



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación



Trabajo Fin de Máster

MICROSISTEMA BASADO EN 802.15.4 PARA SENSÓRICA

Autor: D. JOSUÉ DANIEL VIERA RIVERO
Tutores: Dr. D. ROBERTO ESPER-CHAÍN FALCÓN
Dr. D. FÉLIX TOBAJAS GUERRERO
Fecha: Julio 2011



t +34 928 451086 | iuma@iuma.ulpgc.es
f +34 928 451083 | www.iuma.ulpgc.es

Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación



Trabajo Fin de Máster

MICROSISTEMA BASADO EN 802.15.4 PARA SENSÓRICA

HOJA DE FIRMAS

Alumno: D. JOSUÉ DANIEL VIERA RIVERO Fdo.:

Tutor: Dr. D. ROBERTO ESPER-CHAÍN FALCÓN Fdo.:

Tutor: Dr. D. FÉLIX TOBAJAS GUERRERO Fdo.:

Fecha: Julio 2011



t +34 928 451086 iuma@iuma.ulpgc.es
f +34 928 451083 www.iuma.ulpgc.es

Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria



UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
Instituto Universitario de Microelectrónica Aplicada
Sistemas de información y Comunicaciones

Máster en Tecnologías de Telecomunicación



Trabajo Fin de Máster

MICROSISTEMA BASADO EN 802.15.4 PARA SENSÓRICA

HOJA DE EVALUACIÓN

Calificación:

Presidente . Fdo.:

Secretario . Fdo.:

Vocal . Fdo.:

Fecha: Julio 2011



t +34 928 451086 iuma@iuma.ulpgc.es
f +34 928 451083 www.iuma.ulpgc.es

Campus Universitario de Tafira
35017 Las Palmas de Gran Canaria

Agradecimientos

Quiero dar mis más sinceros agradecimientos a Roberto y a Félix, por haber dirigido este Trabajo Fin de Máster. Gracias por vuestro tiempo, dedicación y por la confianza que habéis depositado en mí para la realización del proyecto. Y sobretodo, por darme la oportunidad de hacer lo que más me gusta, que es aprender, utilizar lo que he aprendido, y “cacharrear” con la electrónica.

Agradecer a toda mi familia, por su sacrificio, paciencia y cariño. Ellos me han transmitido los valores de honestidad, humildad y esfuerzo para poder conseguir lo que uno quiere.

A Nano y a Juan, por ser mis mejores amigos en estos últimos años, y espero que durante muchos más. Gracias por compartir conmigo aficiones, reflexiones, curiosidades y aventuras. Gracias por vuestro apoyo en los momentos más duros, peleando juntos en todas las batallas que hemos librado, ganásemos o perdiésemos.

Al resto de mis compañeros de Teleco y amigos: gracias por esos ratos que hemos compartido durante todos estos años, en especial esos momentos de alegrías, aventuras e ilusiones. Aunque con el tiempo cada uno va haciendo su camino, siempre guardo con especial aprecio vuestro cariño y todos los momentos que hemos vivido.

Reconocer también el mérito a los profesores que realmente tienen interés en enseñar. Darles mi apoyo por el enorme sacrificio que conlleva la docencia en una juventud con menos interés por aprender. Gracias por su dedicación y los conocimientos que me han aportado durante todos estos años.

Para terminar, incluir también, por que no, a todas aquellas personas del pasado que han puesto de su parte para que yo pudiera llegar hasta este punto y ser la persona que soy.

A todos, muchas gracias.

Josué Daniel Viera Rivero

Los espíritus vulgares no tienen destino.

Platón

*Paso a paso. No concibo ninguna otra manera
para lograr las cosas.*

Michael Jordan

*Solo los moteros saben porque los perros
sacan la cabeza por la ventanilla.*

Un motero

Resumen

El objetivo principal de este Trabajo Fin de Máster consiste en el desarrollo de un sistema para aplicaciones con sensores. En este Trabajo Fin de Máster (TFM) se abordan las tareas de estudiar el estándar de comunicaciones IEEE 802.15.4 y su implementación en placas de desarrollo. La especificación de este TFM se orienta a una aplicación de sensórica, como la detección de presencia o la captura de variables como temperatura, luminosidad, etc.

Para comunicarse con los nodos, programarlos o poder visualizar la información recibida, se ha desarrollado un software visor (Interfaz Gráfica de Usuario). De esta manera, la programación y la visualización de los datos se realiza en tiempo real, de una manera clara y sencilla.

Abstract

The main objective of this Master's Thesis is the development of a sensor application system. This Master's Thesis (TFM) focuses in studying the IEEE 802.15.4 communications standard and its implementation in a development board. The specification is focused to a sensor application, such as presence detection or the capture of variables such as temperature, light, etc..

To communicate with the nodes, programming them or to view the information received, a viewer software (Graphic User Interface) has been developed. By this way, programming and the visualization of data is done in real time, in a clear and simple way.

Índice General

Índice General	I
Índice de Tablas	III
Índice de Figuras	V
Índice de Listados	VII
Acrónimos	IX
I MEMORIA	1
1. Introducción	3
1.1. Antecedentes	3
1.2. Objetivos	5
2. Estado del Arte de las Redes de Sensores	7
2.1. Redes de Sensores	7
2.1.1. Características Generales	8
2.2. Tipos de Comunicaciones Inalámbricas	11
2.2.1. Redes Inalámbricas para WSN	13
2.3. Hardware y Software	21
3. Estudio de la Solución Adoptada	31
3.1. RZRAVEN	31
3.2. IEEE 802.15.4	34
3.2.1. Llamadas a la Capa de Datos (MCPS)	36

3.2.2. Llamadas a la Capa de Gestión (MLME)	38
3.2.3. Descripción de Tareas	42
3.3. Pila ATMEL–MAC	47
4. Diseño e Implementación	49
4.1. Diseño Placa Auxiliar	49
4.2. Modificación de los Ficheros AVRRAVEN	53
4.3. Coordinador	56
4.3.1. Ejemplo Básico	56
4.3.2. Modificaciones	60
4.4. Nodo Final	64
4.5. Aplicación Visor	67
5. Verificación	73
5.1. Verificación del Sistema	75
5.2. Análisis de Tramas	80
5.2.1. Formato Trama MAC	80
5.2.2. Capturas del Sniffer	81
6. Conclusiones y Líneas Futuras	89
6.1. Conclusiones	89
6.2. Líneas Futuras	90
Bibliografía	94

Índice de Tablas

2.1. Resumen de Tipos de Comunicación Inalámbrica	12
3.1. Primitivas <i>MLME-SAP</i>	39
4.1. Pines ATmega1284P	50
4.2. Trama Visor-RZUSBSTICK	62
4.3. Trama Comando <i>L</i>	62
4.4. Estructura de las Instrucciones de Comunicación	63
4.5. Instrucción Enviada por el Coordinador	63
4.6. Instrucción Enviada por un Nodo Final	64
5.1. Formato de <i>Frame Control</i>	81
5.2. Tipos de Trama MAC	81
5.3. Datos Enviados por el Primer Nodo	86

Índice de Figuras

2.1. Red de Sensores	8
2.2. Comparativa de Tipos de Comunicación Inalámbrica	12
2.3. Topología en Estrella	16
2.4. Redes Punto a Punto	17
2.5. Clúster	17
2.6. Comunicación con un Coordinador en una red <i>NonBeacon</i>	18
2.7. Comunicación con un Coordinador en una red <i>Beacon</i>	19
2.8. Petición de Datos al Coordinador en una red <i>Beacon</i>	19
2.9. Supertrama	20
2.10. Plataforma Rene	22
2.11. Plataforma Mica	24
2.12. Mota Blue	25
2.13. Plataforma Waspote	26
2.14. Microchip MRF24J40MA	27
2.15. Atmel STK600	28
2.16. Funcionalidad	29
3.1. RZRAVEN	31
3.2. Diagrama de Bloques de la tarjetas AVRRAVEN	33
3.3. Diagrama de Bloques del dispositivo RZUSBSTICK	33
3.4. Primitivas entre Capas	34
3.5. Modelo de la subcapa MAC	36
3.6. Primitivas <i>MCPS-DATA</i>	38
3.7. Mensajes y Primitivas para la Asociación	40
3.8. Secuencia de Mensajes <i>MLME-START</i>	41

3.9. Secuencia de Mensajes <i>MLME-POLL</i>	42
3.10. Primitivas para la Creación de una red PAN	46
4.1. Esquemático de la Placa Auxiliar Diseñada	51
4.2. Ruteado de la Placa Auxiliar Diseñada	52
4.3. Placa Auxiliar Implementada	53
4.4. Diagrama de Estados del Ejemplo	57
4.5. Diagrama de Estados del Coordinador	61
4.6. Diagrama de Estados del Nodo Final	65
4.7. Panel Frontal de la Aplicación Desarrollada en LabView	68
4.8. Conexionado de Bloques en LabView	68
4.9. Código para el Puerto Serie	69
4.10. Interfaz Gráfica para el Puerto Serie	69
4.11. Interfaz Gráfica para Enviar Instrucción	70
4.12. Código para el Refresco de la Pantalla	71
4.13. Código del Conversor	71
4.14. Botón STOP/Instrucción Salir	72
5.1. Sistema Implementado	74
5.2. JTAGICE mkII	74
5.3. Visor Conectado	75
5.4. RZUSBSTICK Recibiendo Datos	76
5.5. Nodo 1 Enviando Datos	77
5.6. Nodos 1 y 2 Enviando Datos	77
5.7. Envío Fallido al Nodo 7	78
5.8. Envío Instrucción ‘Encender LED’ al Nodo 1	79
5.9. Envío Instrucción ‘Apagar LED’ al Nodo 1	79
5.10. Tipos de Trama MAC	80
5.11. Terminal en Modo <i>Promiscuo</i>	82
5.12. Tipos de Comando	83

Índice de Listados

3.1. Pritimiva <i>MCPS-DATA.request</i>	37
4.1. <i>pal_config.h</i>	53
4.2. <i>pal_board.c</i>	54
4.3. <i>usr_mlme_reset_conf</i>	58
4.4. <i>usr_mlme_set_conf</i>	59
4.5. <i>usr_mlme_start_req</i>	59
4.6. Comando L	66

Acrónimos

Acrónimo	Significado
ACK	Acknowledgement
ADC	Analog to Digital Converter
ASCII	American Standard Code for Information Interchange
CSMA-CA	Carrier Sense Multiple Access with Collision Avoidance
DSN	Data Sequence Number
DSSS	Direct-Sequence Spread Spectrum
FFD	Full-Function Device
FHSS	Frequency-Hopping Spread Spectrum
GTS	Guaranteed Time Slots
GUI	Graphical User Interface o <i>Interfaz Gráfica de Usuario</i>
I/O	Input / Output
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control
MAC-PIB	MAC-PAN Information Base
MCPS	MAC Common Part Sublayer
MLME	MAC subLayer Management Entity
MSDU	MAC Service Data Unit
PAN	Personal Area Network
PCB	Printed Circuit Board

Acrónimo	Significado
PHY	Physical Layer
PIB	PAN Information Base
PIC	Programmable Interface Controller
PLME	Physical Layer Management Entity
QPSK	Quadrature Phase Shift Keying
RISC	Reduced Instruction Set Computer
RFC	Request For Comments
RFD	Reduced-Function Device
SAP	Service Access Point
SPI	Serial Peripheral Interface
TFM	Trabajo Fin de Máster
UART	Universal Asynchronous Receiver-Transmitter
USB	Universal Serial Bus
USB-CDC	USB Communications Device Class
WLAN	Wireless Local Area Network
WPAN	Wireless Personal Area Network
WSN	Wireless Sensor Network

Parte I

MEMORIA

Capítulo 1

Introducción

1.1. Antecedentes

Hoy día, las redes de sensores han tomado gran relevancia debido a la creciente necesidad en ciertas aplicaciones de tomar valores en diversos puntos simultáneamente frente a una única localización y transmitirlos a un nodo central. La gestión medioambiental, el control de constantes vitales de pacientes y el hogar inteligente son sólo algunas de estas aplicaciones. En esta última es muy importante el control del gasto energético de los sistemas de climatización. Un sistema que permita identificar la distribución de temperaturas dentro de un recinto es indispensable para obtener los parámetros clave de los algoritmos de gestión del recinto [1][2][3].

Otras aplicaciones tienen la necesidad de detectar un cambio en alguna variable y alertar al sistema si se supera un determinado nivel umbral. En el campo de la seguridad es necesario proteger zonas de acceso restringido. Un intruso que acceda a ellas podría activar un nodo con sensores de movimiento, que generaría un mensaje de alarma con su localización y lo transmitiría a la red. Otro ejemplo claro son los sistemas orientados a la detección de conatos de incendios, activándose al sobrepasar un límite preestablecido de temperatura o de gases [4][5][6].

Estas aplicaciones han sido posible gracias a los avances en la electrónica y las comunicaciones de los últimos años. Con la miniaturización, tanto de componentes electrónicos como de los circuitos integrados, unido a la mejora de los algoritmos y de los dispositivos para la gestión del consumo de energía, ha sido posible alargar la vida

útil de los dispositivos. Las comunicaciones cableadas limitan el área de actuación y fuerzan a una ubicación fija. Además, tienen asociado un alto coste de despliegue, de mantenimiento, y son vulnerables a cortes del cableado, produciéndose en este último caso, una interrupción en las comunicaciones de difícil diagnóstico y arreglo.

Por ello, las nuevas aplicaciones de sensores buscan utilizar las ventajas que brinda las comunicaciones inalámbricas. Un beneficio inmediato de una tecnología inalámbrica es la reducción drástica en los costes de cable y la sencillez y rapidez en su despliegue. Otro beneficio es su capacidad para poder llegar a lugares inaccesibles o complicados. Un ejemplo claro es la recogida de datos en ambientes peligrosos, remotos o incluso especialmente hostiles. Otra ventaja es la movilidad; si al sistema de medida, además de disponer de la característica inalámbrica, se le dota de un sistema de energía autónomo, el sistema de medida podría ser móvil. El único requisito exigible sería el volumen y el peso del propio sistema de medida. Un nodo lo suficientemente pequeño y discreto, permitiría el control, seguimiento y localización de animales o personas [7][8].

Por todo ello, las redes de sensores inalámbricos permiten la captura, procesamiento y tratamiento de datos de una manera eficaz. Las redes se basan en la disposición de un número elevado de nodos independientes que gracias a sus sensores, capturan los datos de interés, y mediante un microprocesador, son capaces de pre-procesar esos datos para su posterior envío a un nodo central. El estándar IEEE 802.15.4 se desarrolló específicamente para los requisitos y funcionalidades requeridas por las redes de sensores [9][10]. Una baja transferencia de datos, la capacidad de sincronización de los nodos y unas radios con bajos consumos, permiten implementar los nodos requeridos para las aplicaciones comentadas. Uno de los fabricantes que facilitan kits de desarrollo con estas capacidades inalámbricas, amplia documentación y herramientas libres, es Atmel [11].

1.2. Objetivos

El objetivo principal de este Trabajo Fin de Máster (TFM) es crear un microsis-tema para aplicaciones con sensores. La especificación de este TFM se orienta a una aplicación de detección de presencia.

Los requisitos que ha de cumplir el sistema a desarrollar son los siguientes:

- Serán posibles las comunicaciones inalámbricas de varios nodos.
- Se desarrolla a partir de kits de desarrollo de Atmel.
- Se implementa sobre el estándar IEEE 802.15.4.
- Se podrán monitorizar los mensajes de los nodos y los sensores en una aplicación software visor desarrollada en LabView.
- Posibilitará el futuro desarrollo, ampliación e investigación de aplicaciones de redes de sensores

Capítulo 2

Estado del Arte de las Redes de Sensores

2.1. Redes de Sensores

Una red de sensores es una aplicación en la que varios dispositivos de medida independientes entre sí se intercomunican con el objeto de realizar una tarea común. Por lo general, se basa en un elevado número de nodos o dispositivos provistos de sensores, que pueden abarcar grandes extensiones y que son capaces de transmitir datos en una red *ad-hoc* sin infraestructura física preestablecida. Por lo tanto, es capaz de autoorganizar y autoconfigurar una red entre los distintos nodos, así como de encaminar los mensajes hasta su destino sin tablas de encaminamiento estáticas.

Las redes de sensores inalámbricas, o Wireless Sensor Networks (WSN) [7], tienen además unos requerimientos específicos de reducido coste, bajas transferencias de datos, mínimo consumo, coberturas de medio o gran alcance, y movilidad. Su principal interés es su utilidad en investigaciones científicas para la recopilación de un gran número de datos de una nube de puntos dispersos, aunque otras áreas, como la seguridad y los hogares inteligentes también se han beneficiado de los avances y descubrimientos en este área. Un ejemplo de un despliegue se puede observar en la Figura 2.1.

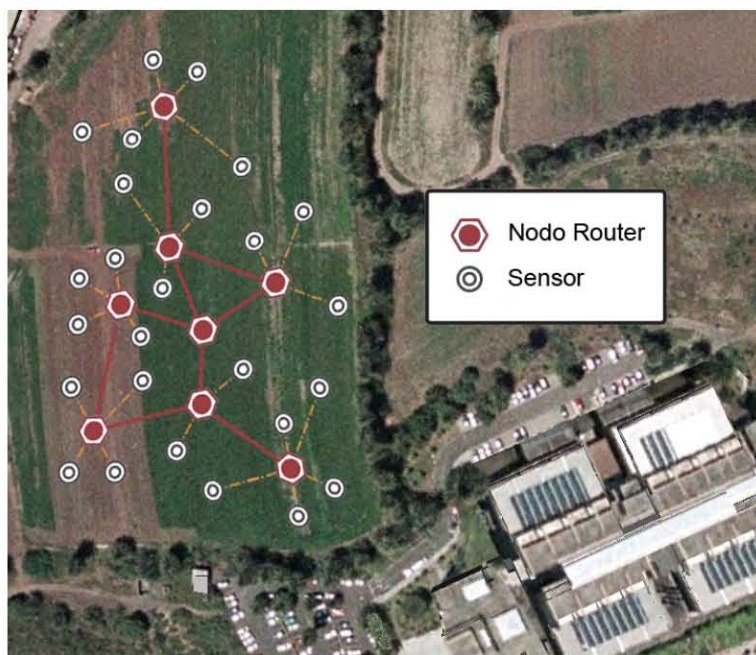


Figura 2.1: Red de Sensores

2.1.1. Características Generales

En general, las redes de sensores inalámbricas abarcan grandes extensiones, pueden tener capacidades de autoorganización, creación automática de reglas de encaminamiento, y son capaces de modificar su comportamiento ante ciertos eventos programados. Estas características se deben a su propia naturaleza frente a las redes de comunicación tradicionales, ya que no tienen una infraestructura física que limite su topología y extensión. Su misión principal es tener la capacidad de medir mediante sensores un entorno o lugar, y a continuación almacenar o transmitir los mensajes mediante un medio inalámbrico, en una red con múltiples nodos o dispositivos. Como se ha dicho, en la actualidad el ámbito de aplicación de las redes de sensores se reduce, bien a la investigación de parámetros físicos distribuidos, o bien a aplicaciones específicas en el campo de la ingeniería de control. Los principales factores que frenan su implantación son la baja capacidad de cómputo, la capacidad de memoria para almacenar los datos, el tamaño físico del dispositivo, la seguridad de las comunicaciones y de los datos, el coste, y el más importante: la energía disponible. Este último factor es el determinante para la duración de la vida del dispositivo. Aunque lo lógico sea pensar solamente en la capacidad de la batería, hay otros factores como la velocidad de cómputo del dispo-

sitivo y los periodos de desconexión, que en conjunto, determinan la duración final de días, meses o años del nodo.

Una red de sensores puede realizar una o varias de las siguientes tareas [8]:

1. **Monitorización:** Debe capturar y almacenar o transmitir el dato de una variable, desde una localización o un área de extensión. Se puede realizar de manera puntual o con tomas periódicas.
2. **Detección de eventos:** Debe capturar un evento o cambio de una variable, e informar de ello.
3. **Clasificación de objetos:** Identificar un objeto o evento. La identificación debe de ser correcta, tomando varias tomas de datos, realizando un análisis con datos compartidos con varios nodos adyacentes, y llegando a una conclusión.
4. **Seguimiento de objetos:** Fijar un objeto y hacer un seguimiento de sus movimientos y de su posición en el área de cobertura de la red.

Fijadas las tareas que realiza la red de sensores, se diseña la aplicación objetivo. La aplicación más inmediata para un sensor inalámbrico sería la de sustituir su homólogo cableado. Sería simplemente convertir una red de sensores cableada en inalámbrica. Pero la capacidad inalámbrica añade la posibilidad de cubrir campos o entornos totalmente nuevos. El despliegue rápido e inmediato, en cualquier tipo de terreno o ambiente peligroso, remoto o incluso especialmente hostil, lo hacen muy interesante para aplicaciones militares o científicas que requieren recuperar datos alterando lo menos posible los datos de las variables a observar. Los tipos de variables sólo dependen de la existencia de un sensor capaz de recoger la temperatura, humedad, presión, aceleración, luz, sonido, campos magnéticos, radiación, localización, componentes químicos, gases, etc. A medida que aumente la capacidad y velocidad de cómputo, se podría preprocesar los datos en los nodos, antes de enviarlos a un nodo central, permitiendo nuevos tipos de análisis y conclusiones. Finalmente, la disminución de las dimensiones físicas, gracias a la miniaturización de los componentes y los sensores, permitiría abarcar nuevos espacios o aplicarlos a seres vivos para aumentar su estudio sin alterar su calidad de vida y sus condiciones. Un listado de posibles aplicaciones se detalla a continuación:

- **Militar:** El rápido e inmediato despliegue de los sensores a través del aire permitiría la recopilación de datos de interés del terreno o del enemigo. Esto serviría para dar apoyo o complementar otra información disponible. Permitiría abarcar, desde la inteligencia militar, al reconocimiento o la vigilancia, y establecer objetivos o blancos.
- **Hogar Inteligente:** Permiten la recogida de datos para el control del hogar. Los sistemas de calefacción, aire acondicionado, ventilación, iluminación, seguridad y simulación de presencia necesitan una gran cantidad de variables de varios puntos de cada estancia para calcular la acción más correcta y la solución más eficiente desde el punto de vista energético.
- **Control Medioambiental e Incendios:** El despliegue de miles de nodos permite cubrir grandes extensiones de campo. En esas extensiones se puede controlar la humedad y la temperatura de las plantaciones, el estado y el comportamiento de la vida animal [12] o la detección de desastres naturales como conatos de incendios, terremotos, inundaciones o tsunamis.
- **Control y Monitorización de Entornos Industriales:** Permiten el control, realizar pruebas y la monitorización de la maquinaria industrial, abarcando toda la fábrica, las líneas de montaje, etc.
- **Seguridad:** La rapidez, su pequeño tamaño y sencillez en su despliegue, es idóneo para zonas de vigilancia y control. Los lugares de instalación incluyen, desde casas, oficinas, fábricas, lugares públicos, hospitales, a lugares restringidos como bases militares, aeropuertos, etc. Hoy día, es muy común encontrar sistemas de detección de intrusos de bajo costo e inalámbricos, en los que, por ejemplo, cada sensor piroeléctrico posee su propio microsistema que determina cuando está o no el sistema comprometido, alertando por radio de dicho evento.
- **Control de Tráfico:** Permite el control del tráfico, del ruido y de la contaminación para su estudio y tomar medidas correctoras. Las redes también podrían actuar en el control y el direccionamiento del tráfico con carteles inteligentes si se detectan retenciones o accidentes en la vía.

- **Salud:** El control de la medicación necesaria para un paciente, o la información sobre su estado de salud se podría ampliar con una toma de datos simultánea en varios puntos del cuerpo.

Todas estas complejidades técnicas y requisitos dificultan el desarrollo de las redes de sensores. Gracias a la innovación de los sensores inalámbricos y las tecnologías asociadas, se van aumentando las prestaciones y capacidades de los nodos, por lo que en un futuro próximo existirán nodos que soporten todas las características necesarias para que en conjunto puedan comportarse como una red de sensores totalmente operativa. A pesar de todas estas dificultades, en la actualidad existen muchas redes de sensores específicas para la recolección de datos en experimentos científicos, como por ejemplo, la monitorización de volcanes o fallas tectónicas en la superficie terrestre. Sin embargo, aún hay campos que están en pleno desarrollo y presentan potenciales beneficios económicos: desde control de invernaderos, a entornos industriales, identificación, y localización y seguimiento del estado de salud de animales y del estado de alimentos durante su transporte.

2.2. Tipos de Comunicaciones Inalámbricas

Las tecnologías inalámbricas han tenido un rápido desarrollo durante los últimos años. Su desarrollo y explotación ha sido espectacular gracias a la aparición incesante de nuevas aplicaciones. Las redes cableadas solamente permiten topologías fijas, con nula o baja movilidad y su modificación conlleva un gran sacrificio, tanto económico como técnico, frente a las inalámbricas, con total flexibilidad, fácil despliegue y bajo coste. Los tipos de tecnologías inalámbricas disponibles en la actualidad y en desarrollo van desde la transmisión infrarroja IrDA, a redes inalámbricas de área local WLAN, comunicaciones punto a punto, y de difusión y comunicaciones con satélites. Hay que tener en cuenta que cada una de estas tecnologías de comunicación está específicamente orientada a un tipo de aplicación (transferencia de datos a alta velocidad, alcance, bajo consumo, entre otros). Como se observa en la Figura 2.2, en un extremo se encuentran aplicaciones de grandes anchos de banda como redes de datos, de televisión o dispositivos móviles, mientras que en el otro existen aplicaciones que necesitan reducidas tasas

de transferencia, bajo coste y de mínimo consumo. En este ámbito se recogen las redes de sensores inalámbricas. En la Tabla 2.1 se detalla una clasificación de las tecnologías inalámbricas disponibles [7].

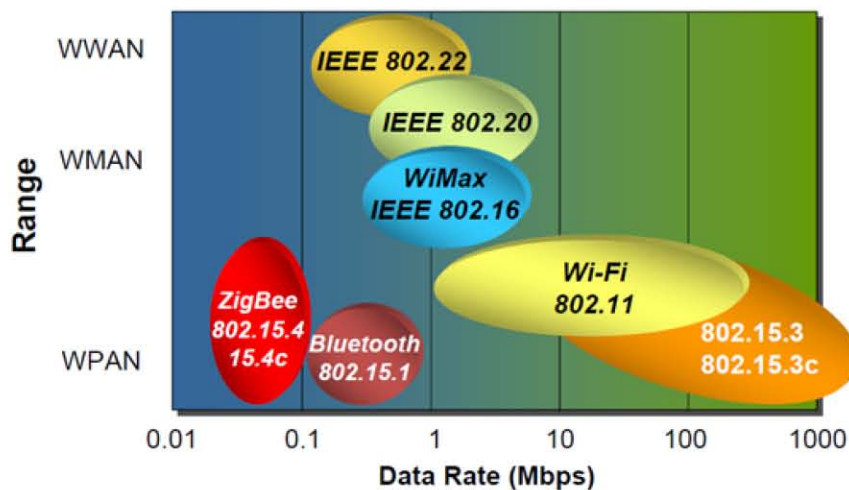


Figura 2.2: Comparativa de Tipos de Comunicación Inalámbrica

Tipo	Velocidad de Transferencia	Cobertura	Aplicación	Tecnologías
WWAN	< 10 Mbps	> 10 km	Telefonía, Internet móvil	GSM, UMTS, Satélites
WMAN	< 100 Mbps	< 10 km	Internet Banda Ancha	IEEE 802.16, HIPERMAN
WLAN	< 100 Mbps	< 100 m	Redes de Ordenadores	IEEE 802.11, HIPERLAN/2
WPAN	< 10 Mbps	< 10 m	Transferencia de datos personales	IEEE 802.15.3, Bluetooth
WSN	< 1 Mbps	< 1 km	Monitorización, control	IEEE 802.15.4, RFID, Propietario

Tabla 2.1: Resumen de Tipos de Comunicación Inalámbrica

2.2.1. Redes Inalámbricas para WSN

El *Institute of Electrical and Electronics Engineers* (IEEE) define dos estándares que son aptos para ser utilizados por las redes de sensores [13]. El IEEE 1451, que especifica una interfaz para sensores y actuadores inteligentes, y el IEEE 802.15 para redes inalámbricas de área personal. La primera se desarrolló a partir del *National Institute of Standards and Technology* en 1994 para comunicar sensores y actuadores de varios fabricantes entre sí. Define una capa de estandarización (TEDS) para controlar, identificar, calibrar, y guardar la información del sensor. La comunicación se realiza de manera cableada o inalámbrica, ya que el estándar no especifica la capa física (PHY) ni el control del acceso al medio (MAC), aunque se dispone de varias interfaces estandarizadas a tecnologías típicas como CAN, 802.11 [14], 801.15.1 y 802.15.4 [9].

El segundo estándar orientado a redes inalámbricas de área personal tiene varias vertientes importantes: Bluetooth (802.15.1), la de redes de alta velocidad WPAN (802.15.3) y la de tasa baja de transmisión de datos (LR-WPAN 802.15.4). Esta última es la que más interés y expectativa tiene para el desarrollo de las redes de sensores inalámbricos, ya que se desarrolló con la premisa de tener una baja complejidad, bajo coste y conectividad inalámbrica de baja potencia. El estándar IEEE 802.15.4 fue aprobado el verano de 2003, y su posterior implementación como especificación fue realizada por la Alianza ZigBee (*ZigBee Alliance*) en diciembre de 2004. El estándar 802.15.4 solamente define la capa física y la del control del acceso al medio, mientras que ZigBee define las capas superiores de encaminado y de aplicaciones para la interoperabilidad entre varios fabricantes.

En los siguientes apartados se detallan estas tecnologías factibles para las redes de sensores inalámbricos en mayor extensión.

Bluetooth

Permite la creación de enlaces de datos entre dispositivos fijos o móviles a distancias cortas, así como de pequeñas redes de área personal. Su intención inicial era el reemplazo del cableado serie en los dispositivos electrónicos. Publicaciones de revisiones posteriores incorporan características como Calidad de Servicio (QoS), mayores tasas de transferencia de datos, seguridad, y menores consumos. El estándar Bluetooth

define, desde la capa física, hasta los perfiles de servicio. La existencia de varios perfiles se debe a que es necesario cubrir una amplia variedad de aplicaciones, y hay que definir los mensajes, procedimientos y protocolos de comunicación para cada uno de ellas. Las redes se denominan *piconets* y se componen como máximo de un maestro y siete esclavos, pudiendo ser la comunicación punto a punto o punto a multipunto.

La banda utilizada es la banda ISM de 2,4 GHz. La banda de 2,4 GHz a 2,4835 GHz se divide en 79 canales con anchos de banda de 1MHz para cada uno. Usándose la técnica de espectro ensanchado por salto de frecuencia (FHSS), se permite cambiar hasta 1600 veces de canal para evitar interferencias y colisiones, además de aumentar la seguridad de la comunicación. La velocidad inicial fue de 1 Mbps, aunque las revisiones aprobadas permiten hasta 3 Mbps. El rango de cobertura depende de la potencia del transceptor, definiéndose las clases 1, 2 y 3. Las comunes son la 2 y la 3, con 2,5 mW (4 dBm) y 1 mW(0 dBm) de potencia, respectivamente, permitiendo rangos de cobertura típicos de 10 m y 1 m. La clase 1 trabaja con 100 mW de potencia (20 dBm) y el alcance es de 100 m [15].

Estándar IEEE 802.11

El estándar 802.11 [14] se publicó por primera vez en 1997 para la creación de redes inalámbricas. Su objetivo era sustituir las redes cableadas Ethernet (IEEE 802.3) para permitir la movilidad y simplificación de las instalaciones, y posibilita la interoperabilidad con las redes cableadas existentes. Lo que se define en este estándar son las capas física (PHY) y de acceso al medio (MAC). Ante la gran diferencia que existe entre un medio físico cableado y el aire, este estándar constituye una innovación, ya que este nuevo medio está sometido a un gran número de interferencias, colisiones en las transmisiones, rango de cobertura, movilidad de los dispositivos y gran variabilidad del medio. Se han propuesto muchas revisiones que han añadido políticas nuevas de seguridad (WEP, RC4, WPA, AES), mayor velocidad de transferencia (desde 11Mbit/s a 600 Mb/s), bandas de frecuencias (2,4 GHz, 5 GHz), mayor cobertura, etc.

La Alianza Wi-Fi (*Wi-Fi Alliance*) es una asociación sin ánimo de lucro que se creó para promocionar las redes inalámbricas y certificar los productos comerciales que cumplan con el estándar IEEE 802.11. Los productos certificados pueden llevar el logo Wi-Fi que certifica que es compatible con otros productos Wi-Fi, tiene el rendimiento

especificado, cumple las políticas de seguridad, y soporta interoperabilidad con las redes inalámbricas y cableadas Ethernet.

Estándar IEEE 802.15.4

Inicialmente publicado en 2003, y posteriormente revisado en 2006 [9] y en 2007 [10]. Este estándar es la base de las comunicaciones de baja velocidad para las redes inalámbricas de área personal. En él se definen el nivel físico (PHY) y el control de acceso al medio (MAC) que serán implementados en dispositivos de bajo coste y de bajo consumo, generalmente alimentados con baterías. Es por ello que los protocolos y procedimientos de asociación y formación de las redes son simples y flexibles. Se permite el uso de varias frecuencias y velocidades, además de varias topologías, así como la selección del canal más adecuado midiendo la energía existente.

Existen las bandas de frecuencias 868 MHz para Europa y Japón, con un solo canal disponible y una tasa de transferencia de 20 kbps. Para Estados Unidos la banda 915 MHz dispone de 10 canales y de tasas de 40 kbps. Finalmente, la banda mundial de 2,4 GHz dispone de 16 canales y tasas de 250 kbps. Los canales en este caso utilizan la técnica de espectro ensanchado por secuencia directa (DSSS), con modulación Offset-QPSK y se utiliza el protocolo de acceso múltiple por detección de portadora con evitación de colisiones (CSMA-CA). Los dispositivos se identifican con direcciones de 64 bits, aunque cuando forman parte de una red se permite utilizar direcciones de 16 bits para la comunicación. Se puede activar encriptación AES de 128 bits y el tamaño máximo de datos en la trama física es de 127 bytes.

Se admiten dos tipos de dispositivos, un dispositivo con funcionalidad completa (*Full-Function Device, FFD*) o un dispositivo con funcionalidad reducida (*Reduced-Function Device, RFD*). El primero de ellos puede operar con 3 modos distintos: como coordinador PAN (*Personal Area Network*), como coordinador, o como dispositivo final. El segundo solo puede ser un dispositivo final. Los FFDs pueden comunicarse con otros FFDs o con los RFDs, mientras que estos últimos solo pueden conectarse con un FFD. Esto es debido a que los RFD son dispositivos sencillos, que solamente tienen que transmitir pocos datos y se asocian a un único FFD. Por ello, se pueden implementar en plataformas con capacidades de memoria y de cálculo mínimas, y con alimentación limitada (baterías). Por el contrario, los FFD disponen de todas las funcionalidades de

la especificación y por lo tanto sus requerimientos de cálculo, memoria y de alimentación son mayores.

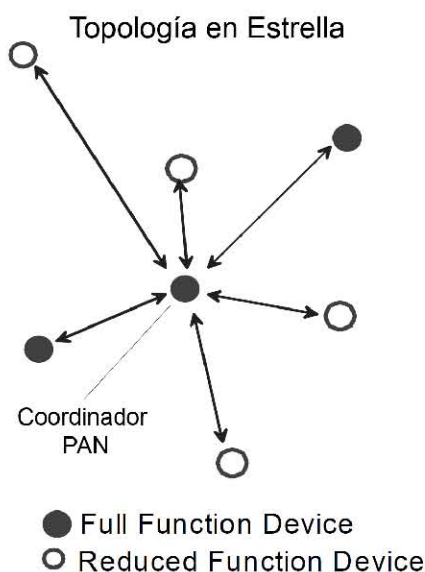


Figura 2.3: Topología en Estrella

Toda red WPAN tiene un FFD actuando como coordinador PAN, el cual inicia los parámetros de la red y generalmente actúa de pasarela a otras redes. Los coordinadores encaminan los datos hacia el coordinador PAN y realizan tareas de organización y reparación de la red. Finalmente, los dispositivos finales solamente pueden comunicarse con coordinadores o con el coordinador PAN, y no son capaces de encaminar información. Estas diferencias llevan a la existencia de dos tipos de topologías de red, de estrella o punto a punto (*peer-to-peer*). En la primera de ellas, existe el coordinador PAN y el resto de nodos serán dispositivos finales que se comunican directamente con él, como se puede observar en la Figura 2.3. Estos dispositivos finales pueden ser FFD o RFD, dependiendo de los recursos hardware del dispositivo. Este tipo de red generalmente se utiliza en redes pequeñas y con rangos de distancia reducidos, ya que los nodos tienen que ser capaces de encontrarse en la cobertura del coordinador PAN.

Para las redes punto a punto, está presente el coordinador PAN, varios coordinadores y varios dispositivos finales. Como se observa en la Figura 2.4, los RFD necesariamente tienen que comunicarse con el FFD al que están asociados, mientras que los FFDs pueden o no comunicarse entre ellos. Este tipo de red presenta numerosas ventajas, ya que la organización es totalmente libre y existen varias opciones de rutas

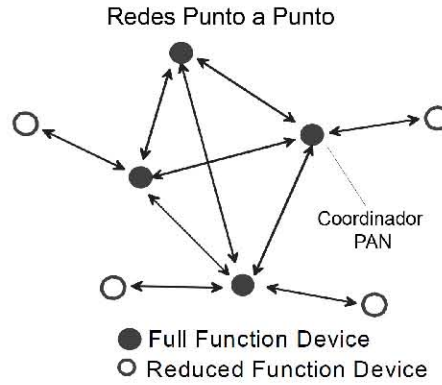


Figura 2.4: Redes Punto a Punto

para prevenir caídas o interrupciones en los coordinadores. Cabe destacar que el enca-
minamiento y el envío de mensajes a través de varios saltos no están definidos en este
estándar, y si se desea, deberán ser añadidos en capas superiores. Las redes punto a
punto se pueden diseñar para ser redes de árboles clúster (véase la Figura 2.5), con el
fin de aumentar el rango de cobertura y crear redes más complejas y mejor organizadas.

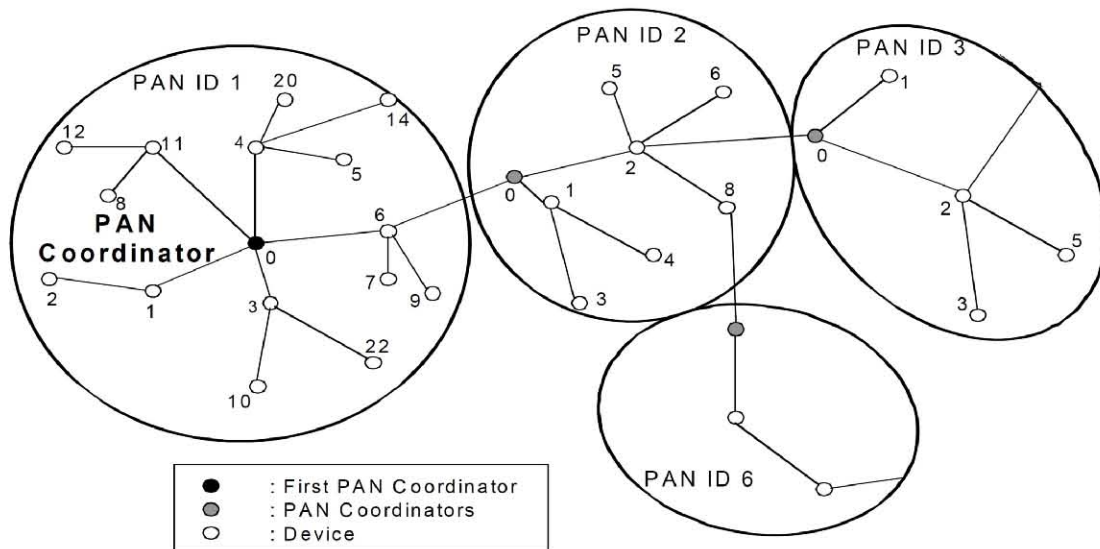


Figura 2.5: Clúster

Para establecer la comunicación, se puede operar en modo no-balizado (*NonBeacon*)
o balizado (*Beacon*). El primero de ellos (*NonBeacon*) se basa en CSMA-CA sin slots
de tiempo. Cuando el nodo tiene algo que transmitir, espera un tiempo aleatorio. Tras

este periodo de *backoff*, simplemente escucha si el canal está libre y, en tal caso, realiza la transmisión. Si el canal no estaba libre, esperará otro periodo aleatorio antes de reintentar la transmisión. En el modo de baliza (*Beacon*), existen unas supertramas con una baliza de *Inicio* que permiten la sincronización de los nodos, que es generada por el coordinador PAN implementando CSMA-CA con slots de tiempos definidos. De esta manera, si un nodo quiere transmitir, el tiempo de espera de *backoff* sólo está definido a un número discreto de slots disponibles. Por lo tanto, si comprueba que el canal está libre, empezará la transmisión justo en la frontera del siguiente slot. Para ambos casos, los reconocimientos (ACKs) son enviados sin usar CSMA-CA, al igual que en el caso de las balizas.

La comunicación entre coordinadores y dispositivos finales en cada sentido puede hacerse con diferentes opciones. Si se está en *NonBeacon*, simplemente después de seguir el procedimiento anteriormente explicado de espera aleatoria, se realiza la transmisión por parte del nodo como se observa en la Figura 2.6. Si se está en el modo *Beacon*, tras la señalización del coordinador, se realizará el envío de los datos por parte del dispositivo final como se muestra en la Figura 2.7. En ambos modos, si un dispositivo final requiere una confirmación por parte del coordinador, puede solicitarla opcionalmente.

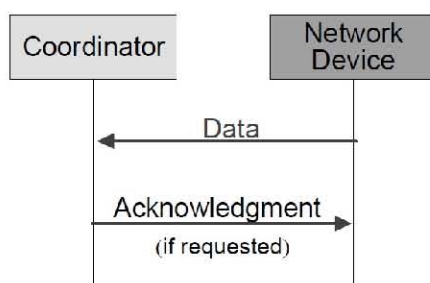


Figura 2.6: Comunicación con un Coordinador en una red *NonBeacon*

En el otro sentido de comunicación, si el coordinador tiene que enviar información a algún nodo, esto se indica en la baliza en el modo *Beacon*. En ese caso, el dispositivo solicita con un *Data Request* que le sea enviada la información. El coordinador entonces envía un ACK y a continuación la información. Si el dispositivo la recibe correctamente, tiene que hacérselo saber al coordinador mediante un ACK para validar la transferencia. Todo este proceso se puede observar de forma más clara en la Figura 2.8. Si se utiliza

el modo *NonBeacon*, el dispositivo tiene que estar siempre despierto para poder recibir de manera directa los datos, o si pasa a modo reposo, encuestar al coordinador si tiene información nueva para él. En este caso, la Figura 2.8 es válida eliminando la traza del *Beacon* generado por el coordinador.

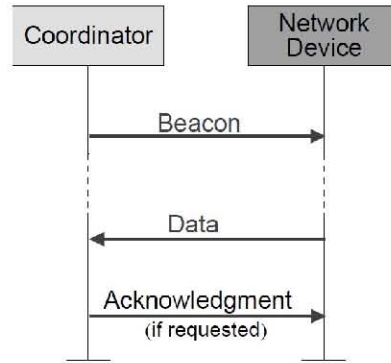


Figura 2.7: Comunicación con un Coordinador en una red *Beacon*

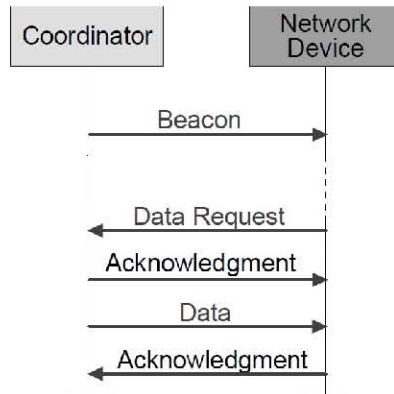


Figura 2.8: Petición de Datos al Coordinador en una red *Beacon*

Si se utiliza el modo con baliza, se utilizan las supertramas (véase la Figura 2.9). Éstas constan de 16 slots de tiempo, o parte activa, siendo el primero de ellos utilizado para las balizas. Las 15 restantes se pueden dividir en dos periodos, el primero de ellos siempre está presente y es el periodo de contención de acceso (*Contention Access Period, CAP*) y son slots ranurados para realizar las transmisiones. Un segundo periodo libre de contención (*Contention Free Period, CFP*) consta de uno o varios slot de tiempos garantizados (*Guaranteed Time Slots, GTS*) y se utiliza para garantizar la transmisión

de ciertos nodos. Finalmente, se puede definir un tiempo de reposo antes de volver a iniciar la supertrama con la siguiente baliza.

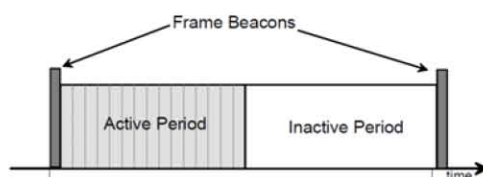


Figura 2.9: Supertrama

Como conclusión, en el caso del modo *NonBeacon*, los coordinadores siempre tienen que estar activos, ya que pueden recibir información en cualquier instante. En el modo *Beacon*, si se permite el tiempo de inactividad en la supertrama, los coordinadores pueden ponerse en reposo. En cambio, para los dispositivos finales, pueden trabajar en modo reposo y activarse solamente para recibir la baliza y si necesitan enviar/-recibir información, o en *NonBeacon* activarse cíclicamente para sincronizarse con el coordinador y recibir información de configuración.

6LowPAN

Es la implementación de IP6 sobre redes personales de bajo consumo. El primer estudio se publicó en agosto del 2007 en la RFC 4919. Posteriormente se publicó la RFC 4944 que define las capas superiores de encapsulado y paquetes para permitir el ruteado con direcciones IP. Este protocolo trabaja sobre la capa física y la capa MAC definidas por el estándar IEEE 802.15.4. Las capas superiores permiten adaptar paquetes IPv6 con longitud máxima 1280 bytes a los 127 bytes del estándar 802.15.4, y el direccionamiento se realiza sobre direcciones IP de 128 bits. El documento RFC 4944 define las direcciones y el formato de las tramas, pero no define el método de encaminamiento y la interacción entre nodos adyacentes. En julio de 2009 se ha publicado un documento borrador donde se especifican los problemas y requisitos para la formación de redes *mesh* y encaminado de paquetes sobre 6LowPAN.

ZigBee

Es una especificación creada por la alianza ZigBee (*ZigBee Alliance*) en diciembre de 2004 [16]. Se desarrolla para permitir aplicaciones de alto nivel sobre dispositivos que se basen en el estándar 802.15.4. Su principal cometido es promover características adicionales con capas superiores sobre la capa física y de acceso al medio que define el 802.15.4 para redes WPAN. Se desarrolló una revisión en 2006 [17] y la versión actual 2007 [18] identifica dos tipos de perfiles, uno normal y otro PRO.

ZigBee viene a cubrir las cuestiones que dejaba en el aire el estándar 802.15.4, tales como encaminamiento, seguridad, perfiles en la capa de aplicación, y certificación privada de los productos. Se mantienen los dos tipos de dispositivos, FFD y RFD, mientras que los dispositivos de la red cambian de nombre: el coordinador PAN pasa a denominarse *ZigBee Coordinator*, los coordinadores por *ZigBee Routers*, y los dispositivos finales como *ZigBee End-Devices*. Las tres topologías de red se siguen manteniendo: en estrella, punto a punto, y cluster-tree, además de los modos *Beacon* y *NonBeacon*. Por encima de la capa MAC existe la capa de red (NWK), que se encarga de la organización, la sincronización, el descubrimiento de los nodos, y el reenvío de mensajes. Las tablas de encaminamiento se calculan con el vector de distancias bajo demanda (*Ad hoc On-Demand Distance Vector, AODV*), por lo que en las redes *mesh* los *routers* pueden decidir qué ruta utilizar para enviar la información desde los *end-devices*, al coordinador. La seguridad se consigue con encriptación AES de 128 bits y la opción de listas de control de acceso.

Se observa que las premisas de bajo coste, ahorro de energía y bajas tasas de transferencia siguen vigentes, y se añade la capacidad de encaminamiento con múltiples saltos, la creación y organización de la red, y los perfiles de aplicación.

2.3. Hardware y Software

Las redes de sensores inalámbricos tienen amplios estudios teóricos respecto a su diseño, sus características principales, sus ventajas e inconvenientes, y otros aspectos, pero su implementación no es una tarea trivial y es por ello que numerosos grupos de investigación se han dedicado de lleno a su desarrollo y verificación. El primer grupo de

investigación, que inició la implementación de prototipos para formar una red de sensores, fue el Departamento de Ingeniería Eléctrica e Informática (*Electrical Engineering and Computer Science*) de la Universidad de California, Berkeley, en 1999. El Doctor David Culler ha sido el impulsor del desarrollo de las motas y del sistema operativo TinyOS (*Tiny Microthreading Operating System*) [19]. Los primeros prototipos de las plataformas Rene y Mica sentaron las bases para la definición de la arquitectura física, el futuro desarrollo de nuevas motas, y la búsqueda del perfeccionamiento y mejoras a aplicar a este primer diseño. El sistema operativo TinyOS es de código abierto y diseñado específicamente para las redes de sensores inalámbricos. Su particularidad es que al tener en cuenta el tamaño limitado de memoria y velocidad de las motas, utiliza un sistema operativo basado en eventos para poder gestionar las tareas necesarias de captura, procesamiento y almacenamiento en los sensores. TinyOS está escrito en una variante de C, *nesC*, diseñada específicamente para poder ejecutarse con las limitaciones de cálculo, memoria y energía de los sensores inalámbricos [20][21].

Desde entonces, la progresión del desarrollo e investigación de las redes de sensores ha ido creciendo exponencialmente, aunque las premisas siguen siendo las mismas desde hace una década. Los primeros prototipos partían de los módulos COTS y weC. Estos prototipos básicos y con pequeñas prestaciones, sentaron las bases para el desarrollo de los primeros prototipos de investigación funcionales. La plataforma se desarrolla [20] tomando como base el diseño hardware de la mota weC, pero modificándola para cumplir unos requisitos definidos de energía, potencia mínima de cálculo para soportar el TinyOS, un subsistema de comunicaciones por radio, un puerto de expansión y soportar alimentación autónoma. En la Figura 2.10 se muestran algunas de las motas empleadas en esta plataforma.



Figura 2.10: Plataforma Rene

Las características principales de la plataforma Rene son las siguientes:

- Microcontrolador de 8 bits ATMEL 90LS8535.
- Velocidad: 4MHz.
- 8KB de memoria Flash.
- 512 bytes de memoria RAM.
- 32KB memoria EEPROM externa.
- Radio TR1000 916.5 MHz AM On–Off keying a 10kpbs.
- Coprocesador para reprogramación ATMEL AT90LS2343.
- Sistema operativo: TinyOS.

Esta plataforma sirvió para probar en un sistema físico el sistema operativo TinyOS, desarrollado por Culler, para ver los problemas y las limitaciones en el diseño hardware de la plataforma Rene. Los informes generados a partir de estas pruebas, sirvieron para disponer de una plataforma totalmente funcional y operativa para la comunidad científica.

Las conclusiones de los informes anteriores llevaron al desarrollo de la plataforma Mica [21][22], que ha tenido un gran éxito mundial. En ella se han incluido aceleradores hardware para poder incrementar la velocidad de transmisión y aumentar la precisión de la sincronización temporal (la Figura 2.11 muestra la versión final de la monta implementada).

Sus características principales son las siguientes:

- Microcontrolador de 8 bits ATMEL ATMega 103L o ATMega 128.
- Velocidad: 4MHz con 4MIPS.
- 128KB de memoria Flash.
- 4 Kbytes de memoria RAM.
- 1 MB Atmel AT45DB041B EEPROM externa.

- Radio RFM TR1000 916.5 MHz AM On-Off keying a 10kpbs.
- Coprocesador para reprogramación ATMEL AT90LS2343.
- ROM para Número de Serie, Maxim DS2401.
- Maxim1678 DC-DC Boost pump converter para alimentación de 3.3V.
- Sistema operativo: TinyOS.
- 51 pines en el puerto de expansión (8 analógicas, 8 de potencia, I2C, SPI, puerto serie, programación).
- Dimensiones: 57mm x 32mm.



Figura 2.11: Plataforma Mica

La empresa Crossbow ha comercializado esta plataforma con el nombre de Mica [23], junto a varias placas de expansión con múltiples sensores para distintas aplicaciones.

A partir del gran éxito que tuvo Mica, varias empresas empezaron a desarrollar sus propias motas tomando como partida Mica, además de la propia Universidad de Berkeley. La primera mota desarrollada por una empresa privada fue 'Blue' (véase la Figura 2.12), creada por Dust Inc. Ésta mantuvo la arquitectura básica y el soporte de TinyOS,

pero modificó el microcontrolador a uno de 16 bits, el Texas Instruments MSP430. Este procesador está diseñado para aplicaciones de bajo consumo, por lo que incluye características tales como poder operar la UART o los conversores ADC mientras la CPU está en reposo, mayor resolución en los conversores ADC (12 bits) y un menor coste. El subsistema de comunicaciones también se cambió a FM con modulación FSK Chipcon CC1000, que tiene un mayor consumo pero permite una transmisión de mayor velocidad (76,8 Kbps) y más inmune a interferencias.



Figura 2.12: Mota Blue

A partir del gran éxito que tuvo Mica, varias empresas empezaron a desarrollar sus motas tomando ésta como partida. Por citar una empresa española, Libelium [24], después de 2 años de desarrollo, finaliza sus nodos Wasp mote (la Figura 2.13 presenta la versión final de la mota). Tienen una amplia gama de placas de sensores, y soporta comunicación externa con un módulo Bluetooth o ZigBee. Para su programación, dispone de un IDE donde se programa su Atmel con lenguaje C, a lo que hay que incluir la disponibilidad de librerías APIs para su mota. Su kit de desarrollo incluye 5 motas con el módulo 802.15.4, con un coste aproximado de 500€. Su características principales son las siguientes:

- Microcontrolador de 8 bits ATMEL ATmega 128l.
- Velocidad: 8MHz.

- 128KB de memoria Flash.
- 8 Kbytes de memoria RAM.
- 4KB de memoria EEPROM interna.
- Sistema operativo: Waspote.
- 7 entradas/salidas analógicas, 8 I/O digitales, I2C.
- Dimensiones: 73mm x 51mm x 13mm.



Figura 2.13: Plataforma Waspote

Microchip

Microchip [25], fue fundada en 1987 con el objetivo fabricar dispositivos programables, de gran versatilidad con módulos de entrada y salida. Sus primeros dispositivos se diseñaron usando instrucciones RISC de 8 bits, con un reloj de 20MHz y a un precio muy inferior a la competencia desde 1989. Hoy día, es uno de los mayores fabricantes de microcontroladores, de memorias y de dispositivos analógicos [26].

En el área de las comunicaciones inalámbricas para redes de sensores, dispone de distintas soluciones, aunque no ofrece una pila base de 802.15.4. Ofrece varias opciones dentro de Zigbee: como puede ser la pila estándar Zigbee-2006, la superior Zigbee-PRO, la RF4CE y un perfil de aplicación completo “*Smart Energy Profile*” [27].

Microchip también ofrece su protocolo propietario, que implementa la capa superior a la MAC para el encaminamiento de los mensajes, pero sin la complejidad de ZigBee. Ofrece la solución MiWi-Mesh y MiWi-P2P. La librería es de pequeño tamaño y compatible con la mayoría de sus microcontroladores de media y alta gama. Soporta 1024 nodos conectados a 8 coordinadores, con un máximo de 2 saltos hasta el coordinador PAN y 4 en la red. La profundidad máxima de los clusters será de 3 nodos [28][29].

Para implementar estas pilas, ofrece la solución hardware MRF24J40 [30]. El transceptor de radio, MRF24J40, trabaja en la frecuencia 2,4GHz, con un consumo de 23mA en transmisión y con interfaz SPI para la interconexión con los microcontroladores. Se suministra en un kit (véase la Figura 2.14) que incluye la antena PCB, los estabilizadores de tensión, los cristales, y el circuito de adaptación. Este kit hay que utilizarlo con alguno de los kits de desarrollo de la familia de microcontroladores PIC18, PIC24 ó PIC32 para poder usar las mencionadas pilas.

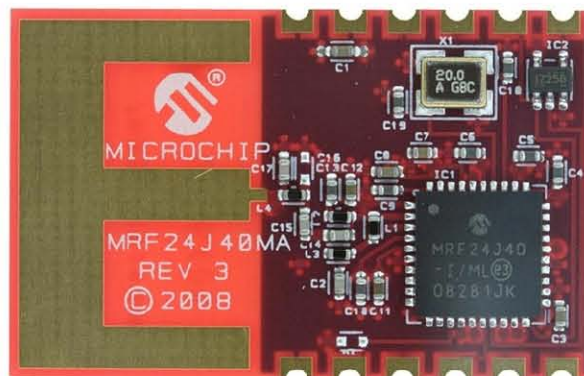


Figura 2.14: Microchip MRF24J40MA

Atmel

La compañía Atmel [11] fue fundada en 1984. Dispone de microprocesadores basados en 8051 y en ARM, aunque también dispone de su arquitectura propia AVR creada por los estudiantes noruegos Alf-Egil Bogen y Vegard Wollan [31]. Atmel dispone de una gran variedad de dispositivos de 8 bits, como el ATMega103 ó el ATMega128. Estos dos son los más utilizados por las motas desarrolladas para investigación, ya que soportan el sistema operativo TinyOS.

Atmel ofrece también transceptores de radio con soporte 802.15.4, como los AT86RF212, AT86RF230 y AT86RF231, que pueden conectarse a una gran cantidad de microcontroladores mediante SPI. Un producto innovador es el microcontrolador con radio integrada ATmega128RFA1. Éste se compone del microcontrolador 1281 y de la radio AT86RF231, todo integrado en el mismo encapsulado.

Los kits de desarrollo disponibles son múltiples, dependiendo de la aplicación final y de la pila soportada. Para el estándar IEEE 802.15.4 se dispone de la pila Atmel-MAC, que incluye las declaraciones de las funciones del estándar, ejemplos sencillos y varia documentación [32]. Para ZigBee, se ofrece la pila privada de BitCloud [33], RF4CE y varios perfiles de aplicación (*Building Automation, Home Automation, Smart Energy*). Existe también la opción RUM (*Route Under MAC*) con soporte IPv6 (6LoW-PAN) [11].

Todas estas pilas pueden implementarse en el kit RZRAVEN [34], específico para el desarrollo, depurado y demostración de comunicaciones inalámbricas de baja potencia. Este kit se compone de dos placas AVRRAVEN y de un USBStick para el PC. Existe también una plataforma de desarrollo más compleja, la STK600 (se muestra en la Figura 2.15), con zócalos de expansión y totalmente modular. Por ello, hay que montar una placa con algún microcontrolador compatible y un kit de comunicaciones RZ600 con los transceptores AT86RF230.



Figura 2.15: Atmel STK600

Conclusiones

Concluido el análisis del estado del arte, se decidió utilizar los kits RZRAVEN de Atmel. Las otras plataformas disponibles utilizan lenguajes de programación especiales o no son soluciones completas. Con el RZRAVEN, se puede empezar inmediatamente a estudiar y comprender la base de las redes de sensores, su funcionamiento e implementarlas de manera práctica. De las pilas Atmel disponibles y compatibles con el kit, se elige la Atmel MAC 802.15.4. Este acercamiento permite empezar a investigar de manera completa el estándar IEEE 802.15.4, que sirve de base para otros estándares más avanzados. Elegir ZigBee BitCloud dificultaría el estudio y comprensión, ya que añade complejidad debido a que cubre otras tareas como encaminar mensajes, políticas de seguridad, etc. El incremento de funcionalidad va unido a una mayor complejidad y tamaño del código, como se muestra en la Figura 2.16.



Figura 2.16: Funcionalidad

Capítulo 3

Estudio de la Solución Adoptada

3.1. RZRAVEN

El kit RZRAVEN consta de dos tarjetas AVRRAVEN y un dispositivo USB RZUSBS-TICK, representado en la Figura 3.1.

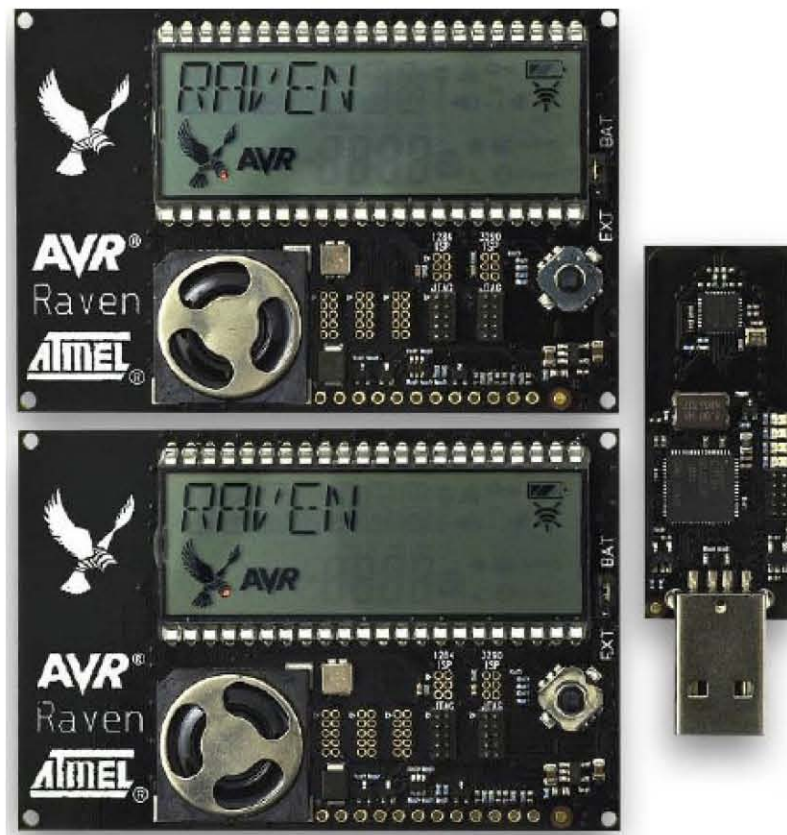


Figura 3.1: RZRAVEN

Como se muestra en la Figura 3.2, las placas AVRRAVEN contienen los siguientes elementos [34]:

- **Transceptor de radio AT86RF230.** Trabaja en la frecuencia de 2,4 GHz, con una potencia de transmisión de +3 dBm, una sensibilidad de -101 dBm y tiene además un bajo consumo (16 mA). La antena tiene una ganancia de 5dB con una impedancia de 100Ω.
- **Microcontrolador ATmega1284P.** Su arquitectura es de tipo Harvard y es el encargado de controlar el chip de radio. Sus características principales son las siguientes:
 - Es un microcontrolador de 8 bits.
 - Velocidad: 20MHz que se corresponden con 20 MIPS.
 - 128KB de memoria Flash para el código del programa.
 - 16KB de memoria SRAM para datos.
 - 4KB memoria EEPROM para almacenar información.
 - Alimentación desde 1,8V a 5,5V.
 - Dispone de un módulo hardware multiplicador integrado.
 - Reloj en tiempo real (RTC) de 32KHz.
 - Timers de 8 y 16 bits.
 - 6 canales para PWM.
 - 10 conversores A-D.
 - Un comparador analógico.
 - Posee varios protocolos de comunicación: SPI, I^2C , USART.
- **Microcontrolador ATmega3290P.** Maneja el display LCD, las E/S y el joystick. Es similar al ATmega1284P, pero con menos memoria. Su particularidad es que tiene una salida dedicada para control de displays LCD.

Adicionalmente, se disponen de varios puertos de expansión (*Headers*), aunque tienen una separación de 50-mil ó 1.27mm. Mediante ellos, se realiza la programación de

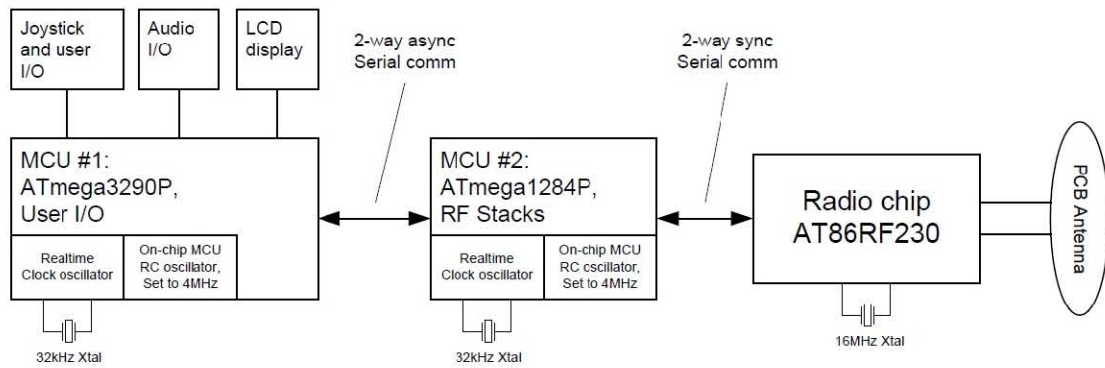


Figura 3.2: Diagrama de Bloques de la tarjetas AVRRAVEN

ambos microcontroladores mediante JTAG o con un programador ISP, o se puede acceder a los puertos analógicos o digitales de los microcontroladores. Para la alimentación, se dispone de la posibilidad de alimentar con dos pilas de 1.5V LR44, o a través del conector *EXT_PWR*, que admite tensiones de 3,6V a 12V, ya que se utiliza el regulador LDO LP2985A-33DBV de 3,3V.

Por otro lado, como se muestra en la Figura 3.3, el dispositivo RZUSBSTICK consta del microcontrolador AT90USB1287 y de la radio AT86RF230. Se comunican de nuevo por la interfaz SPI. Este microcontrolador tiene integrado el bus USB, para poder conectarlo a un PC. La antena está integrada en la propia PCB, siendo de tipo dipolo doblada y con una ganancia de 0dB. De nuevo, dispone de un puerto JTAG para su programación y de 4 LEDs de estado.

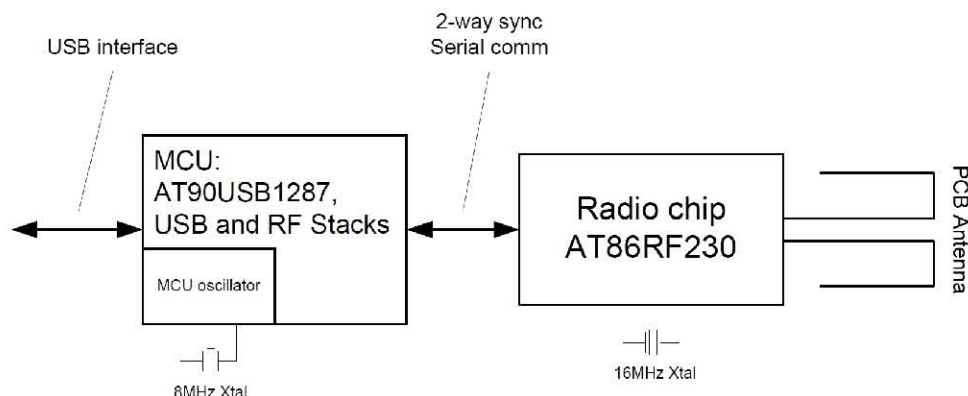


Figura 3.3: Diagrama de Bloques del dispositivo RZUSBSTICK

3.2. IEEE 802.15.4

El estándar IEEE 802.15.4 se desarrolló en 2003 para cumplir los requisitos y las funcionalidades requeridas por las redes de sensores: baja transferencia de datos, capacidad de sincronización y comunicación de los nodos, manteniendo consumos mínimos. Fue sometido a una importante revisión en 2006 [9]. El estándar, de 320 páginas, especifica las topologías válidas, la capa física y la capa de enlace, además de información adicional de seguridad, vectores de test y coexistencia con otros estándares.

Este TFM se centra en las reglas básicas de creación, asociación y comunicación en las redes de sensores, y en las especificaciones de la subcapa MAC. El estándar define los métodos y pasos para la gestión de las redes, además de las primitivas que se tienen que implementar en los nodos para gestionar la capa física, la subcapa MAC, y el paso de mensajes a la capa de aplicación.

Las primitivas permiten realizar acciones e interacciones entre dos capas. Los servicios de una capa son las capacidades que se ofrecen al usuario de la capa superior mediante la declaración de funciones en los servicios de la capa inferior. Este concepto se ilustra en la Figura 3.4, mostrando las primitivas existentes y la relación entre el Usuario- N (N -User) y la Capa- N asociada (N -layer) [9].

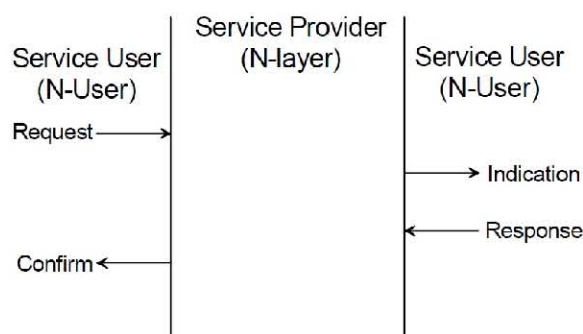


Figura 3.4: Primitivas entre Capas

Los servicios se especifican describiendo el flujo de información entre el N -User y la N -Layer. Este flujo de información se modela por eventos que caracterizan la prestación de un servicio. Cada evento consiste en hacer pasar una primitiva de una capa a la otra a través de una capa SAP asociada al N -User. Un servicio se especifica mediante la descripción de las primitivas y los parámetros que la caracterizan. Cada

primitiva puede tener cero o más parámetros que transmiten la información necesaria para la prestación del servicio.

Una primitiva puede ser de uno de los siguientes cuatro tipos:

- **Petición (*Request*)**: La primitiva de petición se pasa del *N-User* a la *N-Layer* para solicitar el inicio de un servicio.
- **Indicación (*Indication*)**: Esta primitiva se pasa de la *N-Layer* al *N-User* para indicar un evento ocurrido en la *N-Layer* que puede ser de interés para el *N-User*.
- **Respuesta (*Response*)**: La primitiva de respuesta se pasa del *N-User* a la *N-Layer* para completar un método previamente invocado por una primitiva de tipo ‘Indication’.
- **Confirmación (*Confirm*)**: La primitiva de confirmación se pasa de la *N-Layer* al *N-User* para comunicar los resultados de una primitiva de tipo ‘Request’.

En el capítulo 7 del estándar IEEE 802.15.4 se especifica la subcapa MAC [9]. Después de una lectura detallada, se van detallar únicamente los puntos claves de este capítulo. La subcapa MAC maneja todo el acceso al canal de radio físico y es responsable de las siguientes tareas:

- La generación de balizas (*Beacons*) si el dispositivo es un coordinador.
- Sincronización de las balizas de la red.
- Control de la asociación y disociación.
- Seguridad del dispositivo.
- Utilizar el mecanismo CSMA-CA para el canal de acceso.
- Manejo del mecanismo de GTS.
- Proporcionar un vínculo fiable entre dos entidades MAC.

Un dato muy importante es el uso de constantes y atributos en las primitivas. Éstos están completamente definidos en el estándar. Además, se caracterizan por que las constantes tienen el prefijo ‘a’, mientras que los atributos usan el prefijo ‘mac’.

La subcapa MAC provee una interfaz entre la capa de aplicación de usuario (SSCS) y la capa física (PHY). La subcapa MAC conceptualmente incluye una entidad de gestión llamada MLME (*MAC subLayer Management Entity*). Esta entidad proporciona las interfaces de servicio a través de las cuales pueden invocarse las funciones de la capa de gestión. La entidad MLME también es responsable de mantener una base de datos de los objetos gestionados que pertenezcan a la subcapa MAC, conocida como MAC-PIB (*MAC-PAN Information Base*).

La subcapa MAC proporciona dos servicios, accesibles a través de dos SAP (*Service Access Point*):

- El servicios de datos, accesible a través de la subcapa *MAC Common Part Sublayer data SAP* (MCPS-SAP)
- El servicio de gestión, a través de *MLME-SAP*.

Estos dos servicios proporcionan la interfaz entre la capa de aplicación del usuario y la capa física (los servicios *PD-SAP* y *PLME-SAP* pertenecen a la PHY). Estas relaciones, componentes y servicios se pueden observar en la Figura 3.5.

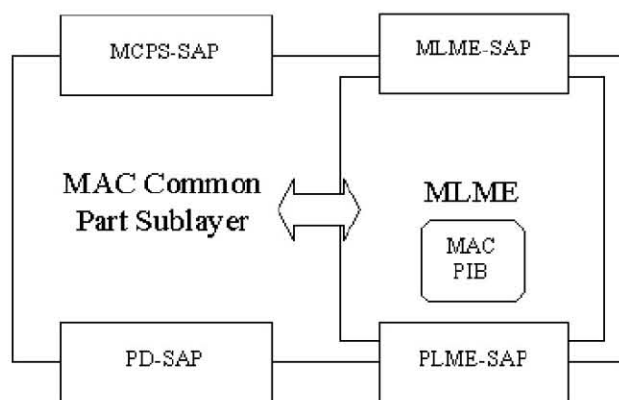


Figura 3.5: Modelo de la subcapa MAC

3.2.1. Llamadas a la Capa de Datos (MCPS)

El primer servicio, el relativo a los datos (MCPS), tiene dos entidades. La primera de ellas, *MCPS-DATA*, es relativa a eventos con los datos, mientras que la segunda,

MCPS-PURGE, se utiliza para purgar o eliminar tramas pendientes en la cola. Solamente se explicará en detalle la primera de ellas debido a que la segunda sólo se utiliza en casos muy puntuales.

MCPS-DATA

La entidad de datos sirve para transmitir datos y tiene asociadas tres primitivas: *Request*, *Confirm* e *Indication*. La primera, *MCPS-DATA.request*, inicia la transmisión, provocando la llamada a la segunda, *MCPS-DATA.indication*, en el receptor. En el coordinador, se informa del resultado de la transferencias de datos mediante *MCPS-DATA.confirm*.

La primitiva *MCPS-DATA.request* se utiliza para solicitar la transferencia de datos MSDU (*MAC Service Data Unit*). Entre los parámetros que son requeridos, uno de ellos (*SrcAddrMode*) sirve para definir el tamaño de las direcciones (direcciones cortas de 16 bits o direcciones extendidas de 64 bits). A continuación, son necesarias la dirección de destino y el Identificador PAN de la red de destino (*DstPANId*). Los datos (*msdu*) tendrán un tamaño definido (*msduLength*), un identificador *msduHandle*, un nivel de seguridad, y 3 bits de *TxOptions* en la que se puede decidir si se quiere confirmar la recepción del mensaje en destino con un ACK, y realizar transmisiones directas o indirectas. La declaración completa de esta primitiva se puede observar en el Listado 3.1.

```
MCPS-DATA.request (
    SrcAddrMode ,
    DstAddrMode ,
    DstPANId ,
    DstAddr ,
    msduLength ,
    msdu ,
    msduHandle ,
    TxOptions ,
    SecurityLevel ,
    KeyIdMode ,
    KeySource ,
    KeyIndex
)
```

Listado 3.1: Pritimiva *MCPS-DATA.request*

El estándar explica todos los casos posibles asociados a esta primitiva. Por ejemplo, en primer lugar se comprueba si la dirección tiene formato correcto. Si están fijadas a 0x00, se invoca a la primitiva de confirmación *MCPS-DATA.confirm* con el estado de dirección errónea (*INVALID_ADDRESS*). La capa continúa con una serie de comprobaciones, tales como las opciones, el modo de envío, etc. Si hay algún error, se genera la primitiva *MCPS-DATA.confirm* con un estado que corresponda al fallo, y solamente en el caso de enviar correctamente los datos, el estado será el de ‘SUCCESS’.

Por lo tanto, la primitiva *MCPS-DATA.confirm* informa al solicitante del *Data.Request*, del desenlace y estado final de su petición de envío de datos. Los parámetros incluyen el identificador *msduHandle*, el estado, y un *Timestamp*. Finalmente, se produce una primitiva *MCPS-DATA.indication* para indicar que se ha recibido un dato y éste tiene que ser transferido de la subcapa MAC a la de aplicación.

Todo los eventos generados, y sus respectivas primitivas, se pueden observar en la Figura 3.6.

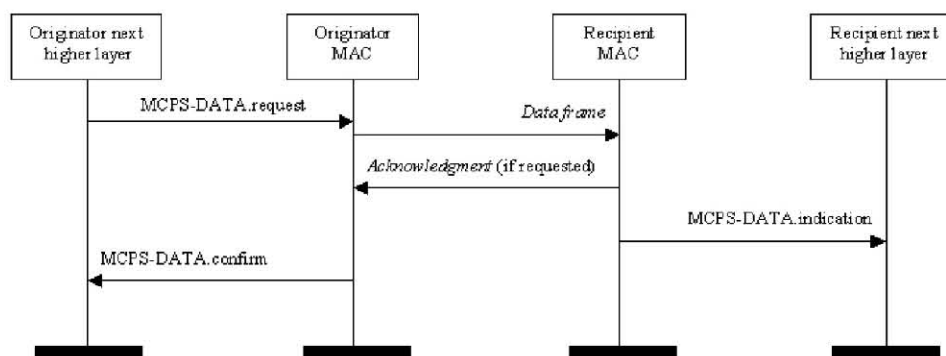


Figura 3.6: Primitivas *MCPS-DATA*

3.2.2. Llamadas a la Capa de Gestión (MLME)

El segundo servicio se corresponde con la capa de gestión MAC, MLME. El número de entidades es mucho mayor, como se puede observar en la Tabla 3.1, ya que hay que controlar la asociación y disociación de los nodos, modificar atributos de la base de datos PIB, iniciar la red, escanear los canales, y varios eventos adicionales que están totalmente definidos en el estándar. En este TFM se revisan sólo las más relevantes y que son las mínimas necesarias para la existencia de una red de sensores.

Name	Request	Indication	Response	Confirm
MLME-ASSOCIATE	7.1.3.1	7.1.3.2	7.1.3.3	7.1.3.4
MLME-DISASSOCIATE	7.1.4.1	7.1.4.2		7.1.4.3
MLME-BEACON-NOTIFY		7.1.5.1		
MLME-GET	7.1.6.1			7.1.6.2
MLME-GTS	7.1.7.1	7.1.7.3		7.1.7.2
MLME-ORPHAN		7.1.8.1	7.1.8.2	
MLME-RESET	7.1.9.1			7.1.9.2
MLME-RX-ENABLE	7.1.10.1			7.1.10.2
MLME-SCAN	7.1.11.1			7.1.11.2
MLME-COMM-STATUS		7.1.12.1		
MLME-SET	7.1.13.1			7.1.13.2
MLME-START	7.1.14.1			7.1.14.2
MLME-SYNC	7.1.15.1			
MLME-SYNC-LOSS		7.1.15.2		
MLME-POLL	7.1.16.1			7.1.16.2

Tabla 3.1: Primitivas *MLME-SAP*

MLME-ASSOCIATE

La primitiva *MLME-ASSOCIATE.request* se utiliza en un nodo con una dirección extendida fijada, para solicitar la asociación a un coordinador deseado con una determinada dirección, *PAN_ID*, y canal. Al recibir el coordinador una solicitud de asociación, se invoca la primitiva *MLME-ASSOCIATE.indication* para que la petición pueda ser tratada. En la función asociada, el coordinador deberá comprobar si tiene en su Tabla de direcciones, alguna dirección libre sin asignar. En caso afirmativo, tendrá que llamar a la primitiva *MLME-ASSOCIATE.response* para poder enviar al nodo la dirección asignada. En el nodo, la primitiva *MLME-ASSOCIATE.confirm* se ejecutará para validar la asociación y poder continuar con el proceso de creación de la red. De esta manera, se finaliza la asociación. El procedimiento de disociación es muy parecido, ya que es necesario llevar a cabo un intercambio de mensajes para desconectar el nodo y liberar su dirección. Todo los eventos generados en la asociación y sus respectivas primitivas se pueden observar en la Figura 3.7.

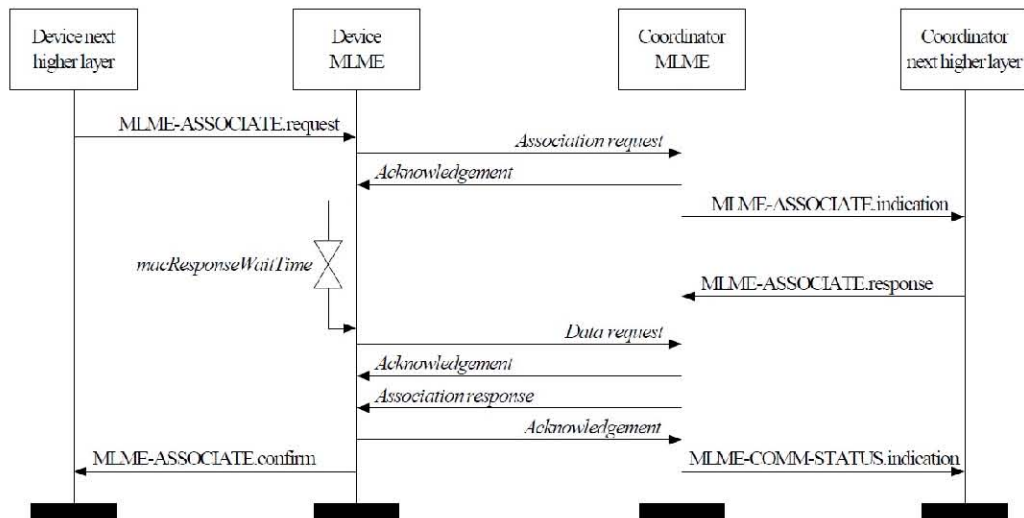


Figura 3.7: Mensajes y Primitivas para la Asociación

MLME-RESET

Otra primitiva imprescindible es *MLME-RESET.request*, con su único parámetro fijado a 'TRUE'. Es la primera que debe ser llamada, ya que se encarga de inicializar todos los atributos a su valor por defecto en el estándar. Si se realiza correctamente, se invoca a *MLME-RESET.confirm* con el estado de 'SUCCESS', y el usuario ya puede iniciar la creación de la red.

MLME-SCAN

Uno de los pasos necesarios antes de iniciar una determinada red, es escanear el medio para evaluar la energía y la ocupación de los canales. Para ello, se utiliza la primitiva *MLME-SCAN.request*, indicando en sus parámetros el tipo de escaneo, los canales a analizar, y la duración, entre otros. Los resultados o errores serán comunicados mediante la correspondiente primitiva de confirmación *MLME-SCAN.confirm*.

MLME-GET y MLME-SET

Las primitivas *MLME-GET.request* y *MLME-SET.request* son las que permiten leer o escribir los atributos de la PIB, respectivamente. En la primera de ellas, hay que indicar el atributo PIB y su correspondiente índice, que se pueden obtener de la tabla definida en el estándar. Si el atributo está definido y existe en la base de

datos, se generará la primitiva *confirm* con el valor correspondiente. En caso de error, se genera la confirmación, pero su variable de estado indicará la constante asociada al fallo. Para escribir mediante la primitiva *MLME-SET.request*, hay que indicar el atributo, su índice y el valor nuevo. Si la escritura se realiza de manera correcta, se ejecuta la primitiva *MLME-SET.confirm* con el estado 'SUCCESS'. Si el atributo era de "Sólo Lectura", o si no existe en la base de datos, se informa mediante las constantes 'READ_ONLY' o 'INVALID_PARAMETER', respectivamente.

MLME-START

El comando que permite iniciar una nueva red PAN es *MLME-START.request*. Además del identificador PAN, es necesario una serie de parámetros para definir si es modo *Beacon* o *NonBeacon*, el canal, la seguridad, etc. Si se crea correctamente la PAN, se confirma con una primitiva *MLME-START.confirm* con la variable *status* a 'SUCCESS'. El intercambio de mensajes se observa en la Figura 3.8.

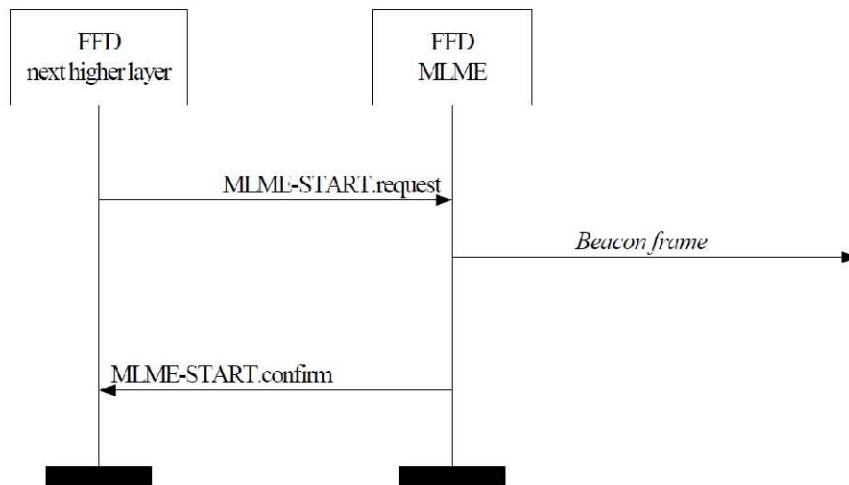


Figura 3.8: Secuencia de Mensajes *MLME-START*

MLME-POLL

La primitiva *MLME-POLL.request* debe ser llamada por un nodo para lanzar una pregunta al coordinador, por si tiene datos suyos almacenados. Si no existen, el coordinador enviará un ACK con el campo *FP* (*Frame Pending*) a '0'. En caso afirmativo,

el coordinador enviará un ACK con el campo *FP* a '1', indicando que sí existen datos, y seguidamente los envía. Esta secuencia se recibe en el nodo mediante una primitiva *MLME-POLL.confirm* con 'SUCCESS', y una directiva *MLME-DATA.indication*, para que los datos sean procesados. Este intercambio de mensajes se puede observa en la Figura 3.9.

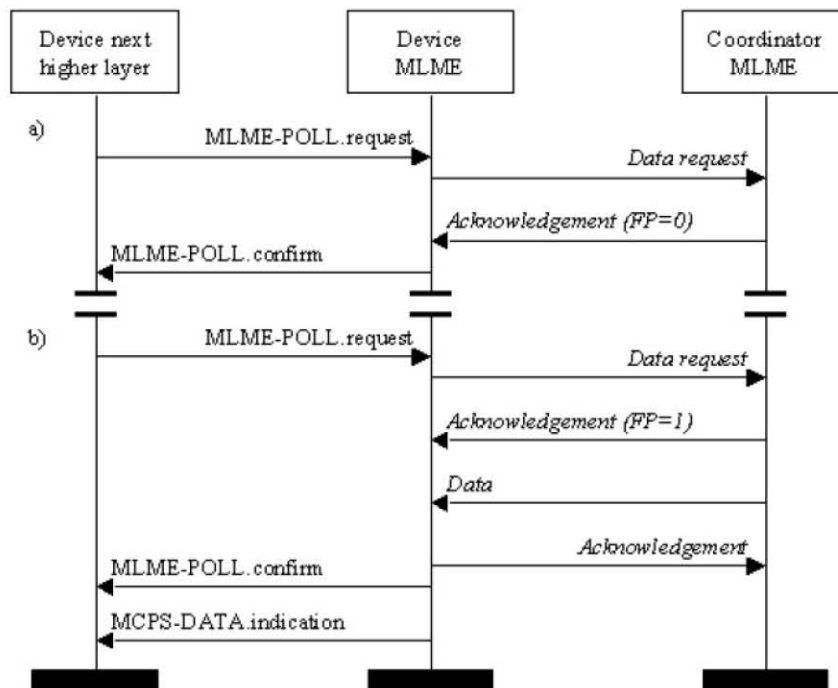


Figura 3.9: Secuencia de Mensajes *MLME-POLL*

3.2.3. Descripción de Tareas

Después de tener especificadas las capas y sus correspondientes directivas, el estándar hace una descripción funcional de la subcapa MAC. Primero explica el acceso al medio mediante el algoritmo CSMA-CA y el uso de los tiempos GTS. A continuación, describe diferentes tareas para poder crear correctamente una red basada en 802.15.4. Hay que resaltar que no se especifica un algoritmo o secuencia de pasos obligatorios a seguir. Lo que hace es explicar diferentes estados o acciones posibles que se pueden realizar, y las dependencias que existen entre ellas.

Creación de la red PAN

El primer grupo de tareas serían aquellas para que permiten empezar y mantener una red PAN. Para ello, el primer paso es realizar un *MLME-RESET.request* para inicializar todos los atributos correctamente. Si el nodo es un coordinador, se explica que a continuación puede realizar un escáner de detección de energía (*ED scan*) para detectar los picos de energía en los diferentes canales de frecuencia. Esto se realiza con la primitiva *MLME-SCAN.request*. Una vez finalizado el escaneo, se invocará la primitiva *MLME-SCAN.confirm* con los resultados, y pudiéndose entonces escoger el canal menos ocupado, o utilizar uno ya predefinido. El siguiente paso es común a los coordinadores y a los nodos, ya que ambos deberán realizar un escáner activo (o pasivo con otras condiciones) para poder localizar coordinadores PAN, que se irán almacenando en la lista de *MLME-SCAN.confirm*. A continuación, si el dispositivo es un coordinador, debe comprobar que la red que quiere inicializar no tenga un *PAN_ID* igual a alguna ya existente, o se tendrá que seguir el método de resolución de conflicto explicado en el estándar. Una vez se tenga escogido el *PAN_ID*, el coordinador debe llamar a la primitiva *MLME-START.request* con el parámetro *PANCoordinator = TRUE*. Al ejecutarse la correspondiente función de retorno (*callback*) *MLME-START.confirm*, y si su variable *status* es igual a 'SUCCESS', estará operativa la red PAN.

Asociación

El estándar especifica que sólo tras haber realizado un *reset* y tras haber realizado un escáner activo, es cuando se puede iniciar la asociación. Para ello, hay que utilizar la primitiva *MLME-ASSOCIATE.request* con uno de los coordinadores PAN encontrados, con su correspondiente canal, *PAN_ID* y la dirección corta del coordinador. Hay que tener en cuenta el parámetro *macAssociationPermit* del coordinador debe estar a *TRUE*, y que es necesario un tiempo para que el coordinador pueda consultar su tabla de direcciones de nodos para poder asignar una dirección al nodo. Si el coordinador no responde antes del tiempo de espera *macResponseWaitTime*, se invoca la primitiva *MLME-ASSOCIATE.confirm* e indica error. En caso contrario, se recibe la dirección corta y debe almacenarla en la variable *macShortAddress*.

Transmisión y Recepción

El grupo de tareas más frecuente será el de transmisión y recepción de mensajes. Los valores más importantes para la directiva *MCPS-DATA.request* son la dirección origen, la de destino, el *PAN_ID*, y el mensaje. Otro requisito adicional es el estado de la radio, activando el envío con 'TX_ON' en la capa PHY, y la recepción con 'RX_ON'. Como en todo sistema de comunicaciones, la recepción generalmente es la parte más compleja. En la subcapa MAC se realizan una serie de niveles de filtrado hasta que se acepte la trama. El primer nivel se encarga de descartar aquellas tramas que tengan un valor incorrecto de FCS. El segundo nivel se encarga de aceptar sólo aquellas tramas que cumplan una serie de requisitos:

- El tipo de trama no podrá ser uno del tipo reservado.
- El campo *Frame Version* no podrá tener un valor reservado.
- Si el identificador PAN de destino está incluido en la trama, deberá coincidir con el *macPANId* o corresponder al valor (0xFFFF) asociado al identificador PAN de *broadcast*.
- Si una dirección corta de destino está incluida en la trama, deberá coincidir con la *macShortAddress* o la dirección de *broadcast* (0xFFFF). De lo contrario, si es extendida deberá coincidir con la *aExtendedAddress*.
- Si es una trama baliza (*Beacon*), el identificador PAN deberá coincidir con *macPANId* o ser igual a 0xFFFF.
- Si sólo existe la dirección de origen, la trama sólo se aceptará si el dispositivo es el coordinador PAN y el *PAN_ID* de origen coincide con *macPANId*.

Si no se cumple alguno de estos requisitos, la trama se descarta. Si en cambio, se cumplen todos, la trama es aceptada por el dispositivo a falta de comprobar los parámetros de seguridad. Una vez superado el control de seguridad, se activa la primitiva *MCPS-DATA.indication*, informando al dispositivo del mensaje recibido y pendiente de procesar por la capa superior. Este segundo nivel de filtrado no se tiene en cuenta si se activa el modo promiscuo (*macPromiscuousMode*). En este último caso, todas las

tramas se aceptan con sólo superar el primer filtrado. Este modo se explicará posteriormente.

Confirmaciones ACKs

Otro tema que queda pendiente son los *Acknowledgments* o ACKs. El estándar obliga a que todas las tramas de tipo baliza, los propios ACKs, o tramas con direcciones de *broadcast*, deberán tener el campo de *Acknowledgement Request* desactivado ('0'). En cambio, en las tramas con datos o comandos, el usuario podrá decidir si desea activar (*Acknowledgement Request* = '1') la transmisión de confirmaciones por parte del receptor para poder asegurar que los mensajes enviados llegan a su destino. El ACK contendrá el mismo DSN de la trama recibida. En caso de que estén activadas las confirmaciones y de no se recibe el ACK, transcurrido un tiempo de espera definido en *macAckWaitDuration*, se deberá retransmitir la trama enviada. Esto es debido a que, probablemente, ha existido un problema en la transmisión hacia el destino. El número de retransmisiones a realizar por parte del transmisor se fija con el atributo *macMaxFrameRetries*, con un valor predeterminado de '3'. Estas retransmisiones son automáticas por parte de la subcapa MAC, y en caso de no realizarse correctamente, se indica con un error. Este procedimiento de retransmisiones no se realiza si el transmisor es un coordinador y se está operando en modo indirecto. En este caso, la trama se queda almacenada en la cola y el nodo solicitante deberá resolicitar la trama con otro *MLME-POLL.request*.

Modo Promiscuo

El estándar explica otros grupos de tareas relativos a las supertramas, tiempos de slots GTS, etc. El último que se va a detallar es el modo *promiscuo*. Si se activa con el atributo *macPromiscuousMode*, la subcapa MAC aceptará todas las tramas recibidas, como se explicó anteriormente. El requisito que impone 802.15.4 es que se active la recepción en el nivel PHY. Este modo es útil para estudiar y analizar todo el tráfico de tramas que existe en un determinado canal.

Finalmente, en la Figura 3.10 se incluye un diagrama con varias de las primitivas y tareas comentadas anteriormente para iniciar la creación de una red PAN. También se

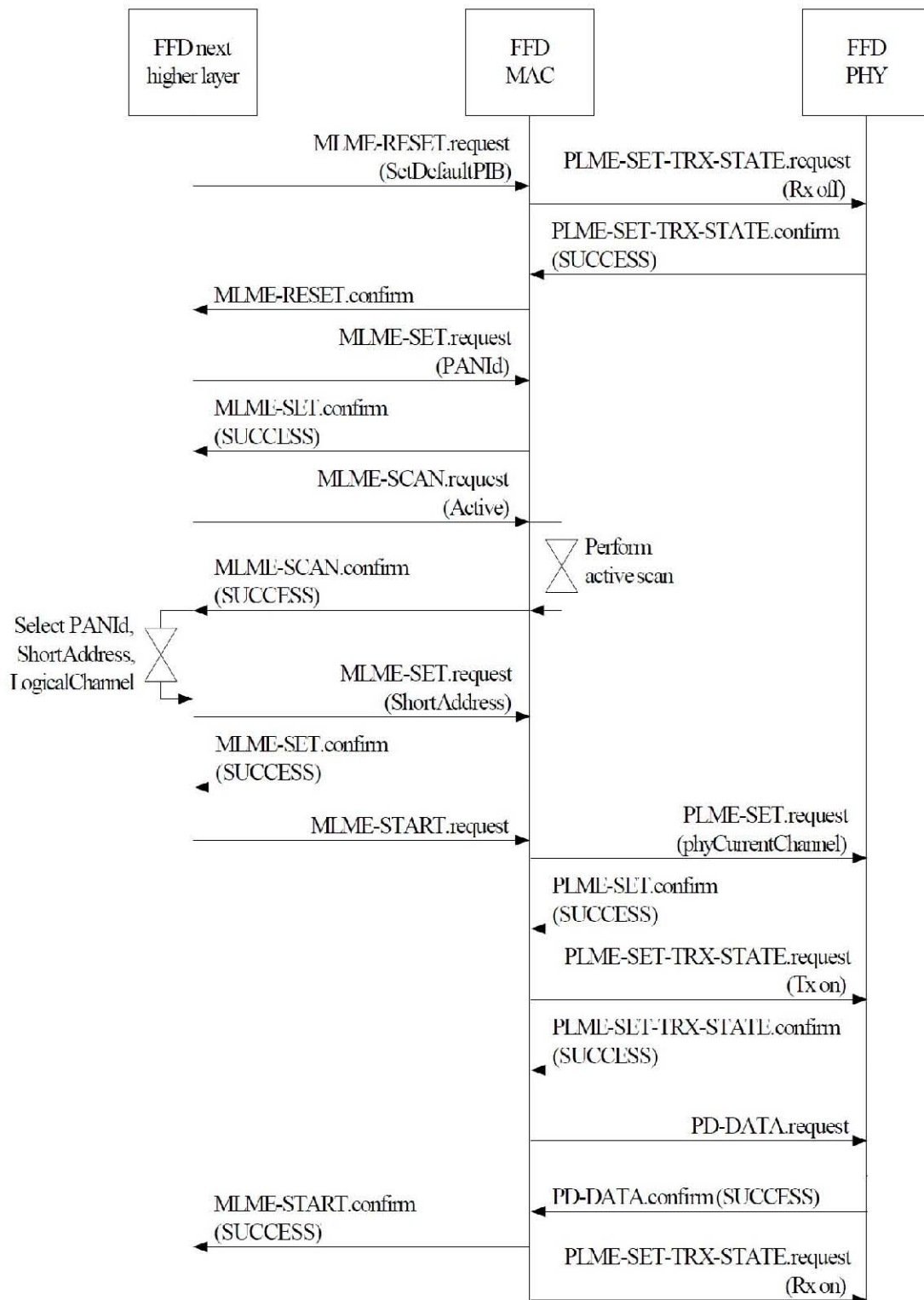


Figura 3.10: Primitivas para la Creación de una red PAN

incluyen los intercambios de mensajes entre la subcapa MAC y la capa física PHY. El estándar sigue definiendo tareas relativas a la seguridad y varios anexos que incluyen los vectores de test, las modulaciones, y la coexistencia con otras redes, pudiéndose acudir al propio estándar para su lectura en detalle [9].

3.3. Pila ATMEL–MAC

Atmel ha desarrollado una pila MAC del estándar IEEE 802.15.4 para distintas plataformas y transceptores de radio. Se basa en una nueva estructura de ficheros que facilita la portabilidad y la configuración al mismo tiempo que optimiza el uso de los recursos. La documentación y la pila [32] dan soporte a dos IDEs: *IAR Embedded Workbench* y *AVR Studio/WinAVR*. Se utilizará el segundo por ser de libre distribución. La pila está estructurada en carpetas que dan soporte a la capa física, a la subcapa MAC, un directorio con los ficheros de configuración de las plataformas, una capa de abstracción de los transceptores de radio, otra para las plataformas y otras adicionales para la seguridad y control. Después de una lectura completa de la Guía de Usuario que acompaña la pila [35], se adquieren los conocimientos y las directrices para poder compilar el ejemplo que está disponible con el fin de poder evaluar una red básica para el kit RZRAVEN. Destacar que en las placas AVRRAVEN, solamente se utiliza el dispositivo ATmega1284P, ya que es el que se programa con la pila para manejar el chip de radio.

Capítulo 4

Diseño e Implementación

La pila IEEE 802.15.4 ha sido implementada por Atmel y está disponible en su web como fichero ‘AVR2025_MAC_v2_5_2_raven.zip’ [32]. Ésta dispone de un ejemplo orientado a poder evaluar una red básica para el kit RZRAVEN. Una vez compilado y comprobado, se verifica el correcto funcionamiento de las placas AVRRAVEN y del dispositivo RZUSBSTICK. El coordinador inicia una red PAN de tipo sin baliza *Non-Beacon*, y sólo admite la asociación de dos nodos. Los nodos, al asociarse, envían datos aleatorios cada 2 segundos. Este ejemplo, por lo tanto, tiene un claro fin de demostración. Este TFM modificará este ejemplo para conseguir una mayor funcionalidad y poder disponer de una aplicación de sensórica.

4.1. Diseño Placa Auxiliar

El dispositivo RZUSBSTICK tiene varios LEDs que sirven para indicar el estado de la red, y para avisar del envío o recepción de mensajes. Estos indicadores visuales no están presentes en la tarjeta AVRRAVEN. Además, en la placa AVRRAVEN, existen varios sensores analógicos, como una resistencia NTC y un joystick, pero sólo están conectados a los pines del microcontrolador ATmega3290.

Debido a ello, se diseña y se fabrica una pequeña placa auxiliar que permita tener al ATmega1284P de recursos similares a los anteriores. Usando un esquema similar a RZUSBSTICK, se utilizarán tres LEDs y un sensor analógico, que se conectarán al microcontrolador ATmega1284P. Antes de diseñar el esquemático y decidir los componentes necesarios en la placa auxiliar, se consulta la hoja de características del dispositivo

RZRAVEN [34][36] y del ATmega1284P [37].

Todo circuito electrónico debe ser alimentado correctamente. La placa AVRRAVEN dispone de un conector de 12 pines para varias entradas. Los dos primeros se corresponden con la alimentación externa EXT_PWR, que admite tensiones de 3,6V a 12V, ya que se utiliza el regulador LDO LP2985A-33DBV de 3,3V. Por ello, se utilizará una batería que proporcione una tensión en ese rango. Para poder conectar los LEDs al dispositivo ATmega1284P y los sensores, se estudia los GPIO o pines de Entrada/Salida disponible en los tres conectores de 10 pines J201, J202 y J203. Este último conector está ruteado hacia las entradas ADC del microcontrolador, considerándose utilizar 4 para dar flexibilidad a este TFM. Adicionalmente, se utilizarán salidas digitales para poder encender o apagar los LEDs. Observando el esquemático del stick RZUSBS-TICK, y la propia placa, existen un LED azul, uno rojo, uno verde y uno naranja. Se conservará esta distribución para poder visualizar el estado de funcionamiento y la conexión de la tarjeta AVRRAVEN en la red. Así, las entradas y salidas utilizadas se detallan en la Tabla 4.1.

Pines ATmega1284P	Conexión Header	Comentario
PA0	J203-1	Canal 0 ADC
PA1	J203-2	Canal 1 ADC
PA2	J203-3	Canal 2 ADC
PA3	J203-4	Canal 3 ADC
PD4	J202-5	DIO Led Azul/Blanco
PD5	J202-6	DIO Led Rojo
PB2	J202-7	DIO Led Verde
PD7	J202-8	DIO Led Naranja

Tabla 4.1: Pines ATmega1284P

Teniendo en cuenta estas premisas, se diseña con el software *IDE Altium Designer* el esquemático de la placa auxiliar, que contará con 4 LEDs, un sensor de temperatura, un potenciómetro y una LDR. La alimentación se realiza mediante una batería de Ión-Litio de 3,7V y 2000mAh, con el fin de garantizar una autonomía suficiente para las pruebas. Esta alimentación también se utiliza para alimentar los componentes de la placa auxiliar mediante un estabilizador de 3,3V. El esquemático se muestra en la Figura 4.1.

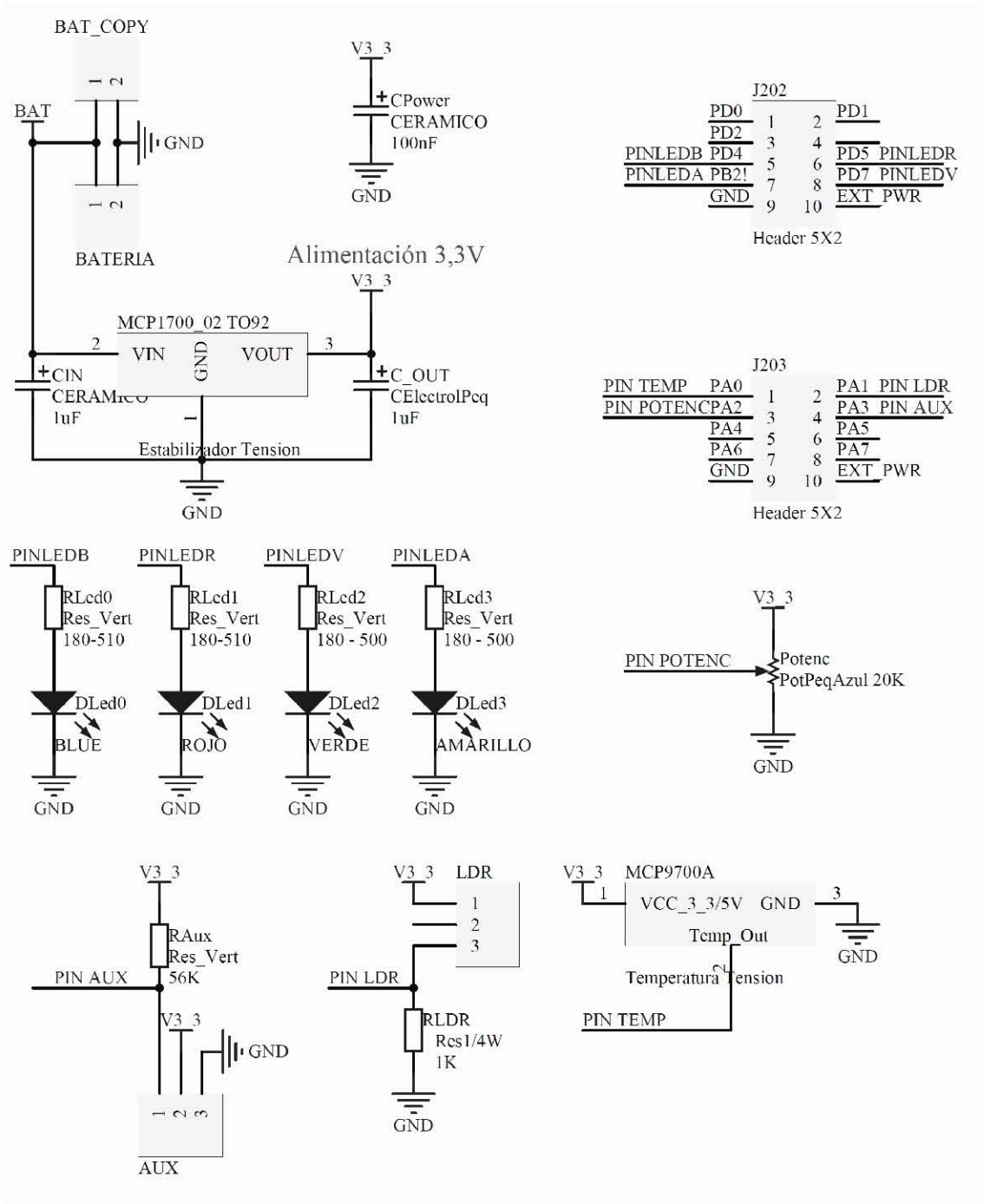


Figura 4.1: Esquemático de la Placa Auxiliar Diseñada

A continuación, se genera el fichero PCB. Se procede a colocar los componentes de manera similar al esquemático, para intentar que las conexiones generasen pistas directas y de pequeña longitud. Intentando mantener las dimensiones al mínimo posible, se intenta colocar en los bordes de la placa las conexiones, los LEDs de estado, y el conector de la batería. En la Figura 4.2 se muestra el ruteado final de la placa auxiliar diseñada.

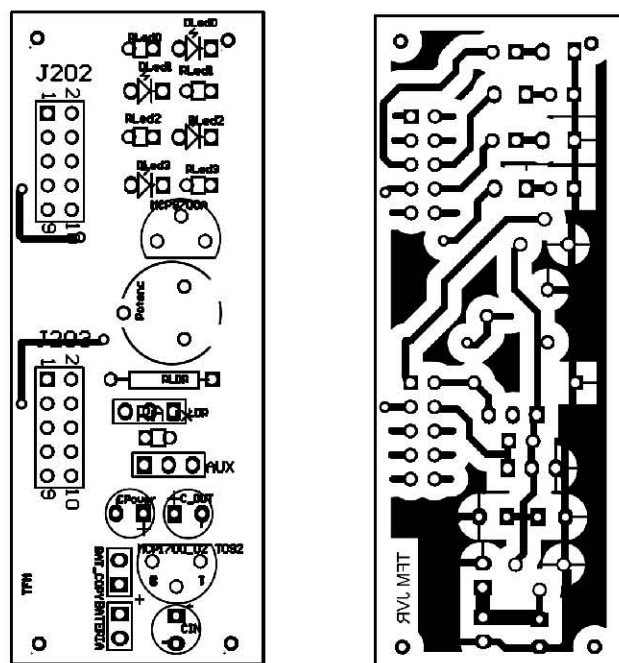


Figura 4.2: Ruteado de la Placa Auxiliar Diseñada

Debido a las técnicas utilizadas en el diseño y el ruteado, se obtiene una placa de pequeñas dimensiones. Esto permite fabricar la placa de manera artesanal, sin necesidad de recurrir a una fabricación industrial o utilizar un laboratorio de prototipado. Una vez fabricada, se procede a montar los componentes de inserción. Posteriormente, se comprueba manualmente la continuidad de las pistas de la placa. En la Figura 4.3 se observa la placa terminada y montada.

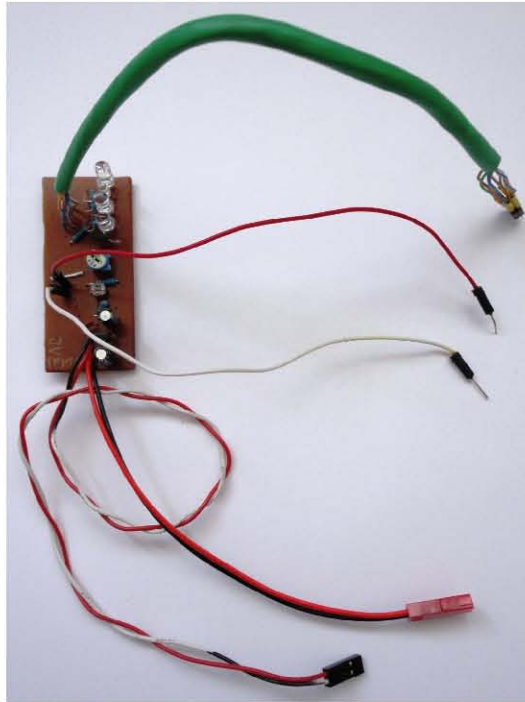


Figura 4.3: Placa Auxiliar Implementada

4.2. Modificación de los Ficheros AVRRAVEN

Como se explica en la Guía de Usuario [35], hay que realizar una serie de cambios en varios ficheros. Una vez completada la implementación de la placa auxiliar, se procede a modificar los ficheros de la pila para poder disponer de la funcionalidad deseada en este TFM. Para poder utilizar los LEDs de la placa auxiliar, se modifica el fichero de configuración de pines del AVRRAVEN. Éste se encuentra en la carpeta `PAL\AVR\ATMEGA1284P\Boards\Raven`, debido a que la capa PAL (*Platform Abstraction Layer*) es la que especifica las interfaces de las plataformas. Para ello, se modifica el fichero `pal_config.h` incluyendo los LEDs y los pines analógicos. Las modificaciones realizadas se detallan en el Listado 4.1.

```
typedef enum led_id_tag
{
    LED_0,
    LED_1,
    LED_2
} led_id_t;

#define NO_OF_LEDS          (3)
```

```
//LED_0 Blanco/azul  LED_1 Rojo  LED_2 Verde
//PORT LEDs conectados
#define LED_PORT_LED_0      (PORTD)
#define LED_PORT_DIR_LED_0  (DDRD)
#define LED_PORT_LED_1      (PORTD)
#define LED_PORT_DIR_LED_1  (DDRD)
#define LED_PORT_LED_2      (PORTD)
#define LED_PORT_DIR_LED_2  (DDRD)

//PINs LEDs conectados
#define LED_PIN_0           (PIND4)
#define LED_PIN_1           (PIND5)
#define LED_PIN_2           (PIND7)

// ADC pines
#define PIN_TEMP             (PINA0)
#define PIN_LDR              (PINA1)
#define PIN_POTEN            (PINA2)
#define PIN_AUX              (PINA3)

// ADC canales
#define CANAL_TEMP           (0)
#define CANAL_LDR            (1)
#define CANAL_POTENC         (2)
#define CANAL_AUX            (3)
```

Listado 4.1: *pal_config.h*

El siguiente fichero a modificar es *pal_board.c*. Este fichero se encarga de la funcionalidad específica PAL para cada plataforma. En este caso, se incluyen las funciones para controlar los LEDs que se muestran en el Listado 4.2.

```
//LED_0 Blanco/azul  LED_1 Rojo  LED_2 Verde
//@brief Initialize LEDs
void pal_led_init(void)
{
    LED_PORT_LED_0    &= ~(1 << LED_PIN_0); // high active
    LED_PORT_LED_1    &= ~(1 << LED_PIN_1); // high active
    LED_PORT_LED_2    &= ~(1 << LED_PIN_2); // high active
    LED_PORT_DIR_LED_0 |= (1 << LED_PIN_0);
    LED_PORT_DIR_LED_1 |= (1 << LED_PIN_1);
    LED_PORT_DIR_LED_2 |= (1 << LED_PIN_2);
}
```

```

//@brief Control LED status
// param[in] led_no LED ID
// param[in] led_setting LED_ON, LED_OFF, LED_TOGGLE
void palled(led_id_t led_no, led_action_t led_setting)
{
    uint8_t pin;
    uint8_t *port;
    bool active_dir;

    switch (led_no)
    {
        case LED_0: /* high active */
            pin = LED_PIN_0;
            port = (uint8_t *)&LED_PORT_LED_0;
            active_dir = HIGH;
            break;

        case LED_1: /* high active */
            pin = LED_PIN_1;
            port = (uint8_t *)&LED_PORT_LED_1;
            active_dir = HIGH;
            break;

        case LED_2: /* high active */
            pin = LED_PIN_2;
            port = (uint8_t *)&LED_PORT_LED_2;
            active_dir = HIGH;
            break;

        default: return;
    }

    switch (led_setting)
    {
        case LED_ON:
            if (active_dir == HIGH)
            {
                *port = *port | (1 << pin);
            }
            else
            {
                *port = *port & ~(1 << pin);
            }
            break;

        case LED_OFF:
            if (active_dir == HIGH)
            {

```

```
        *port = *port & ~(1 << pin));
    }
    else
    {
        *port = *port | (1 << pin);
    }
    break;

case LED_TOGGLE:
    if ((*port & (1 << pin)) == 0x00)
    {
        *port = *port | (1 << pin);
    }
    else
    {
        *port = *port & ~(1 << pin);
    }
    break;
default: /* do nothing */ break;
}
}
```

Listado 4.2: *pal_board.c*

4.3. Coordinador

4.3.1. Ejemplo Básico

Como coordinador se utilizará el dispositivo RZUSBSTICK con el fin de poder utilizar el puerto USB para las comunicaciones. En la carpeta `App_1_Nobeacon\Coordinator\AT86RF230B_AT90USB1287_USBSTICK_C` se encuentra el proyecto asociado al ejemplo 1 de una red *NonBeacon*. Al abrirlo, se ejecuta la aplicación *Atmel Studio*, mostrándose el código principal, los ficheros, y las librerías que tienen dependencia con el proyecto.

El ejemplo básico ejecuta las primitivas explicadas en el capítulo 3. Para facilitar la comprensión del código del ejemplo básico, en la Figura 4.4 se muestra el diagrama de estados del ‘Coordinador’ implementado en él.

El código empieza con la función *wpan_init()*, que se encarga de inicializar la capa MAC y sus capas subyacentes, como PAL, TAL, etc. Si el inicio es correcto, se

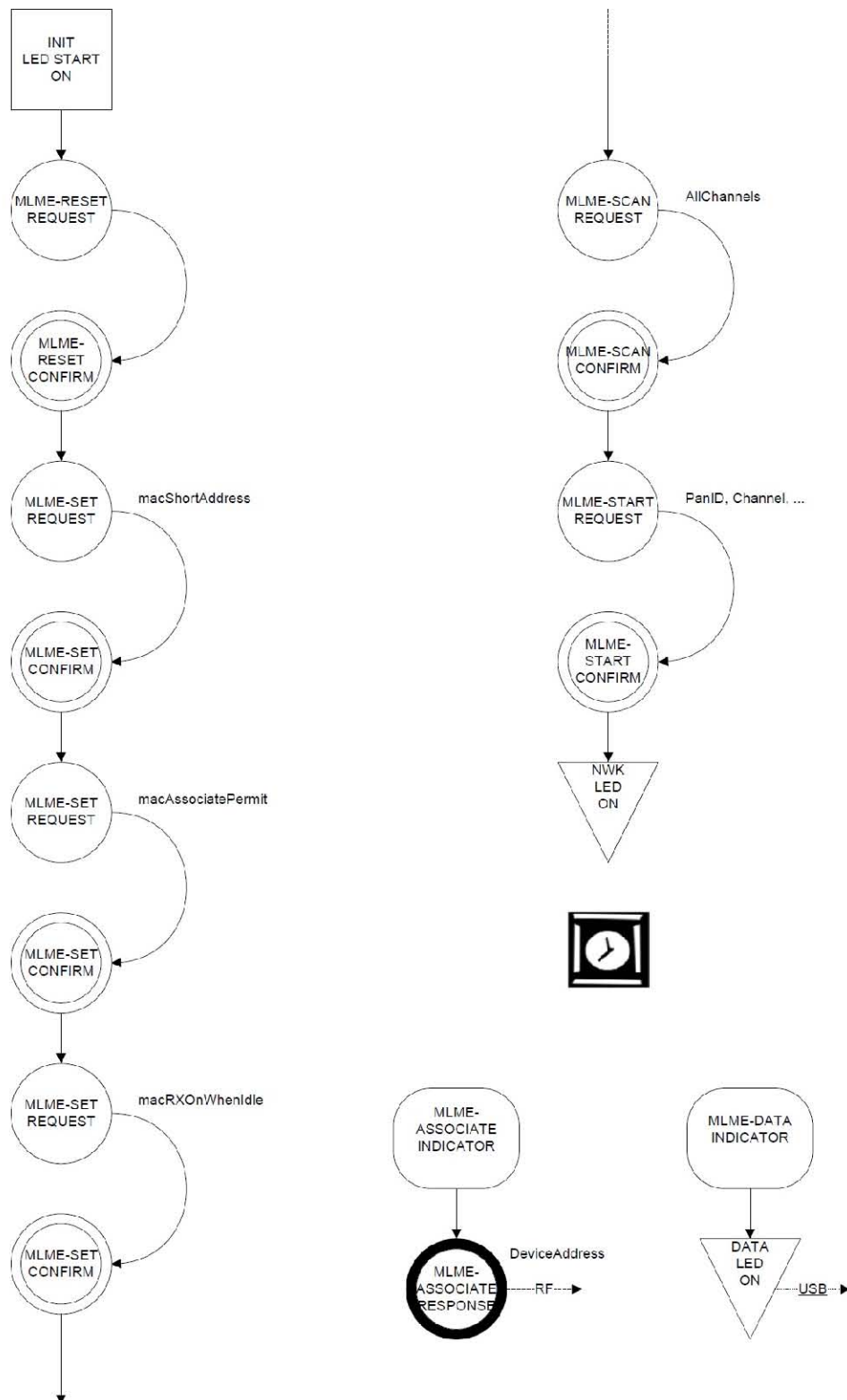


Figura 4.4: Diagrama de Estados del Ejemplo

enciende el LED_0 (Led Azul) del dispositivo RZUSBSTICK y el resto permanece apagado. Entonces se llama a la primitiva *MLME-RESET.request*. Atmel utiliza una nomenclatura muy similar a la del estándar, correspondiéndose con la función *wpan_mlme_reset_req(true)*. A partir de aquí, el código principal se queda en un bucle infinito ejecutando el planificador de tareas *wpan_task()* a la espera de ejecuciones de primitivas o la función de retorno (callbacks).

Cuando la capa MAC termine de ejecutar la primitiva *MLME-RESET.request*, se invoca su función de retorno, *usr_mlme_reset_conf()*. Si la inicialización se realizó de manera correcta, la variable de estado *status* tendrá el valor 'SUCCESS' y se establecerá el atributo PIB correspondiente a la dirección del coordinador, pasando como parámetros el índice del atributo, y el valor (0x002A), ya que la función es de escritura o 'SET'. Si falló el *reset*, se vuelve a ejecutar la primitiva *MLME-RESET.request*. El procedimiento completo se puede ver en el Listado 4.3.

```
void usr_mlme_reset_conf(uint8_t status)
{
    if (status == MAC.SUCCESS)
    {
        // Set the short address of this node.
        uint8_t short_addr[2];
        short_addr[0] = (uint8_t)COORD.SHORT_ADDR;
        short_addr[1] = (uint8_t)(COORD.SHORT_ADDR >> 8);
        wpan_mlme_set_req(macShortAddress, short_addr);
    }
    else
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
    }
}
```

Listado 4.3: *usr_mlme_reset_conf*

Cuando la solicitud para establecer la dirección corta del coordinador sea procesada, la capa MAC llamará a la función de confirmación *usr_mlme_set_conf()*. En ella, se procesa si el atributo se ha modificado correctamente para poder continuar con el establecimiento de la red PAN, habilitando la asociación de los nodos. Para ello, se vuelve a llamar a la función *MLME-SET.request*, pero en esta ocasión modificando el atri-

buto *macAssociationPermit*. La función de retorno será de nuevo *usr_mlme_set_conf()*, pero mediante otra función de control ‘if’, se verifica esta nueva solicitud y se inicia un escáner activo. Estos pasos se ven claramente en el Listado 4.4.

```
void usr_mlme_set_conf(uint8_t status, uint8_t PIBAttribute)
{
    if ((status == MAC.SUCCESS) && (PIBAttribute == macShortAddress))
    {
        //Allow other devices to associate to this coordinator.
        uint8_t association_permit = true;
        wpan_mlme_set_req(macAssociationPermit,
                        &association_permit);
    }
    else if ((status == MAC.SUCCESS) && (PIBAttribute == macAssociationPermit))
    {
        //Initialize an active scan over all channels to determine
        //which channel to use.
        wpan_mlme_scan_req( MLME_SCAN_TYPE_ACTIVE,
                        SCAN_ALL_CHANNELS,
                        SCANDURATION.COORDINATOR);
    }
    else
    {
        // something went wrong; restart
        wpan_mlme_reset_req(true);
    }
}
```

Listado 4.4: *usr_mlme_set_conf*

Una vez terminado el escáner, se invoca la función *usr_mlme_scan_conf()*. En la variable *ResultList* se tendrán los coordinadores encontrados con sus datos. En esta ocasión, se ignoran estos resultados porque se va a establecer una red nueva con valores predefinidos, como es el identificador PAN (0xBABE) y el canal 20. Para ello, se ejecuta la primitiva *MLME-START.request* como se muestra en el Listado 4.5.

```
wpan_mlme_start_req(DEFAULT_PAN_ID,
                    DEFAULT_CHANNEL,
                    DEFAULT_CHANNEL_PAGE,
                    15, 15,
                    true, false, false);
```

Listado 4.5: *usr_mlme_start_req*

Una vez concluida la creación de la red, el coordinador enciende el LED_1 (Led Rojo) para indicarlo y se queda a la espera de la asociación de los nodos. Cuando algún nodo lo solicite se ejecuta la función *usr_mlme_associate_ind()*. En ella, se busca en la tabla de direcciones y si existe alguna dirección libre, se le otorga al nodo mediante la primitiva *MLME-ASSOCIATE.response* y el estado 'ASSOCIATION_SUCCESSFUL'. En caso contrario, se le deniega la solicitud con la misma primitiva, pero con el estado de red PAN llena, 'PAN_AT_CAPACITY'. A partir de este punto, ese nodo podrá enviar datos y el coordinador se queda a la espera de alguna nueva asociación o la entrada de datos. Cuando algún nodo envíe datos, la primitiva *MCPS-DATA.indication*, o en este caso, la función *usr_mcps_data_ind()*, se ejecuta, encendiéndose el LED_2 (Led Verde) para indicar la recepción del dato.

4.3.2. Modificaciones

La funcionalidad explicada anteriormente es la que implementa el ejemplo que trae la pila de Atmel. Debido a su sencillez, en este TFM se van a añadir varias funciones adicionales. El diagrama final de estados del 'Coordinador' implementado en este TFM se muestra en la Figura 4.5.

Una función necesaria es realizar comandos a los nodos por parte del coordinador. Éstos se recibirán por el puerto USB, siendo enviados desde un software visor. El comando recibido se procesa y se enviará la instrucción asociada al nodo correspondiente. Para realizar este proceso, se utilizará un puerto COM virtual del tipo USB-CDC (*USB Communications Device Class*). La pila de Atmel incluye esta librería, la cual hay que añadir al proyecto de *AVR Studio* e inicializar el USB del dispositivo RZUSBSTICK.

Para ello, consultando la Guía de Usuario [35], y después de muchas pruebas, se consigue añadir la librería mediante la instrucción *#include "sio_handler.h"*. Posteriormente, hay que modificar el fichero para la compilación *Makefile* con el fin de incluir la ruta de la librería y los flags de definición para indicar que el puerto de comunicaciones es el USB. Destacar que esta librería no soporta la salida formateada del tipo *printf*, por lo que dificulta el código. En el código de inicialización, hay que añadir la función *pal_sio_init(SIO_CHANNEL)* que se encarga de activar el bus USB y la enumeración por parte del PC. A continuación, se crea un *timer* periódico con un tiempo de refresco

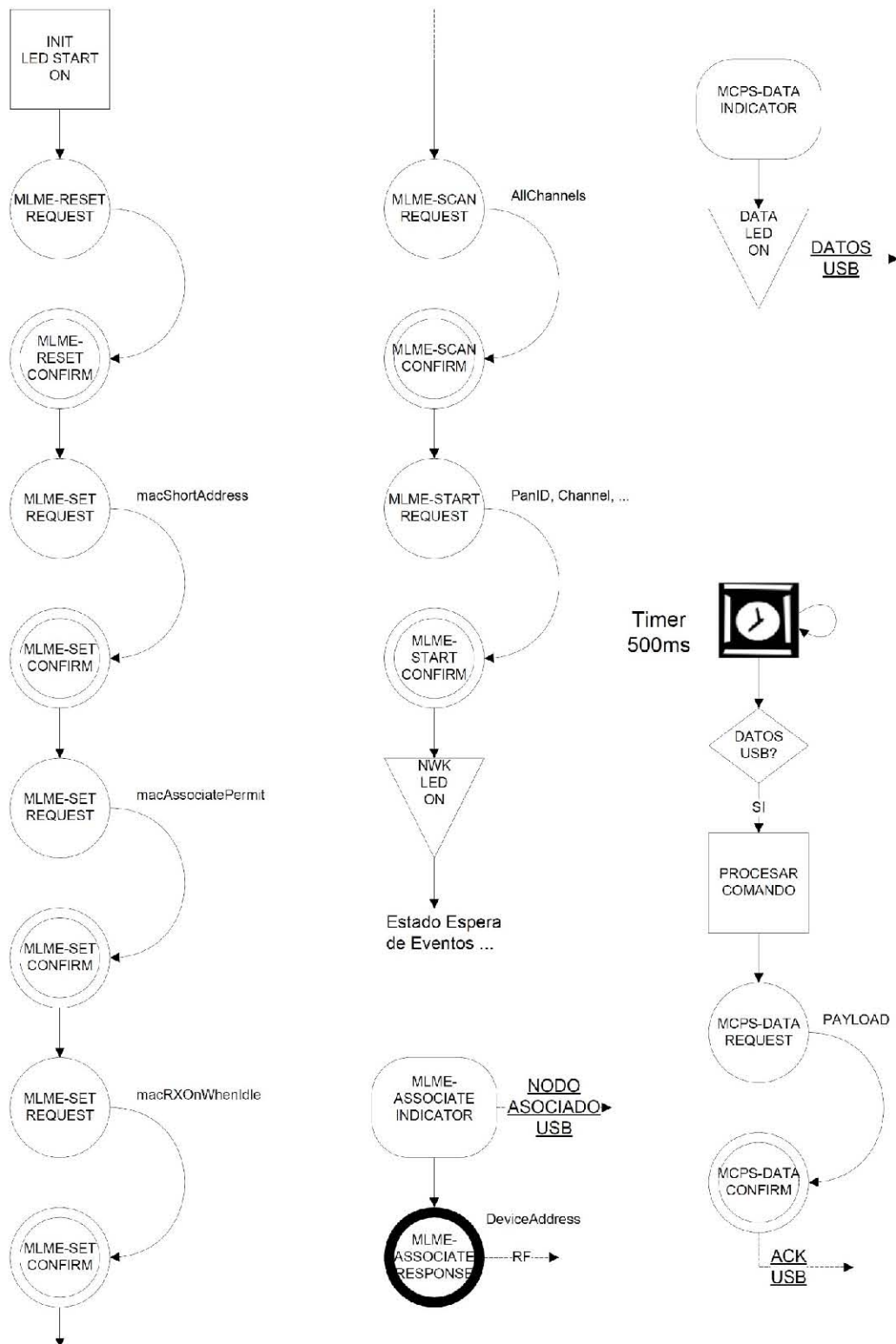


Figura 4.5: Diagrama de Estados del Coordinador

de 500ms para comprobar si existen caracteres enviados por el software visor y que tienen que ser procesados.

Trama Visor–RZUSBSTICK

Para la comunicación entre el visor y del dispositivo RZUSBSTICK, se ha decidido diseñar una trama que permita la transferencia del comando, de la dirección y de varios parámetros al RZUSBSTICK. El formato de la trama se muestra en la Tabla 4.2.

Inicio de Instrucción (\$)	Comando	Byte 0	Byte 1 ...
0x24	Comando	Direccion_Destino	Parámetro(s)

Tabla 4.2: Trama Visor–RZUSBSTICK

De esta manera, el coordinador es capaz de identificar el inicio de una instrucción USB, el comando, el destinatario, y los parámetros. Las instrucciones se leen con la función `number_of_bytes_usb_rx = pal_sio_rx(SIO_CHANNEL, sio_rx_data, MAX_APP_DATA_LENGTH)`; Si el número de bytes recibidos es mayor que cero, entonces se procede a validar la Trama USB y se decodifica el comando. El comando que se ha implementado es la posibilidad de encender o apagar un LED de los nodos finales, codificándose como ‘L’. De esta manera, se consigue la función de enviar comandos a los nodos, que ejecutan una tarea o acción. Destacar que se podría implementar cualquier función deseada en los nodos según la aplicación final. La Trama USB que se envía desde el visor es la mostrada en la Tabla 4.3.

Inicio de Instrucción (\$)	Comando	Byte 0	Byte 1 ...
0x24	‘L’	Direccion_Destino	Encender/Apagar

Tabla 4.3: Trama Comando *L*

El último byte sirve para indicar el estado del LED. Una vez procesada la trama, la siguiente tarea es enviarla al nodo destinatario. Las instrucciones enviadas a los nodos también se estructuran con un formato que facilite la identificación del nodo origen, el comando y los parámetros. La estructura que se utiliza para las comunicaciones entre el coordinador y los nodos, o viceversa, presenta la estructura mostrada en la Tabla 4.4.

Byte 0	Byte 1	Byte 2	Byte 3
Inicio de Instrucción (\$)	ID_Nodo_Origen	Dirección_Origen	Dirección_Destino
Byte 4	Byte 5	Byte 6 ...	Último Byte
Separador ()	Comando	Parámetro(s)	Fin de Instrucción (%)

Tabla 4.4: Estructura de las Instrucciones de Comunicación

La primera parte de la instrucción sirve para la identificación y la definición de las direcciones, mientras que la segunda es el comando que debe procesar el nodo. El campo *ID_Nodo* corresponde con un entero, utilizándose en este TFM el número 5 para el Coordinador, con la dirección corta 0x002A. El resto de nodos finales utilizarán del entero 11 en adelante, y las direcciones cortas serán las que les asigne el coordinador. A continuación, en la Tabla 4.5 se muestra cómo queda la instrucción de Encender o Apagar el LED enviada por el coordinador al nodo final con *ID_Nodo* ‘11’.

Byte 0	Byte 1	Byte 2	Byte 3
Inicio de Instrucción (\$)	ID_Nodo_Origen	Dirección_Origen	Dirección_Destino
0x24	5	0x2A	Dirección_Destino
Byte 4	Byte 5	Byte 6	Byte 7
Separador ()	Comando	Parámetro(s)	Fin de Instrucción (%)
0x7C	‘L’	1/0	0x25

Tabla 4.5: Instrucción Enviada por el Coordinador

Sólo queda pendiente enviar esta instrucción mediante la primitiva *wpan_mcps_data_req()*. El mensaje enviado se indica a través del mensaje “Coord> Enviando datos a: *Dirección_Destino*”, que se envía con la función *pal_sio_tx()* al visor para que sea mostrado por pantalla con el fin de facilitar el control y la gestión de la red. Si se realiza la entrega de la trama al nodo correspondiente, al recibirse el ACK, se invoca la función *wpan_mcps_data_conf()*, indicándose mediante el mensaje “Coord> ACK_OK” en el visor. Si el nodo destino no existe, no se recibirá la confirmación de entrega y por lo tanto, no se mostrará este mensaje. De esta manera, el usuario podrá conocer el resultado del envío.

Otras modificaciones que se han realizado en el código son la de incluir varios

mensajes de estado o eventos de la red cuando se ejecutan ciertas primitivas. Un ejemplo es cuando se realiza correctamente la asociación de un nodo, enviándose al visor el texto “Coord> Nodo Asociado num: *Dirección_Asignada*”, para que sea mostrado por pantalla con el fin de facilitar el control y la gestión de la red. Otra tarea pendiente es que cuando se reciban datos provenientes de los nodos con el formato de instrucción explicado anteriormente, estos se envíen por el puerto USB para que el visor los pueda procesar y mostrar por pantalla.

4.4. Nodo Final

En el caso de los nodos finales o ‘*End Devices*’, el código que trae el ejemplo base es similar al explicado en la sección del Coordinador. Lo que es diferente es que en vez de crear la red, después del escáner, el nodo intenta asociarse al coordinador predefinido. Si lo consigue, empezará a enviar datos aleatorios cada 2 segundos.

El diagrama de estados del Nodo Final implementado en este TFM se muestra en la Figura 4.6.

En este TFM, se envían datos periódicamente cada 10 segundos. Estos datos tendrán el formato mostrado en la Tabla 4.4. En este caso, a la codificación del comando se le ha asignado la letra ‘P’ de prueba, enviándose como parámetros el valor del canal analógico número 0. En la Tabla 4.6 se presenta la instrucción que se enviaría al Coordinador si el nodo tiene asociado el *ID_Nodo* ‘11’, y el valor del canal ADC_0 fuese 0x1234.

Byte 0	Byte 1	Byte 2	Byte 3	...
Inicio de Instrucción (\$)	ID_Nodo_Origen	Dirección_Origen	Dirección_Destino	...
0x24	11	Dirección_Origen	0x2A	...
Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Separador ()	Comando	Parámetro 1	Parámetro 2	Fin de Instrucción (%)
0x7C	‘P’	0x12	0x34	0x25

Tabla 4.6: Instrucción Enviada por un Nodo Final

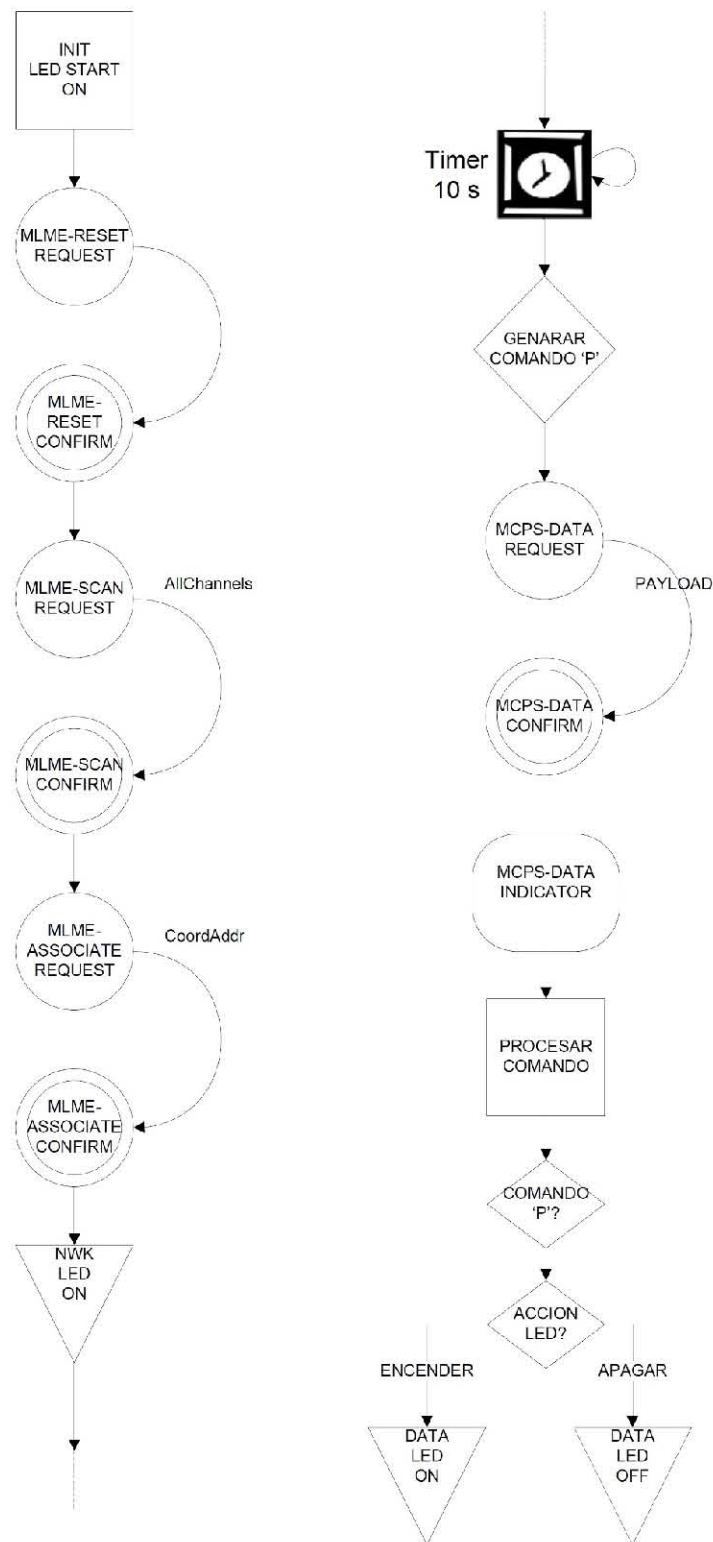


Figura 4.6: Diagrama de Estados del Nodo Final

Esta instrucción se pasa como parámetro a la primitiva *MCPS-DATA.request*, que se corresponde con la función *wpan_mcps_data_req()* para enviar los datos al Coordinador.

Para poder recibir datos, hay que activar la recepción de datos en los nodos. Para ello, se activa el estado *macRXOnWhenIdle*. Ahora, ante la llegada de una trama a un nodo, al invocarse la función *wpan_mcps_data_ind()* es necesario procesar la instrucción para saber el tipo de comando a ejecutar. En este TFM sólo se ha implementado la función de apagar o encender un LED, por lo que el comando será necesariamente 'L'. Destacar que la inclusión de otros comandos se puede hacer de manera trivial y totalmente flexible en función de la aplicación final deseada. El procesado de las instrucciones se realiza con el código mostrado en el Listado 4.6.

```
//PROCESAR INSTRUCCION
inicio_cabecera = msdu[0];    // '$ '
ID_NODO_Origen  = msdu[1];
direcc_origen   = msdu[2];    // direcc origen coord/device
direcc_destino  = msdu[3];    //direcc destino: es la de este device
// 4 es separador '|'
comando = msdu[5];
//datos:
parametro = msdu[6];
// otros parametros ...
final_cabecera = msdu[PAYLOAD.SIZE-1];    //'%'

if (inicio_cabecera == '$' && final_cabecera == '%')
{ //Inicio y Final de instrucción correcta

    if (comando == 'L')
    { //Comando L: Encender/Apagar LED:
        if ( parametro == 1)
        {
            pal_led(LED_DATA, LED_ON);
        }
        else
        {
            pal_led(LED_DATA, LED_OFF);
        }
    } //end if(comando='L'
} //end if(inicio_cabecera == '$' && '%')
```

Listado 4.6: Comando L

4.5. Aplicación Visor

La aplicación Visor, o Interfaz Gráfica de Usuario (GUI), se ha desarrollado utilizando el IDE del software *LabView*. LabView, desarrollado por National Instruments [38], es un entorno de programación gráfica que permite desarrollar sistemas sofisticados de medida, pruebas y control. Se basa en un lenguaje gráfico propietario, donde se utilizan bloques, o iconos, e interconexiones para relacionarlos entre ellos, formando así un diagrama de flujo de datos. Estos diagramas de bloques se integran en un Instrumento Virtual (VI), que contiene el código y un panel frontal visual para poder interactuar con el usuario u otros VI. El desarrollo de aplicaciones por lo tanto, se realiza sin necesidad de escribir líneas de código, sino uniendo los objetos proporcionados mediante bibliotecas.

Una de las ventajas de este lenguaje es que se ofrecen cientos de VI de dispositivos hardware ya existentes. Su principal uso es conectar dispositivos o instrumentos de medida, y permitir la comunicación o interacción entre ellos, de manera personalizada y automatizada. LabView por lo tanto, permite la simple monitorización de uno o varios equipos, el registro de medidas en ficheros para su posterior análisis, o la creación de entornos de test automatizados para verificar el correcto comportamiento de los sistemas.

El visor en este TFM permite realizar las comunicaciones y la gestión de la red. Éste se ha diseñado para representar la información de interés en ventanas de fácil lectura y comprensión para el usuario. El panel frontal del visor desarrollado se muestra en la Figura 4.7.

En la Figura 4.8 se muestra cómo la programación se basa en el conexionado de los bloques de un sistema.

Puerto COM–USB

El código realizado en LabView tiene varias partes diferenciadas. La primera parte contiene la configuración del puerto serie USB–CDC (véase la Figura 4.9). Entre los datos más importantes, está el propio puerto, 8 bits de datos sin paridad y un bit de *stop*. El panel frontal resenta un control para seleccionar el puerto COM que se crea al conectar el dispositivo RZUSBSTICK al puerto USB, y un botón para iniciar la

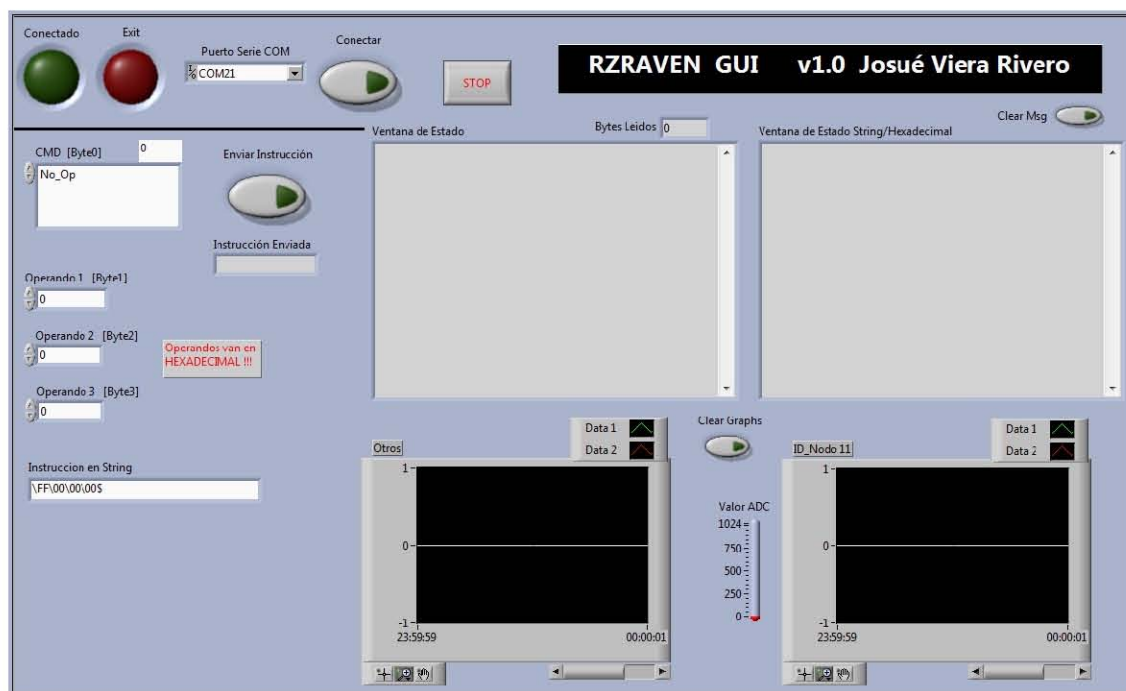


Figura 4.7: Panel Frontal de la Aplicación Desarrollada en LabView

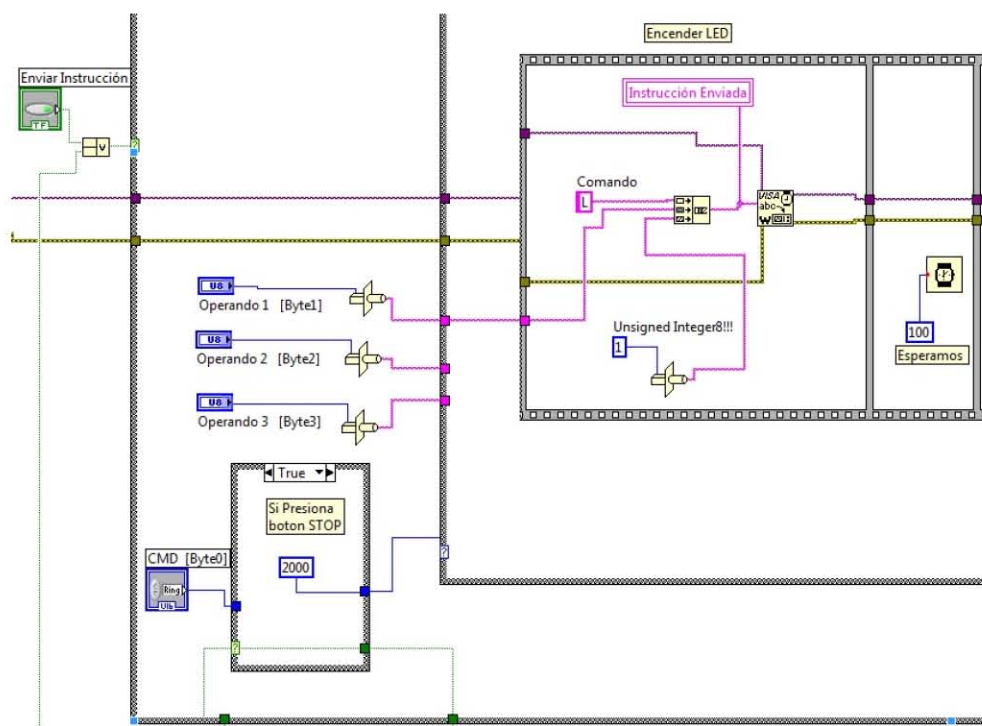


Figura 4.8: Conexionado de Bloques en LabView

comunicación, como se muestra en la Figura 4.10.

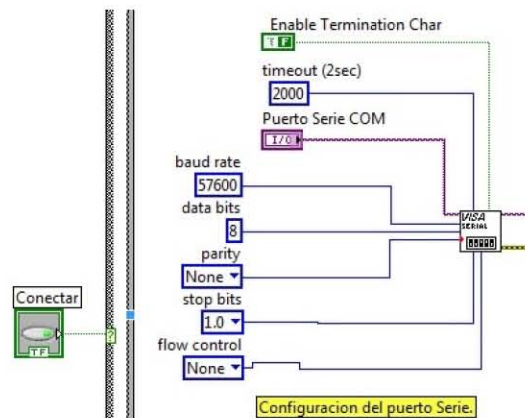


Figura 4.9: Código para el Puerto Serie



Figura 4.10: Interfaz Gráfica para el Puerto Serie

Menú de Instrucciones

A continuación, se dispone de un menú con las instrucciones disponibles, como se observa en la Figura 4.11. En un menú desplegable, se han incluido las instrucciones de *Encender* o *Apagar Led*, la posibilidad de enviar datos en hexadecimal, y la opción salir. Solamente hay que seleccionar la instrucción deseada, rellenar los parámetros si hiciesen falta, y pulsar el botón ‘Enviar Instrucción’. El código asociado es el mostrado en la Figura 4.11, donde se identifica la instrucción y se ejecuta el código asociado. En este caso, al ejecutar la instrucción *Encender Led* ‘L’, el código tiene el formato Trama USB correspondiente a esta instrucción.

Lectura de los Datos

Otra parte importante es la lectura de datos del puerto USB. Éstos son enviados por el Coordinador mediante la función *pal_sio_tx()* al visor para que sean mostrados por pantalla. Si existen datos nuevos, se leen y se muestran en la ventana de estado. Éstos

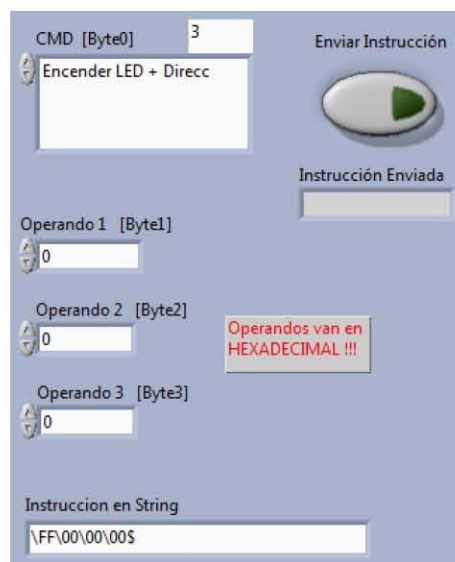


Figura 4.11: Interfaz Gráfica para Enviar Instrucción

podrán ser tramas que han enviado los nodos o los mensajes de estado que enviaba el Coordinador “Coord>” para facilitar el control y la gestión de la red. Al mismo tiempo, estos datos pasan por un conversor que analiza si los datos corresponden con una instrucción válida. Destacar que existen dos gráficas, una asociada al nodo con *ID_Nodo* a ‘11’, que es el que tiene conectado la placa auxiliar, y otra asociada al resto de nodos. Debido a ello, se diferencian los tipos de comandos y mediante la etiqueta *ID_Nodo* se identifica de qué nodo provienen, para actualizar la gráfica correspondiente. Si se desean borrar las trazas de las gráficas, o de las ventanas de estado, se puede realizar pulsando los botones correspondientes, que borran el historial. Todo esto se muestra en la Figura 4.12

Conversor.VI

El procesamiento de las tramas se realiza en otro VI: el conversor (véase el código en la Figura 4.13). La trama se procesa separando el texto entre los caracteres de inicio y de fin, como es el *ID_Nodo*, el Comando, y el payload (parámetros). En este caso, al ser un comando ‘P’, se corresponde al comando de prueba que envía la temperatura en los dos primeros bytes del payload. A este valor se le aplica una sencilla conversión a un entero de 16 bits para que pueda ser utilizado en las gráficas del panel frontal del visor. Este es el dato que se utiliza para las gráficas anteriores.

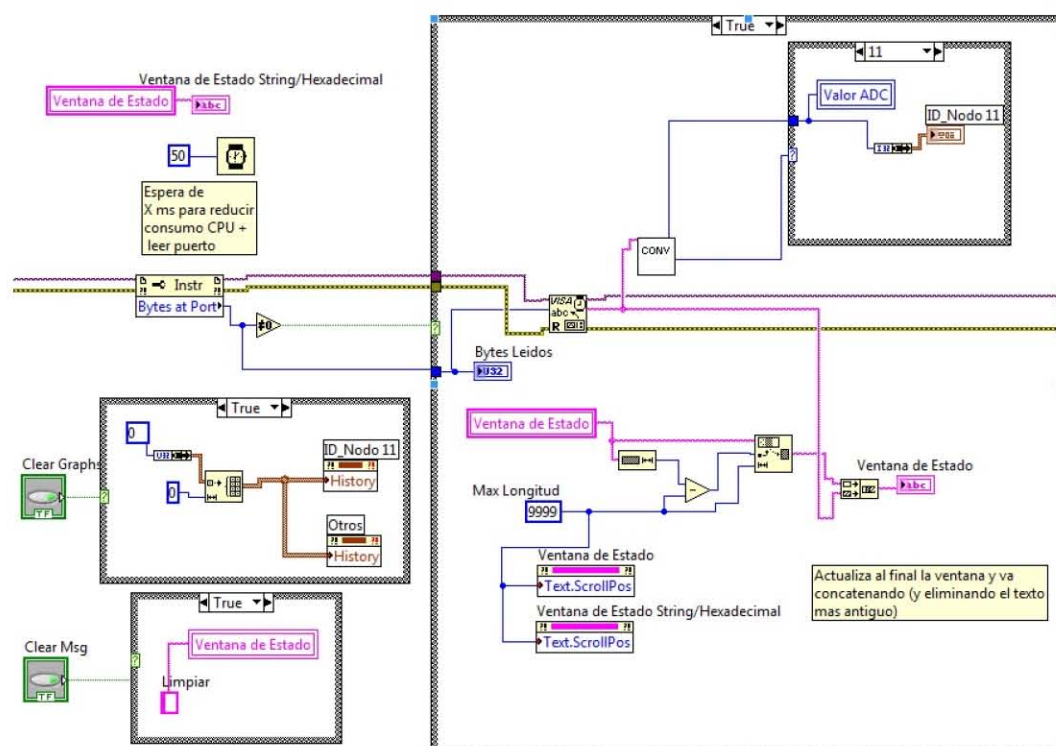


Figura 4.12: Código para el Refresco de la Pantalla

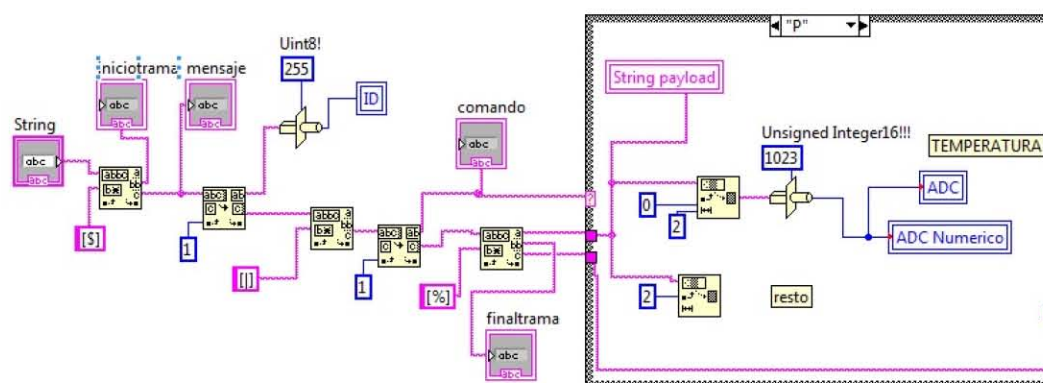


Figura 4.13: Código del Conversor

Finalmente, para detener la comunicación con el dispositivo RZUSBSTICK, se debe utilizar el botón STOP o la instrucción Salir (para más detalle véase la Figura 4.14), que cierran y el puerto serie correctamente y finalizan la ejecución de la Interfaz Gráfica de Usuario.

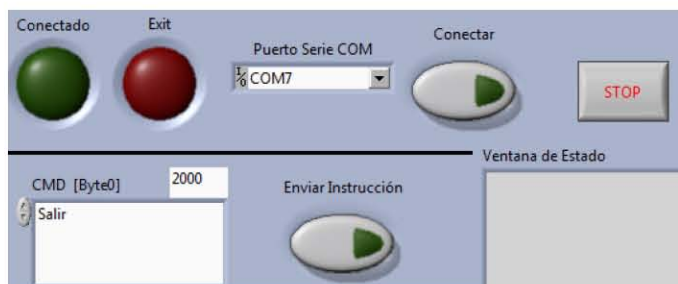


Figura 4.14: Botón STOP/Instrucción Salir

Capítulo 5

Verificación

El proceso de verificación busca asegurar que el sistema cumple los requisitos funcionales planteados inicialmente. Las funciones de la aplicación deben comprobarse mediante pruebas individuales y conjuntas con el fin de validar la funcionalidad y garantizar su correcto comportamiento. En este TFM se plantea un banco de pruebas basado en diferentes estados de ejecución de la pila y el envío de instrucciones a los nodos.

Una vez comprobada la continuidad y funcionamiento de la placa auxiliar, se procede a montar el sistema final compuesto por dos placas AVRRAVEN, una de ellas con la placa auxiliar, y el dispositivo RZUSBSTICK conectado al PC (véase la Figura 5.1).

A continuación, se procede a programar los dispositivos con el código desarrollado en la pila de Atmel, utilizando un programador KEE AVR JTAGICE mkII [39] por el puerto JTAG (véase la Figura 5.2).

Para poder verificar toda la aplicación, se han ido realizando pruebas incrementales del código. Cada función se ha verificado de manera independiente, con el fin de poder comprobar su correcto funcionamiento. La librería USB-CDC, la recepción de las tramas por parte de los nodos y del Coordinador, el procesamiento de las tramas y el control del led, etc. La ejecución de varios programas, incrementando su complejidad y duración, ha servido para validar el funcionamiento completo del sistema. A continuación se detalla la ejecución del programa creado para la aplicación final.

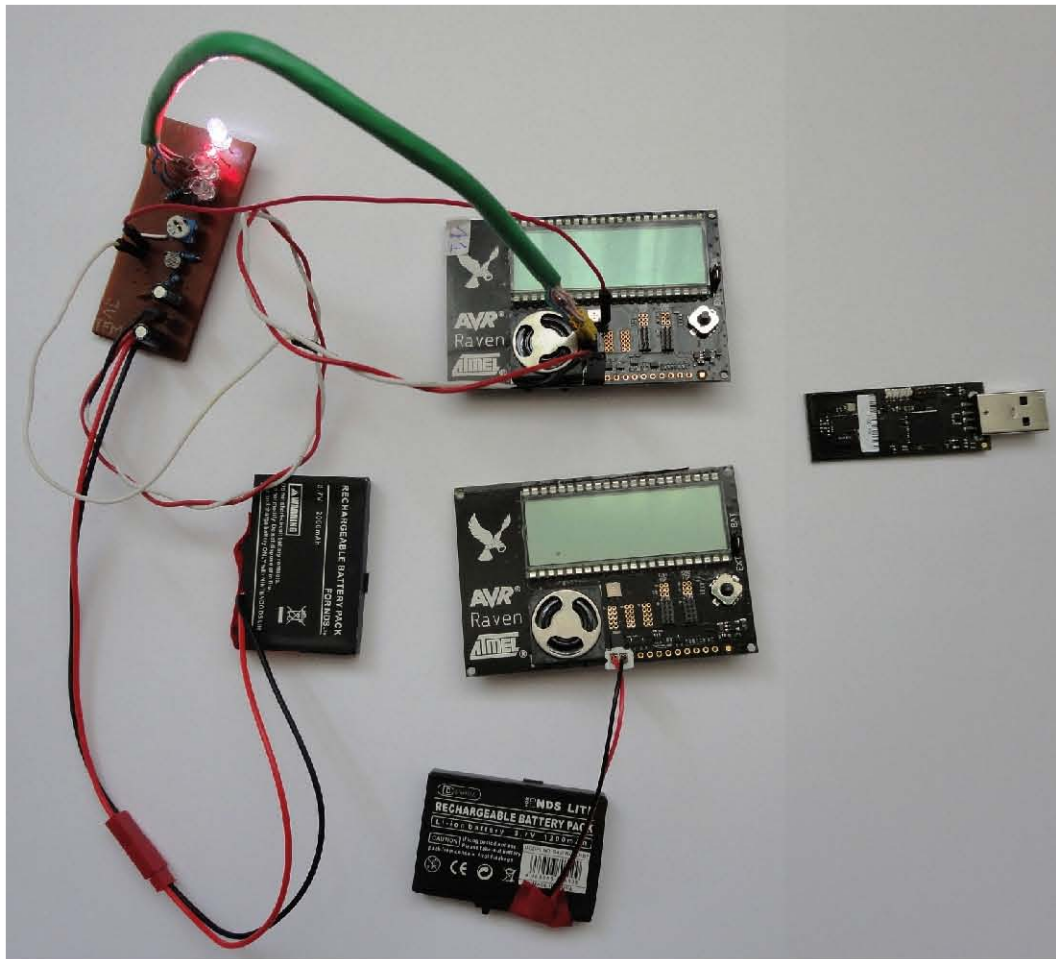


Figura 5.1: Sistema Implementado



Figura 5.2: JTAGICE mkII

5.1. Verificación del Sistema

El funcionamiento del sistema se ha probado en el laboratorio de Instrumentación Electrónica ubicado en el Pabellón A del Edificio de Electrónica y Telecomunicación de la ULPGC. El sistema final se compone de dos placas AVRRAVEN como nodos finales, una de ellas con la placa auxiliar, y del dispositivo RZUSBSTICK conectado al PC en modo coordinador. El primero paso es conectar el stick RZUSBSTICK y conocer el puerto USB-COM asignado. Para ello, se utiliza el Administrador de Dispositivos de Windows. En la rama del árbol ‘Puertos (COM y LPT)’ aparece el dispositivo AT90USB. En esta ocasión tiene el puerto COM 21. Con este dato, se procede a ejecutar el visor desarrollado en LabView. Se selecciona de la lista el puerto COM21 y se pulsa sobre el botón conectar. Si todo es correcto, el LED verde ‘Conectado’ se enciende, como se muestra en la Figura 5.3.

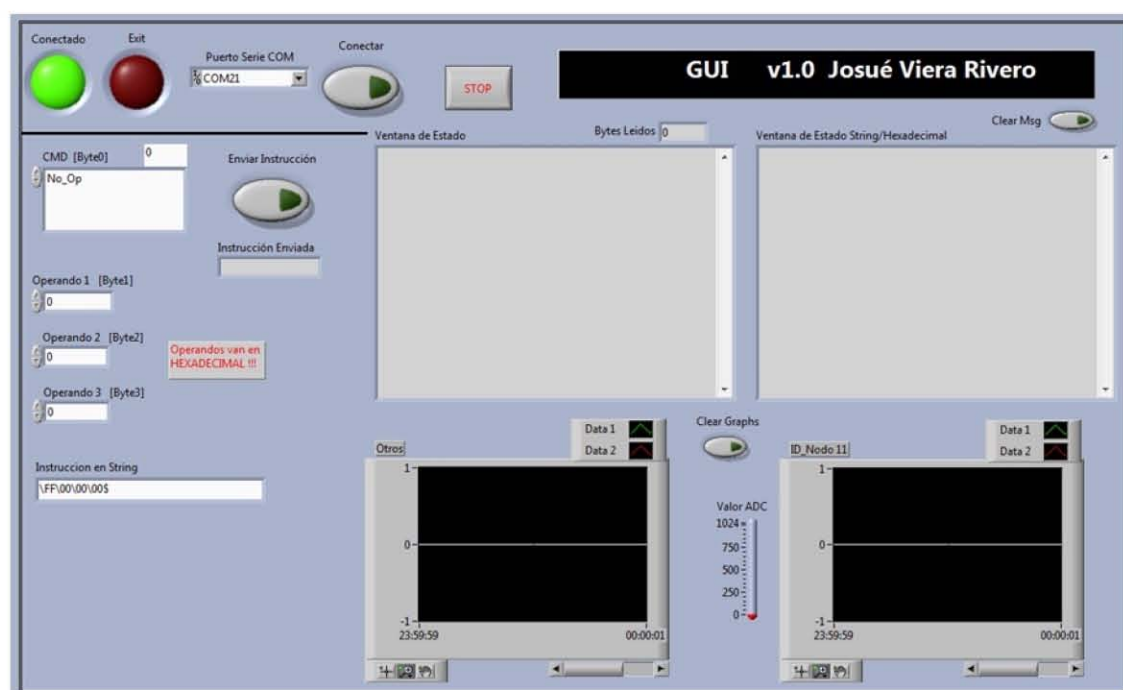


Figura 5.3: Visor Conectado

Cuando el Coordinador RZUSBSTICK tenga los dos primeros LEDs encendidos, la red PAN estará correctamente establecida y se permite la asociación de los nodos.

Encendido del Primer Nodo

El siguiente paso es alimentar el primer nodo que tiene conectada la placa auxiliar. Su primer LED se encenderá mientras que el segundo parpadeará. Cuando el primer nodo consiga asociarse al coordinador, el segundo LED permanecerá fijo y se iniciará el proceso de enviar la instrucción ‘P’ cada 10 segundos. Cada vez que se reciba una trama, el coordinador enciende su LED verde, como se muestra en la Figura 5.4.

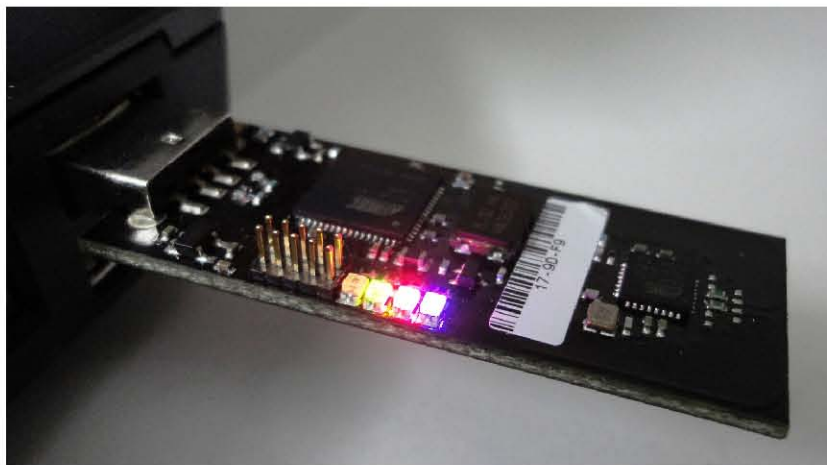


Figura 5.4: RZUSBSTICK Recibiendo Datos

Al mismo tiempo, como se puede observar en la Figura 5.5, el visor recoge el mensaje del coordinador de un nuevo nodo asociado y los datos recibidos. Al ser el primer nodo y tener el *ID_Nodo* al valor ‘11’, se actualiza su correspondiente gráfica.

Encendido del Segundo Nodo

Después de un tiempo, se procede a alimentar el segundo nodo. Éste realizará el mismo proceso que el primero, asociándose y enviando sus datos cada diez segundos. En el visor (Figura 5.6) se pueden apreciar los mensajes de estado del coordinador, los mensajes de los nodos y las gráficas con los datos.

Envío de Instrucciones a los Nodos

A continuación, se procede a enviar un mensaje desde el coordinador a uno de los nodos. Para ello, se selecciona de la lista de instrucciones ‘*Encender LED + Dirección*’. La dirección de destino hay que introducirla en la casilla ‘Operando 1’. Para comprobar

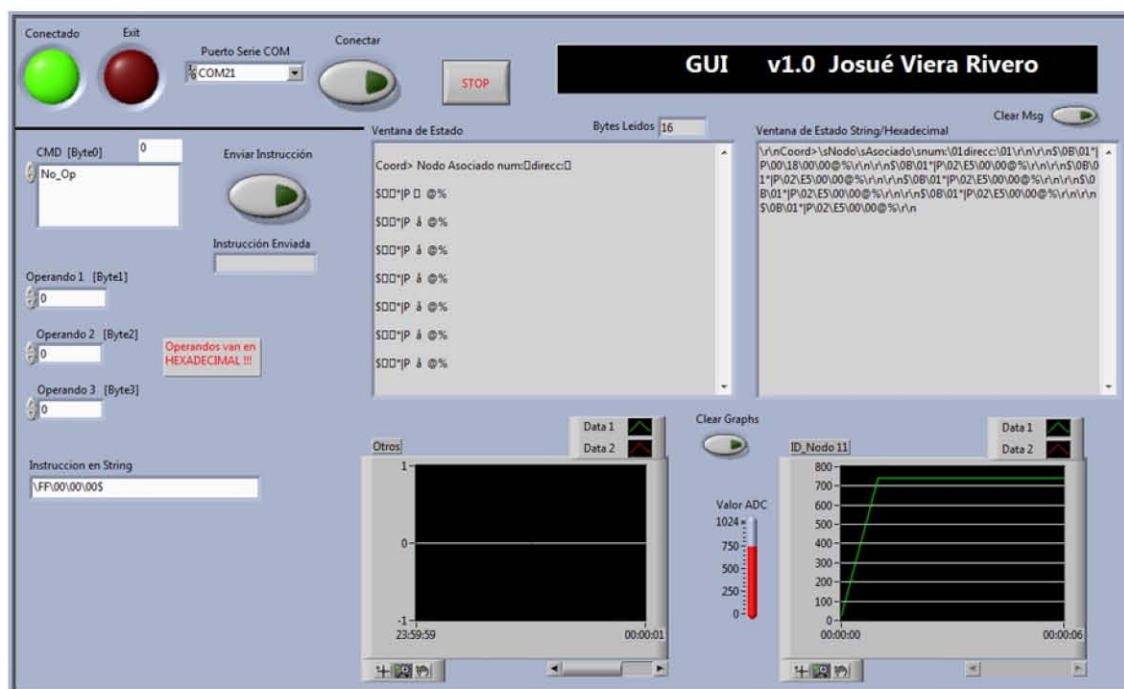


Figura 5.5: Nodo 1 Enviando Datos

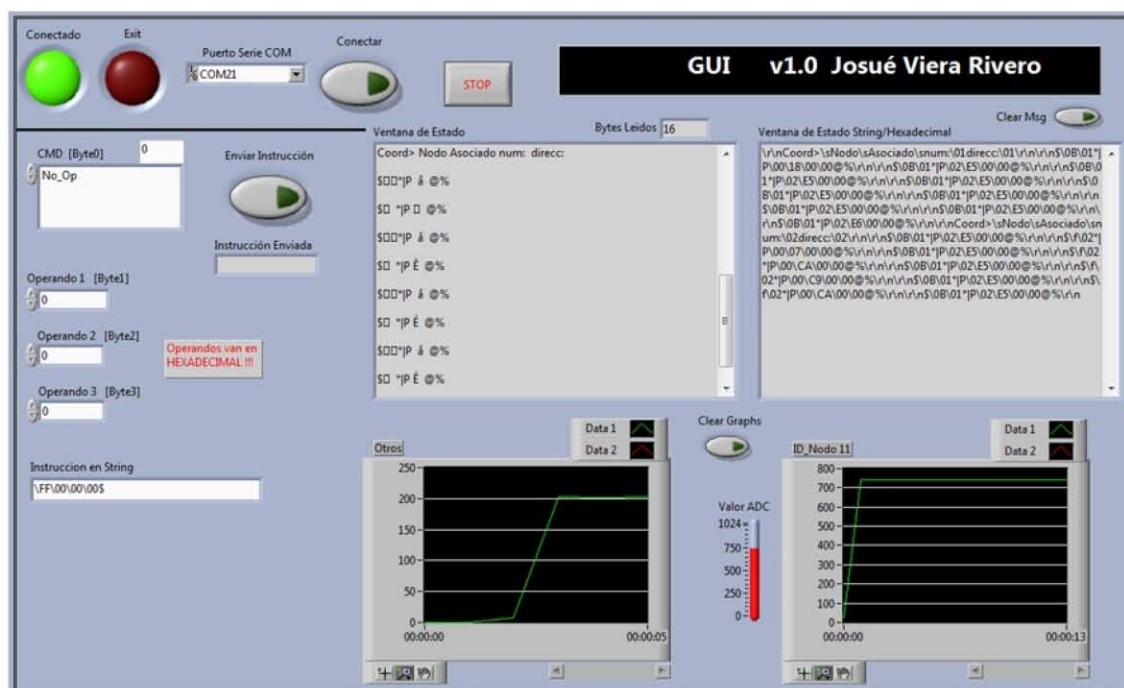


Figura 5.6: Nodos 1 y 2 Enviando Datos

cómo al realizar un envío a un nodo inexistente falla la comunicación, se introduce, por ejemplo, la dirección 7. Al pulsar en ‘Enviar Instrucción’, en la ventana de estado (Figura 5.7) se observa sólo ‘Coord> Enviando datos a (7)’. Al no recibirse un mensaje de confirmación ACK por parte del nodo, no se ha invocado la primitiva *MCPS-DATA.confirm* y por lo tanto, no se imprime el mensaje de mensaje “ACK_OK”.

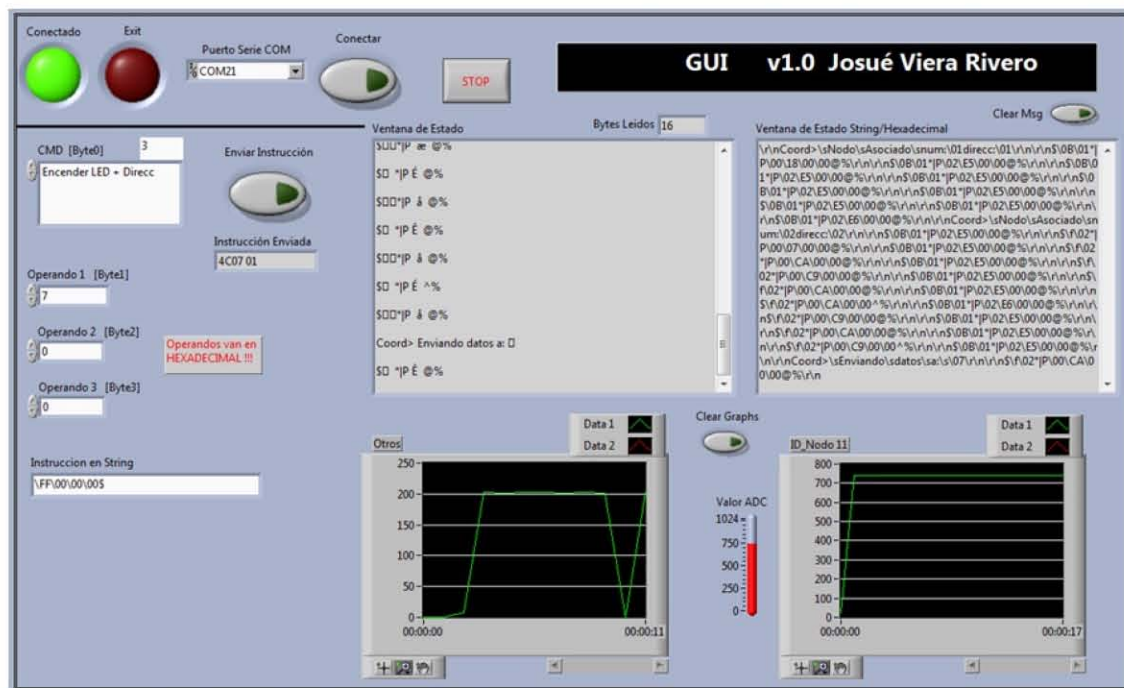


Figura 5.7: Envío Fallido al Nodo 7

Si se intenta la misma acción, pero con destinatario al nodo 1, el resultado es distinto. Al realizarse correctamente el envío, el nodo responde con un ACK. Por ello, se ejecuta la función *wpan_mcps_data_conf()* y se muestra en la ventana de estado ‘Coord> ACK_OK’ (véase la Figura 5.8). Al recibir la instrucción, el nodo inmediatamente enciende el LED en la placa auxiliar.

Para anular la orden anterior, se envía su antónima, ‘Apagar LED + Dirección’ al nodo 1. El resultado es el apagado del LED en la placa auxiliar y los mensajes que se muestran en la ventana de estado mostrada en la Figura 5.9. Durante todo el transcurso del proceso de verificación, los nodos siguen enviando correctamente datos al coordinador.

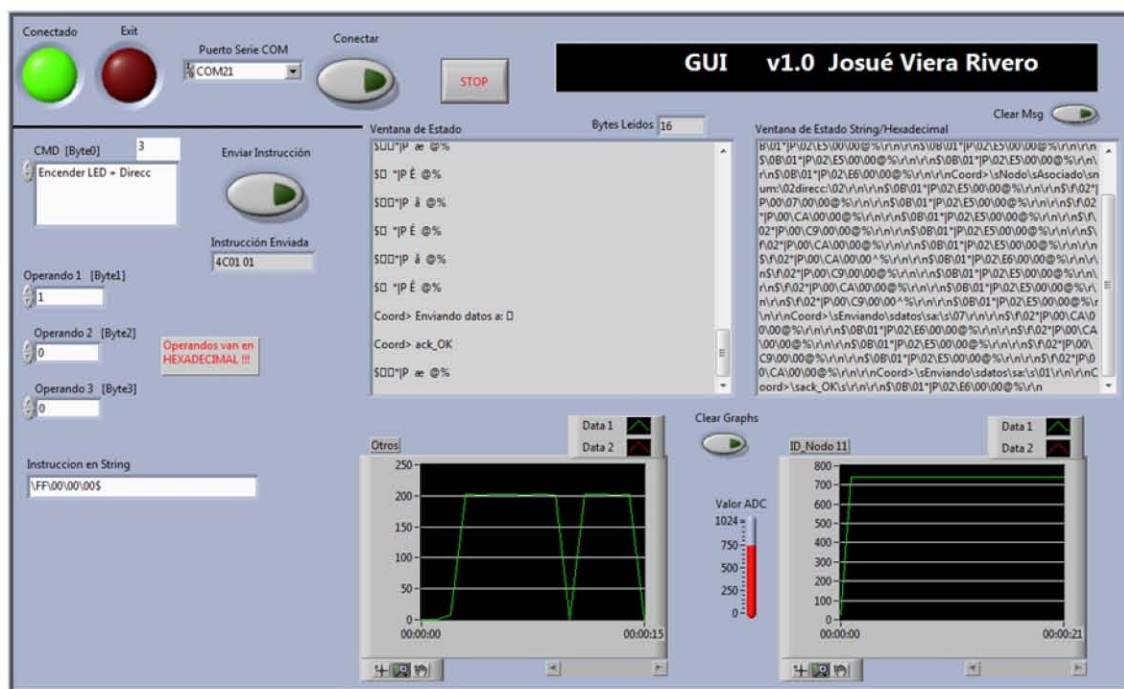


Figura 5.8: Envío Instrucción 'Encender LED' al Nodo 1

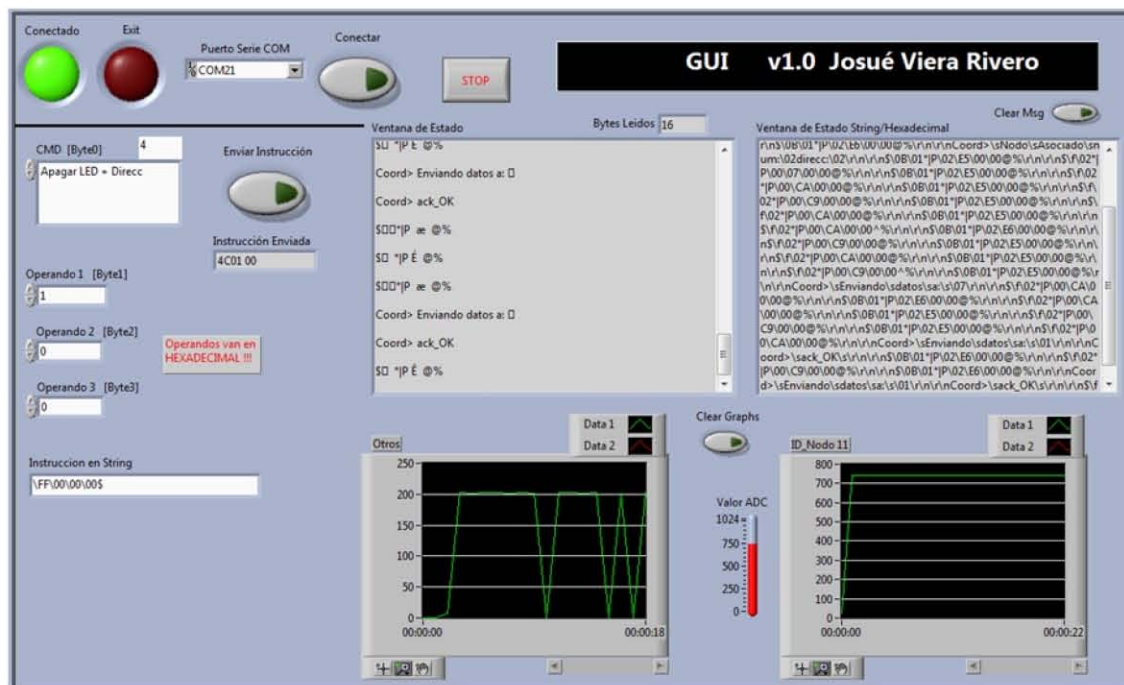


Figura 5.9: Envío Instrucción 'Apagar LED' al Nodo 1

5.2. Análisis de Tramas

Para realizar un estudio más detallado se va a utilizar un capturador de tramas o ‘*sniffer*’. Este dispositivo permite capturar y mostrar las tramas que se envían durante las comunicaciones. Para ello, en un dispositivo RZUSBSTICK, se procede a cargar la aplicación ‘*Promiscuous Mode Demo*’ que proporciona la pila ATMEL–MAC. Este ejemplo implementa el modo *promiscuo*, que al no realizar la subcapa MAC ningún filtro y por lo tanto no desecha ninguna trama, conlleva que se pasen todas a la capa aplicación.

5.2.1. Formato Trama MAC

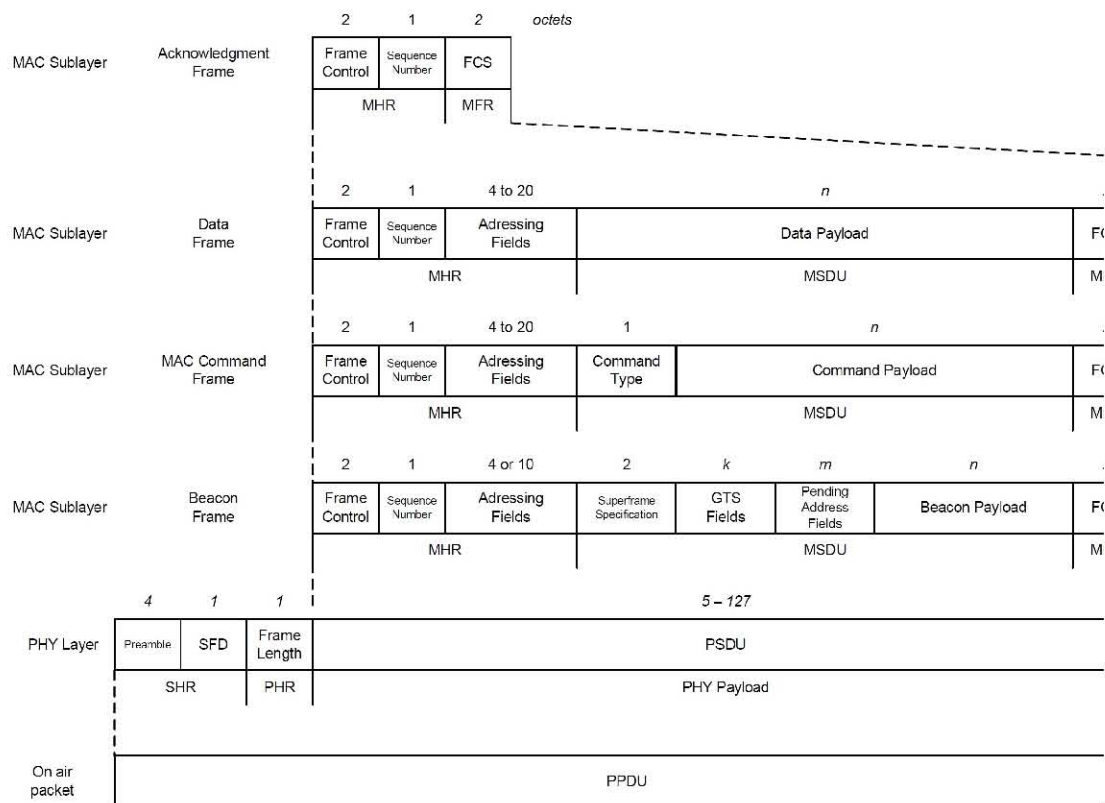


Figura 5.10: Tipos de Trama MAC

Antes de analizar las tramas, hay que conocer los tipos y campos que están definidos en el estándar 802.15.4 [9]. Como se puede observar en la Figura 5.10, la cabecera (*MAC header* o *MHR*) contiene el campo ‘*Frame Control*’, el número de secuencia, y el campo de direcciones. *MAC payload* contiene el tipo de comando y su payload. El final (*MFR*)

siempre se compone de 2 bytes de *checksum* para la detección y corrección de errores mediante el siguiente polinomio:

$$G_{16}(x) = x^{16} + x^{12} + x^5 + 1$$

Bits 0–2	3	4	5	6	7–9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	ACK Request	PAN ID Compr.	Reserv.	Dest. Addr. Mode	Frame Version	Source Addr. Mode

Tabla 5.1: Formato de *Frame Control*

El campo ‘*Frame Control*’ consta de 2 bytes que contienen varios campos, como se observa en la Tabla 5.1. Uno de los más importantes es el primero, que identifica el tipo de trama, existiendo cuatro como se muestra en la Tabla 5.2.

Frame type value b2 b1 b0	Descripción
000	Beacon
001	Data
010	Acknowledgment
011	MAC command
100–111	Reserved

Tabla 5.2: Tipos de Trama MAC

5.2.2. Capturas del Sniffer

Una vez explicados brevemente los tipos de trama, se procede al análisis de las tramas de la verificación. Para esta prueba, se realizarán los mismo pasos anteriores, con las dos placas AVRRAVEN y un dispositivo RZUSBSTICK actuando como Coordinador. Para este apartado, se dispone conectado otro dispositivo RZUSBSTICK en modo *promiscuo* capturando los paquetes. Después de conectarlo, es necesario abrir un programa terminal, utilizándose en este TFM el software ‘*Hercules SETUP utility*’.

Una vez abierto *Hercules* y, al pulsar cualquier tecla, aparece el menú que se muestra en la Figura 5.11. Se selecciona el canal 20 correspondiente a la red implementada y el ‘*sniffer*’ empezará a capturar los paquetes, formateando visualmente las tramas para mostrarlas correctamente por la ventana del terminal.

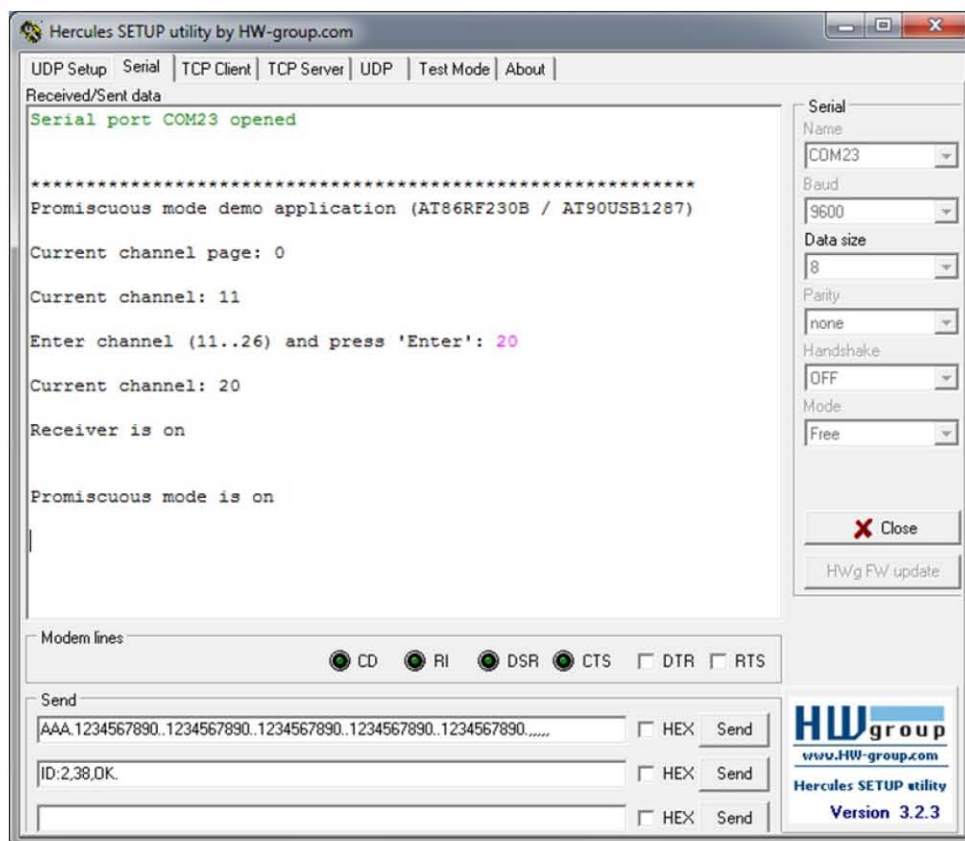


Figura 5.11: Terminal en Modo *Promiscuo*

Inicialización del Coordinador

El primer paso es conectar el dispositivo RZUSBSTICK Coordinador. Una vez que se ha inicializado correctamente, provoca el envío de una trama similar a la siguiente:

No. 1 Cmd: 03 08 C0 FF FF FF FF 07 4B 2E

Destacar que está ordenado en *little endian*, correspondiéndose los dos primeros bytes a : 0000 1000 0000 0011. Con esto se pueden identificar los tres bits de menor peso correspondientes a una trama de tipo *comando*, y el bit 12 (junto al 13) correspondiente con la versión 2006 del estándar. El resto de bits están a cero, el siguiente byte es el

número de secuencia, seguido por 4 bytes de dirección que indican que el identificador del coordinador es de *broadcast* (0xFFFF), y la propia dirección de 16 bits de *broadcast* (0xFFFF). Los dos últimos bytes se corresponden con FCS. Queda por analizar el byte restante, o tipo de comando. Observando la Figura 5.12, el valor de éste coincide con ‘*Beacon Request*’, indicando que el coordinador está realizando un escáner de tipo activo asociado a la directiva *MCPS-SCAN.request*. Como no recibe respuesta de nadie, continuará la ejecución del código e iniciará su propia PAN.

Table 82—MAC command frames

Command frame identifier	Command name	RFD		Subclause
		Tx	Rx	
0x01	Association request	X		7.3.1
0x02	Association response		X	7.3.2
0x03	Disassociation notification	X	X	7.3.3
0x04	Data request	X		7.3.4
0x05	PAN ID conflict notification	X		7.3.5
0x06	Orphan notification	X		7.3.6
0x07	Beacon request			7.3.7
0x08	Coordinator realignment		X	7.3.8
0x09	GIS request			7.3.9
0x0a–0xff	Reserved			—

Figura 5.12: Tipos de Comando

Inicialización del Nodo en Ausencia de Coordinador

Desconectando el Coordinador, y alimentando la primera placa AVRRAVEN, se observa que cada 15 segundos el nodo final envía las siguientes tramas:

No. 2 Cmd: 03 08 52 FF FF FF FF 07 0F 61

No. 3 Cmd: 03 08 62 FF FF FF FF 07 DF A6

No. 4 Cmd: 03 08 72 FF FF FF FF 07 6F E4

No. 5 Cmd: 03 08 82 FF FF FF FF 07 CC 24

En esta ocasión, las tramas siguen siendo de control y de ‘*Beacon Request*’, ya que el nodo también está realizando un escáner activo para localizar coordinadores. De

la misma manera, no recibe respuesta, por lo que seguirá repitiendo el escáner como está contemplado en el código de la función *usr_mlme_scan_req()* del Nodo Final.

Inicialización Correcta de la Red

Si se empieza de nuevo, conectando primero el Coordinador, y al cabo de unos segundos, el primer nodo, la respuesta es distinta. En esta ocasión, el Coordinador sí responderá al '*Beacon Request*' del nodo, generando un *Beacon* (bits de tipo de trama están a 000b).

No. 1 Cmd: 03 08 97 FF FF FF FF 07 FB 72

No. 2 Cmd: 03 08 4C FF FF FF FF 07 1D 1A

No. 3 Beacon: 00 80 60 BE BA 2A 00 FF CF 00 00 2A 35

Consultando el estándar, los datos más interesantes del *Beacon* son, el identificador PAN (0xBABE), la propia dirección corta que tiene el coordinador (0x002A), y que la red es de tipo *NonBeacon*, ya que no se especifica la supertrama de baliza. Continuando con la comunicación, el coordinador enviará una serie de comandos que son recibidos y confirmados por el nodo mediante ACKs.

No. 4 Cmd: 23 C8 53 BE BA 2A 00 FF FF A3 5B 4B 5F 1E 07 E9 4A 01 80 E2 F9

No. 5 Ack: 02 00 53 A6 D5

No. 6 Cmd: 63 C8 54 BE BA 2A 00 A3 5B 4B 5F 1E 07 E9 4A 04 0B FA

No. 7 Ack: 12 00 54 8C 24

No. 8 Cmd: 63 CC 9E BE BA A3 5B 4B 5F 1E 07 E9 4A 1D 66 E3 2F 75 0B CB 24
02 01 00 00 06 1C

No. 9 Ack: 02 00 9E 4F C8

Se observa que los ACKs siempre tienen el número de secuencia igual a la trama que confirman. El último comando de datos envía la dirección asignada al nodo, que se corresponde con la 0x0001. A partir de este punto, no se van a analizar bit a bit las tramas del test, ya que es una tarea trivial coger el estándar y analizar los bits de las tramas. Simplemente se expondrá lo necesario para ver el estado de la red y el flujo de comunicaciones.

Envío de Datos

Llegados a este punto, la asociación ha sido correcta y el primer nodo tiene la dirección corta 0x0001. Por lo tanto, empezará a enviar datos cada 10 segundos al coordinador. Esto se observa en la secuencia de tramas capturadas:

No. 10 Data: 61 88 55 BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 C2 00 00 40
25 F8 0E

No. 11 Ack: 02 00 55 90 B0

No. 12 Data: 61 88 56 BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 94 00 00 40
25 C8 FD

No. 13 Ack: 02 00 56 0B 82

No. 14 Data: 61 88 57 BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 BD 00 00 40
25 74 5F

No. 15 Ack: 02 00 57 82 93

No. 16 Data: 61 88 58 BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 AD 00 00 40
25 F4 B1

No. 17 Ack: 02 00 58 75 6B

En ellas, se observa el campo de tipo de trama es de datos (001b), el Identificador PAN (0xBABE), y la dirección de destino (0x002A), siendo la de origen la 0x0001. El resto de campos se pueden identificar con la Figura 5.10, pero se observa que el *payload* sigue el formato utilizado para las instrucciones de comunicaciones (consultar la Tabla 4.4) con el delimitador de inicio 0x24 = \$ y el de fin 0x25 = %.

24 0B 01 2A 7C 50 02 AD 00 00 40 25

Realizando un extracto de los datos de la instrucción, queda como se muestra en la Tabla 5.3.

Asociación del Segundo Nodo

Al realizar la puesta en marcha del segundo nodo, el procedimiento es muy similar al explicado anteriormente, con la respuesta del *Beacon* y la asignación de la dirección si la asociación es correcta:

Byte 0	Byte 1	Byte 2	Byte 3	...
Inicio de Instrucción (\$)	ID_Nodo_Origen	Dirección_Origen	Dirección_Destino	...
0x24	0x0B = 11	0x01	0x2A	...
Byte 4	Byte 5	Byte 6	Byte 7 ...	Byte 11
Separador ()	Comando	Parámetro 1	Parámetro 2 ...	Fin de Instrucción (%)
0x7C	0x50 = 'P'	0x02	0xAD ...	0x25

Tabla 5.3: Datos Enviados por el Primer Nodo

```

No. 28  Cmd: 03 08 89 FF FF FF FF 07 E9 09
No. 29  Beacon: 00 80 61 BE BA 2A 00 FF CF 00 00 D7 78
No. 30  Cmd: 23 C8 90 BE BA 2A 00 FF FF 4F 7A 7D 78 95 2F B1 3D 01 80 B1 EB
No. 31  Ack: 02 00 90 31 21
No. 32  Cmd: 63 C8 91 BE BA 2A 00 4F 7A 7D 78 95 2F B1 3D 04 ED 58
No. 33  Ack: 12 00 91 2D B5
No. 34  Cmd: 63 CC 9F BE BA 4F 7A 7D 78 95 2F B1 3D 1D 66 E3 2F 75 0B CB
24 02 02 00 00 A1 4A
No. 35  Ack: 02 00 9F C6 D9

```

Ahora el segundo nodo tiene la dirección corta 0x0002 y empieza a enviar datos cada 10 segundos al Coordinador. En las siguientes tramas se puede observar cómo se intercalan las tramas de los nodos con las direcciones 0x0001 y 0x0002, con destino al Coordinador:

```

No. 36  Data: 61 88 5E BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 E2 00 00 40
25 AF 69
No. 37  Ack: 02 00 5E 43 0E
No. 38  Data: 61 88 92 BE BA 2A 00 02 00 24 0C 02 2A 7C 50 00 08 00 00 40
25 42 B6
No. 39  Ack: 02 00 92 23 02
No. 40  Data: 61 88 5F BE BA 2A 00 01 00 24 0B 01 2A 7C 50 02 F7 00 00 40
25 F2 68
No. 41  Ack: 02 00 5F CA 1F

```

No. 42 Data: 61 88 93 BE BA 2A 00 02 00 24 0C 02 2A 7C 50 00 C8 00 00 40
25 7C 6E

No. 43 Ack: 02 00 93 AA 13

Envío de Instrucciones a los Nodos

Finalmente, se realiza la comunicación en el sentido contrario, enviando una instrucción desde el Coordinador a un nodo. El test inicialmente intentaba enviar la instrucción “*Encender Led*” a la dirección inexistente 7. Como se comentó anteriormente, no se recibió ningún ACK. Lo que no se podía observar en el visor eran los reintentos realizados por la capa MAC, como está definido en el atributo *macMaxFrameRetries*, que tiene un valor predeterminado de ‘3’. El envío y las tres tramas de reintento se muestran a continuación:

No. 92 Data: 61 88 A1 BE BA 07 00 2A 00 24 05 2A 07 7C 4C 01 00 00 00 01
25 52 AB

No. 93 Data: 61 88 A1 BE BA 07 00 2A 00 24 05 2A 07 7C 4C 01 00 00 00 01
25 52 AB

No. 94 Data: 61 88 A1 BE BA 07 00 2A 00 24 05 2A 07 7C 4C 01 00 00 00 01
25 52 AB

No. 95 Data: 61 88 A1 BE BA 07 00 2A 00 24 05 2A 07 7C 4C 01 00 00 00 01
25 52 AB

En cambio, cuando se envió el mensaje “*Encender Led*” a la dirección 1, si se consiguió una recepción afirmativa. Efectivamente, a continuación se observa la respuesta de confirmación por parte del nodo :

No. 114 Data: 61 88 A2 BE BA 01 00 2A 00 24 05 2A 01 7C 4C 01 00 00
00 01 25 D4 8E

No. 115 Ack: 02 00 A2 A0 33

El último grupo de tramas analizadas se corresponden con el envío del mensaje “*Apagar Led*” a la dirección 1, que de nuevo se confirma mediante la recepción de un ACK. La diferencia de la trama de datos con la anterior está en el número de secuencia, el FCS y en el *payload*, ya que ahora este parámetro en vez de ser ‘1’, es nulo.

No. 122 Data: 61 88 A3 BE BA 01 00 2A 00 24 05 2A 01 7C 4C 00 00 00
00 01 25 B6 19

No. 123 Ack: 02 00 A3 29 22

Capítulo 6

Conclusiones y Líneas Futuras

6.1. Conclusiones

Este Trabajo Fin de Máster ha sido un gran desafío. Se han tenido que abordar múltiples tareas que han supuesto un incremento en las capacidades, conocimientos y práctica que conlleva un proyecto de este calibre. El estudio del estándar IEEE 802.15.4 ha permitido conocer la estructura, la nomenclatura, la definición de las primitivas, y las funciones asociadas al estándar 802.15.4. De esta manera, se ha adquirido el conocimiento y comprensión del estándar en profundidad, además de poder realizar en un futuro la lectura de otros estándares con una menor dificultad.

Este aprendizaje ha sido apoyado y reforzado debido a la utilización práctica de una pila desarrollada por Atmel y el kit de desarrollo RZRAVEN. La pila da soporte a las primitivas de la capa MAC del estándar y al estar estructurada en varias capas de aplicación, permite la portabilidad y la modificación de las configuraciones de varias plataformas, sin afectar al código principal. Con respecto al kit, su diseño proporciona una gran libertad al usuario al brindar los puertos de expansión y una puesta en funcionamiento muy rápida. Esto permite utilizar los sensores que sean necesarios para una aplicación determinada, además de un tiempo corto de desarrollo y de pruebas. Esta versatilidad posibilita un gran abanico de aplicaciones y de usuarios.

Por otra parte, el visor desarrollado permite el envío de comandos y la representación en tiempo real de los datos recibidos. Este Visor, o Interfaz Gráfica de Usuario, es visualmente atractivo, de fácil lectura y utilización gracias al menú de instrucciones

y los botones incluidos. Las ventanas de estado permiten conocer en cada momento el estado de la red y los mensajes recibidos. Las gráficas reflejan al mismo tiempo, los datos de manera clara y directa. Estas dos secciones permiten un análisis inmediato de los datos y la detección de problemas, posibilitando la gestión y toma de decisiones de manera rápida y eficaz.

Finalmente, el estudio de las tramas proporcionó un mayor detalle sobre el intercambio de mensajes entre los dispositivos. Esto permitió verificar el correcto funcionamiento de la red y que las directrices del estándar se cumplieran.

Con este conjunto, se ha conseguido comprender la puesta en marcha de una red de sensores. La implementación de primitivas adicionales amplía la funcionalidad básica de la pila, y con la incorporación de la comunicación USB, se consigue un sistema con gran potencial. El sistema es fácilmente ampliable, pudiéndose aumentar la funcionalidad mediante la incorporación de nuevas instrucciones o sensores adicionales. Esto permite adaptarlo a un gran número de aplicaciones prácticas. Un ejemplo es la automatización del almacenamiento de los datos recibidos en una base de datos, para su disponibilidad y uso en GIS. La implementación del sistema cumple los objetivos planteados inicialmente en este Trabajo Fin de Máster.

Las redes de sensores están en fase de desarrollo, aportando en la actualidad grandes ventajas, sin acabar de mostrar todo su potencial. Además, el rango de aplicaciones es muy amplio y cada día aparecen nuevos campos de investigación gracias a las redes de sensores. Es por esta razón que su implantación debe generalizarse, ya que su mayor uso en líneas de investigación, ya sea en el ámbito universitario, en pequeños grupos de investigación o en empresas privadas, permitirá lograr más avances y mejoras en este campo.

6.2. Líneas Futuras

El presente Trabajo Fin de Máster consistió en el desarrollo de un sistema para aplicaciones con sensores. Debido a la naturaleza de un TFM, las tareas están acotadas en el tiempo para garantizar la calidad y finalización del mismo. Por ello, se plantean algunas líneas de actuación a abordar en el futuro:

- Aumentar la funcionalidad del código añadiendo directivas adicionales.

- Incrementar el número de instrucciones de los nodos finales.
- Enviar datos entre nodos y utilizar los datos para cambiar el comportamiento.
- Diseñar y fabricar un prototipo compatible con la pila ATMEL–MAC.
- Realizar un despliegue a mayor escala.

Bibliografía

Bibliografía

- [1] J. K. Hart and K. Martinez, “Environmental Sensor Networks: A revolution in the earth system science?,” *Earth-Science Reviews*, vol. 78, pp. 177–191, 2006.
- [2] R. Shorey *et al.*, *Mobile, Wireless, and Sensor Networks: Technology, Applications, and Future Directions*. Wiley-IEEE Press, 2006.
- [3] J. A. Stankovic *et al.*, “Wireless Sensor Networks for In-Home Healthcare: Potential And Challenges,” in *High Confidence Medical Device Software and Systems (HCMDSS) Workshop*, 2005.
- [4] W. T. R. Solutions, “The WTRS Wireless Sensor Network Technology Trends,” *LLC*, p. 97, 2009.
- [5] S. Rackley, *Wireless Networking Technology: From Principles to Successful Implementation*. Newnes, 2007.
- [6] Wireless Sensor Network, [Online].
http://en.wikipedia.org/wiki/Wireless_sensor_network. Last visit: 06/2011.
- [7] M. Kuorilehto *et al.*, *Ultra-Low Energy Wireless Sensor Networks in Practice: Theory, Realization and Deployment*. Wiley, 2008.
- [8] M. Ilyas and I. Mahgoub, *Handbook of sensor networks: compact wireless and wired sensing systems*. CRC Press, 2004.
- [9] IEEE Computer Science, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs),” *IEEE 802.15.4-2006*, 2006.

- [10] IEEE Computer Science, “Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs), Amendment,” *IEEE 802.15.4-2007*, 2007.
- [11] Atmel, [Online]. <http://www.atmel.com>. Last visit: 06/2011.
- [12] N. Wang, N. Zhang, and M. Wang, “Wireless sensors in agriculture and food industry - Recent development and future perspective,” *Computers and Electronics in Agriculture*, vol. 50, no. 1, pp. 1–14, 2006.
- [13] P. Barontib *et al.*, “Wireless Sensor Networks: A Survey on the State of the Art and the 802.15.4 and ZigBee Standards,” *Computer Communications*, vol. 30, no. 7, pp. 1655–1695, 2007.
- [14] IEEE Computer Science, “Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,” *IEEE Std 802.11-2007*, 2007.
- [15] J. Leonard, “How to compare proprietary and standards-based radio solutions,” *EETimes*, 2006.
- [16] ZigBee Alliance, “ZigBee Specification-v1.0,” 2004.
- [17] ZigBee Alliance, “ZigBee Specification-2006,” 2006.
- [18] ZigBee Alliance, “ZigBee Specification-2007,” 2007.
- [19] P. Levis, J. Polastre, J. Hill, and D. Culler, “TinyOS: An operating system for sensor networks,” in *Ambient Intelligence*, Springer Verlag, 2004.
- [20] J. Hill, “System Architecture for Wireless Sensor Networks,” 2003.
- [21] J. Hill and D. Culler, “MICA: A wireless platform for deeply embedded networks,” vol. 22, pp. 12–24, 2002.
- [22] J. Hill, D. Culler, and M. Horton, “MICA, The Commercialization of Microsensor Motes,” vol. 19, pp. 40–48, 2002.
- [23] CrossBow Tecnology, “Datasheet Mica, Wireless Measurement System.”

- [24] Libelium: Waspote, [Online]. <http://www.libelium.com/products/waspote>. Last visit: 06/2011.
- [25] Microchip, [Online]. <http://www.microchip.com>. Last visit: 06/2011.
- [26] E. Howell and D. Cecic, “Microchip Corporate Summary,” 2007.
- [27] Microchip ZigBee, [Online]. <http://www.microchip.com/ZigBee>. Last visit: 06/2011.
- [28] Microchip MiWi, [Online]. <http://www.microchip.com/MiWi>. Last visit: 06/2011.
- [29] Microchip, “Datasheet AN1066 Microchip MiWi Wireless Networking Protocol Stack.”
- [30] Microchip MRF24J40, [Online].
<http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en535967>. Last visit: 06/2011.
- [31] Wikipedia, [Online]. <http://www.wikipedia.com>. Last visit: 06/2011.
- [32] Atmel, “Datasheet and Firmware Atmel IEEE 802.15.4 MAC Stack.”
- [33] Atmel, “Datasheet AVR2052: BitCloud Quick Start Guide.”
- [34] Atmel, “Datasheet and Firmware AVR2015: RZRAVEN Quick Start Guide.”
- [35] Atmel, “Datasheet AVR2025: IEEE 802.15.4 MAC Software Package - User Guide.”
- [36] Atmel, “Datasheet AVR2016: RZRAVEN Hardware User’s Guide.”
- [37] Atmel, “Datasheet Atmel ATmega1284P.”
- [38] National Instruments, [Online]. <http://www.ni.com>. Last visit: 06/2011.
- [39] KEE AVR JTAGICE mkII, [Online]. <http://keeelectronics.com/mkii.htm>. Last visit: 06/2011.
- [40] R.H. Barnett, L. O’Cull and S. Cox, *Embedded C programming and the Atmel AVR*. Thomson Delmar Learning, 2006.

