounded Face Routing*) que consiste en limitar el área de búsqueda a una elipse, como se explicó anteriormente, con el algoritmo BFR y si no se consigue alcanzar el destino se dobla el tamaño de la elipse de búsqueda, como se explicó anteriormente, con el algoritmo AFR. Por tanto, *Other Adaptive Face Routing* (OAFR) consiste en utilizar OBFR adaptando el tamaño de la elipse cuando sea necesario con AFR.

Greedy Other Adaptive Face Routing (GOAFR) se forma combinando *Greedy Routing* y OAFR. Se utiliza *Greedy* hasta que se alcanza el destino o un mínimo local, que es un nodo sin ningún vecino más cercano al destino que él mismo. Previamente, se define la elipse de búsqueda. Si el siguiente salto excede los límites de la elipse, se define la nueva zona de búsqueda como el área que se obtiene doblando la longitud de su eje mayor. Cuando se alcanza un mínimo local, se ejecuta OAFR sólo en la primera cara, empezando a recorrerla por el mínimo local. El algoritmo termina cuando se alcanza el destino mediante OAFR. Si se detecta una

desconexión, OAFR debe informar al nodo origen. Si no, se continúa con *Greedy* desde el nodo más cercano al destino encontrado por OAFR.

2.3.4 *Geographical Adaptive Fidelity* (GAF)

Geographical Adaptive Fidelity (GAF) (18) es un algoritmo que reduce el consumo de energía en redes inalámbricas ad-hoc (aunque también puede ser aplicable en WSN). El mecanismo que se utiliza para conservar energía es identificar nodos equivalentes desde la perspectiva del encaminamiento y apagar los nodos innecesarios, manteniendo un nivel constante de fidelidad de encaminamiento.

El procedimiento consiste en que cada nodo de la red utiliza información de localización para asociarse a una cuadrícula virtual (*grid*), donde todos los nodos situados en el mismo cuadrado de la cuadrícula son equivalentes desde la perspectiva del reenvío de paquetes. La cuadrícula virtual se corresponde con el área geográfica de la red de sensores. Los nodos equivalentes interactúan para decidir cuál de ellos utiliza el modo "dormir" y durante cuánto tiempo. Pasado ese periodo de tiempo, se intercambia el papel con otro nodo para equilibrar el consumo de potencia. La información de localización se obtiene mediante GPS u otros sistemas de localización.

En GAF, los nodos pueden estar en tres estados diferentes: dormido, en modo descubrimiento o activo. El diagrama de estados se muestra en la Figura 2.16:

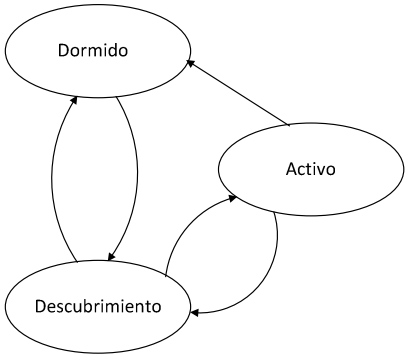

Figura 2.16. Diagrama de estados en GAF.

Inicialmente los nodos se encuentran en el estado de descubrimiento. Aquí intercambian mensajes de descubrimiento con otros nodos, para encontrar cuáles están situados en su misma cuadrícula. Cada nodo envía en *broadcast* su mensaje de descubrimiento. Transcurrido un tiempo, el nodo entra en estado activo. Permanece en este estado durante un tiempo establecido, enviando su mensaje de descubrimiento en intervalos regulares. Tras este período de tiempo el nodo regresa al estado de descubrimiento. Un nodo que se encuentre en el estado activo o en el de descubrimiento puede pasar al estado dormido si encuentra otro nodo equivalente que ejecute el encaminamiento. Los nodos negocian entre sí para elegir cuál es el encargado de gestionar el encaminamiento. Cuando un nodo despierta del estado dormido, vuelve al estado de descubrimiento. De esta forma se consigue alargar el tiempo de vida de la red porque permite a los nodos ahorrar energía mientras se encuentran en el estado dormido. Cuando éstos despiertan, uno de ellos se convierte en activo para que el nodo que estaba activo anteriormente pueda pasar al estado dormido.

El modelo ideal es que haya un nodo activo por cada cuadrícula. Sin embargo, si un nodo cambia de ubicación puede ocurrir que el nodo activo abandone la cuadrícula, reduciendo la fidelidad del algoritmo. Para solucionar este problema, se incluye información sobre movilidad en los mensajes de descubrimiento.

2.3.5 *Geographic and Energy Aware Routing* (GEAR)

Este algoritmo eficiente energéticamente para redes de sensores consiste en propagar el paquete por la región geográfica adecuada, sin inundar toda la red. GEAR (19) selecciona a los vecinos en base al ahorro de energía para encaminar el paquete hacia la región destino y, una vez dentro de la región destino, utiliza técnicas de reenvío geográfico recursivo para difundir el paquete. GEAR alarga la vida de las redes más que otros algoritmos de encaminamiento geográfico, especialmente cuando se trata de distribuciones de tráfico no uniformes. La información de localización se obtiene mediante GPS u otros sistemas de localización. Cada nodo conoce su localización y su nivel de energía restante y las localizaciones y los niveles de energía de sus vecinos, mediante un simple protocolo de descubrimiento.

El procedimiento que lleva a cabo este protocolo se divide en dos fases:

- Reenvío de paquetes hacia la región destino. GEAR utiliza métodos heurísticos basados en información geográfica y ahorro de energía para encaminar el paquete hacia la región destino. Cuando existe un nodo más cercano al destino, se selecciona como siguiente salto. Si hay varios nodos más cercanos al destino, la elección del siguiente salto es un compromiso entre elegir el nodo más cercano al destino y equilibrar el consumo de energía. Si todos los vecinos están más lejos significa que hay un hueco en la red. En este caso, se selecciona como siguiente salto el vecino que minimice el coste energético.

- Difusión del paquete dentro de la región destino. En esta fase se utiliza un algoritmo de reenvío geográfico recursivo. Sin embargo, si la densidad de la red es baja, este algoritmo no siempre consigue llegar al destino. En este caso, se puede hacer uso de la inundación restringida.

2.3.6 *Distance Routing Effect Algorithm for Mobility* (DREAM)

El protocolo DREAM (20) se basa en tablas de localización, pero teniendo en cuenta el efecto distancia. Cada nodo envía periódicamente un paquete de control en *broadcast* especificando sus coordenadas. Para incluir el efecto distancia, a cada paquete de control se le asigna un tiempo de vida basado en la distancia geográfica recorrida por el paquete. Cuando un nodo recibe un paquete de control calcula la distancia hasta el emisor del paquete y si es mayor que el tiempo de vida, el paquete no se reenvía. La frecuencia con la que un nodo envía paquetes de control depende de su movilidad. La idea del tiempo de vida es que cada nodo conozca los nodos cercanos a él, mientras que la información de los nodos lejanos se actualiza con menor frecuencia.

Cuando un nodo envía un mensaje a un destino, busca en su tabla información relativa al destino. Basándose en esta información, selecciona todos aquellos vecinos que se encuentran

en la dirección del destino y les envía el mensaje. Cada uno de estos nodos lleva a cabo el mismo procedimiento, hasta que se consigue alcanzar el destino deseado, si es posible.

Si el emisor no dispone de información sobre la localización del destino, se utiliza un método alternativo para enviar el mensaje. Por ejemplo, se puede enviar el mensaje inundando la red o utilizar la inundación para determinar una ruta hacia el destino.

2.3.7 *Trajectory Based Forwarding* (TBF)

Trajectory Based Forwarding (TBF) (21) es un método para reenviar paquetes en redes ad-hoc densas que permite encaminar el paquete a lo largo de una curva predefinida. Es un híbrido entre encaminamiento basado en origen y reenvío cartesiano. Al igual que en encaminamiento basado en origen, se establece la trayectoria en el origen, pero sin especificar todos los nodos intermedios. Al igual que en reenvío cartesiano, las decisiones de reenvío tomadas en cada nodo son *greedy*, pero no se basan en la distancia al destino sino en la distancia a la trayectoria deseada. TBF requiere que los nodos conozcan su posición respecto a un sistema de coordenadas. Para esto, se puede utilizar GPS pero también son válidos otros métodos de posicionamiento.

En una red en la que las posiciones de los nodos son conocidas, el paquete se reenvía al vecino más cercano a la trayectoria especificada por el origen. Si el nodo destino es conocido, la trayectoria seguida por el paquete será una línea y el método se reduce a reenvío cartesiano.

2.4 IEEE 802.15.4

El estándar IEEE 802.15.4 (22), publicado en mayo de 2003 (23), define una capa de comunicación en el nivel 2 del modelo OSI. Su objetivo principal es permitir la comunicación entre dos dispositivos.

2.4.1 Motivación del estándar

El estándar 802.15.4 (24) surge de la necesidad de llenar el hueco existente entre los estándares para redes inalámbricas, entre los que se incluyen:

- IEEE 802.15.1: *Bluetooth*, que es una tecnología de red inalámbrica de baja potencia y baja tasa para comunicaciones punto a punto.
- IEEE 802.15.3: WPAN (*Wireless Personal Area Network*) de alta tasa de datos.

IEEE 802.15.3 se utiliza en aplicaciones que requieren alta tasa de datos o una gran cobertura, lo que supone soluciones complejas con elevado consumo de potencia. Sin embargo, hay aplicaciones que requieren transmitir una pequeña cantidad de datos en un área restringida. *Bluetooth* no está diseñado para soportar la comunicación entre redes de varios nodos, por tanto, se necesita un nuevo estándar (802.15.4) que cumpla con los siguientes criterios:

- Baja complejidad.
- Muy bajo consumo de energía.
- Baja tasa de datos.
- Radio de cobertura relativamente pequeño.
- Uso de bandas de radiofrecuencia sin licencia.
- Fácil instalación.
- Bajo coste.

El requisito fundamental del estándar 802.15.4 es un consumo de potencia extremadamente bajo. Esto permite utilizar dispositivos inalámbricos alimentados por baterías, de fácil instalación y bajo coste, en localizaciones en las que es difícil o imposible realizar una instalación por cable. El inconveniente es que, debido al bajo consumo de potencia, el radio de cobertura se ve reducido.

2.4.2 Áreas de aplicación

IEEE 802.15.4 se utiliza en gran número de aplicaciones del ámbito doméstico e industrial. Es una excelente solución para aplicaciones cuyos requisitos coincidan con los expuestos en la sección 2.4.1.

Algunos de los campos de aplicación de IEEE 802.15.4 son los siguientes:

- Automatización en el hogar y seguridad: se utiliza para controlar electrónicamente la calefacción, el aire acondicionado, las luces, o para seguridad contra robos o contra incendios.
- Bienes de consumo: se utiliza para establecer un control remoto en sistemas de entretenimiento en el hogar, cuyos componentes se encuentran distribuidos por varias habitaciones, y también para reemplazar enlaces de cable por enlaces inalámbricos en juguetes y ordenadores, como por ejemplo, entre ratón y ordenador.
- Sanidad: se emplean sensores y dispositivos de diagnóstico conectados mediante una WPAN, para aplicaciones como monitorización durante pruebas de esfuerzo, por ejemplo.
- Control de vehículos: los vehículos contienen muchos sensores y dispositivos de diagnóstico, en los que se utilizan WPANs. Un ejemplo son los sensores de presión en los neumáticos, que no pueden ser conectados mediante cables.
- Agricultura: se utilizan este tipo de redes inalámbricas para monitorizar condiciones ambientales que permitan optimizar el rendimiento de la tierra. En este caso, se necesita amplia cobertura geográfica; se soluciona utilizando topologías de red que permitan la retransmisión de mensajes a través de la red.

2.4.3 Bandas de frecuencia y tasas de datos

IEEE 802.15.4 está diseñado para utilizar bandas de frecuencia sin licencia. Las bandas sin licencia no son las mismas en todo el mundo, por tanto, se utilizan tres bandas de frecuencia diferentes centradas en 868, 915 y 2400 MHz. En la Tabla 2.1 se muestran las distintas bandas de frecuencia utilizadas, sus características y las áreas geográficas de aplicación:

Banda RF	Rango de frecuencias (MHz)	Tasa de datos (Kbps)	Número de canal	Área geográfica
868 MHz	868,3	20	0 (1 canal)	Europa
915 MHz	902-928	40	1-10 (10 canales)	América, Australia
2400 MHz	2405-2480	250	11-26 (16 canales)	Todo el mundo

Tabla 2.1. Bandas de frecuencia utilizadas por el estándar IEEE 802.15.4 (24).

La banda de 2400 MHz es la más utilizada por las siguientes razones:

- Uso sin licencia disponible en todo el mundo.
- Tasa de datos más alta y mayor número de canales.
- Menor consumo de potencia (debido a que se tarda menos tiempo en enviar y recibir porque la tasa de datos es más alta).
- Banda de frecuencias comúnmente empleada en el mercado (también utilizada por *Bluetooth* y el estándar IEEE 802.11).

2.4.4 Arquitectura

La arquitectura definida en el estándar IEEE 802.15.4 se divide en dos niveles: capa física y subcapa MAC (junto con la subcapa LLC). El conjunto de subcapa MAC y subcapa LLC se conoce como capa de enlace de datos. La arquitectura descrita se muestra en la Figura 2.17:

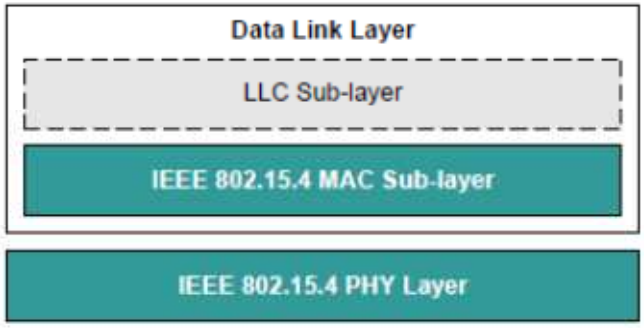

Figura 2.17. Arquitectura de IEEE 802.15.4 (24).

A continuación se definen las funciones y servicios de ambas capas.

➢ Capa física

La capa física actúa como interfaz con el medio físico de transmisión, radio en este caso, e intercambia bits de datos con el medio y con la capa superior, la subcapa MAC.

Las funciones de la capa física con el medio son las siguientes:

- Estimación del canal.
- Comunicaciones a nivel de bit (modulación y demodulación de bits y sincronización de paquetes).

La capa física ofrece a la subcapa MAC los siguientes servicios:

- *PHY Data Service*: proporciona un mecanismo de envío de datos a la subcapa MAC.
- *PHY Management Services*: proporciona mecanismos para controlar la configuración y la funcionalidad de las comunicaciones radio a la subcapa MAC.

La información necesaria para gestionar la capa física se almacena en una base de datos llamada *PHY PIB*.

➢ Subcapa MAC

Las funciones principales de la subcapa de control de acceso al medio (MAC) son las siguientes:

- Proporcionar servicios para que los dispositivos puedan asociarse o desasociarse de la red.
- Proporcionar control de acceso a los canales compartidos.
- Generación de *beacons,* si procede.
- Gestión de *Guaranteed Timeslot* (GTS), si procede.

La subcapa MAC ofrece a la capa superior los siguientes servicios:

- *MAC Data Service (MCPS)*: proporciona un mecanismo de envío de datos a la capa superior.
- *MAC Management Services (MLME)*: proporciona mecanismos para controlar la configuración y la funcionalidad de las comunicaciones radio y de red de la capa superior.

La información necesaria para gestionar la subcapa MAC se almacena en una base de datos llamada *MAC PIB*.

2.4.5 Direccionamiento de dispositivos

Cada dispositivo en una red IEEE 802.15.4 puede tener direcciones de dos tipos:

- Dirección IEEE (MAC): es una dirección de 64 bits asignada por el IEEE, que identifica el dispositivo de manera única. Ningún dispositivo en el mundo puede tener la misma dirección IEEE que otro. También se denomina dirección extendida.
- Dirección corta: es una dirección de 16 bits que identifica de manera local un nodo en una red, es decir, dos nodos en redes distintas pueden tener la misma dirección corta.

Se prefiere el uso de direcciones cortas frente a direcciones IEEE porque así la longitud de los paquetes es menor y se optimiza el uso del ancho de banda de la red. Si un dispositivo no tiene dirección corta asignada, se utiliza su dirección IEEE.

2.4.6 Seguridad

La subcapa MAC del estándar IEEE 802.15.4 ofrece tres modos de seguridad:

- Modo inseguro
- Modo ACL (*Access Control List*)
- Modo seguro

El modo inseguro no toma medidas de seguridad. Los otros dos modos se explican a continuación.

➢ Modo ACL

En este modo cada nodo tiene una lista de control de acceso que contiene las direcciones de los nodos con los que se permite la comunicación. La dirección del nodo origen del mensaje se busca en la lista y el resultado de la búsqueda se comunica a las capas superiores, en las que se decide si se acepta o se rechaza el mensaje.

➢ Modo seguro

En el modo seguro se utilizan distintas medidas de seguridad, que son las siguientes:

- Control de acceso: el procedimiento es similar al del modo ACL, a diferencia de que cuando el origen del mensaje no se identifica, éste no se envía a las capas superiores.
- Cifrado: los datos son cifrados en el origen y descifrados en el destino con la misma clave, de tal manera que sólo los dispositivos que tengan la clave correcta pueden descifrar el mensaje.
- Integridad: se añade al mensaje un código de integridad del mensaje (*Message Integrity Code*, MIC) que permite detectar la manipulación del mensaje por parte de dispositivos que no poseen la clave de cifrado correcta.
- Frescura secuencial: se añade al mensaje un contador de trama. Este valor se compara con el valor de la última trama recibida, almacenado en el dispositivo. Indica el orden de las tramas, pero no contiene información temporal. Se utiliza para identificar mensajes antiguos reenviados.

Hay siete conjuntos de seguridad disponibles en este modo. Cada uno de ellos utiliza diferentes combinaciones de las medidas de seguridad anteriores, pero en todos se emplea un algoritmo de cifrado AES. En la Tabla 2.2 se muestran los diferentes conjuntos con las opciones que se utilizan en cada uno:

Conjunto de seguridad	**Longitud del MIC (bits)**	**Medidas de seguridad**			
		Control de acceso	**Cifrado**	**Integridad**	**Frescura secuencial**
AES-CTR	0	Sí	Sí	No	Opcional
AES-CCM-128	128	Sí	Sí	Sí	Opcional
AES-CCM-64	64	Sí	Sí	Sí	Opcional
AES-CCM-32	32	Sí	Sí	Sí	Opcional
AES-CBC-MAC-128	128	Sí	No	Sí	No
AES-CBC-MAC-64	64	Sí	No	Sí	No
AES-CBC-MAC-32	32	Sí	No	Sí	No

Tabla 2.2. Conjuntos de seguridad en el modo seguro de IEEE 802.15.4 (24).

2.5 Sensores Jennic JN5139

JN5139 (25) es una familia de microcontroladores inalámbricos que cumplen con el estándar IEEE 802.15.4. Los dispositivos incluyen un transceptor inalámbrico que trabaja en la banda de frecuencias ISM (2,4 – 2,5 GHz), un procesador RISC de 32 bits, una memoria ROM de 192 KB, una selección de memorias RAM desde 8 KB hasta 96 KB y un amplio conjunto de periféricos analógicos y digitales. El transceptor cuenta con aceleradores de cifrado AES de 128 bits.

En la Figura 2.18 se muestra un esquema de los dispositivos:

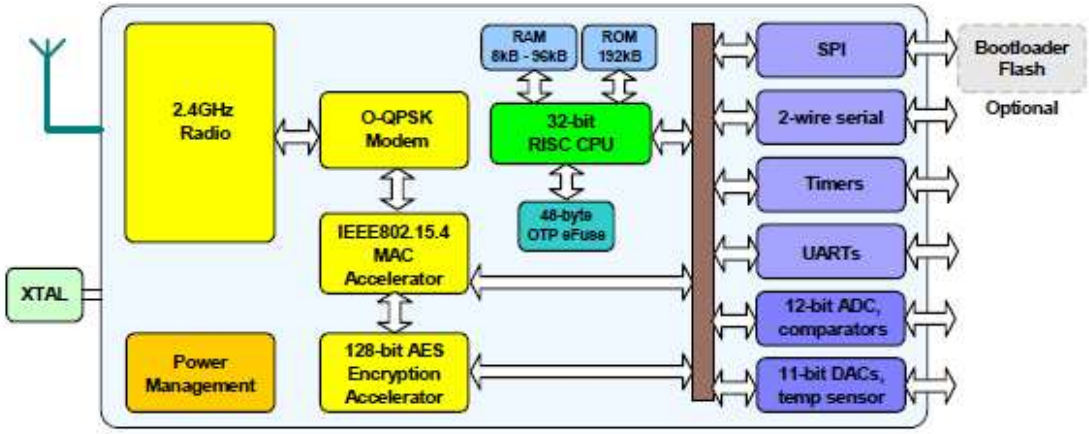

Figura 2.18. Esquema de los dispositivos JN5139 (25).

Los periféricos disponibles son los siguientes:

- Puerto SPI.
- Interfaz serie de dos cables.
- Dos temporizadores/contadores, dos *Sleep Timers* y un *Tick Timer* programables.
- Dos UARTs.
- Conversor analógico-digital de 12 bits.
- Dos comparadores analógicos programables.
- Dos conversores digital-analógico de 11 bits.
- Sensor de temperatura interna y monitor de batería.
- Veintiún GPIO.

El kit de desarrollo de software proporcionado por el fabricante Jennic, denominado JN5139-EK000 IEEE 802.15.4/JenNet Evaluation Kit (26), está formado por los siguientes elementos:

Figura 2.19. Componentes del kit JN5139-EK000 IEEE 802.15.4/JenNet Evaluation Kit (26).

1) Dispositivo *Controller board* con módulo pre-instalado con conector SMA.
2) Dispositivos *Sensor boards* (dos con módulo pre-instalado con conector SMA y dos con módulo pre-instalado con antena cerámica integrada).
3) Dos módulos *plug-in* de alta potencia.
4) Tres antenas con conector SMA.
5) Dos cables puerto USB a puerto serie (FTDI).
6) Un paquete de 10 pilas AAA para las placas.
7) Un CD-ROM con el software y la documentación en archivos pdf.

La documentación actualizada también está disponible en la página web de Jennic, en el área de soporte: www.jennic.com/support.

Para el desarrollo de este proyecto se han utilizado dos kits completos, con objeto de poder construir una red formada por diez dispositivos.

A continuación se explican las características hardware de ambos tipos de dispositivos:

- *Controller board* (27):

  El diseño de la placa se muestra en la Figura 2.20:

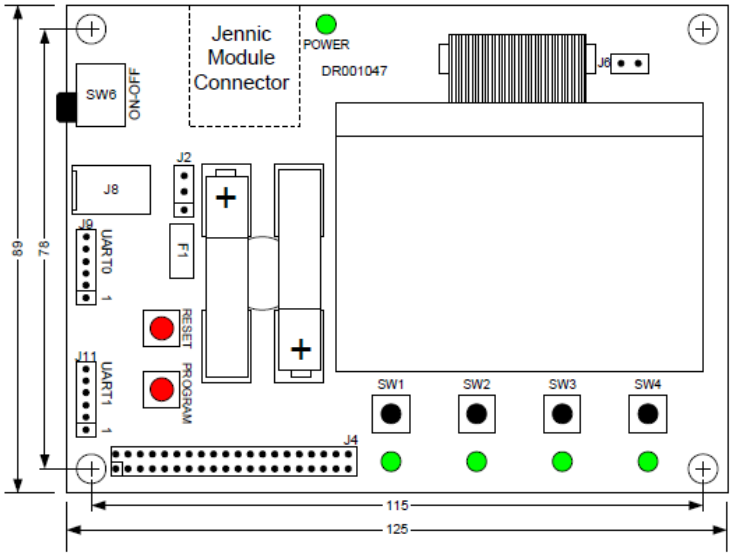

Figura 2.20. Diseño de la placa *controller board* (27).

Está formada por los siguientes elementos:

- Pantalla LCD de 128 x 64 píxeles.
- Interfaces UART para comunicaciones y descarga de programas.
- Sensor de luz.
- Sensor de temperatura.
- Sensor de humedad.
- EEPROM.
- Cuatro interruptores pulsadores, SW1-SW4.
- Cuatro indicadores LED, D1-D4.
- Interruptor SW5, Reset.
- Interruptor SW7, botón del modo programación.
- LED de estado, D9, para indicar cuándo está encendida.

- *Sensor board* (28):

El diseño de la placa se muestra en la Figura 2.21:

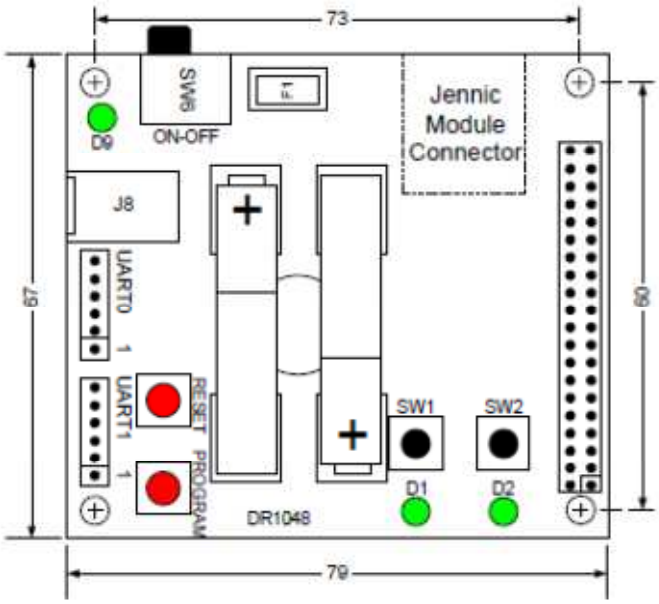

Figura 2.21. Diseño de la placa *sensor board* (28).

Está formada por los siguientes elementos:

- Interfaces UART para comunicaciones y descarga de programas.
- Sensor de luz.
- Sensor de temperatura.
- Sensor de humedad.
- EEPROM.
- Dos interruptores pulsadores, SW1 y SW2.
- Dos indicadores LED, D1 y D2.
- Interruptor SW5, Reset.
- Interruptor SW7, botón del modo programación.
- LED de estado, D9, para indicar cuándo está encendida.

Por tanto, las diferencias entre ambas placas son que la *controller board* dispone de una pantalla LCD, cuatro interruptores y cuatro LEDs mientras que la *sensor board* no posee pantalla LCD y sólo tiene dos interruptores y dos LEDs.

El rango de cobertura de estos dispositivos es de centenas de metros, dependiendo de las condiciones (29).

A la hora de construir la red de sensores inalámbricos todos los dispositivos se han considerado funcionalmente iguales, ya que no se está utilizando la pantalla LCD y el interruptor SW3 de las *controller board* cumple una función meramente informativa y que no afecta al funcionamiento.

En el Anexo se da información más detallada acerca de la instalación del software, la alimentación de las placas y el procedimiento de programación de las mismas.

Capítulo 3

Diseño

3.1 Introducción

En este capítulo se describe la estructura en capas del proyecto, qué se hace en cada capa, las funciones necesarias para llevar a cabo dichas acciones y cómo se relacionan las funciones de diferentes capas entre sí.

Se ha diseñado una pila de protocolos que se divide en tres capas o niveles: nivel de enlace/físico, nivel de red y nivel de aplicación. El nivel de enlace/físico se corresponde con el estándar IEEE 802.15.4. Además, se dispone de una librería de hardware auxiliar, cuyas funciones sirven de apoyo a los tres niveles. En la Figura 3.1 se muestra la pila de protocolos desarrollada:

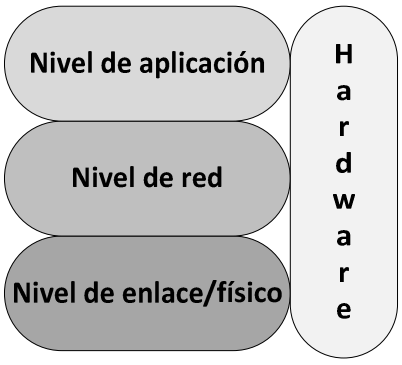

Figura 3.1. Pila de protocolos desarrollada.

En el nivel de aplicación se ha desarrollado una aplicación llamada “ping-pong”, que permite evaluar el funcionamiento del encaminamiento geográfico en la red de sensores.

En el nivel de red se lleva a cabo el descubrimiento de nodos y el encaminamiento geográfico, mediante *Greedy Forwarding*.

El nivel de enlace/físico hace posible el envío y recepción de mensajes por parte de los dispositivos.

La librería de hardware contiene las funciones necesarias para interactuar con los dispositivos, tales como controlar el apagado y encendido de los LEDs, gestionar las pulsaciones de los interruptores, imprimir mensajes y los valores de las variables por pantalla, etc.

Todas las funciones correspondientes a cada una de las capas, su descripción y las relaciones entre ellas se explican a continuación. También se muestra un ejemplo gráfico que ilustra las relaciones entre las funciones de las diferentes capas.

3.2 Nivel de aplicación

En el nivel de aplicación se ha desarrollado una aplicación de ejemplo llamada “ping-pong”. Esta aplicación consta de dos tipos de mensajes. La primera parte de la aplicación consiste en enviar un mensaje “ping” a unas coordenadas geográficas aleatorias. Para ello, se debe pulsar un interruptor en uno de los nodos que forman la red. Este nodo es el origen del mensaje. Cuando se envía el mensaje, se enciende un LED en el nodo origen. Si se quiere volver a apagar el LED, se debe pulsar otro interruptor en el nodo. En la segunda parte de la aplicación se confirma que el mensaje ha llegado al destino final (si las coordenadas corresponden a un nodo) o al nodo más cercano, mediante el envío de un mensaje de confirmación “pong” al nodo origen, el cual se imprime por pantalla.

Las funciones que se han desarrollado para llevar a cabo las acciones necesarias en el nivel de aplicación se describen a continuación.

- `PUBLIC void AppColdStart(void)` → Inicialización general. Desde esta función se llama a todas las demás funciones de inicialización. Se ejecuta al encender los dispositivos o cuando éstos despiertan del modo “dormir”, aunque en este proyecto no se utiliza este modo.

- `PUBLIC void AppWarmStart(void)` → Se utiliza cuando los nodos despiertan del modo “dormir” y se desea mantener los datos almacenados en memoria. Tampoco se utiliza este modo, por tanto, se redirige la llamada a la función anterior.

- `PUBLIC void AppButton1 (void)` → Se ejecuta la acción correspondiente a la pulsación del interruptor SW1 (ver Figuras 2.20 y 2.21) en un nodo. En este caso, se apaga el LED D1, mediante una llamada a la correspondiente función del hardware.

- `PUBLIC void AppButton2 (void)` → Se ejecuta la acción correspondiente a la pulsación del interruptor SW2 (ver Figuras 2.20 y 2.21) en un nodo. En este caso, se

generan unas coordenadas aleatorias como destino y se envía un mensaje de tipo "ping" utilizando la función `AppSendMessage()`.

- `PUBLIC void AppSendMessage (tsPosition dest, uint8 appType, uint8 *data, uint8 dataLength)` → Esta función envía un mensaje a la capa de red, mediante una llamada a la función correspondiente de este nivel (`NetworkSendPacket()`), pasándole los parámetros necesarios. Los mensajes del nivel de aplicación tienen el formato mostrado en la Figura 3.2:

uint8 **msgType**	char **msgString**[80]

Figura 3.2. Formato del mensaje del nivel de aplicación.

El primer parámetro que compone el mensaje es un identificador llamado "msgType" que indica el tipo de mensaje. Su valor es 0 si se trata de un mensaje de tipo "ping" o 1 en el caso de un mensaje de tipo "pong". El segundo parámetro es una cadena de caracteres llamada "msgString" con un tamaño máximo de 80 caracteres, que contiene el mensaje de confirmación "pong" que se imprime por pantalla en la segunda parte de la aplicación "ping-pong".

- `PUBLIC void AppReceiveMessage (tsPosition src, uint8 appType, uint8 *payload, uint8 payloadLength)` → Se encarga de la recepción de mensajes en la capa de aplicación. Esta función se llama desde el nivel de red cuando se ha encontrado el destino final o el nodo más cercano a él y éste ha recibido el mensaje. Los parámetros que recibe son la posición del nodo origen, el identificador del tipo de aplicación, el mensaje correspondiente al nivel de aplicación y su longitud. El diagrama de flujo de esta función se muestra en la Figura 3.3:

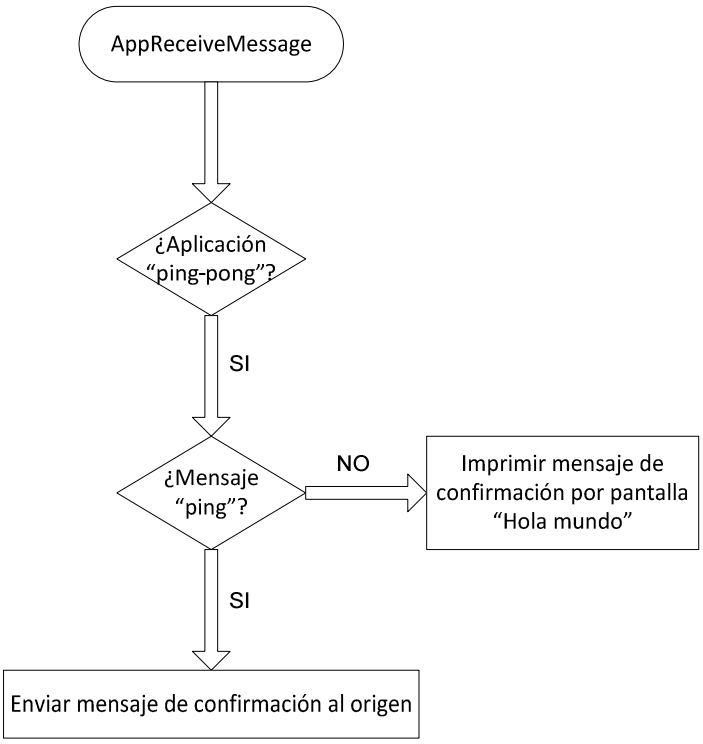

Figura 3.3. Diagrama de flujo de la función AppReceiveMessage() de la capa de aplicación.

3.3 Nivel de red

En el nivel de red se desarrolla la parte más amplia del proyecto.

Por una parte, se construye la red mediante el envío de "hello messages". Cada nodo envía cada 10 segundos un mensaje en *broadcast* formado por sus coordenadas geográficas y su dirección, de tal forma que los nodos receptores puedan identificar al emisor como nodo vecino y aprender sus coordenadas y su dirección. También se almacena información relativa al último "hello message" recibido de cada nodo vecino, con objeto de averiguar si se han producido cambios en la red. Cuando un nodo recibe el primer "hello message" procedente de un vecino se apaga un LED en dicho nodo (que inicialmente se encuentra encendido), para poder comprobar que se están recibiendo.

Por otra parte, el nivel de red es el encargado de ejecutar el encaminamiento geográfico. El protocolo de encaminamiento geográfico que se ha implementado es *Greedy Forwarding*. El procedimiento consiste en que cuando un nodo recibe un mensaje, éste debe buscar cuál de sus nodos vecinos es el más cercano al destino final y reenviarle el mensaje. El siguiente nodo receptor lleva a cabo el mismo procedimiento. Para ello, se calculan las distancias entre el nodo receptor y el destino final, y entre los vecinos y el destino final, utilizando las coordenadas geográficas, y se comparan. Si el nodo receptor es el destino final, o está más cerca del destino final que sus vecinos, se envía el mensaje a la capa de aplicación. Para poder hacer comprobaciones posteriores, siempre que un nodo recibe un mensaje se enciende un LED en dicho nodo.

A continuación, se describen las funciones correspondientes al nivel de red que se han desarrollado para poder llevar a cabo las acciones necesarias.

- `PUBLIC void InitNetwork (void)` → En esta función se inicializa el nivel de red de cada nodo asignándole sus coordenadas geográficas y su dirección. También se inicializa el array de vecinos, colocando en la posición 0 la información relativa al propio nodo. Esta función es llamada desde la capa de aplicación. Las coordenadas están definidas mediante la estructura "tsPosition", como se muestra a continuación:

```
typedef struct
{
    uint8 x;
    uint8 y;
} tsPosition;
```

  En el array de vecinos se almacena la dirección corta y la posición de cada nodo vecino, así como un *timestamp* con información relativa a la última vez que se recibió un "hello message" procedente del vecino en cuestión. Este *timestamp* se utiliza para poder actualizar posteriormente las entradas del array.

- `PUBLIC void NetworkSendHello (uint32 u32Device, uint32 u32ItemBitmap)` → Esta función se utiliza para enviar un "hello message" cada 10 segundos. Se ejecuta cuando se produce la interrupción provocada por el periférico interno *Tick Timer*. Un "hello message" tiene el mismo formato que un paquete de nivel de red (mostrado en la Figura 3.4) a diferencia de que en este caso el campo de datos tiene una longitud de 2 bytes (16 bits) y contiene la dirección corta del

nodo que corresponda. Se envía a través de la correspondiente función del nivel de enlace/físico, `LinkSendFrame()`.

- `PUBLIC void NetworkButton3 (void)` → Se ejecuta la acción correspondiente a la pulsación del interruptor SW3 (ver Figura 2.20) en un nodo. En este caso se utiliza para mostrar el array de vecinos del nodo en un momento dado, con objeto de observar el estado de la red.

- `PUBLIC void NetworkSendPacket (tsPosition dest, uint8 appType, uint8 *data, uint8 dataLength)` → Se utiliza para llevar a cabo el envío de paquetes. Esta función es llamada por el nivel de aplicación, recibiendo como parámetros las coordenadas destino, un identificador que indica la aplicación que se está ejecutando, el mensaje del nivel de aplicación y la longitud de éste. Esta función se encarga de construir el paquete y llamar a la correspondiente función del nivel de enlace/físico, pasándole los parámetros necesarios. El paquete de nivel de red está formado por las coordenadas geográficas del destino y el origen, un identificador que indica la aplicación que se está ejecutando y un campo de datos con un tamaño máximo de 80 bytes en el que se almacena el mensaje correspondiente al nivel de aplicación. El formato del paquete del nivel de red se muestra en la Figura 3.4:

uint8 **xDest**	uint8 **yDest**	uint8 **xSrc**	uint8 **ySrc**	uint8 **appType**	uint8 **payload[80]**

Figura 3.4. Formato del paquete del nivel de red.

- `PUBLIC void NetworkReceivePacket (uint8 payload[], uint8 payloadLength)` → Esta función se encarga de recibir paquetes procedentes del nivel de enlace/físico y procesarlos. Los parámetros que recibe son los datos que forman el paquete de nivel de red y su longitud. El diagrama de flujo de esta función se muestra en la Figura 3.5:

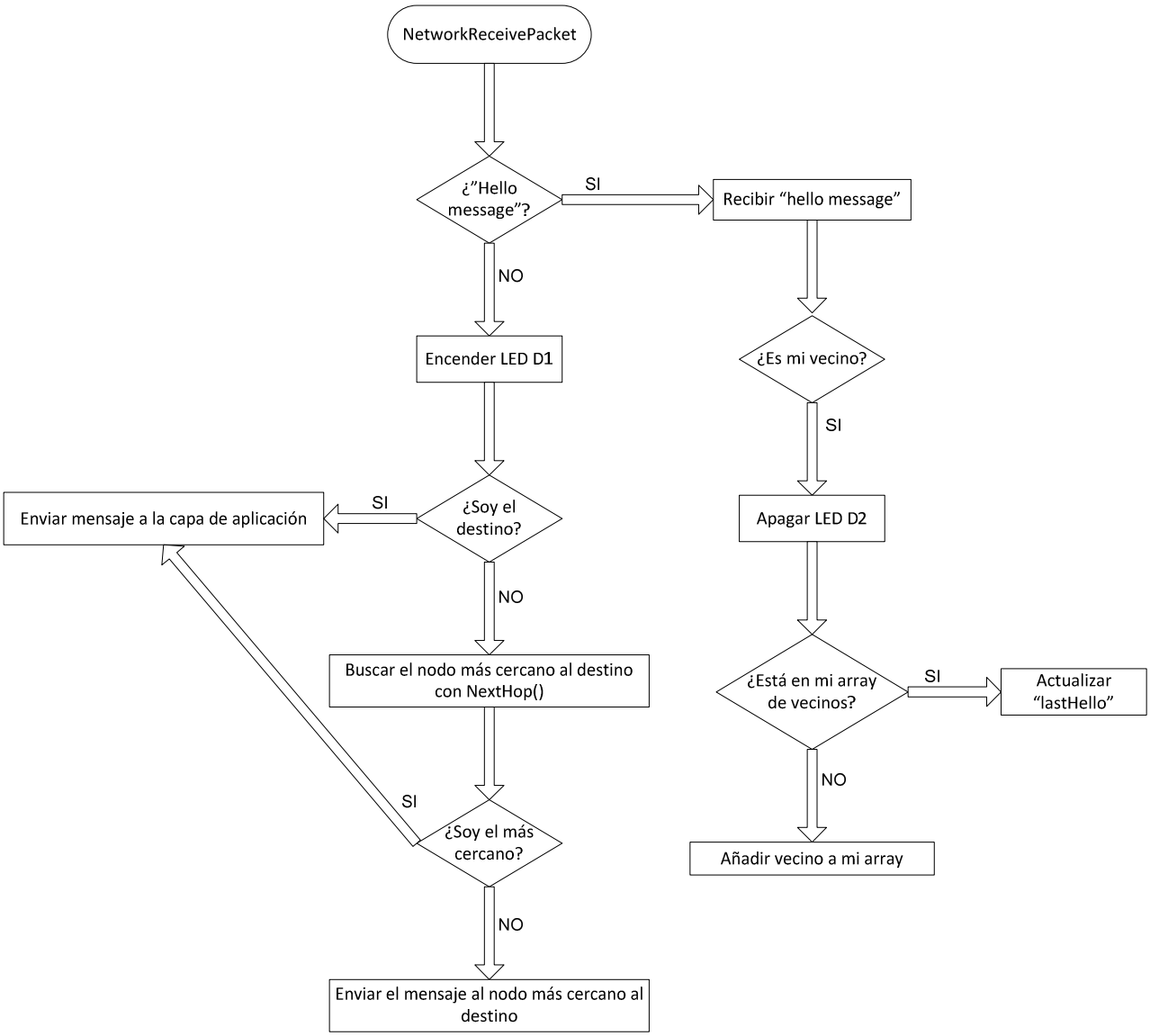

Figura 3.5. Diagrama de flujo de la función NetworkReceivePacket() de la capa de red.

- `PUBLIC uint16 NextHop (tsPosition dest)` → En esta función se busca el nodo más cercano al destino final entre los vecinos del nodo receptor y se devuelve su dirección. El parámetro necesario para llevar a cabo el procedimiento es la posición geográfica del destino. Se calculan las distancias entre el nodo receptor y el destino final, y entre los vecinos y el destino final, utilizando las coordenadas geográficas, y se comparan. El nodo más cercano será aquel que tenga la menor distancia hasta el destino final. También se comprueba que el tiempo transcurrido desde la recepción del último "hello message" no supera un cierto umbral, en este caso, 30 segundos. Si se ha sobrepasado el umbral, se descarta el nodo correspondiente.

- `PUBLIC uint16 CalculateDistanceSquared (tsPosition pos1, tsPosition pos2)` → Esta función se utiliza para calcular la distancia al cuadrado entre dos nodos a partir de sus coordenadas geográficas, las cuales se reciben como parámetros. Como las distancias sólo son necesarias para compararlas entre sí, se utiliza la distancia al cuadrado para evitar la operación de raíz cuadrada.

3.4 Nivel de enlace/físico

A través del nivel de enlace/físico se realiza el envío y la recepción de mensajes.

Para poder realizar estas acciones, se han desarrollado las siguientes funciones:

- `PUBLIC void LinkSendFrame (uint16 u16DestAddr, uint8 data[], uint8 dataLength)` → En esta función se construye la trama y se envía a través del medio físico. Los parámetros necesarios son la dirección destino, el paquete de nivel de red y su longitud.

- `PUBLIC void LinkReceiveFrame (uint8 payload[], uint8 payloadLength)` → Cuando el hardware detecta que se ha recibido un mensaje se llama a esta función pasándole como parámetros los datos que forman la trama y la longitud de la misma. Se encarga de extraer los parámetros necesarios para llamar a la correspondiente función del nivel de red, `NetworkReceivePacket()`.

3.5 Librería de hardware

Para permitir la interacción humana con los dispositivos y la gestión de los LEDs, los interruptores, los temporizadores, etc., son necesarias las siguientes funciones realizadas por el hardware:

- `PUBLIC void vInitSystem(void)` → Inicializa la pila y las interfaces del hardware.

- `PUBLIC void vInitEndpoint(void)` → Inicializa determinados parámetros de los nodos, como habilitar el uso de los interruptores.

- `PUBLIC void vLightBulbInit(void)` → Inicializa los LEDs.

- `PUBLIC void vInitUart(void)` → Inicializa el puerto serie.

- `PUBLIC void InitTimer(void)` → Inicializa el *Tick Timer*.

Todas estas funciones de inicialización son llamadas desde la capa de aplicación.

- `PUBLIC void vProcessIncomingHwEvent(AppQApiHwInd_s *psAHI_Ind)` → En esta función se detecta cuándo se ha pulsado un interruptor en un nodo, se identifica cuál es el interruptor que se ha pulsado y se llama a la correspondiente función del nivel de aplicación o de red, según corresponda.

- `PUBLIC void vProcessIncomingData (MAC_McpsDcfmInd_s *psMcpsInd)` → En esta función se detecta cuándo un nodo recibe un mensaje y se

extraen los parámetros necesarios para llamar a la correspondiente función del nivel de enlace/físico, `LinkReceiveFrame()`.

- `PUBLIC void vPrintString (char *pcMessage)` → Mediante esta función se imprime por pantalla el mensaje que se le pasa como parámetro.

- `PUBLIC void vDisplayHex (uint32 u32Data)` → Esta función se encarga de imprimir por pantalla el valor hexadecimal de la variable que se le pasa como parámetro.

3.6 Ejemplo gráfico

En la Figura 3.6 se representan las relaciones entre las principales funciones de la pila de protocolos. Para simplificar se muestra un ejemplo concreto, que corresponde al caso en que el siguiente salto calculado por el nodo emisor es el destino final del mensaje.

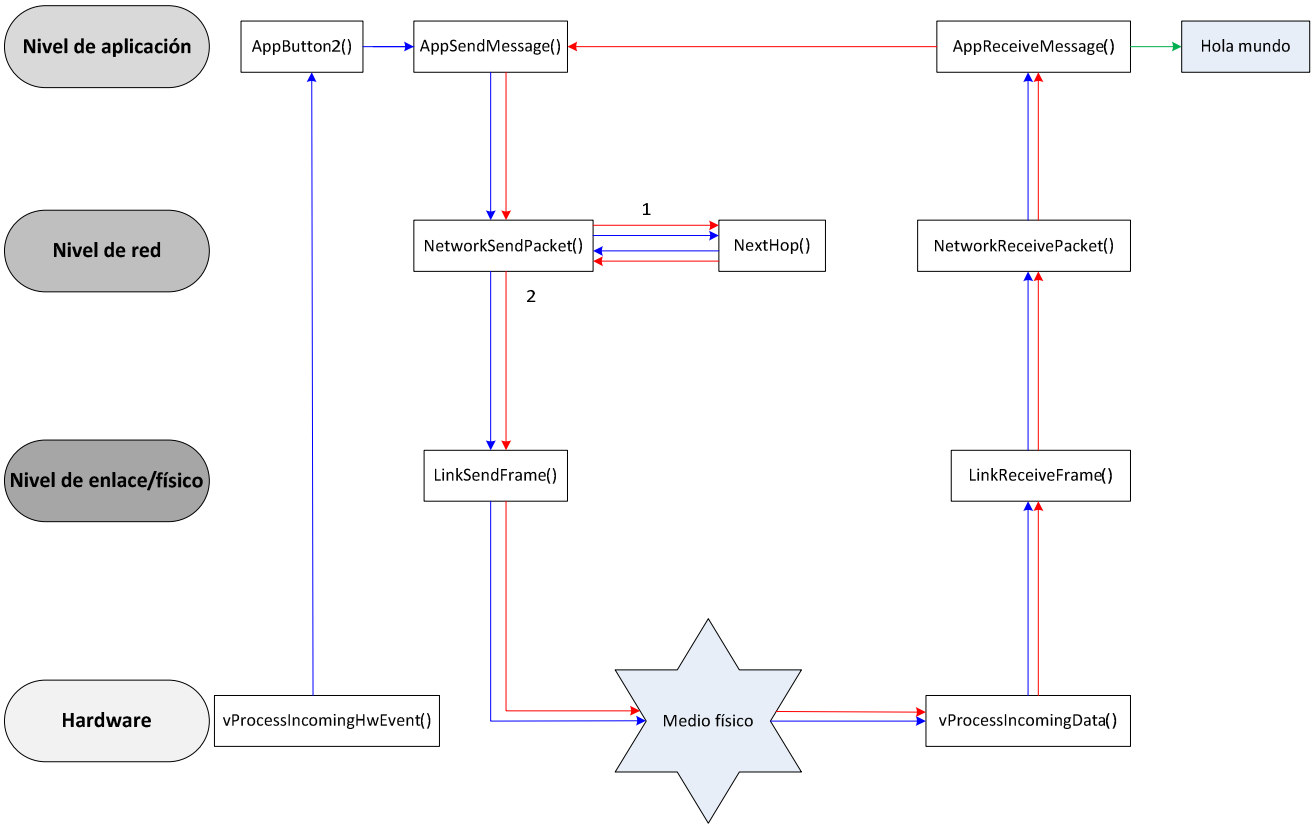

Figura 3.6. Ejemplo gráfico de las relaciones entre las principales funciones.

En la Figura 3.6 se ha representado el camino seguido por el mensaje “ping” en azul y el flujo del mensaje “pong” en rojo, hasta que se recibe y se imprime por pantalla “Hola mundo”. El punto de partida es la función del hardware `vProcessIncomingHwEvent()`que se ejecuta cuando se pulsa el interruptor SW2 en un nodo.

En el Capítulo 4 se explica detalladamente cómo se han implementado todas estas funciones.

Capítulo 4

Implementación y evaluación

4.1 Introducción

La red de sensores inalámbricos que se ha utilizado para la implementación y la evaluación del funcionamiento del encaminamiento geográfico se muestra en la Figura 4.1:

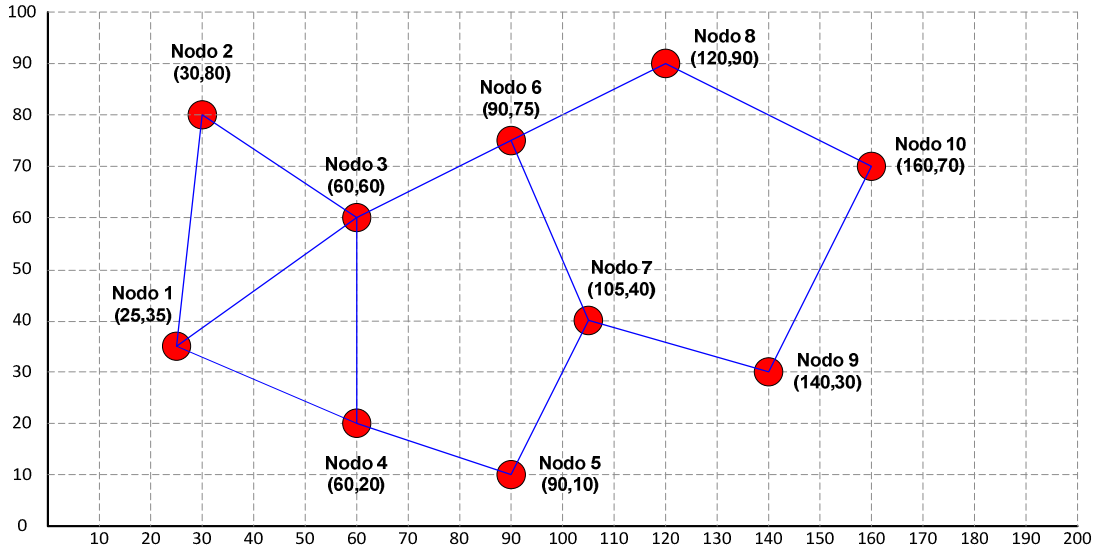

Figura 4.1. Esquema de la red de sensores inalámbricos utilizada.

En el escenario representado en la Figura 4.1 cada círculo rojo representa un nodo de la red de sensores inalámbricos. Como podemos observar, se ha utilizado una red formada por diez nodos. Éstos están distribuidos espacialmente e identificados mediante sus coordenadas cartesianas. Las líneas azules representan las conexiones entre los nodos que son vecinos entre sí, de tal forma que pueden comunicarse directamente entre ellos, mientras que los nodos que no son vecinos necesitan encontrar una trayectoria multisalto para comunicarse. Por comodidad, los nodos han sido etiquetados, para poder referirse a ellos posteriormente, con los nombres de nodo 1, nodo 2, etc.

Los dispositivos que se han utilizado para construir la WSN representada en la Figura 4.1 son los correspondientes al fabricante Jennic, modelo JN5139, descritos en el Capítulo 2. Estos dispositivos están pensados para ser programados con lenguaje C de alto nivel (25), por tanto, se ha utilizado este lenguaje para implementar el proyecto.

La estructura del proyecto consiste en una serie de ficheros .c con el código fuente y sus correspondientes ficheros de cabecera .h (que son necesarios para que cada capa de la pila de protocolos pueda realizar llamadas a las funciones de las capas adyacentes) y un fichero *makefile* que se utiliza para compilar el código. Los ficheros .c se describen brevemente a continuación:

- PingPong.c: en este fichero se han implementado las funciones correspondientes a la aplicación “ping-pong” en el nivel de aplicación.
- ApplicationLayer.c: en este fichero se gestionan las funciones que constituyen el punto de entrada de ejecución cuando se encienden los nodos o cuando despiertan del modo “dormir”. Estas funciones también pertenecen al nivel de aplicación de la pila de protocolos.
- NetworkLayer.c: este fichero está formado por las funciones implementadas correspondientes al nivel de red de la pila de protocolos.
- LinkLayer.c: en este fichero se desarrollan las funciones del nivel de enlace/físico que permiten el envío y la recepción de tramas.
- Hardware.c: en este fichero se encuentran las funciones correspondientes a la librería de hardware, que se utilizan para poder interactuar con los dispositivos.

A cada uno de estos ficheros .c le corresponde un fichero de cabecera .h con el mismo nombre.

En este capítulo se explica en detalle cómo se han implementado las principales funciones necesarias en cada nivel de la pila de protocolos y las pruebas que se han llevado a cabo para comprobar su correcto funcionamiento.

4.2 Nivel de aplicación

Para facilitar la evaluación del funcionamiento del encaminamiento geográfico en la WSN de la Figura 4.1, se ha desarrollado una aplicación de ejemplo llamada “ping-pong”. Como ya se explicó en el Capítulo 3, esta aplicación consiste en generar aleatoriamente unas coordenadas destino y enviar un mensaje de tipo “ping” a esas coordenadas. Cuando el mensaje ha llegado al destino final (si las coordenadas corresponden a un nodo) o al nodo más cercano a las mismas, se envía al origen un mensaje de confirmación de tipo “pong”. Para poder realizar

comprobaciones del camino seguido por el mensaje, cada vez que se envía un mensaje se enciende un LED en el nodo origen, que posteriormente podrá ser apagado pulsando un interruptor.

Para llevar a cabo la aplicación "ping-pong", las funciones principales que se han implementado son las siguientes:

➢ Función AppButton2()

El punto de partida de la aplicación "ping-pong" es la pulsación del interruptor SW2 en uno de los nodos. Cuando el hardware detecta que se ha pulsado este interruptor en uno de los nodos, se realiza una llamada a la función `AppButton2()` del nivel de aplicación.

En esta función se generan unas coordenadas destino de manera aleatoria. Para ello, se utiliza la función `rand()`de la librería `stdlib.h.` La función `rand()` (30) genera números enteros pseudo-aleatorios comprendidos en un rango entre 0 y un número máximo establecido. Se dice que `rand()`devuelve números pseudo-aleatorios porque se repite la misma secuencia de números aleatorios cada vez que se ejecuta la función. Para obtener secuencias distintas se puede utilizar la función `srand()`, también disponible en la librería `stdlib.h`. Esta función establece el valor inicial (semilla) que utiliza la función `rand()`para calcular la secuencia de números aleatorios. Sin embargo, es necesario que esa semilla vaya cambiando, o de lo contrario, al usar la misma semilla con `srand()`, se obtiene todas las veces la misma secuencia de números al ejecutar `rand()`. Por tanto, la solución para obtener una secuencia de números realmente aleatoria es pasarle como parámetro a la función `srand()` un valor distinto para la semilla cada vez que se ejecuta. Se suele utilizar la hora actual como valor para la semilla, ya que este valor siempre será distinto excepto si se ejecuta la aplicación varias veces en el mismo segundo, lo cual es improbable. El problema es que los dispositivos utilizados en este proyecto no disponen de un reloj interno que mida el tiempo. Para solucionar este inconveniente se ha utilizado uno de los periféricos internos de los dispositivos, el *Tick Timer*. El *Tick Timer* (31) es capaz de contar los ciclos de un reloj de 16 MHz hasta alcanzar un valor predefinido. El valor inicial también se puede especificar. Este periférico se inicializa en la función `InitTimer()` de la librería de hardware. Se puede obtener en cualquier momento el valor actual del *Tick Timer* mediante una llamada a la función `u32AHI_TickTimerRead()`, disponible en la API de periféricos internos. Por tanto, para poder generar coordenadas aleatorias, se utiliza como semilla en la función `srand()` el valor que tiene el *Tick Timer* en el momento en que se ejecuta la función `AppButton2()`.

Mediante el proceso que se acaba de explicar ya se puede obtener una secuencia de números aleatorios. En particular, se necesita generar unas coordenadas "x" e "y" que se encuentren dentro del área geográfica que se ha definido. Si se observa la Figura 4.1, la coordenada "x" puede tener un valor comprendido entre 0 y 200, mientras que la coordenada "y" puede tener un valor comprendido entre 0 y 100. Para ello, una vez que se han obtenido los dos números aleatorios, se hace una normalización para que se encuentren dentro del rango deseado.

Una vez que se dispone de las coordenadas destino, se procede a construir el mensaje "ping". Como se vio en el Capítulo 3, los mensajes de la capa de aplicación están formados por un identificador que indica el tipo de mensaje y una cadena de caracteres. En este caso, el identificador de mensaje toma el valor 0, que corresponde al tipo "ping", y la cadena de ejemplo es "Hola mundo". El mensaje se envía utilizando la función

AppSendMessage(), que llama a la correspondiente función del nivel de red, NetworkSendPacket().

En esta función también se enciende el LED D1 del nodo origen del mensaje, mediante la llamada a la función del hardware vLedControl(), definida en la librería de Jennic LedControl.h.

➢ Función AppButton1()

Cuando el hardware detecta que se ha pulsado el interruptor SW1 en uno de los nodos, se llama a esta función. Realizando una llamada a la función del hardware vLedControl()se encarga de apagar el LED D1 del nodo correspondiente.

Esta función se utiliza para facilitar la fase de evaluación del funcionamiento de la red, porque permite apagar los LEDs de los nodos a través de los que se ha transmitido un mensaje sin tener que reiniciar los dispositivos. De esta manera, se puede enviar otro mensaje y observar la trayectoria seguida por éste, sin que se produzca confusión con el camino seguido por el mensaje anterior.

➢ Función AppReceiveMessage()

En esta función se ejecuta la segunda parte de la aplicación "ping-pong". Cuando se recibe un mensaje procedente del nivel de red, se hace un *casting* para convertir los datos recibidos en un mensaje de aplicación con estructura "tsPingPongAppMessage". Esta estructura se define de la siguiente forma:

```
#define MAX_APP_PAYLOAD_SIZE    80

typedef struct
{
    uint8 msgType;
    char msgString [MAX_APP_PAYLOAD_SIZE];
} tsPingPongAppMessage;
```

Se extrae del mensaje el parámetro que indica el tipo de mensaje. Si es de tipo "ping" (el identificador es igual a 0) significa que el mensaje ha alcanzado el destino final o el nodo más cercano a éste y se envía el correspondiente mensaje "pong" al nodo origen utilizando la función AppSendMessage().Este mensaje está formado por el identificador, que toma el valor 1, y la misma cadena de caracteres que contiene el mensaje "ping". El mensaje "pong" se encamina de la misma manera que el mensaje "ping", teniendo en cuenta que ahora las coordenadas destino del mensaje en realidad son las del nodo origen. El camino seguido por el mensaje "pong" no tiene por qué coincidir necesariamente con la secuencia de nodos atravesada por el mensaje "ping". Si, por el contrario, el mensaje recibido es de tipo "pong", se imprime por pantalla la cadena de caracteres que contiene utilizando la función del hardware vPrintString(). La situación que corresponde a este caso es que el mensaje de confirmación ha alcanzado el nodo origen.

4.3 Nivel de red

En el nivel de red se construye la red de la Figura 4.1 y se ejecuta el encaminamiento geográfico. Aquí es donde se desarrolla la parte más importante del proyecto. El protocolo de encaminamiento geográfico que se ha implementado es *Greedy Forwarding* (ver Capítulo 2). Resumidamente, este protocolo consiste en que cada nodo que recibe un mensaje busca entre sus nodos vecinos cuál es el más cercano al destino, calculando la distancia entre ellos y el destino a partir de sus coordenadas geográficas, y le reenvía el mensaje. Se repite el proceso hasta que se alcanza el destino final.

Las características que han determinado la elección de este protocolo de encaminamiento geográfico son que se necesita almacenar muy poca información en los dispositivos para encaminar paquetes (lo que supone un ahorro de memoria), que la información necesaria para tomar las decisiones de reenvío es reducida (con lo que se disminuye la longitud de los paquetes enviados), y que es sencillo de implementar.

Para llevar a cabo las acciones correspondientes al nivel de red se han implementado las siguientes funciones principales:

➢ Función InitNetwork()

El primer paso es gestionar la localización de los sensores. Como los dispositivos no disponen de GPS ni ningún otro mecanismo de localización, se deben asignar sus coordenadas geográficas "a mano". Para ello, se han creado diez ficheros llamados NetworkLayer1.h, NetworkLayer2.h, etc., en los que se definen las coordenadas geográficas y la dirección de cada nodo, respectivamente. Las coordenadas geográficas de los nodos se muestran en la Figura 4.1. Las direcciones que se han asignado son direcciones cortas de 16 bits, como se define en el estándar IEEE 802.15.4. Para facilitar la evaluación del funcionamiento, las direcciones asignadas se corresponden con la etiqueta de cada nodo, es decir, el nodo 1 tiene la dirección "1", el nodo 2 tiene la dirección "2" y así sucesivamente. También se define el identificador de red (PAN_ID) para que los nodos tengan conocimiento de que pertenecen a la misma red y puedan comunicarse entre ellos.

En la función `InitNetwork()`se asigna a cada nodo sus coordenadas geográficas "x" e "y" y su dirección corta. Las coordenadas geográficas se definen mediante una estructura llamada "tsPosition", que se muestra a continuación:

```
typedef struct
{
    uint8 x;
    uint8 y;
} tsPosition;
```

También se define una estructura formada por los campos que se utilizan para almacenar la información necesaria de cada vecino, llamada "tsNeighbour":

```
typedef struct
{
    uint16 u16ShortAddr;
    tsPosition position;
```

```
    uint32 lastHello;
} tsNeighbour;
```

Además, se inicializa el array de vecinos colocando en la posición 0 del array sus propias coordenadas y dirección y el número de "hello messages" enviados. Este campo se inicializa con el valor 0 porque ningún "hello message" ha sido enviado todavía. El número de "hello messages" enviados se utilizará posteriormente para hacer un cálculo aproximado de la última vez que se recibió un "hello message" procedente de cada vecino. El motivo de almacenar las propias coordenadas y dirección en el array de vecinos es hacer más sencilla la búsqueda del nodo más cercano al destino, ya que se debe comparar la distancia entre los vecinos y el destino con la distancia entre el propio nodo y el destino. El resto de posiciones del array se rellenan a medida que se descubren nuevos vecinos, mediante la recepción de "hello messages". Se utiliza un array para almacenar la información relativa a todos los posibles vecinos porque los dispositivos no disponen de una memoria dinámica de fácil acceso.

La posición, la dirección y el array de vecinos de cada nodo se almacenan en una estructura llamada "tsCoordData", que se muestra a continuación:

```
#define MAX_NEIGHBOURS     10

typedef struct
{
     tsPosition position;
     uint16  u16ShortAddr;
     tsNeighbour neighbours[MAX_NEIGHBOURS];
} tsCoordData;
```

➢ Función NetworkSendHello()

Para que los nodos de la red puedan conocer a sus vecinos y aprender su dirección y sus coordenadas, se realiza un algoritmo de descubrimiento mediante el envío de "hello messages". Un "hello message" es un mensaje formado por las coordenadas y la dirección de un nodo y un identificador de aplicación que se utiliza para diferenciar los "hello messages" de los mensajes de la aplicación "ping-pong". Cada nodo envía en *broadcast* su "hello message" cada 10 segundos, utilizando la función `NetworkSendHello()`.

En esta función se construye el "hello message". Para compatibilizar la recepción de "hello messages" con la de mensajes de la aplicación "ping-pong", éstos tienen la siguiente estructura:

```
#define ADDR_LENGTH     2

typedef struct
{
    uint8 xDest;
    uint8 yDest;
    uint8 xSrc;
    uint8 ySrc;
    uint8 appType;
    uint8 payload[ADDR_LENGTH];

} tsNetworkHello;
```

Para indicar que el mensaje se envía en *broadcast,* las coordenadas "x" e "y" destino toman el valor 0xFF. Las coordenadas origen son las del nodo que corresponda. El tipo de aplicación toma el valor 0, para poder distinguir un "hello message" de un mensaje de aplicación "ping-pong", cuyo identificador toma el valor 1. La dirección del nodo correspondiente se almacena en un campo de datos de 16 bits (2 bytes de longitud). Esta estructura se ha definido así para que el compilador reserve exactamente esta cantidad de memoria. De lo contrario, se añadirían bytes de *padding* entre los campos.

Una vez construido el "hello message" se envía utilizando la función `LinkSendFrame()` del nivel de enlace/físico y se incrementa el contador de "hello messages" enviados por el nodo, que se encuentra almacenado en la posición 0 del array de vecinos. Para realizar un envío en *broadcast* se debe enviar la trama a la dirección 0xFFFF, definida en el estándar IEEE 802.15.4 (32).

Para poder enviar un "hello message" cada 10 segundos se utiliza el *Tick Timer*. El procedimiento seguido se explica más adelante, en la función `InitTimer()` del hardware.

➢ Función NetworkButton3()

Esta función sirve para mostrar por pantalla el array de vecinos de un nodo, con objeto de observar cuál es el estado de la red. Recibe la llamada desde el hardware cuando se detecta la pulsación del interruptor SW3 en un nodo, por tanto, sólo puede ser utilizada por las *controller boards* (ver Capítulo 2).

Cuando un nodo recibe un "hello message" procedente de otro nodo, se actualiza el array de vecinos almacenando información sobre su posición y sus coordenadas y actualizando el contador de "hello messages" recibidos.

El propósito de los "hello messages", además de que cada nodo de la red sea capaz de conocer la posición y la dirección de sus vecinos, también es que los nodos puedan detectar si se ha producido algún cambio en la red.

En la red que se está utilizando en este proyecto se asume que todos los nodos son estáticos para simplificar la evaluación del funcionamiento, pero en un escenario real los nodos pueden estar en movimiento. Un nodo en movimiento puede alejarse y salirse del área de alcance de otro nodo que antes era vecino suyo. Además los nodos pueden sufrir fallos y la topología de la red puede cambiar si se añaden nodos nuevos, por ejemplo. Para detectar los cambios producidos en la red, si un dispositivo no ha recibido ningún "hello message" procedente de un vecino determinado durante un período de tiempo establecido, éste se elimina del array de vecinos.

En este proyecto, con el objeto de poder realizar pruebas para comprobar el funcionamiento del encaminamiento geográfico, se han desplegado los dispositivos en un área pequeña. En un escenario real la distancia física entre nodos puede ser del orden de decenas o incluso centenas de metros, por tanto, todos los nodos que estén dentro del rango de cobertura de otro serán vecinos suyos. Para simular un entorno real, se ha definido un umbral para determinar qué nodos están dentro del alcance de otros.

A la hora de implementar esta función se han tenido en cuenta estas consideraciones, de tal forma que se ha definido un umbral de distancia, calculado a partir de las coordenadas geográficas asignadas a los nodos, para que la red se corresponda con la mostrada en la Figura 4.1. Puesto que para comparar las distancias entre nodos se utiliza la distancia al cuadrado, el umbral se corresponde con la distancia máxima al cuadrado entre dos nodos vecinos, siendo su valor 2050 (metros cuadrados, si se considera que las unidades de los ejes cartesianos representados en la Figura 4.1 están en metros). También se ha definido un período de 30 segundos como tiempo máximo transcurrido desde la recepción del último "hello message" antes de que el vecino se elimine del array.

Cuando se ejecuta la función `NetworkButton3()`, primero se muestran por pantalla la dirección y las coordenadas del propio nodo, utilizando las correspondientes funciones del hardware. A continuación se muestran las coordenadas y direcciones de los vecinos que tiene el nodo en ese momento.

Para determinar qué nodos siguen siendo vecinos, hay que comprobar que el tiempo transcurrido desde la recepción del último "hello message" de cada vecino no exceda del umbral de 30 segundos. Como los dispositivos no disponen de un reloj interno para medir el tiempo, se ha utilizado un contador de "hello messages" enviados en el propio nodo y un contador de "hello messages" recibidos de cada vecino. El procedimiento que se lleva a cabo para determinar si la recepción del último "hello message" se produjo dentro del período establecido es el siguiente: cuando un nodo recibe un "hello message" procedente de un vecino, se actualiza el contador de "hello messages" recibidos de este vecino con el valor actual de "hello messages" enviados. Los "hello messages" se envían cada 10 segundos, por tanto, el umbral de 30 segundos es igual a 3 "hello messages" enviados. Para determinar si el último "hello message" se recibió dentro de este período, se restan el valor actual del contador de "hello messages" enviados y el valor actual del contador de "hello messages" recibidos. Si se han enviado menos de 3 "hello messages" desde que se recibió el último "hello message" procedente del vecino, la diferencia entre estos valores será menor que 3 (que es el valor del umbral), lo que significa que el nodo sigue activo. Si, en cambio, la diferencia es mayor o igual que el umbral, se salta esa posición del array porque ese nodo no se considera vecino. Esto equivale a eliminar el nodo del array, pero esta manera es mucho más sencilla de implementar.

Este procedimiento se lleva a cabo para cada vecino almacenado en el array.

➢ Función NetworkSendPacket()

Esta función se utiliza para enviar paquetes. Tras construir el paquete de nivel de red se busca la dirección del siguiente salto utilizando la función `NextHop()`. Si la dirección del siguiente salto coincide con la del nodo emisor, significa que está más cerca del destino que sus vecinos y se transmite el paquete a la capa de aplicación, mediante una llamada a la función `AppReceiveMessage()`, indicando las coordenadas del nodo origen incluidas en el paquete para que la aplicación envíe de vuelta un mensaje de confirmación. Si no, se envía el paquete al siguiente salto utilizando la función del nivel de enlace/físico `LinkSendFrame()`.

La estructura de los paquetes enviados en el nivel de red está formada por las coordenadas del destino y el origen, el tipo de aplicación y un campo de datos de 80 bytes como máximo. Esta estructura se denomina "tsNetworkPacket" y se define a continuación:

```
#define MAX_NET_PAYLOAD_SIZE    80

typedef struct
{
    uint8 xDest;
    uint8 yDest;
    uint8 xSrc;
    uint8 ySrc;
    uint8 appType;
    uint8 payload[MAX_NET_PAYLOAD_SIZE];

} tsNetworkPacket;
```

➢ Función NetworkReceivePacket()

La función `NetworkReceivePacket()`se encarga de recibir paquetes procedentes del nivel de enlace/físico. Primero, se hace un *casting* para convertir los datos recibidos en un paquete de nivel de red con estructura "tsNetworkPacket". A continuación se extraen del paquete los parámetros necesarios. La función debe identificar si el mensaje es un "hello message" o un mensaje de aplicación, mediante el valor del identificador de aplicación.

Si el mensaje es un "hello message" (el identificador es igual a 0), se lleva a cabo el siguiente procedimiento:

- Se calcula la distancia al cuadrado entre el nodo emisor y el nodo receptor. Si ésta es menor que un determinado umbral (2050), se considera que está dentro del alcance y, por tanto, el nodo emisor es identificado como vecino.
- Cuando se recibe el primer "hello message" procedente de un vecino, se apaga el LED D2 en el nodo (que inicialmente se encuentra encendido). Esto sirve para comprobar de una manera sencilla que los "hello messages" se están recibiendo.
- Se comprueba si el vecino ya está en el array.
- Si el vecino ya está almacenado en el array, se actualiza el contador de "hello messages" recibidos de ese vecino con el valor actual del contador de "hello messages" enviados por el nodo receptor.
- Si el vecino no se encuentra almacenado en el array, se actualiza el array almacenando la información del vecino en la primera posición vacía. Se guardan su dirección y sus coordenadas, contenidas en el "hello message", y se actualiza el contador de "hello messages" recibidos de ese vecino con el valor actual del contador de "hello messages" enviados por el nodo receptor.

Por el contrario, si el mensaje recibido no es un "hello message" (el identificador es distinto de 0), se ejecuta el siguiente procedimiento:

- Se enciende el LED D1 en el nodo receptor. Esto sirve para comprobar la trayectoria seguida por el mensaje de forma sencilla y visual, facilitando la tarea de evaluación del funcionamiento del encaminamiento geográfico.
- Se comprueba si el nodo receptor es el destino final, comparando las coordenadas geográficas de ambos.
- Si el nodo receptor es el destino final, se transmite el mensaje a la capa de aplicación mediante una llamada a la función `AppReceiveMessage()`,

indicando las coordenadas del nodo origen, que ahora se convierte en el destino del mensaje de confirmación.
- Si no, se busca el nodo más cercano al destino en el array de vecinos.
- Si no hay ningún vecino más cerca que el propio nodo, significa que se ha alcanzado el nodo más cercano al destino y se traspasa el mensaje a la capa de aplicación, del mismo modo que se ha explicado para el caso en que se alcanza el destino final.
- Si algún vecino está más cerca del destino, se reenvía el mensaje al siguiente salto mediante la función del nivel de enlace/físico `LinkSendFrame()`.

➢ Función NextHop()

Se ha implementado esta función para que se encargue de encontrar el nodo más cercano al destino final en el array de vecinos. Para ello, se lleva a cabo el siguiente proceso:

- Se calcula la distancia al cuadrado entre el destino y el propio nodo, utilizando la función `CalculateDistanceSquared()`.
- Se calcula la distancia al cuadrado entre el destino y los nodos del array de vecinos, utilizando la función `CalculateDistanceSquared()`.
- Se comprueba que la diferencia entre el contador de "hello messages" recibidos de cada vecino y el contador de "hello messages" enviados sea menor que el umbral. Si no es así, se descarta el vecino que corresponda.
- Se comparan la distancia calculada entre el nodo y el destino con la distancia calculada entre cada uno de los vecinos y el destino. El nodo más cercano al destino es aquel que tiene una distancia menor hasta el destino.
- Se devuelve la dirección del nodo más cercano al destino.

➢ Función CalculateDistanceSquared()

Esta función calcula la distancia geográfica al cuadrado entre dos posiciones, especificadas por sus coordenadas geográficas: (x_1, y_1) y (x_2, y_2). Como las distancias sólo son necesarias para compararlas entre sí, se utilizan las distancias al cuadrado para evitar la operación de la raíz cuadrada.

La fórmula que se ha utilizado para calcular la distancia al cuadrado se muestra a continuación:

$$distancia^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

4.4 Nivel de enlace/físico

Esta capa es necesaria para el envío y la recepción de tramas. El nivel de enlace/físico se corresponde con el estándar IEEE 802.15.4. Las características principales de este estándar se presentaron en el Capítulo 2.

La MTU (33) en el nivel físico de IEEE 802.15.4 es de 127 octetos y el tamaño máximo de la cabecera es de 25 octetos. Por tanto, la MTU en el nivel de enlace es de 102 octetos. El tamaño de la cabecera del nivel de enlace dependerá del modo de seguridad empleado.

Las dos funciones pertenecientes a esta capa que se han implementado se describen a continuación:

➢ Función LinkSendFrame()

Recibe llamadas procedentes del nivel de red. Esta función se encarga de construir la trama y enviarla a través del medio físico.

➢ Función LinkReceiveFrame()

Esta función recibe llamadas procedentes del hardware, cuando detecta que se ha recibido una trama, y realiza la llamada a la correspondiente función del nivel de red, `NetworkReceivePacket()`.

4.5 Librería de hardware

Las funciones del hardware hacen posible la interacción con los dispositivos. El hardware se encarga de realizar las funciones de inicialización, de detectar cuándo se ha pulsado un interruptor en un nodo e identificar el interruptor pulsado, de detectar cuándo se ha recibido un mensaje, y de imprimir mensajes y el valor de las variables por pantalla.

A continuación se explican las funciones principales del hardware:

➢ Función InitTimer()

En esta función se inicializa el *Tick Timer*. El *Tick Timer* se utiliza para contar periodos de 10 segundos, necesarios para el envío de “hello messages”.

El *Tick Timer* (31) se puede controlar utilizando las funciones de la API de periféricos internos. Este periférico es capaz de contar los ciclos de un reloj de 16 MHz. Se puede utilizar para programar interrupciones de tiempo, eventos regulares en el tiempo, referencias de tiempo de alta precisión y gestión de tiempos de espera. El proceso que realiza consiste en contar ciclos de reloj hasta que llega a un valor predeterminado desde un valor inicial que también se puede establecer previamente. Una vez que se ha alcanzado el valor deseado, *Tick Timer* puede seguir contando, reiniciar la cuenta desde cero o parar de contar. También se puede generar una interrupción cuando se alcanza el valor de referencia.

El procedimiento que se ha seguido en esta función es el siguiente:

- Se establece el valor inicial de la cuenta a 0.
- Se calcula el número de ciclos que corresponden a un tiempo de 10 segundos mediante una operación sencilla: si un ciclo de reloj se corresponde con 16 MHz,

cada ciclo de reloj son 1/16MHz = 62,5 nanosegundos, por tanto, para contar 10 segundos se necesitará contar 160 millones de ciclos.

- Se habilitan las interrupciones cuando se alcanza el valor deseado y éstas son recogidas en la función `NetworkSendHello()`.
- Se reinicia el temporizador desde 0 cuando ha alcanzado el valor deseado.

A continuación se muestra el código correspondiente a esta función:

```
PUBLIC void InitTimer(void)
{
      vAHI_TickTimerConfigure(E_AHI_TICK_TIMER_DISABLE);

      vAHI_TickTimerWrite(0); //Valor inicial a 0

      vAHI_TickTimerInterval(0x9896800); //Número de ciclos
      de reloj de 16 MHz que corresponden a 10 segundos

      vAHI_TickTimerIntEnable(TRUE); //Habilitar
      interrupciones

      vAHI_TickTimerInit(NetworkSendHello); //Función que
      recoge la interrupción

      vAHI_TickTimerConfigure(E_AHI_TICK_TIMER_RESTART);
      //Reiniciar el temporizador cuando se alcance el
      valor deseado
}
```

➢ Función IncomingHwEvent()

Esta función detecta cuándo se ha pulsado un interruptor en un nodo e identifica cuál es el interruptor que se ha pulsado, para realizar la llamada a la función correspondiente.

Para ello, se utiliza la función `u8ButtonReadFfd() (34)`, definida en la librería de Jennic `Button.h`, y se realiza la llamada a la función que corresponda en cada caso.

➢ Función IncomingData()

Esta función detecta cuándo se ha recibido una trama y llama a la correspondiente función del nivel de enlace/físico, `LinkSendFrame`().

Las demás funciones de la librería de hardware, que se ocupan de la inicialización y la impresión por pantalla, se citan a continuación pero no se explican porque se consideran triviales.

- `PUBLIC void vInitSystem(void)`
- `PUBLIC void vInitEndpoint(void)`
- `PUBLIC void vLightBulbInit(void)`
- `PUBLIC void vInitUart(void)`
- `PUBLIC void vPrintString (char *pcMessage)`

- `PUBLIC void vDisplayHex (uint32 u32Data)`

4.6 Evaluación

La evaluación del funcionamiento en una red de sensores inalámbricos no es intuitiva, ya que no se puede observar el funcionamiento interno de los dispositivos. Para poder comprobar el correcto funcionamiento se deben utilizar recursos visuales como la impresión de mensajes y de los valores de las variables por pantalla, el encendido y apagado de los LEDs, etc.

Se han realizado diferentes pruebas a lo largo de todo el desarrollo del proyecto, hasta llegar a comprobar el funcionamiento global.

Para poder llevar a cabo dichas pruebas, se necesita previamente preparar el escenario. En este caso, se utiliza el escenario mostrado en la Figura 4.1.

A continuación se explica la preparación del escenario de pruebas y las diferentes pruebas que se han realizado de cada parte del proyecto hasta llegar a la demostración final.

➢ Preparación del escenario de pruebas

El primer paso para poder llevar a cabo las pruebas es compilar el código fuente. Hay varias formas distintas de compilar, que se explican con más detalle en el Anexo. En este caso, se ha utilizado un archivo *makefile*. El código se compila a través de la línea de comandos de MS-DOS utilizando el siguiente comando:

```
make –f makefile JENNIC_CHIP=JN5139R1
```

Se debe especificar el nombre del fichero "*makefile*" y también se puede seleccionar el tipo de dispositivo, en este caso, JN5139R1.

De esta forma se crea el archivo binario correspondiente al código. En este caso, el archivo binario contiene la pila de protocolos implementada.

El siguiente paso es descargar el código en los dispositivos. Previamente, se preparan los dispositivos alimentando cada uno con dos pilas tipo AAA y se colocan simulando la topología de red representada en la Figura 4.1, para que sea más fácil identificar a cada nodo de la red. Para descargar el código en los dispositivos se deben conectar uno a uno al ordenador mediante un cable puerto USB a puerto serie, de la manera indicada en la Figura 4.2:

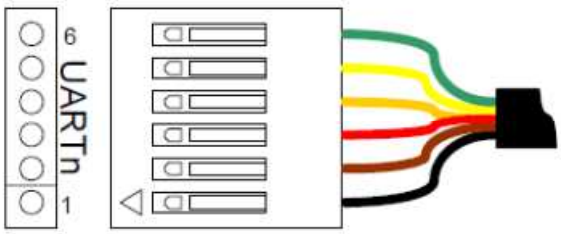

Figura 4.2. Polaridad correcta del cable puerto USB a puerto serie (28).

Cada dispositivo, una vez conectado de esta manera, se debe encender y poner en modo programación. Para ello, hay que seguir los siguientes pasos (28):

- Mantener pulsado el interruptor de programación (SW7).
- Mantener pulsado el interruptor de reset (SW5).
- Soltar el interruptor de reset.
- Soltar el interruptor de programación.

Cuando el dispositivo se encuentra en modo programación se procede a descargar en él el archivo binario utilizando el programa Flash Programmer, proporcionado por Jennic en el kit de desarrollo. Para salir del modo programación se pulsa el interruptor reset o se reinicia el dispositivo. Este proceso se debe hacer para cada uno de los nodos de la red. En el Anexo, se explica en detalle cómo utilizar la aplicación Flash Programmer.

Si se desea hacer cambios en el código fuente, hay que repetir todo este procedimiento para poder hacer pruebas.

Las pruebas que se han realizado durante el desarrollo del proyecto se describen a continuación.

➢ Prueba de la función CalculateDistanceSquared()

Para comprobar que esta función calcula la distancia al cuadrado entre dos nodos correctamente se ha utilizado la impresión de mensajes y variables por pantalla. El terminal en el que se muestra la información se denomina Jen Term, disponible en el software del fabricante Jennic. Para ello, el dispositivo debe estar conectado al ordenador mediante el cable puerto USB a puerto serie, como se vio en la Figura 4.2. Las opciones de configuración de Jen Term se explican en el Anexo.

Esta prueba se puede simplificar realizando un envío estático, por ejemplo, haciendo que el nodo 1 envíe un mensaje al nodo 7. Para comprobar que está funcionando correctamente se calcula a mano, utilizando la fórmula vista anteriormente, la distancia al cuadrado entre el destino y el nodo 1 y entre el destino y sus vecinos, y se imprime por pantalla el valor de las distancias al cuadrado calculadas por la función. Hay que tener en cuenta que la función del hardware imprime por pantalla los valores de las variables en hexadecimal, por tanto, para comparar los dos valores hay que hacer una conversión, ya que ambos deben estar especificados en el mismo sistema (hexadecimal o decimal).

Mediante este procedimiento se ha comprobado que la función calcula la distancia entre dos nodos correctamente.

➢ Prueba de la función NextHop()

Para comprobar que la función NextHop() realmente encuentra el vecino más cercano al destino, también se ha utilizado la impresión de mensajes y variables por pantalla.

En este caso, se imprimen por pantalla las coordenadas destino y la dirección del siguiente salto. Inspeccionando la red de la Figura 4.1 se puede observar si el siguiente salto elegido es realmente el vecino del nodo origen que se encuentra más cerca del destino. También se puede comprobar el funcionamiento calculando a mano la distancia entre el origen y el

destino, y entre sus vecinos y el destino, y compararlas para ver cuál es menor. El nodo que tenga una distancia menor al destino, es el que debe aparecer como siguiente salto.

Mediante este procedimiento se ha comprobado que la función NextHop() funciona correctamente.

➢ Prueba del encaminamiento geográfico

Para facilitar la comprobación del correcto funcionamiento del encaminamiento geográfico, cada vez que un nodo recibe un mensaje se enciende el LED D1 en dicho nodo. En fase de pruebas, es más sencillo utilizar la función `rand()` mientras se realizan las comprobaciones, sin cambiar la semilla. De esta forma, las coordenadas destino generadas aleatoriamente tienen siempre la misma secuencia de valores.

Primero se imprime por pantalla la secuencia de coordenadas destino que se obtiene pulsando el interruptor SW2 varias veces en uno de los nodos, siempre teniendo en cuenta que éstas se muestran en hexadecimal. Una vez conocida la secuencia, se reinician los dispositivos para volver a obtener la misma secuencia de coordenadas. De esta manera es fácil determinar la trayectoria que debe seguir el mensaje para cada destino, observando la red de la Figura 4.1. Solamente hay que comprobar si el camino seguido por el mensaje se corresponde con el esperado, observando cuáles son los dispositivos que tienen el LED D1 encendido.

De esta manera se puede probar el encaminamiento de una forma sencilla y visual probando con todos los nodos como origen.

Mediante esta prueba se demuestra la robustez del encaminamiento geográfico en redes de sensores inalámbricos.

Si se desea, también se puede comprobar el funcionamiento sin utilizar la función `rand()`. Al pulsar un interruptor en un nodo para enviar un mensaje se imprimen las coordenadas destino aleatorias por pantalla y, *a posteriori*, se comprueba que el camino seguido por el mensaje es el correcto, mirando si los nodos que tienen el LED D1 encendido son los que deberían haber encaminado el mensaje hasta el destino o hasta el nodo más cercano a éste.

En determinados casos, el encaminamiento puede no encontrar el destino final. Si se encuentra un hueco en la red el algoritmo falla, como se explicó en el Capítulo 2. Es una de las limitaciones de *Greedy Forwarding.* Mediante el procedimiento que se acaba de exponer, es fácil identificar cuándo nos encontramos en uno de estos casos, por simple inspección de la representación de la red.

Utilizando el procedimiento expuesto se ha comprobado que el encaminamiento geográfico se ejecuta correctamente.

➢ Prueba de la recepción de “hello messages”

Como se ha explicado anteriormente, se utiliza el LED D2 para comprobar que los nodos reciben “hello messages”. Los nodos inicialmente tienen este LED encendido y cuando se recibe el primer “hello message” procedente de un vecino se apaga. Para realizar la

comprobación, se encienden dos nodos vecinos y se observa cómo transcurridos 10 segundos se apagan los LEDs en los nodos.

➢ Prueba de la detección de cambios en la red

La función `NetworkButton3()` muestra los vecinos de un nodo en un momento determinado. Utilizando esta función se puede comprobar que la red se ha construido correctamente. Para ello, se encienden todos los nodos de la red y se espera un tiempo adecuado para que se envíen y reciban los "hello messages". Observando la red de la Figura 4.1, se puede saber quiénes son los vecinos de cada nodo, que deben coincidir con los vecinos mostrados por la función.

Si se apaga uno de los nodos vecinos, se puede comprobar que transcurrido el período umbral de 30 segundos, éste no se muestra por pantalla porque se ha eliminado del array de vecinos.

De esta manera se comprueba fácilmente que los nodos detectan cambios producidos en la red.

➢ Demostración del funcionamiento global

El procedimiento que se lleva a cabo para demostrar el funcionamiento global del proyecto es el siguiente:

- Se pulsa el interruptor SW2 en uno de los nodos, que previamente se conecta mediante el cable puerto USB a puerto serie para poder ver la información mostrada por pantalla. Éste es el nodo origen.
- Se imprime por pantalla un mensaje que indique que los valores mostrados a continuación son los de las coordenadas destino generadas aleatoriamente y se muestran estos valores. Siempre que se muestren por pantalla los valores de las variables hay que tener en cuenta que son valores hexadecimales.
- Se imprime por pantalla un mensaje indicando la dirección y las coordenadas del siguiente salto y se muestran estos valores.
- Para comprobar que el mensaje se está enviando a través de las capas de la pila de protocolos, se muestra por pantalla un mensaje que indique cuándo el mensaje se está enviando o recibiendo en cada capa.
- Una vez que el mensaje ha llegado a su destino o al nodo más cercano a él, se envía un mensaje de confirmación al nodo origen. Cuando el nodo origen recibe este mensaje se muestra por pantalla, por tanto, se imprime "Hola mundo".

Para terminar de comprobar que el encaminamiento se ha ejecutado correctamente, se observa el esquema de la red para determinar por inspección visual cuál es la trayectoria que debe haber seguido el mensaje y se comprueba si los nodos correspondientes a esta secuencia tienen el LED D1 encendido.

Con el procedimiento que se acaba de exponer se ha buscado comprobar el funcionamiento de una manera sencilla y visual.

Este proceso se puede repetir tantas veces como se desee, sin necesidad de reiniciar los dispositivos. El único requisito es apagar los LEDs de los nodos que han participado en el

encaminamiento después de cada envío, utilizando el interruptor SW1, para poder comprobar la trayectoria seguida por el siguiente mensaje.

Durante la fase de desarrollo del proyecto se han realizado otras comprobaciones, tales como determinar si la secuencia de coordenadas generada es realmente aleatoria, mediante la impresión de mensajes y variables por pantalla que se consideran triviales, por lo que no se explican aquí.

Capítulo 5

Planificación de proyecto y presupuesto

5.1 Introducción

A continuación se realiza una descripción de las fases de desarrollo del proyecto y se muestra el presupuesto, calculado a partir de los costes de los recursos materiales y humanos que han sido necesarios.

5.2 Planificación de proyecto

La realización de este proyecto ha conllevado un largo proceso de aprendizaje. Este ha consistido en la familiarización con los dispositivos utilizados y con el lenguaje de programación C utilizado para realizar el desarrollo sobre la plataforma. Durante este proceso se ha pasado por diferentes etapas, hasta llegar al proyecto final. Además, la realización del proyecto se ha compaginado con otras actividades como la asistencia a clases, la vida laboral como becaria a media jornada y ha sido totalmente interrumpida durante los períodos de prácticas y exámenes. También ha tenido en cuenta la disponibilidad de los tutores.

A continuación se explican las diferentes fases de desarrollo del proyecto.

Como primera toma de contacto, se utilizó una aplicación de ejemplo proporcionada por el fabricante Jennic. Ésta consistía en la utilización de dos dispositivos para realizar una aplicación muy sencilla, en la que uno de los nodos actuaba como coordinador y permitía que el otro nodo se asociase a su red, encendiendo un LED en el coordinador para indicar que la asociación se había realizado satisfactoriamente. El objetivo de esta fase fue familiarizarse con la utilización de los dispositivos.

Posteriormente se siguió trabajando con dos dispositivos para continuar con la fase de aprendizaje. En esta fase se introdujo la utilización de los interruptores, el envío de mensajes entre nodos, la impresión por pantalla de mensajes y variables y otras funciones necesarias para el posterior desarrollo del proyecto. Esta fase de aprendizaje comprendió desde septiembre de 2009 hasta febrero de 2010, interrumpiéndose durante los meses de diciembre y enero para la preparación y realización de los exámenes.

La siguiente etapa, que comenzó en marzo de 2010, consistió en la utilización de cuatro nodos sensores para implementar y evaluar el protocolo de encaminamiento geográfico. Esta etapa comprendió hasta octubre de 2010, interrumpiéndose durante los períodos de prácticas y exámenes del segundo cuatrimestre y la convocatoria de exámenes de septiembre.

En la siguiente fase, se construyó el escenario utilizado con diez dispositivos y se evaluó el funcionamiento del encaminamiento geográfico adaptando la implementación, entre los meses de noviembre de 2010 y febrero de 2011. Esta etapa también fue interrumpida para la preparación y realización de los exámenes de la convocatoria de enero.

La última fase de la implementación comprendió dese marzo hasta noviembre de 2011. En ella se añadió la aplicación “ping-pong”, la generación de coordenadas de manera aleatoria, se hizo el diseño y la implementación por capas y se incluyó el envío y recepción de “hello messages” para construir la red y detectar los cambios producidos en ésta de manera dinámica.

Para concluir, la fase de documentación se ha desarrollado entre diciembre de 2011 y febrero de 2012. En ella se ha realizado esta memoria y se ha preparado la presentación.

5.3 Presupuesto

Para calcular el presupuesto se ha utilizado la plantilla para el presupuesto del proyecto fin de carrera (35) disponible en la página web de la Universidad Carlos III. Ésta se divide en costes directos, que incluyen los costes de personal y los del material utilizado, y costes indirectos.

A continuación se realiza un desglose detallado de los costes.

5.3.1 Recursos materiales

Los costes de los recursos materiales utilizados se dividen en costes de hardware y costes de software.

Para calcular los costes de los equipos de hardware se ha utilizado el cálculo de la amortización que viene especificado en la plantilla de presupuesto y se han considerado los costes de los equipos sin IVA. Éstos se detallan en la Tabla 5.1:

Descripción	Cantidad	Coste unitario (euros)	Coste total (euros)	% Uso dedicado al proyecto	Dedicación (meses)	Período de depreciación (meses)	Coste imputable (euros)
Ordenador portátil	1	862	862	70	21	60	211,19
Kit de desarrollo Jennic JN5139	2	250	500	100	18	60	150
						Total	361,19

Tabla 5.1. Costes de hardware.

Los costes de software incluyen los costes de licencia de los programas utilizados. El software proporcionado por Jennic viene incluido en el kit de desarrollo, por tanto, no se contabiliza de nuevo.

Los costes relacionados con el software se muestran en la Tabla 5.2:

Descripción	Coste (euros)
Microsoft Office Word 2007	13,73
Microsoft Office Excel 2007	28,78
Microsoft Office Visio 2007	33,22
Total	75,73

Tabla 5.2. Costes de software.

5.3.2 Recursos humanos

Para calcular los costes de personal se han utilizado las especificaciones que se detallan en la plantilla de presupuesto. La dedicación de cada persona se mide en hombres-mes. Cada hombre-mes equivale a 131,25 horas de trabajo. El coste de cada hombre-mes también se especifica en la plantilla.

En la Tabla 5.3 se muestran los costes de personal:

Apellidos y nombre	Categoría	Dedicación (hombres mes)	Coste / hombre mes	Coste (euros)
López Alvarez, Miriam	Ingeniero	12,8	2.694,39	34.488,19
Cuevas Rumín, Ángel	Ingeniero Senior	0,23	4.289,54	986,59
Urueña Pascual, Manuel	Ingeniero Senior	0,17	4.289,54	729,22
			Total	36.204,01

Tabla 5.3. Costes de personal.

5.3.3 Costes totales

El presupuesto total incluye los costes directos, calculados en las secciones 5.3.1 y 5.3.2, y los costes indirectos tales como el gasto de luz y calefacción en el laboratorio en el que se ha realizado el proyecto, la conexión a Internet, etc. La tasa de costes indirectos especificada en la plantilla de presupuesto es del 20 %.

En la Tabla 5.4 se detalla el presupuesto total:

Costes	Presupuesto total (euros)
Costes de hardware	361,19
Costes de software	75,73
Costes de personal	36.204,01
Costes indirectos	7.328,19
Total	43.969,12

Tabla 5.4. Presupuesto total.

Por tanto, el presupuesto total de este proyecto asciende a la cantidad de 43.969,12 euros.

Capítulo 6

Conclusiones y líneas de trabajo futuras

6.1 Introducción

En este capítulo se exponen las conclusiones obtenidas a partir de la realización de este proyecto y las líneas de trabajo futuras que se pueden llevar a cabo para ampliar el trabajo desarrollado.

6.2 Conclusiones

El principal objetivo de este proyecto, como se vio en el Capítulo 1, era el de implementar un protocolo de encaminamiento geográfico para optimizar el envío de mensajes en una red de sensores inalámbricos. Estas redes se utilizan en multitud de aplicaciones de numerosos ámbitos diferentes, pero tienen el inconveniente de que disponen de una cantidad de recursos de memoria, ancho de banda y cobertura bastante limitados. Además, es imprescindible optimizar su consumo de energía para que puedan operar con baterías durante largos intervalos de tiempo.

Para solucionar estos problemas se pueden utilizar protocolos de encaminamiento geográfico, ya que mediante la optimización del reenvío de mensajes entre los nodos de la red, se optimiza también el consumo de recursos de memoria y energía y la utilización del ancho de banda disponible.

El protocolo de encaminamiento geográfico que se ha implementado es *Greedy Forwarding*. Este protocolo es muy adecuado para redes de sensores inalámbricos porque sólo requiere que los nodos almacenen la información relativa a sus vecinos locales para encaminar los paquetes. Además, sólo se necesitan las coordenadas geográficas del destino del mensaje y las posiciones de los nodos vecinos, que son los candidatos a siguiente salto, para tomar las decisiones de reenvío. Por tanto, el único dato que se precisa añadir al paquete enviado es la posición del destino, que no conlleva un aumento excesivo de la longitud del paquete. Estas características del protocolo suponen un ahorro de memoria y evitan la saturación de la red, que se produce al enviar una gran cantidad de datos de encaminamiento. La construcción y detección de cambios en la red también se realiza de una manera sencilla y eficiente mediante el envío de "hello messages". Como cada nodo sólo necesita transmitir su posición y su dirección, el envío de estos mensajes de señalización tampoco supone una sobrecarga en la red.

En este proyecto se ha implementado una pila de protocolos basada en este protocolo de encaminamiento y se ha evaluado su correcto funcionamiento mediante la ejecución de una aplicación en una red de sensores inalámbricos real. Como primera conclusión, se ha podido observar la robustez del encaminamiento geográfico, ya que nunca comete fallos, dentro de sus propias limitaciones. Dichas limitaciones, que se explicaron en el Capítulo 2, tienen como consecuencia que no siempre sea posible alcanzar el nodo más cercano al destino final, si se tiene una topología de red con huecos. Este comportamiento también se ha observado, pero no se debe considerar como un fallo porque este caso se encuentra contemplado dentro de la definición del protocolo. Si se duplica o se pierde alguna trama esporádicamente se debe a la transmisión por radio, ya que el medio inalámbrico es en sí mismo propenso a fallos.

Otra de las conclusiones que se han extraído de la realización de este proyecto es que, aunque la implementación del protocolo en sí mismo es relativamente sencilla, trabajar con redes de sensores inalámbricos es complejo. Como se explicó en el Capítulo 4, la evaluación del funcionamiento en este tipo de redes no es inmediata y hay que buscar otros recursos para comprobar que la red está realizando el procedimiento deseado. Además, hay que tener en cuenta las limitaciones de hardware de los dispositivos y en consecuencia, buscar métodos alternativos para conseguir el comportamiento requerido. Sin embargo, una vez que se ha comprobado que los dispositivos de la red realizan el procedimiento correctamente, se pueden desplegar garantizando un funcionamiento robusto y sin necesidad de atención o mantenimiento, para ser utilizados por multitud de aplicaciones. Esta es una de las principales ventajas de las redes de sensores inalámbricos.

Por tanto, se ha demostrado que el objetivo principal de este proyecto se ha conseguido con éxito. La implementación de este protocolo de encaminamiento geográfico implica una mejora en el ahorro de recursos y permitirá que estas redes puedan ser utilizadas por un mayor número de aplicaciones de forma más eficiente.

6.3 Líneas de trabajo futuras

Como ya se ha comentado, una de las limitaciones del protocolo de encaminamiento geográfico *Greedy Forwarding* es que no siempre se consigue alcanzar el nodo más cercano al destino en presencia de huecos en la red. Por tanto, una de las líneas de trabajo futuras sería implementar un algoritmo que solucione este inconveniente.

Para solucionar este problema se podría implementar un algoritmo tipo *Face Routing* en cualquiera de sus variantes. En GPSR (*Greedy Perimeter Stateless Routing*) (15) se propone combinar este mecanismo con *Greedy Forwarding* (ver Capítulo 2). Básicamente, el algoritmo de GPSR consiste en ejecutar *Greedy Forwarding* siempre que sea posible y un algoritmo de *Face Routing* en las regiones en las que no es posible el envío con *Greedy*. *Face Routing* se basa en la regla de la mano derecha. Cuando se encuentra una región en la que el reenvío con *Greedy* no es posible, se recorre el perímetro de la región en sentido contrario a las agujas del reloj hasta encontrar un nodo más cercano al destino que aquel donde *Greedy* no pudo continuar. De esta forma, se puede volver a utilizar *Greedy* a partir del nodo encontrado por *Face Routing*. Una línea de trabajo futura sería implementar el algoritmo GPSR completo. De esta forma, se solucionarían las limitaciones de *Greedy Forwarding*.

Para evitar pérdidas o duplicados en las tramas enviadas, una línea de trabajo futura sería implementar un protocolo de nivel de transporte que realice las gestiones de control necesarias.

Uno de los problemas que se han tenido que resolver en este proyecto es la localización de los dispositivos. Mediante la utilización de dispositivos con GPS integrado, se solucionaría el problema de que los nodos fuesen capaces de conocer su posición. El trabajo pendiente consistiría en buscar en el mercado dispositivos de este tipo y decidir si compensa asumir el coste, dependiendo de la aplicación para la que se vayan a utilizar.

En este proyecto se ha implementado una aplicación muy sencilla cuyo principal propósito era evaluar fácilmente el funcionamiento del encaminamiento geográfico. Por tanto, en el futuro se puede desarrollar una amplia gama de aplicaciones más complejas, que se beneficien de la utilización del protocolo de encaminamiento desarrollado. Esta línea de trabajo queda totalmente abierta porque las aplicaciones que se desarrollen en el futuro pueden tener cabida en multitud de ámbitos distintos.

Glosario

- ACL, *Access Control List*: lista de permisos asociados a un objeto. Especifica qué usuarios o qué procesos tienen el acceso permitido a un objeto y las acciones permitidas en dicho objeto (36).

- AES, *Advanced Encryption Standard*: esquema de cifrado por bloques adoptado como un estándar de cifrado por el gobierno de los Estados Unidos (36).

- API, *Application Programming Interface*: conjunto de funciones que ofrece una librería para ser utilizadas por otro software (36).

- Banda ISM, *Industrial, Scientific and Medical*: banda de radiofrecuencia reservada internacionalmente para uso industrial, científico y médico. Se han popularizado por su uso en comunicaciones inalámbricas (36).

- CD-ROM, *Compact Disc – Read Only Memory*: disco compacto con permisos de sólo lectura (36).

- CLI, *Command Line Interface*: método que permite a las personas dar instrucciones a un programa informático por medio de líneas de texto simples (36).

- Conector SMA, *SubMiniature version A*: conectores coaxiales de radiofrecuencia desarrollados en los años 60 como interfaces de conexión mínimas para cable coaxial con mecanismo de acoplamiento tipo tornillo. Estos conectores tienen una impedancia de 50 Ω y ofrecen excelentes prestaciones eléctricas desde corriente continua hasta 18 GHz (36).

- CPU, *Central Processing Unit*: se traduce como unidad central de procesamiento y es el componente de los ordenadores y otros dispositivos programables encargado de interpretar las instrucciones y procesar los datos (36).

- Domótica: este término se refiere al conjunto de sistemas capaces de automatizar una vivienda para ofrecer servicios de seguridad, bienestar, comunicación y gestión

energética, que pueden estar integrados mediante redes de comunicación cableadas o inalámbricas interiores y exteriores y que disponen de un control desde dentro y fuera del hogar (36).

- EEPROM, *Electrically Erasable Programmable Read-Only Memory*: memoria que puede ser borrada y modificada eléctricamente (36).

- FTDI, *Future Technology Devices International*: compañía privada escocesa del negocio de los dispositivos semiconductores especializada en la tecnología USB. Desarrolla, fabrica y da soporte a dispositivos y sus correspondientes drivers para convertir transmisiones serie RS-232 o TTL en señales USB, para permitir la compatibilidad de los dispositivos con los ordenadores modernos (36).

- GPIO, *General Purpose Input/Output*: pin genérico de entrada o de salida en un chip, cuyo comportamiento puede ser programado (36).

- GPS, *Global Positioning System*: sistema global de navegación por satélite (GNSS) que permite conocer la posición de una persona o un objeto en cualquier parte del mundo, con una precisión de algunos metros. Fue desarrollado por el Departamento de Defensa de los Estados Unidos (36).

- GTS, *Guaranteed Timeslot*: método de calidad de servicio que asigna a cada nodo específico un período de tiempo en el que puede actuar sin sufrir latencia (37).

- IDE, *Integrated Development Environment*: entorno de desarrollo integrado.

- IEEE, *Institute of Electrical and Electronic Engineers*: asociación técnico-profesional mundial, sin ánimo de lucro, dedicada a la estandarización, entre otras cosas (36).

- Inmótica: incorporación de sistemas de gestión técnica automatizada para mejorar el confort y la seguridad y reducir el consumo de energía en edificios de uso terciario o industrial como oficinas, edificios corporativos, hoteleros o empresariales, etc. (36).

- IVA, Impuesto sobre el Valor Añadido: impuesto indirecto financiado por el consumidor final (36).

- *Jenie*: es una capa que permite fácil interacción con JenNet, un protocolo propiedad de Jennic que se ofrece como una alternativa más simple que ZigBee (38).

- LCD, *Liquid Crystal Display*: pantalla de cristal líquido.

- LED, *Light Emitting Diode*: diodo emisor de luz.

- LLC, *Logical Link Control*: subcapa del nivel de enlace responsable del control de errores, control de flujo, entramado, control de diálogo y direccionamiento de la subcapa MAC (36).

- MAC, *Media Access Control*: subcapa de control de acceso al medio.

- MAC PIB, *PAN Information Base* de la subcapa MAC: conjunto de parámetros utilizados por la subcapa MAC, que describen la PAN en la que se encuentra el nodo (39).

- MIT, *Massachussetts Institute of Technology*: el Instituto Tecnológico de Massachussetts es una institución privada americana dedicada a la investigación y a la educación científica y tecnológica (36).

- MS-DOS, *MicroSoft Disk Operating System*: sistema operativo de disco de Microsoft para ordenadores basados en x86 (36).

- MTU, *Maximum Transmission Unit:* tamaño en bytes de la unidad de datos mayor que una capa de un protocolo de comunicaciones puede transmitir (36).

- OSI, *Open System Interconnection*: el modelo de interconexión de sistemas abiertos es un marco de referencia para la definición de arquitecturas de interconexión de sistemas de comunicaciones, creado por la Organización Internacional para la Estandarización (36).

- PAN, *Personal Area Network*: red de área personal.

- PHY PIB, *PAN Information Base* de la capa física: conjunto de parámetros utilizados por la capa física, que describen la PAN en la que se encuentra el nodo (39).

- PSU, *Power Supply Unit*: elemento que convierte corriente alterna en corriente continua de baja tensión adecuada para los componentes internos de un ordenador (36).

- RAM, *Random Access Memory*: memoria de acceso aleatorio.

- RF: radiofrecuencia.

- RISC, *Reduced Instruction Set Computer*: tipo de procesador cuyas características fundamentales son que las instrucciones son de tamaño fijo y con un número reducido de formatos, y sólo acceden a memoria las instrucciones de carga y almacenamiento (36).

- ROM, *Read- Only Memory*: memoria de sólo lectura, es decir, no se pueden modificar los datos (36).

- SDK, *Software Developers's Kit*: kit de desarrollo de software.

- SPI, *Serial Peripheral Interface*: estándar de comunicaciones usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos (36).

- UART, *Universal Asynchronous Receiver-Transmitter*: circuito integrado para comunicaciones serie sobre el puerto serie de un ordenador o un periférico. Son muy comunes en microcontroladores.

- USB, *Universal Serial Bus*: puerto que sirve para conectar periféricos a un ordenador.

- WPAN, *Wireless Personal Area Network*: red inalámbrica de área personal. Es una red de dispositivos sin conexión por cables con un alcance de unas decenas de metros (40).

- *ZigBee*: especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radiodifusión digital de bajo consumo, basada en el estándar IEEE 802.15.4. Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío y maximización de la vida útil de sus baterías (36).

Referencias

(1). **Aakvaag, Niels y Frey, Jan-Erik.** *Redes de sensores inalámbricos: Nuevas soluciones de interconexión para la automatización industrial.* s.l. : Revista ABB, 2006.

(2). RIMSI Research Project. [En línea] 10 de abril de 2007. [Citado el: 21 de diciembre de 2011.] http://www.motas.es/rimsi/index.php?option=com_content&task=view&id=2&Itemid=3.

(3). **Fernández Barcell, Manuel.** Wireless Sensor Network. [En línea] [Citado el: 21 de diciembre de 2011.] http://www.mfbarcell.es/conferencias/wsn.pdf.

(4). **Escolar Díaz, M. Soledad.** Wireless Sensor Networks: Estado del arte e investigación. [En línea] [Citado el: 21 de diciembre de 2011.] http://www.arcos.inf.uc3m.es/~sescolar/index_files/presentacion/wsn.pdf.

(5). **Gómez Mula, Francisco.** Redes de sensores inalámbricos: Las tecnologías de la información y de las comunicaciones en el ámbito de la atención socio-sanitaria. [En línea] 13 de julio de 2007. [Citado el: 20 de diciembre de 2011.] http://atc.ugr.es/~aprieto/TIC_socio_sanitario/A11_4_05_Redes_sensores.pdf.

(6). **Ortiz Tapia, Francisco.** Redes de sensores inalámbricos. [En línea] [Citado el: 20 de diciembre de 2011.] http://profesores.elo.utfsm.cl/~tarredondo/info/networks/Presentacion_sensores.pdf.

(7). **National Instruments.** ¿Qué es una Red de Sensores Inalámbrica (WSN)? [En línea] [Citado el: 27 de diciembre de 2011.] http://zone.ni.com/devzone/cda/tut/p/id/9507.

(8). Cebras con GPS. *El País.* 2008.

(9). **MIT.** The Cricket Indoor Location System. [En línea] [Citado el: 27 de diciembre de 2011.] http://nms.lcs.mit.edu/projects/cricket/.

(10). **Universidad de Berkeley.** Fire Information and Rescue Equipment (FIRE): Enhanced Decision-Making and Situational Awareness for Urban/Industrial Firefighting. [En línea] [Citado el: 27 de diciembre de 2011.] http://fire.me.berkeley.edu/about_fire.htm.

(11). **PID Systems.** Portable Intruder Detection. [En línea] [Citado el: 26 de diciembre de 2011.] http://www.pid-systems.co.uk/.

(12). **Harvard Sensor Networks Lab.** CodeBlue: Wireless Sensors for Medical Care. [En línea] [Citado el: 26 de diciembre de 2011.] http://fiji.eecs.harvard.edu/CodeBlue.

(13). **Martos García, Gema.** Diseño, desarrollo y validación de un protocolo de encaminamiento bidireccional sobre 802.15.4 para redes de sensores inalámbricos. [En línea] 2011. [Citado el: 9 de enero de 2012.] http://e-archivo.uc3m.es/bitstream/10016/12040/1/PFC_Gema_Martos_Garcia.pdf.

(14). **Palma Gómez, Antonio Manuel.** Análisis de protocolos de enrutamiento para redes de sensores inalámbricas. [En línea] 18 de diciembre de 2009. [Citado el: 9 de enero de 2012.] http://e-archivo.uc3m.es/bitstream/10016/8493/1/PFC_AntonioManuel_Palma_Gomez.pdf.

(15). **Karp, Brad y Kung, H.T.** *GPSR: Greedy Perimeter Stateless Routing for Wireless Networks.* En Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom '00), agosto 2000.

(16). **Xing, Guoliang, y otros.** *Impact of sensing coverage on greedy geographic routing algorithms.* En Parallel and Distributed Systems, IEEE Transactions, abril 2006.

(17). **Kuhn, Fabian, Wattenhofer, Roger y Zollinger, Aaron.** Worst-Case Optimal and Average-Case Efficient Geometric Ad-Hoc Routing. En Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (MobiHoc '03), 2003.

(18). **Xu, Ya, Heidemann, John y Estrin, Deborah.** *Geography-Informed Energy Conservation for Ad Hoc Routing.* En Proceedings of the 7th annual international conference on Mobile computing and networking (MobiCom '01), 2001.

(19). **Yu, Yan, Govindan, Ramesh y Estrin, Deborah.** Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks. [En línea] 14 de agosto de 2001. En UCLA Computer Science Department Technical Report, UCLA-CSD TR-01-0023, 2001.

(20). **Basagni, Stefano, y otros.** A Distance Routing Effect Algorithm for Mobility (DREAM). En Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom '98), 1998.

(21). **Niculescu, Dragos y Nath, Badri.** Trajectroy Based Forwarding and its applications. En Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03), 2003.

(22). **Gascón, David.** 802.15.4 vs ZigBee. [En línea] Wireless Sensor Networks Research Group, 17 de noviembre de 2008. [Citado el: 19 de diciembre de 2011.] http://www.sensor-networks.org/index.php?page=0823123150.

(23). **Grupo de Redes y Arquitecturas de Altas Prestaciones UCLM.** Redes de sensores inalámbricas. [En línea] [Citado el: 19 de diciembre de 2011.] http://www.dsi.uclm.es/asignaturas/42650/PDFs/Tema2.pdf.

(24). **Jennic.** IEEE 802.15.4 Wireless Networks User Guide. [En línea] 6 de octubre de 2006. [Citado el: 8 de diciembre de 2011.] http://www.jennic.com/files/support_files/JN-UG-3024-IEEE802.15.4-1v1.pdf.

(25). **Jennic**. Data Sheet-JN513x. [En línea] 26 de octubre de 2007. [Citado el: 1 de diciembre de 2011.] http://www.jennic.com/files/support_files/JN-DS-JN5139-001-1v10.pdf.

(26). **Jennic**. JN5139-EK000 IEEE 802.15.4/JenNet Evaluation Kit Getting Started. [En línea] 2009. [Citado el: 5 de diciembre de 2011.] http://www.jennic.com/files/support_files/JN-UG-3029-JN5139-EK000-Getting-Started-1v3.pdf.

(27). **Jennic**. DR1047 Controller Board Reference Manual. [En línea] 13 de febrero de 2009. [Citado el: 1 de diciembre de 2011.] http://www.jennic.com/files/support_files/JN-RM-2029-DR1047-Controller-Board-1v2.pdf.

(28). **Jennic**. DR1048 Sensor Board Reference Manual. [En línea] 16 de junio de 2010. [Citado el: 1 de diciembre de 2011.] http://www.jennic.com/files/support_files/JN-RM-2030-DR1048-Sensor-Board-1v4.pdf.

(29). **Jennic**. What distance can the radio cover? [En línea] [Citado el: 25 de enero de 2012.] http://www.jennic.com/support/solutions/00011.

(30). **Linux man page.** rand(). [En línea] [Citado el: 24 de enero de 2012.] http://linux.die.net/man/3/rand.

(31). **Jennic.** JN51XX Integrated Peripherals API User Guide. [En línea] 30 de junio de 2011. [Citado el: 24 de enero de 2012.] http://www.jennic.com/files/support_files/JN-UG-3066-Integrated-Peripherals-API.pdf.

(32). **Jennic**. ZigBee Stack Advanced User Guide. [En línea] 6 de marzo de 2008. [Citado el: 26 de enero de 2012.] http://www.jennic.com/files/support_files/JN-UG-3045-ZigBee-Advanced-User-Guide-1v2.pdf.

(33). **Montenegro, G., y otros.** Transmission of IPv6 Packets over IEEE 802.15.4 Networks. [En línea] septiembre de 2007. [Citado el: 4 de febrero de 2012.] http://www.hjp.at/doc/rfc/rfc4944.html.

(34). **Jennic.** *Board API Reference Manual.* 2007.

(35). **Universidad Carlos III de Madrid.** Plantilla presupuesto proyecto fin de carrera. [En línea] [Citado el: 28 de enero de 2012.] http://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera.

(36). Wikipedia. [En línea] http://www.wikipedia.org/.

(37). **Kinney, Patrick.** *ZigBee Technology: Wireless Control that Simply Works.* 2003.

(38). **Jennic.** Software Dveloper's Kit Installation Guide. [En línea] 21 de mayo de 2008. [Citado el: 29 de enero de 2012.] http://www.jennic.com/files/support_files/JN-UG-3035-SDK-Installation-1v3.pdf.

(39). **Jennic**. 802.15.4 Stack API Reference Manual. [En línea] 10 de enero de 2007. [Citado el: 4 de febrero de 2012.] http://www.jennic.com/files/support_files/JN-RM-2002-802.15.4-Stack-API-1v8.pdf.

(40). **Kioskea.net.** WPAN (Wireless Personal Area Network). [En línea] [Citado el: 8 de diciembre de 2011.] http://es.kioskea.net/contents/wireless/wpan.php3.

(41). **Jennic.** CodeBlocks IDE User Guide. [En línea] 6 de marzo de 2008. [Citado el: 30 de enero de 2012.] http://www.jennic.com/files/support_files/JN-UG-3028-CodeBlocks-1v8.pdf.

(42). **Jennic**. JN51xx Flash Programmer Application User Guide. [En línea] 29 de febrero de 2008. [Citado el: 30 de enero de 2012.] http://www.jennic.com/files/support_files/JN-UG-3007-Flash-Programmer.pdf.

Anexo

Instalación y utilización del entorno de desarrollo de Jennic

En este anexo se explican los pasos necesarios para instalar el software proporcionado por Jennic y cómo utilizar los componentes necesarios para crear, compilar y ejecutar aplicaciones. También se explican las opciones disponibles para alimentar las placas.

A.1 Instalación de Jennic SDK

La instalación del kit de desarrollo de software (*Software Developer's Kit*, SDK) de Jennic (38) se divide en dos partes, ambas disponibles en el CD contenido en el kit de desarrollo y en el área de soporte de la página web de Jennic: www.jennic.com/support.

Por una parte, se instalan las herramientas de software de Jennic necesarias para desarrollar aplicaciones para redes inalámbricas, que incluyen herramientas de desarrollo, compilador y herramientas de programación Flash.

Por otra parte, se instalan las librerías de software de Jennic, que incluyen las APIs de los protocolos IEEE 802.15.4, ZigBee y Jenie. Estas pilas de protocolos se muestran en la Figura A.1:

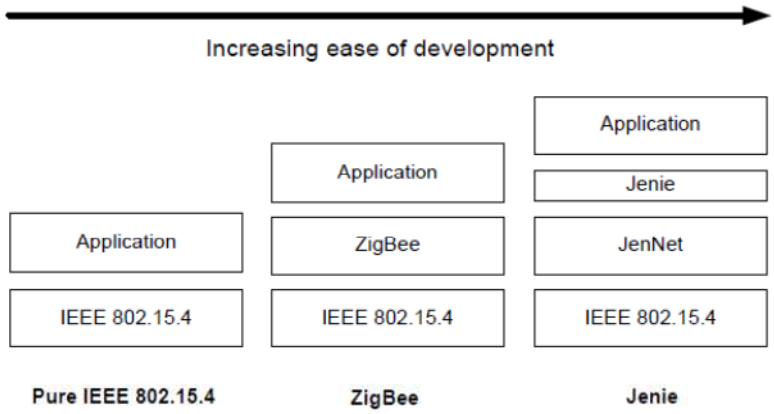

Figura A.1. Pilas de protocolos disponibles en Jennic SDK (38).

A.1.1 Requisitos previos

Antes de instalar el SDK, se debe comprobar que se dispone de los siguientes elementos:

- Una máquina con las siguienes especificaciones:
  - Sistema operativo Windows Vista, XP o 2000.
  - Al menos 240 MB de espacio libre en el disco duro.
- Derechos de administrador en dicha máquina.
- Los siguientes instaladores del SDK disponibles en el CD del kit de desarrollo o en la página web de Jennic:
  - JN-SW-4031-SDK-Toolchain-vX.Y.exe.
  - JN-SW-4030-SDK-Libraries-vX.Y.exe.

A.1.2 Instalación de las herramientas del SDK

Los componentes que se instalan son los siguientes:

- Cygwin: entorno de desarrollo Unix para Windows.
- Code::Blocks: entorno de desarrollo.
- Flash Programmer: programa necesario para descargar el código compilado en la memoria Flash de los dispositivos.
- Herramientas del compilador de Jennic: incluyen el compilador y el enlazador, que también son imprescindibles para el desarrollo.

Para instalar las herramientas del SDK, se deben seguir los siguientes pasos:

- Paso 1: Eliminar de la máquina cualquier versión anterior de Jennic SDK, mediante la opción “Agregar o quitar programas” del Panel de control.
- Paso 2: Comenzar la instalación de las herramientas del SDK.
  - Si se están instalando desde el CD, insertar el CD en la máquina y seguir las instrucciones mostradas en la pantalla. Si se está conectado a Internet, la página web de Jennic comprobará si existe una versión más reciente del instalador. Seleccionar el instalador “*SDK Toolchain*”.

- Si se va a descargar desde la página web de Jennic, descargar y guardar el archivo "JN-SW-4031-SDK-Toolchain-vX.Y.exe" en la máquina y ejecutar el instalador.

- Paso 3: Seguir las instrucciones del *wizard* mostrado en pantalla, hasta llegar a la pantalla de selección de componentes, que se muestra en la Figura A.2:

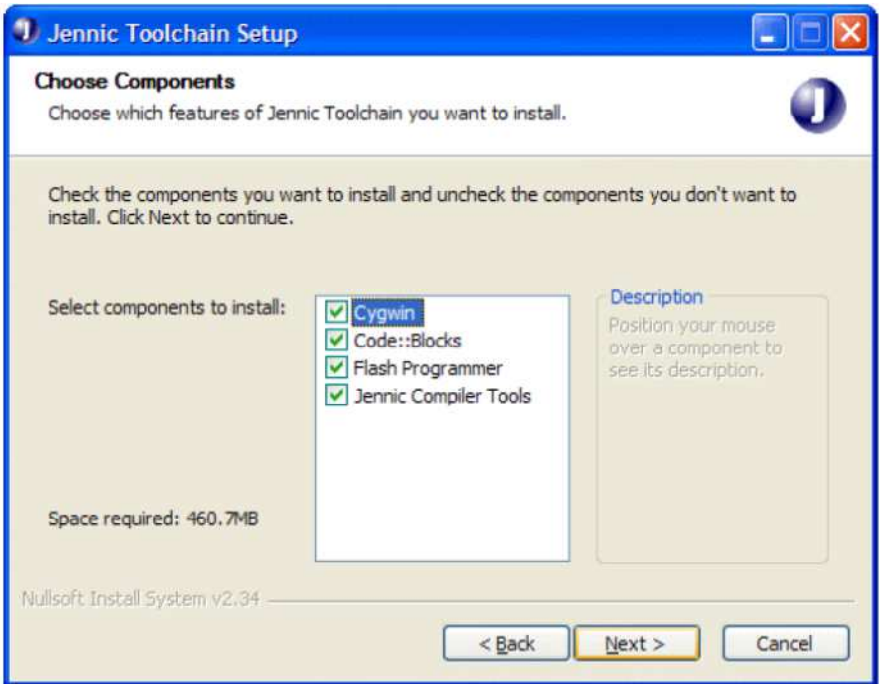

Figura A.2. Pantalla de selección de componentes, de "Jennic Toolchain Setup" (38).

Todos los componentes están seleccionados por defecto. Los componentes que no se deseen instalar se deben desmarcar. En particular, se debe desmarcar:

- Cygwin, si ya se dispone de una instalación de Cygwin que se desee conservar.
- Code::Blocks, si no se desea desarrollar las aplicaciones utilizando este entorno gráfico.

Hacer click en "Next" para continuar.

- Paso 4: En la siguiente pantalla (que se muestra en la Figura A.3) seleccionar la localización en la que se desea guardar las herramientas:

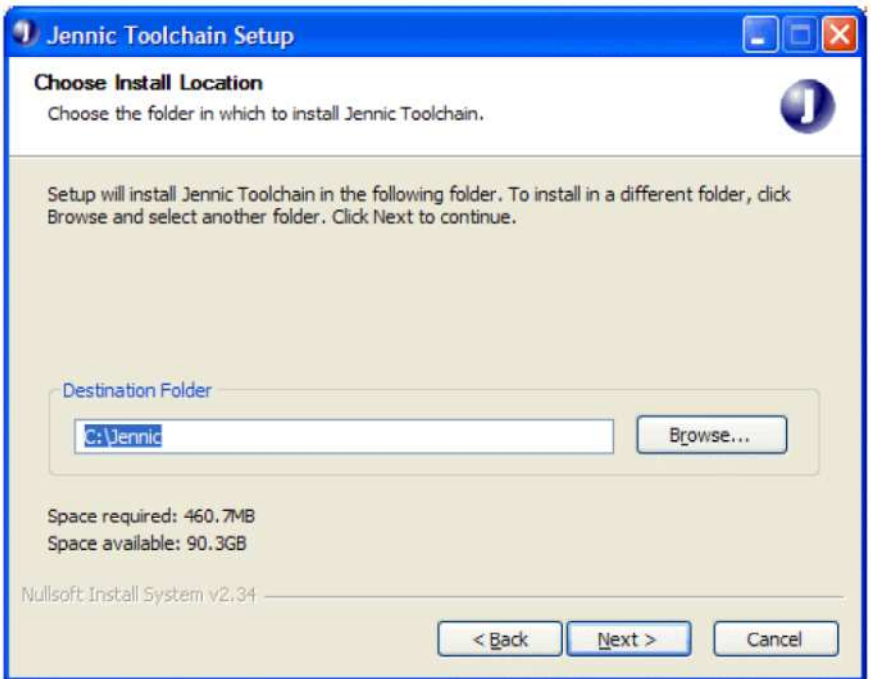

Figura A.3. Pantalla para elegir localización, de "Jennic Toolchain Setup" (38).

Por defecto, el directorio especificado es C:\Jennic. Si se desea, se puede especificar otro disco pero se debe mantener el nombre del directorio, por ejemplo, D:\Jennic. Hacer click en "Next" para continuar.

- Paso 5: En la siguiente pantalla, mostrada en la Figura A.4, especificar el nombre de la carpeta en la que se desea que aparezcan las herramientas en el menú de Inicio de Windows. Por defecto, se crea una carpeta denominada Jennic.

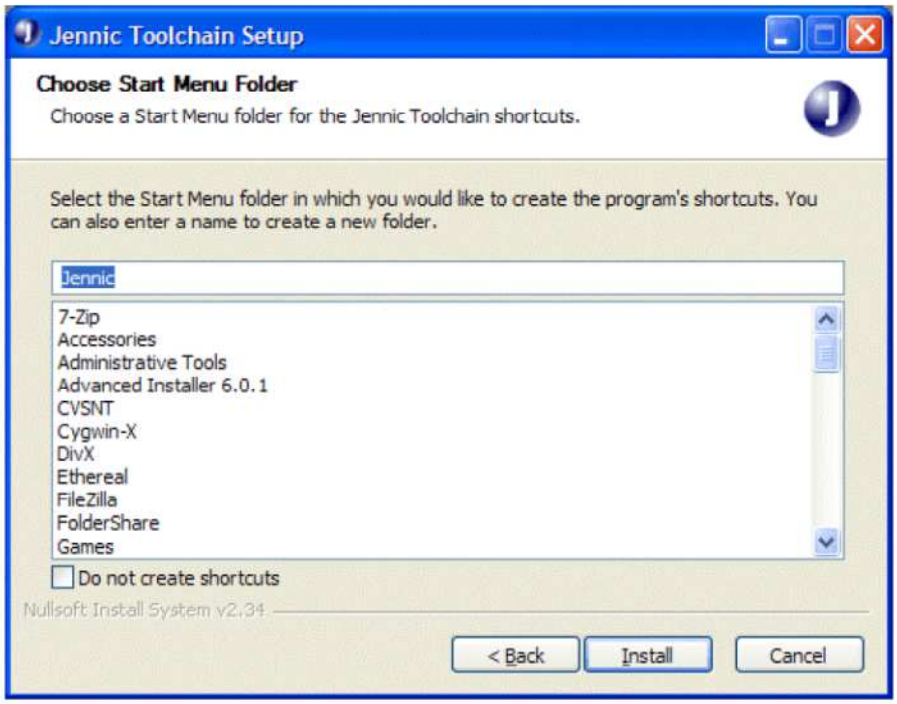

Figura A.4. Pantalla para elegir la carpeta del menú de inicio, de "Jennic Toolchain Setup" (38).

Hacer click en "Install".

- Paso 6: Esperar a que se complete la instalación (este proceso puede tardar unos minutos) y, cuando haya finalizado, hacer click en “Next” y a continuación en “Finish”.
- Paso 7: Reiniciar la máquina cuando sea requerido.

A.1.3 Instalación de las librerías del SDK

Para instalar las librerías del SDK se deben seguir los siguientes pasos:

- Paso 1: Asegurarse de que se han instalado las herramientas del SDK, como se describe en la sección A.1.2.
- Paso 2: Comenzar la instalación de las librerías.
    - Si se están instalando desde el CD: insertar el CD en la máquina y seguir las instrucciones mostradas en la pantalla. Si se está conectado a Internet, la página web de Jennic comprobará si existe una versión más reciente del instalador. Seleccionar el instalador “*SDK Libraries*”.
    - Si se va a descargar desde la página web de Jennic, descargar y guardar el archivo “JN-SW-4030-SDK-Libraries-vX.Y.exe” en la máquina y ejecutar el instalador.
- Paso 3: Seguir las instrucciones del *wizard* mostrado en pantalla. Se instalarán todos los componentes, ya que no se permite seleccionar componentes individualmente.
- Paso 4: En la siguiente pantalla (que se muestra en la Figura A.5) seleccionar la localización en la que se desea guardar las librerías:

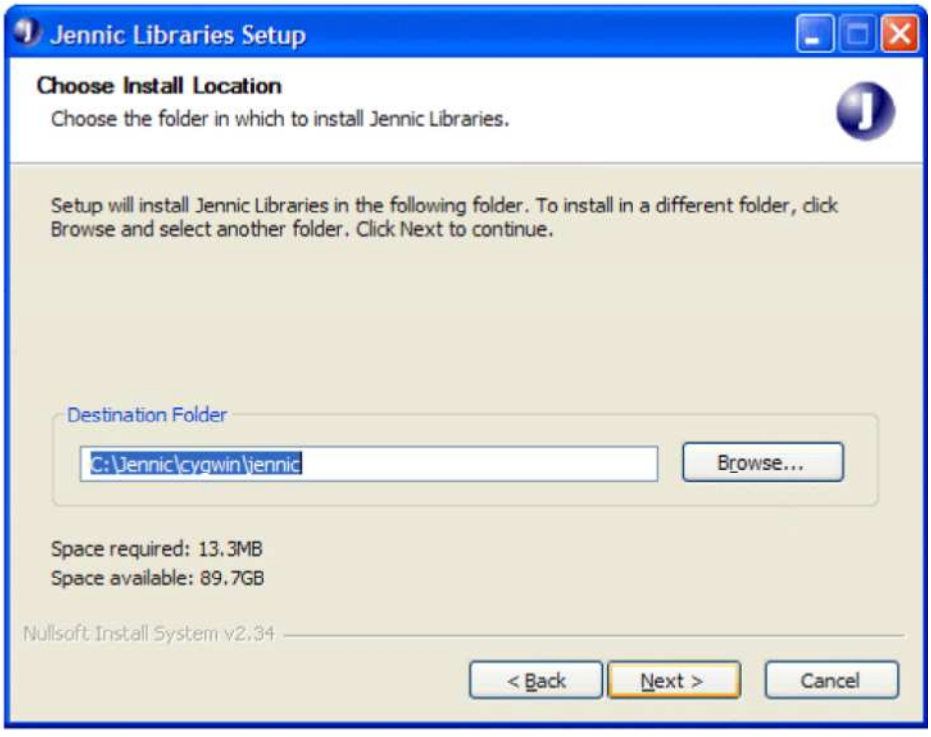

Figura A.5. Pantalla para elegir localización, de “Jennic Libraries Setup” (38).

Por defecto, el directorio especificado es C:\Jennic\cygwin\jennic. Si se desea, se puede especificar otro disco pero se debe mantener el nombre del directorio, por ejemplo, D:\Jennic\cygwin\jennic. Hacer click en “Next” para continuar.

- Paso 5: En la siguiente pantalla, mostrada en la Figura A.6, especificar el nombre de la carpeta en la que se desea que aparezcan las librerías en el menú de Inicio de Windows. Por defecto, se selecciona la carpeta Jennic.

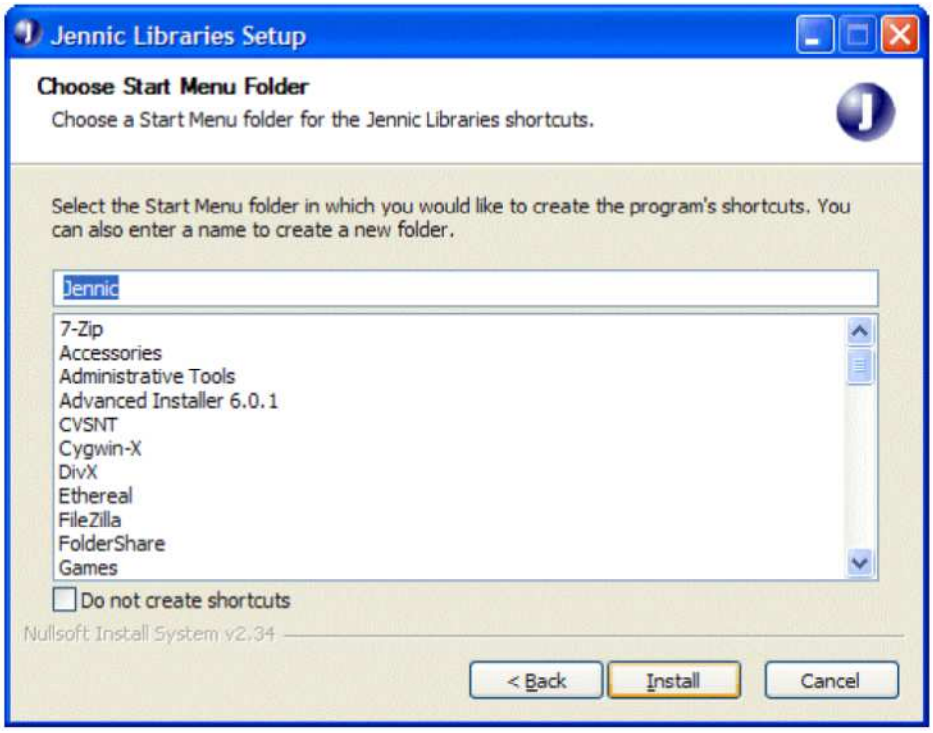

Figura A.6. Pantalla para elegir la carpeta del menú de inicio, de "Jennic Libraries Setup" (38).

  Hacer click en "Install".
- Paso 6: Esperar a que se complete la instalación y, cuando haya finalizado, hacer click en "Next" y a continuación en "Finish".

A.2 Entorno de desarrollo

Las herramientas de software disponibles en el SDK de Jennic incluyen dos tipos de entorno de desarrollo:

- Code::Blocks IDE (*Integrated Development Environment*): proporciona el software necesario para el desarrollo del código de las aplicaciones en una interfaz de usuario gráfica que incorpora todas las herramientas necesarias. La versión proporcionada ha sido especialmente adaptada por Jennic y ésta es la que se debe utilizar en el desarrollo de aplicaciones para los microcontroladores inalámbricos Jennic.
- Cygwin CLI (*Command Line Interface*): proporciona el software necesario para desarrollar el código de las aplicaciones utilizando herramientas mediante líneas de comando en el emulador Cygwin Linux (para Windows).

El entorno Cygwin se debe instalar siempre porque es necesario para las herramientas del compilador de Jennic y para Code::Blocks.

A.3 Utilización de Jennic Code::Blocks

A continuación se explica cómo crear aplicaciones utilizando Code::Blocks (41), si se ha elegido utilizar este entorno de desarrollo y se ha instalado.

A.3.1 Creación de proyectos

Para crear una aplicación, primero se debe crear un proyecto. Para crear un proyecto, se deben seguir los siguientes pasos:

- Paso 1: En Code::Blocks ir al menú File>New>Project. Se muestra la pantalla "New from template".
- Paso 2: En esta pantalla seleccionar el icono de Jennic y hacer click en "Go".

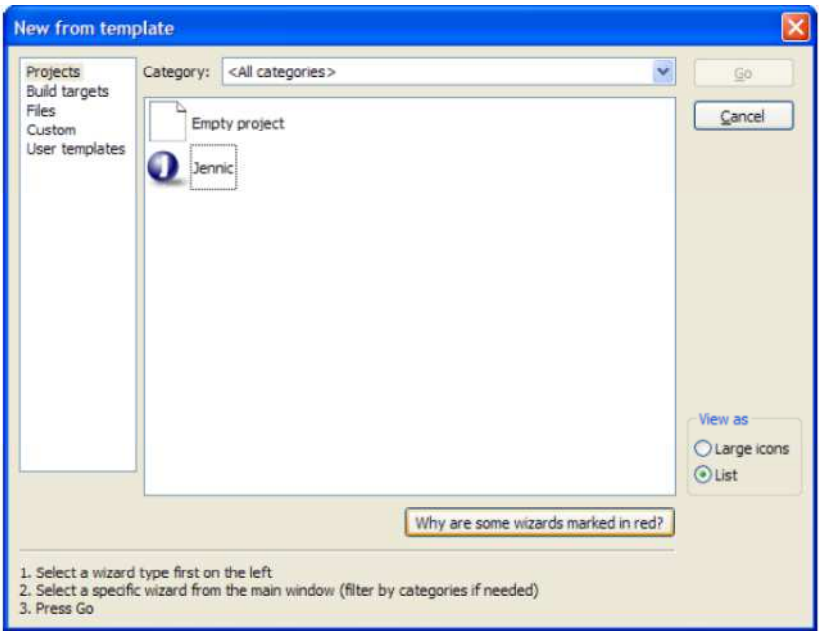

Figura A.7. Pantalla "New for template", en Jennic Code::Blocks (41).

- Paso 3: En la primera pantalla mostrada por el *wizard* hacer click en "Next" y seleccionar la opción para saltar esta página la próxima vez, si se desea.
- Paso 4: En la siguiente pantalla (mostrada en la Figura A.8), introducir la información relativa al proyecto: nombre del proyecto y nombre de la carpeta en la que se va a crear el proyecto. Si se está utilizando la última versión de Jennic SDK el directorio utilizado normalmente es: C:\Jennic\cygwin\jennic\SDK\. Si se está utilizando una versión anterior, el directorio utilizado normalmente es: C:\Jennic\cygwin\jennic\developer\. Los demás campos son completados por el *wizard* automáticamente.

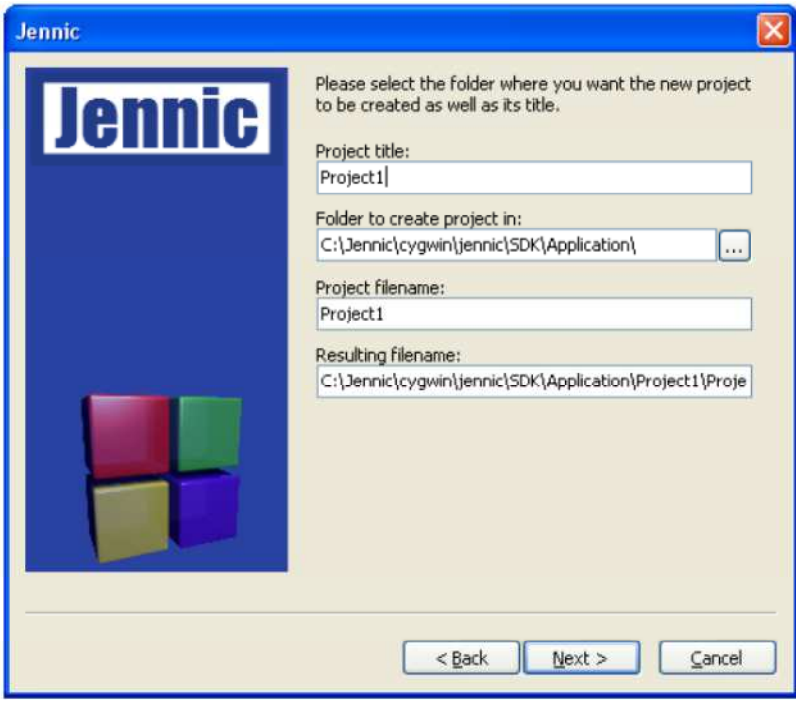

Figura A.8. Pantalla del *wizard* de Jennic Code::Blocks (41).

Hacer click en "Next".

- Paso 5: En la siguiente pantalla, mostrada en la Figura A.9, seleccionar JN51xx y hacer click en "Next".
- Paso 6: Seleccionar el entorno en el que se desarrolla la aplicación. Si se está utilizando la última versión de Jennic SDK el directorio en el que se trabaja normalmente es: C:\Jennic\cygwin\jennic\SDK\. Si se está utilizando una versión anterior, el directorio en el que se trabaja normalmente es: C:\Jennic\cygwin\jennic\developer\. Hacer click en "Next".

Figura A.9. Siguiente pantalla del *wizard* de Jennic Code::Blocks (41).

- Paso 7: En la siguiente pantalla elegir el tipo de aplicación entre los mostrados y hacer click en “Next”.
- Paso 8: Seleccionar el tipo de placas en las que se va a descargar la aplicación de entre los mostrados y hacer click en “Next”.
- Paso 9: En la siguiente pantalla, seleccionar el tipo de microcontrolador de entre los siguientes:
    - JN5121.
    - JN513x (seleccionar esta opción para JN5139).
    - JN513xR1 (seleccionar esta opción para la primera versión del JN5139).

  Hacer click en “Finish”.

A.3.2 Edición de proyectos

Para editar el código utilizando Code::Blocks como editor se deben seguir los siguientes pasos:

- Paso 1: Abrir el proyecto mediante el menú File>Open buscando el archivo .cbp del proyecto que corresponda en el directorio en el que se ha guardado.
- Paso 2: Hacer click en la pestaña “Projects” en el panel “Management” situado a la izquierda. El proyecto aparece en la lista titulada “Workspace”.
- Paso 3: Navegar a través de la estructura en árbol del proyecto hasta alcanzar el fichero .c situado en “Sources”. Inicialmente no tiene el mismo nombre que el proyecto. Si se desea, se puede renombrar haciendo click con el botón derecho en la pestaña “Project” y seleccionando “Rename file”. Se introduce el nuevo nombre y se hace click en “OK”.
- Paso 4: Para mostrar el código en el panel principal, hacer doble click en el fichero .c en la pestaña “Project”. Editar el código en el panel principal.
- Paso 5: Cuando se haya terminado de editar el fichero .c, guardar los cambios mediante el menú File>Save y cerrarlo mediante el menú File>Close file.
- Paso 6: Cuando se haya terminado de trabajar en el proyecto, guardar los cambios mediante el menú File>Save project y cerrarlo mediante el menú File>Close project.

A.3.3 Compilación de proyectos

Primero se deben especificar las rutas necesarias para el compilador y el enlazador. Para hacer esto se deben seguir los siguientes pasos:

- Paso 1: Ir al menú Project>Build options. Se muestra la pantalla “Project build options”.
- Paso 2: Hacer click en el nombre del proyecto en el panel de la izquierda y después hacer click en la pestaña “Directories”. Se obtiene la pantalla mostrada en la Figura A.10.

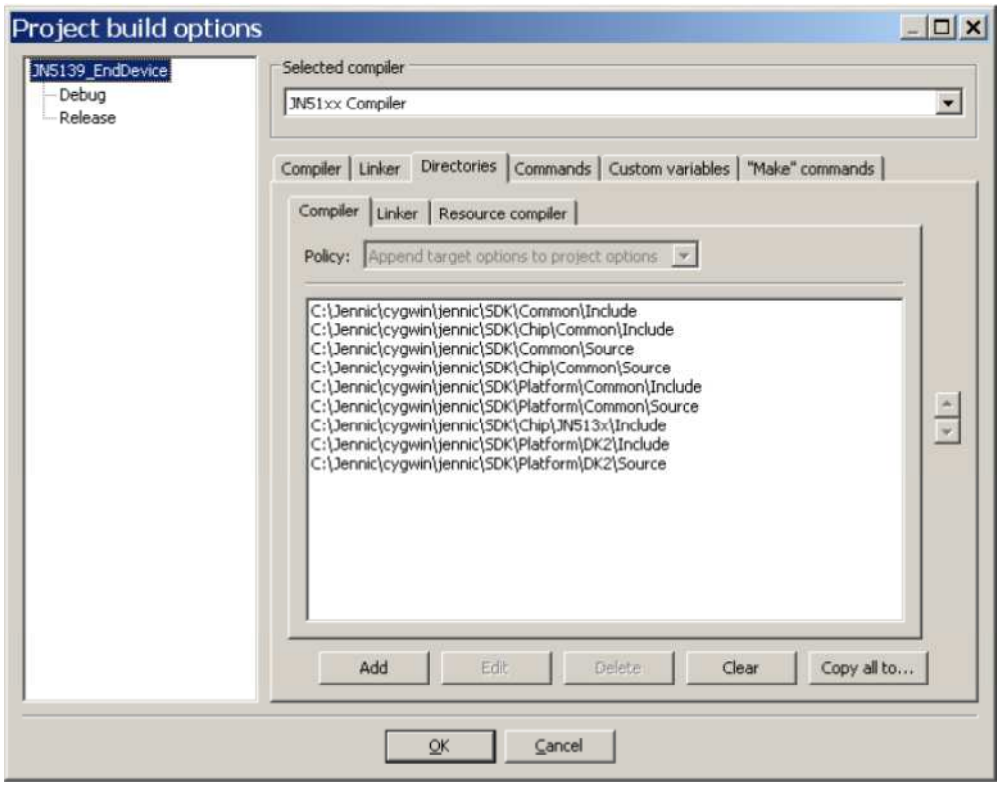

Figura A.10. Pantalla "Project build options", pestaña "Directories", de Code::Blocks (41).

- Paso 3: Hacer click en la subpestaña "Compiler" y asegurarse de que todas las rutas necesarias están especificadas. Si se desea añadir una ruta, hacer click en el botón "Add" para especificarla.
- Paso 4: Hacer click en la subpestaña "Linker" y asegurarse que la ruta está especificada de la siguiente forma:
    - Si se está trabajando en el directorio "SDK", la ruta depende del tipo de dispositivo:
        - Para JN5139: C:\Jennic\cygwin\jennic\SDK\Chip\JN513x\Build
        - Para JN5139R1: C:\Jennic\cygwin\jennic\SDK\Chip\JN513xR1\Build
        - Para JN5121: C:\Jennic\cygwin\jennic\SDK\Chip\JN5121\Build
    - Si se está trabajando en el directorio "developer", la ruta es: C:\Jennic\cygwin\jennic\developer\Build.
- Paso 5: Hacer click en la pestaña "Linker" para obtener la pantalla mostrada en la Figura A.11:

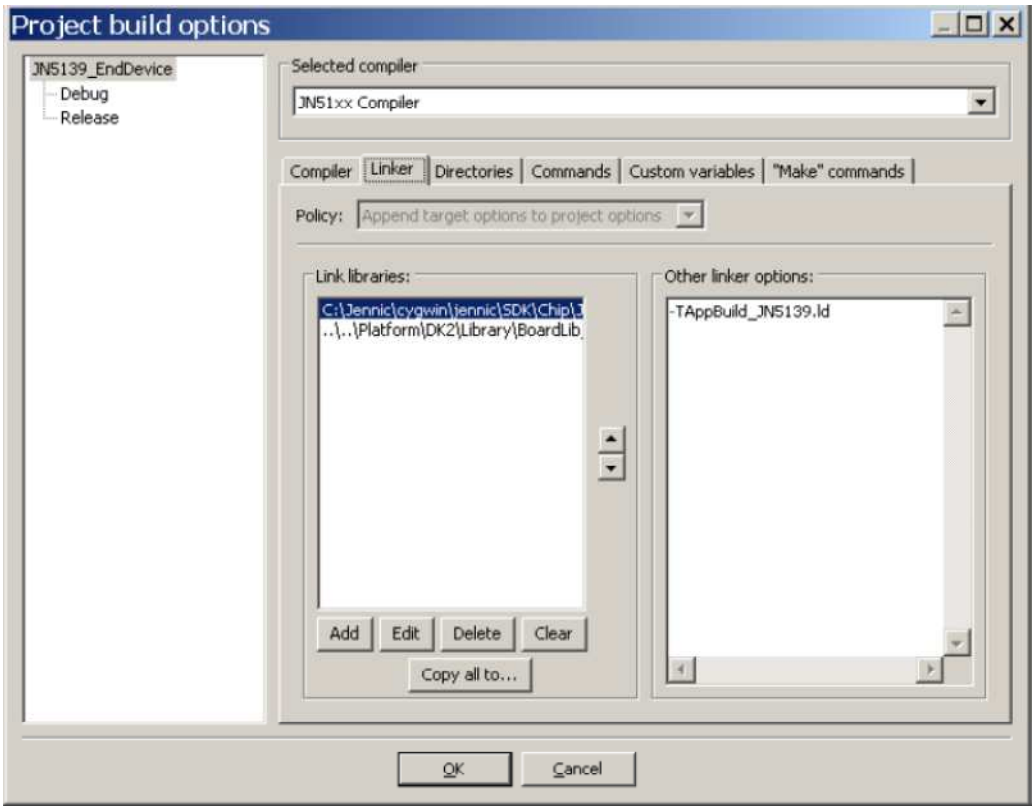

Figura A.11. Pantalla "Project build options", pestaña "Linker", de Code::Blocks (41).

Asegurarse de que las rutas a todas las librerías necesarias se especifican en el campo "Link libraries". Si se desea añadir una ruta, hacer click en el botón "Add" para especificarla.

- Paso 6: Hacer click en "OK" para guardar los cambios y salir de la pantalla "Project build options".

A continuación, para compilar un proyecto se deben seguir los siguientes pasos:

- Paso 1: Abrir el proyecto, hacer click en la pestaña "Projects" y seleccionar el proyecto que corresponda.
- Paso 2: En el menú Build>Select target seleccionar "Debug" si se está compilando con propósito de depurar el código o "Release" si se está compilando una versión final. También se puede realizar este paso utilizando la lista desplegable en la barra de herramientas de Code::Blocks.
- Paso 3: Compilar el proyecto mediante el menú Build>build. El archivo binario .bin resultante se guarda en una subcarpeta creada durante el proceso de compilación llamada "JN5121_Build" o "JN5139_Build" (dependiendo del dispositivo), si se está trabajando en el directorio "SDK", o "Build" si se está trabajando en el directorio "developer". En ambos casos, los archivos binarios se dividen en dos subcarpetas más, "Debug" o "Release", dependiendo de la opción seleccionada.

A.3.3.1 Compilación utilizando *makefile*

También se puede compilar utilizando un archivo *makefile*, como se explicó en el Capítulo 4. El archivo *makefile* debe estar colocado en la carpeta "Build". Se inicia cygwin y se especifica la ruta hasta el directorio "Build". El tipo de dispositivo se debe indicar mediante línea de comandos, por ejemplo, si se está utilizando la primera versión de los dispositivos JN5139 el

comando que se debe invocar es: “make –f makefile JENNIC_CHIP=JN5139R1”. El archivo binario se crea en la carpeta “Build”.

A.4 Utilización de Flash Programmer

Una vez que el código está compilado, se debe utilizar el programa Flash Programmer (42) para descargar el archivo binario en la memoria Flash de los microcontroladores de los dispositivos. También se puede utilizar Flash Programmer integrado en Code::Blocks, pero aquí no se explica esta forma porque no es la que se ha utilizado.

El programa se ejecuta haciendo doble click en el archivo “FlashGUI.exe”. Flash Programmer presenta la interfaz de usuario mostrada en la Figura A.12, que permite descargar el archivo .bin especificado.

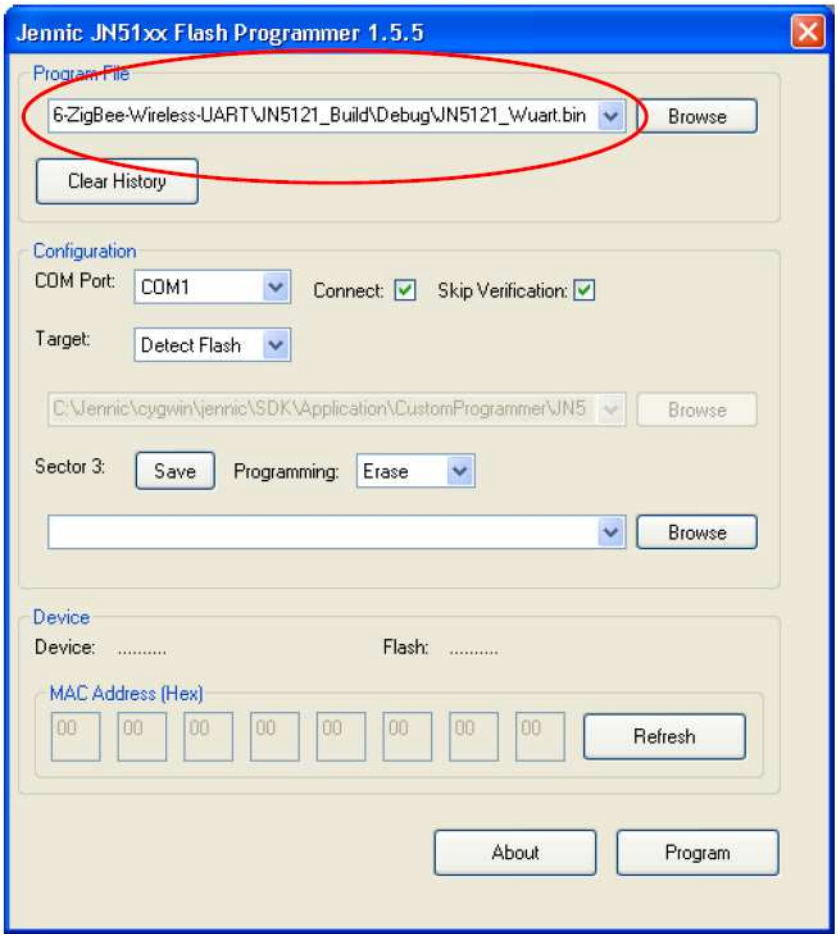

Figura A.12. Interfaz de usuario de Flash Programmer (42).

Para descargar el archivo binario en las placas, se debe llevar a cabo el siguiente procedimiento:

- Paso 1: Conectar el ordenador al dispositivo utilizando el cable puerto USB a puerto serie, disponible en el kit proporcionado por Jennic, con la polaridad correcta en el puerto serie UART del dispositivo, como se vio en el Capítulo 4.

- Paso 2: Ejecutar la aplicación Flash Programmer y seleccionar el puerto de comunicaciones serie del ordenador al que está conectado la placa.
- Paso 3: Poner el dispositivo en modo programación. Para ello se realizan los siguientes pasos:
    - Mantener pulsado el interruptor de programación en la placa.
    - Pulsar y soltar el interruptor de reset.
    - Soltar el interruptor de programación.
- Paso 4: En la interfaz de Flash Programmer, se utiliza el botón "Browse" para buscar y seleccionar el archivo binario que se desea descargar.
- Paso 5: Una vez que se ha seleccionado el fichero, hacer click en el botón "Program" para iniciar la descarga. Mientras la memoria Flash se está programando aparece la pantalla mostrada en la Figura A.13, en la que se puede observar el progreso de la descarga.

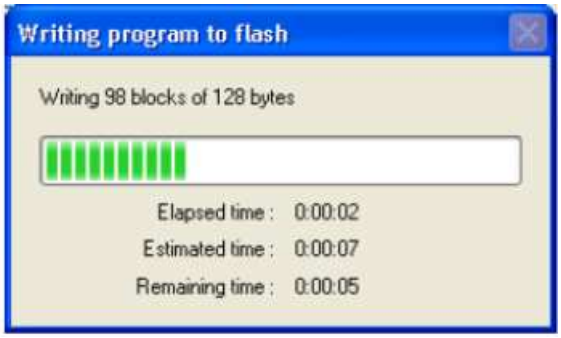

Figura A.13. Pantalla mostrada durante la descarga del archivo binario en la memoria Flash (42).

Cuando finaliza la descarga, se muestra si el resultado ha sido satisfactorio o se han producido errores. Si se ha producido algún error, se debe intentar de nuevo el proceso de descarga.

- Paso 6: Cuando la descarga se ha completado satisfactoriamente se debe desconectar el cable y resetear la placa. El código descargado se ejecutará automáticamente.

A.4.1 Conexión de dispositivos

El cuadro de diálogo "Connect", mostrado en la Figura A.14, se utiliza para conectar un dispositivo en el puerto serie seleccionado. Para que Flash Programmer libere ese puerto, se debe desmarcar esa opción. De esta forma, el puerto serie puede ser utilizado por otras aplicaciones (como por ejemplo, Jen Term). Para que Flash Programmer vuelva a utilizar este puerto se debe volver a marcar esta casilla.

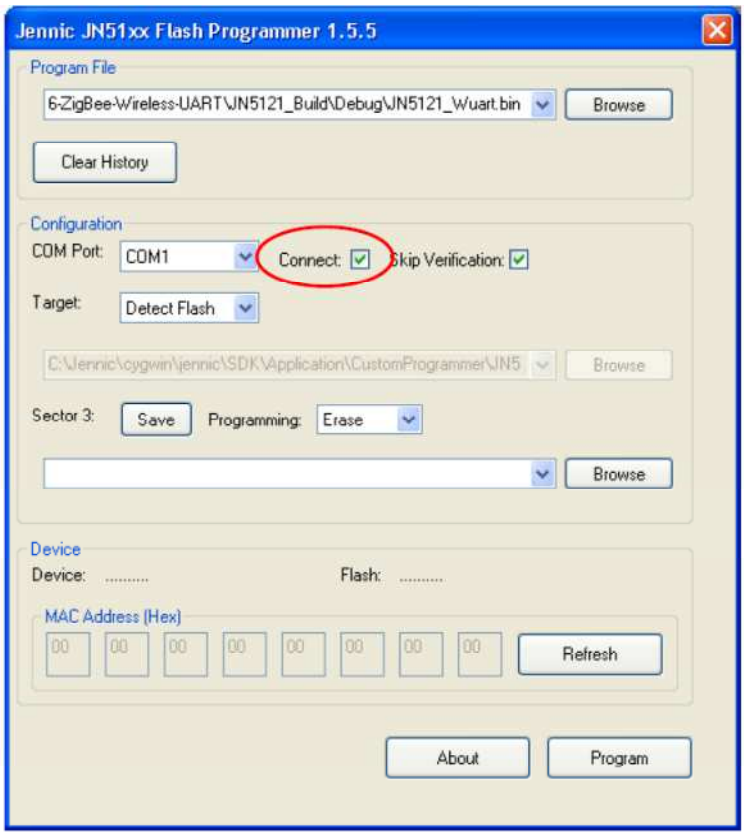

Figura A.14. Cuadro de diálogo "Connect" en la interfaz de usuario de "Flash Programmer" (42).

Si Flash Programmer no puede detectar el dispositivo, se muestra una advertencia por pantalla para indicar que se debe comprobar la alimentación del dispositivo y el cable y que el dispositivo se debe poner en modo programación.

Para verificar la conectividad, se pulsa el botón "Refresh", indicado en la Figura A.15. Se muestra la información relativa al dispositivo (tipo y dirección MAC) en el campo "Device".

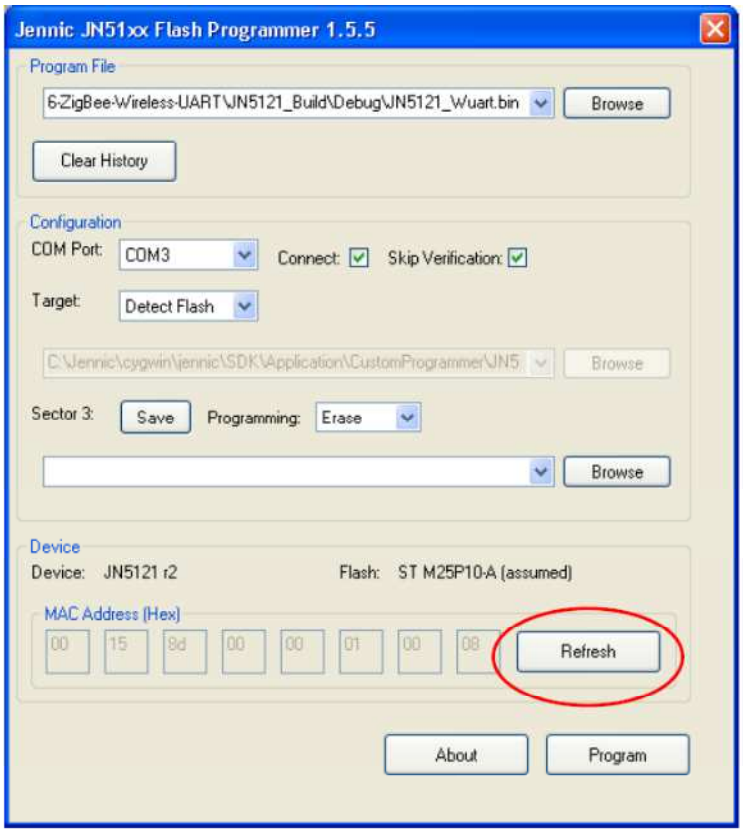

Figura A.15. Botón "Refresh" en la interfaz de usuario de "Flash Programmer" (42).

A.4.2 Identificación del puerto de comunicaciones utilizado

Para averiguar a través de qué puerto de comunicaciones del ordenador está conectada la placa, se pueden seguir los siguientes pasos:

- Paso 1: En el menú de inicio de Windows seleccionar "Panel de control" y a continuación "Sistema". Aparece la pantalla de "Propiedades del sistema".
- Paso 2: En esta pantalla seleccionar la pestaña "Hardware" y hacer click en el botón "Administrador de dispositivos". Aparece la pantalla de "Administrador de dispositivos".
- Paso 3: En la pantalla de "Administrador de dispositivos", buscar en la lista la pestaña "Puertos" y desplegarla. Identificar el puerto que está conectado a la placa, por ejemplo, "Puerto de comunicaciones (COM1)".

A.5 Utilización de Jen Term

Dentro de Flash Programmer, otro de los componentes instalados es Jen Term. Se ejecuta haciendo doble click en el archivo "JenTerm.exe". La pantalla que aparece al ejecutarse se muestra en la Figura A.16:

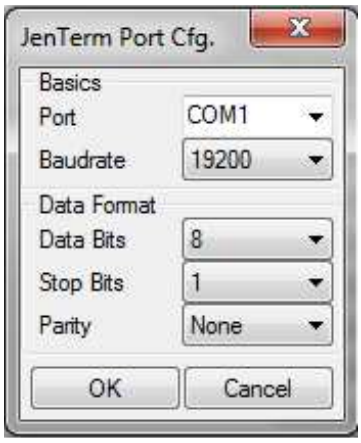

Figura A.16. Interfaz de usuario de Jen Term.

En ella se debe seleccionar el puerto de comunicaciones en el que está conectado el dispositivo, la velocidad de transmisión (en el caso de JN5139 son 19200 baudios) y hacer click en el botón "OK". Los valores de los demás campos son los que aparecen por defecto.

A continuación aparece la pantalla del terminal del puerto correspondiente. Ésta se utiliza para imprimir por pantalla mensajes y los valores de las variables que se deseen mostrar.

A.6 Alimentación de los dispositivos

Las placas pueden ser alimentadas por dos pilas AAA o por una fuente de alimentación externa (PSU, *Power Supply Unit*).

El tipo de fuente de alimentación se selecciona mediante el *jumper* J2 :

- Para utilizar pilas, los pines 1 y 2 del *jumper* J2 deben estar cerrados.
- Para utilizar una fuente de alimentación externa, los pines 2 y 3 del *jumper* J2 deben estar cerrados.

Para encender las placas se utiliza el interruptor SW6, independientemente de la fuente de alimentación utilizada.