UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UN SENSOR INALÁMBRICO CON MODULACIÓN DE ESPECTRO ENSANCHADO PARA COMUNICACIÓN DE LARGO ALCANCE

DANIEL GALA MONTES

2018

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

Reunido el tribunal examinador en el día de la fecha, constituido por

Presidente:	Dr D.	PRESIDENTE
Vocal:	Dr D.	VOCAL
Secretario:	D.	SECRETARIO
Suplente:	Dr D.	SUPLENTE

para juzgar el Trabajo Fin de Máster titulado:

DISEÑO E IMPLEMENTACIÓN DE UN SENSOR INALÁMBRICO CON MODULACIÓN DE ESPECTRO ENSANCHADO PARA COMUNICACIÓN DE LARGO ALCANCE

del alumno D. DANIEL GALA MONTES
dirigido por D. ÁLVARO GUTIÉRREZ MARTÍN

Acuerdan otorgar la calificación de: _____

Y, para que conste, se extiende firmada por los componentes del tribunal, la presente diligencia

Madrid, a 15 de Julio de 2018

El Presidente	El Vocal	El Secretario
Fdo: _____	Fdo: _____	Fdo: _____

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE SISTEMAS ELECTRÓNICOS

TRABAJO FIN DE MÁSTER

DISEÑO E IMPLEMENTACIÓN DE UN SENSOR INALÁMBRICO CON MODULACIÓN DE ESPECTRO ENSANCHADO PARA COMUNICACIÓN DE LARGO ALCANCE

DANIEL GALA MONTES

2018

Resumen

El propósito de este Trabajo Fin de Máster consiste en el diseño e implementación de un sistema empotrado autónomo, capaz de monitorizar variables físicas del entorno y de transmitir los datos obtenidos de forma inalámbrica. Para lograr este objetivo, se ha desarrollado un prototipo empleando un diseño basado en plataformas que ha permitido reducir los riesgos y costes de desarrollo. Se ha diseñado y fabricado una tarjeta electrónica de expansión, incluyendo los componentes específicos de la aplicación necesarios para dotar de funcionalidad añadida a una tarjeta electrónica de control comercial. Por último, debido a la gran importancia del consumo energético en este sistema, los módulos software se han desarrollado siguiendo una estrategia de bajo consumo.

Abstract

The aim of this thesis is to design and implement an autonomous embedded system, capable of monitoring physical environmental variables and transmitting the obtained data wirelessly. In order to do so, a prototype has been developed following a platform-based design approach to save time and reduce development's risks and costs. An expansion board, including all the application specific components, has been designed and manufactured to add the required functionality to a commercial electronic control board. Lastly, power consumption plays a very important role in the system. Therefore, software modules have been implemented following a power saving strategy.

Palabras clave: sensor inalámbrico, autónomo, monitorización, bajo consumo, microcontroladores, STM32, LoRa, tiempo real.

Keywords: wireless sensor, autonomous, monitoring, low power, microcontrollers, STM32, LoRa, real-time.

Agradecimientos

Índice general

Índice de figuras

Índice de tablas

Lista de Acrónimos

ADC: Analog to Digital Converter

AHB: Advanced High-performance Bus

AOI: Automated Optical Inspection

APB: Advanced Peripheral Bus

ARM: Advanced RISC Machines

AVR: Advanced Virtual RISC

BOM: Bill Of Materials

BOR: Brown-Out Reset

CAD: Computer Aided Design

CAM: Computer Aided Manufacturing

CPU: Central Processing Unit

CRC: Cyclic Redundancy Check

DAC: Digital to Analog Converter

DMA: Direct Memory Access

DRC: Design Rules Check

DSP: Digital Signal Processor

DVS: Dynamic Voltage Scaling

ERC: Electrical Rules Check

ESD: ElectroStatic Discharge

FIFO: First In First Out

FPGA: Field Programmable Gate Array

FPU: Floating Point Unit

FSK: Frequency Shift Keying

FSM: Finite State Machine

GFSK: Gaussian Frequency Shift Keying

GPIO: General Purpose Input/Output

HAL: Hardware Abstraction Layer

IOT: Internet Of Things

IRQ: Interrupt Request

ISM: Industrial, Scientifical and Medical

I2C: Inter-Integrated Circuit

JTAG: Joint Test Action Group

LPWAN: Low-Power Wide Area Network

MEMS: MicroElectroMechanical System

MOS: Metal-Oxide-Semiconductor

MOSFET: MOS Field Effect Transistor

MPU: Memory Protection Unit

OCD: On-Chip Debugger

OOK: On-Off Keying

PCB: Printed Circuit Board

PDF: Portable Document Format

PGA: Programmable Gate Array

PLL: Phase-Locked Loop

PWM: Pulse Width Modulation

RISC: Reduced Instruction Set Computer

RTC: Real Time Clock

RTOS: Real Time Operative System

RTU: Remote Terminal Unit

SPI: Serial Peripheral Interface

SRAM: Static Random Access Memory

UART: Universal Asynchronous Receiver-Transmitter

USB: Universal Serial Bus

WSN: Wireless Sensor Networks

Capítulo 1

Introducción y Objetivos

En la actualidad, se está viviendo una nueva era de la información gracias a la proliferación de paradigmas de aprendizaje automático como el *Big Data*, que permite procesar cantidades masivas de datos con un coste computacional relativamente reducido. Esta revolución está generando un creciente interés a todos los niveles por la adquisición de datos de diversas naturalezas de manera eficiente, autónoma y conectada, dando lugar a nuevas tendencias en el mercado entre las que destaca el llamado "Internet de las Cosas".

La principal diferencia entre "el Internet" y "el Internet de las Cosas" (IoT) (Gubbi et al., 2013) es que, en IoT, un dispositivo dispone generalmente de recursos más reducidos: menor memoria, menor capacidad de procesamiento, menor ancho de banda y, por supuesto, menor energía disponible. Esto es debido a que los dispositivos IoT se alimentan a base de baterías, por lo que es prioritario maximizar su autonomía; o bien a que están destinados a una producción de escala (se estima que para el año 2020 habrá cincuenta mil millones de dispositivos conectados (Evans, 2011)), por lo que se busca minimizar costes.

En este contexto, son especialmente relevantes las redes de sensores inalámbricas ó sistemas WSN (*Wireless Sensor Networks*). Estos sistemas consisten en redes compuestas por numerosos nodos autónomos distribuidos espacialmente, que se sirven de sensores para obtener información de su entorno y, típicamente, disponen de radiotransceptores que emplean para transmitir dicha información de forma cooperativa a un servidor central. El diseño de estos nodos es, por tanto, un elemento clave en la implementación de sistemas WSN y será el objetivo fundamental de este Trabajo Fin de Máster.

1.1. Estado del arte en sistemas WSN

De la definición de los sistemas WSN se desprende lo siguiente: los elementos fundamentales de un sistema WSN son los nodos. Es decir, para describir con precisión una red inalámbrica de sensores, es imprescindible describir sus nodos y todos los aspectos que los conciernen: la topología de interconexión, los protocolos de comunicación, la dispersión espacial, la cantidad y la capacidad sensorial individual de cada nodo. El trabajo académico realizado a la hora de evaluar dichos aspectos y clasificar estos sistemas es extenso y está ampliamente documentado (Akyildiz et al., 2002). Por consiguiente, y puesto que este Trabajo Fin de Máster no es un trabajo teórico de carácter científico-técnico sino un proyecto de ingeniería sobre el diseño y la fabricación de un equipo, en esta sección se presentan únicamente aquellos aspectos que han tenido que ser considerados en el diseño del sistema desarrollado.

1.1.1. Topologías de red en sistemas WSN

En general, un sistema WSN puede estar compuesto de cientos o miles de sensores inalámbricos desatendidos y desplegados arbitrariamente, con una probabilidad no despreciable de fallo para un nodo dado. Esta situación implica la necesidad de arquitecturas de red variables y una cierta capacidad de autoorganización en los nodos, que permita incrementar la tolerancia a fallos del sistema, por lo que es necesario recurrir a topologías complejas de tipo malla híbrida, con paso de mensajes y agregadores de información. En contraposición con esta situación, también puede darse un sistema WSN con un número reducido de nodos, en el que la arquitectura de interconexión sea fija y conocida, pudiendo aplicarse topologías de red inalámbrica convencionales (ver Figura 1.1) asegurando que exista un mantenimiento regular de la topología para salvaguardar la fiabilidad del sistema de los fallos inevitables.

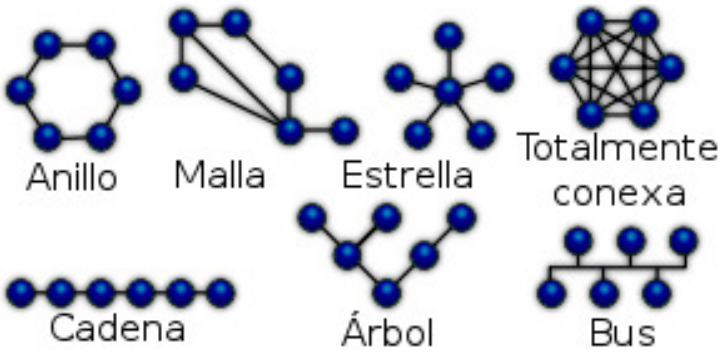

Figura 1.1: Topologías de red convencionales.

1.1.2. Comunicaciones inalámbricas en sistemas WSN

Hasta la fecha, se han estudiado e implementado numerosas tecnologías de comunicación para *IoT*. Como denominador común a todas ellas se encuentra su orientación a bajo consumo y comunicaciones inalámbricas, aunque mayormente se pueden agrupar (Augustin et al., 2016) en dos categorías:

- Redes de área local de baja potencia, con alcance menor a 1000m. Esta categoría incluye tecnologías como *Zigbee* o *BlueTooth/LE*, que son directamente aplicables en redes personales de área local o, si se organizan en topología de malla, también en áreas mayores.
- Redes LPWAN, con alcance mayor a 1000m. En esencia, versiones de bajo consumo de redes celulares, donde cada celda agrupa miles de dispositivos finales. Esta categoría incluye tecnologías como *LoRaWAN*.

A continuación se expone un resumen de las principales alternativas tecnológicas para comunicaciones inalámbricas en sistemas WSN: *Bluetooth/LE*, *Zigbee* y *LoRaWAN*. Asimismo, las características de las tres tecnologías se recogen en la Tabla 1.1.

Bluetooth/LE

Blueetooth Smart o *Bluetooth Low-Energy* es el nombre que recibe la tecnología de red inalámbrica de área personal desarrollada por el "Grupo de Interés Especial en Bluetooth" a partir de su especificación 4.0. En esta versión (Townsend et al., 2014) la tecnología *Bluetooth* soporta una mayor tasa binaria (hasta 2Mbps) e incluye una extensión de bajo consumo. Comparada con versiones anteriores, *Bluetooth/LE* ofrece funciones de establecimiento de enlace más rápidas, implementa cifrado AES de 128 bits y permite intercambiar tasa binaria (hasta 125kbps) por un menor consumo (menor de 15mA de pico), manteniendo un alcance teórico máximo de 100m. Por último, el número máximo de esclavos conectados a la red no está definido en la especificación y depende de las necesidades de comunicación de la red, sirviendo como referencia el máximo de 7 en las revisiones anteriores.

Zigbee

Zigbee (Zigbee, 2018) es el nombre bajo el que se recogen una serie de protocolos de alto nivel de comunicación inalámbrica, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal y con orientación a bajo consumo, permitiendo la existencia de dispositivos durmientes y captación de energía en la red. Esta tecnología opera en las bandas sin licenciar de 2.4GHz (globalmente), 915MHz (en las Americas) y 868MHz (en Europa), ofreciendo tasas binarias de hasta 250kbps, 10kbps y 100kbps respectivamente. Su rango de transmisión se sitúa entre 10 y 100 metros en la banda de 2.4GHz y hasta 1km en el resto. De nuevo, esta tecnología ofrece soporte a la encriptación AES de 128 bits y, teóricamente, el número de nodos asociados a una red *Zigbee* está en torno a las centenas dependiendo de la configuración de la red y la banda de frecuencia utilizada.

LoRaWAN

La especificación *LoRaWAN* es un protocolo abierto de comunicación para redes inalámbricas LPWAN, promovido por la *LoRa Alliance*, diseñado para conectar "cosas" alimentadas por baterías a Internet en redes regionales, nacionales o globales y cumpliendo los requisitos clave del *IoT* como: comunicaciones bidireccionales, seguridad extremo a extremo, movilidad y servicios de localización. Este protocolo (LoRa, 2018) se basa en la modulación *LoRa*, una tecnología propietaria que ofrece tasas binarias bajas, desde 0.3kbps hasta 50kbps, mediante una modulación de espectro

ensanchado y frecuencia pulsada que minimiza el consumo energético intercambiando alcance por tasa binaria. Esta tecnlogía ofrece soporte a la encriptación AES de 128 bits y opera en las bandas ISM de 433MHz, 868MHz o 915MHz con un rango de transmisión de hasta 3km, permitiendo topologías de red en malla con miles de nodos conectados.

	Bluetooth/LE	*Zigbee*	*LoRaWAN*
Tasa binaria [kbps]	125-2000	10-250	0.3-50
Alcance máximo [m]	100	100	3000
Bandas de frecuencia [MHz]	2400	868-915-2400	433-868-915
Tamaño máximo [nodos]	< 10	< 1000	> 1000

Tabla 1.1: Tecnologías de comunicación inalámbrica para sistemas WSN.

1.1.3. Sensores empleados en sistemas WSN

En las últimas décadas, los avances tecnológicos en diseño microelectrónico y en micromecanizado han permitido la miniaturización de la tecnología electrónica, logrando una reducción drástica del tamaño, del coste y del consumo de elementos computacionales y sensores. Los principales exponentes de estos avances son los denominados sistemas microelectromecánicos (MEMS) (Gardner et al., 2001). Dentro de este campo se encuentran los microsensores, término empleado para referirse a dispositivos miniaturizados capaces de convertir variables no eléctricas en señales eléctricas. Estos dispositivos a menudo ofrecen rendimientos similares o incluso superiores a aquéllos ofrecidos por los transductores convencionales. Son ejemplos de estos dispositivos: sensores de temperatura, acelerómetros, giróscopos, sensores de presión, micrófonos, detectores de radiación, sensores magnéticos, sensores de flujo y sensores bioquímicos.

A pesar de las numerosas ventajas que ofrecen los sistemas MEMS, su selección requiere la consideración de diversos factores, especialmente determinantes cuando el dispositivo debe incluirse en un sistema WSN: volúmen, consumo de potencia, compatibilidad con el ciclo de alimentación, fabricación y compatibilidad en el montaje con otros componentes del sistema, necesidades de embalaje y necesidades de contacto con el ambiente (Warneke and Pister, 2002). Por ejemplo, un sensor bioquímico debe estar en contacto con la muestra para obtener una medición. De igual manera, un sensor de concentración gaseosa no puede integrarse en un dispositivo cuyo embalaje lo aisle del exterior. Estas consideraciones afectan directamente a la idoinedad de un sensor y pueden ser determinantes en la selección de un dispositivo MEMS frente a un transductor convencional.

1.2. Red de monitorización *Monitoring Robolabo*

Este Trabajo Fin de Máster ha sido desarrollado en el Laboratorio de Robótica y Control ("Robolabo") de la ETSIT-UPM. Entre los proyectos más relevantes desarrollados en el "Robolabo" se encuentra la red de monitorización *Monitoring Robolabo*: originalmente desarrollada para la competición "*Solar Decathlon Europe*"[1] y posteriormente ampliada por los integrantes del "Robolabo".

La red *Monitoring Robolabo* es un sistema de tiempo real que incluye todos los elementos fundamentales de una red de monitorización industrial: desde los sensores y tarjetas electrónicas de adquisición, hasta las herramientas de procesado y presentación de los datos, siendo estos accesibles a través de una página web[2] (ver Figura 1.2) actualizada en tiempo real y alojada en los servidores del "Robolabo". Dentro de este sistema, se monitorizan variables de naturalezas muy diversas, como pueden ser: temperatura, humedad, flujo térmico, incidencia solar, concentración de CO_2, luminosidad, precipitaciones, generación de energía fotovoltaica, potencia instantánea o energía consumida. Los datos obtenidos de los sensores se transifieren a los servidores del "Robolabo", donde son procesados y almacenados en bases de datos estructuradas, sirviendo de base para la realización de diversos estudios, tanto académicos como de aplicación industrial.

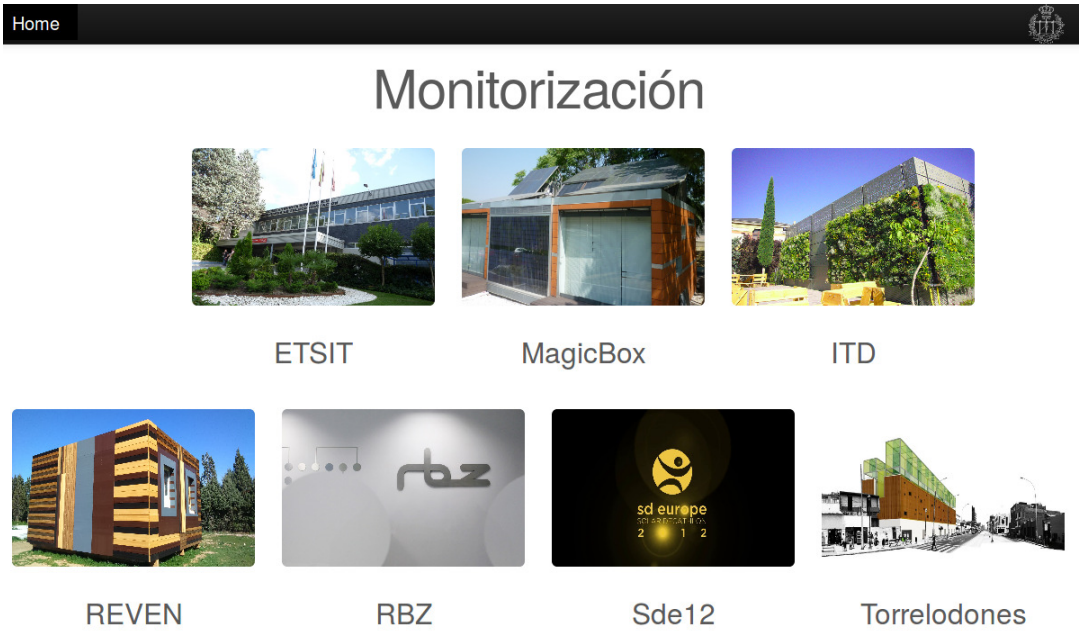

Figura 1.2: Web de monitorización - *Monitoring Robolabo*

[1] https://www.solardecathlon.gov/international-europe.html
[2] http://monitoring.robolabo.etsit.upm.es/

1.2.1. Arquitectura de la red

La arquitectura *hardware* original de la red estaba compuesta por tres bloques principales, como se muestra en la Figura 1.3:

- *TJ Monitor (TJMON).*
- Cuatro módulos *SDIN.*
- Cinco medidores de potencia.

Los módulos *SDIN* son tarjetas electrónicas de adquisición de datos que, en la arquitectura original, se encargaban de obtener medidas de todas aquellas variables no eléctricas (temperatura, humedad, CO_2). El *TJMON* es el núcleo de procesamiento local en la red: una plataforma con un sistema operativo *µLinux* embebido que, en la arquitectura original, obtenía la información de los medidores de potencia y los módulos *SDIN*, la almacenaba en una memoria SD interna y enviaba los datos al servidor *Monitoring Server*. Esta arquitectura ha seguido siendo desarrollada en proyectos posteriores dentro del "Robolabo", creciendo tanto en tamaño de red como en capacidad sensorial, con el desarrollo de módulos *SDIN* para medidas eléctricas (Herreros Elorza, 2016), pero manteniendo siempre la misma topología: las tarjetas de adquisición transfieren sus medidas a un dispositivo *TJMON*, que actúa de concentrador, generando los ficheros de datos y envíandolos al servidor central.

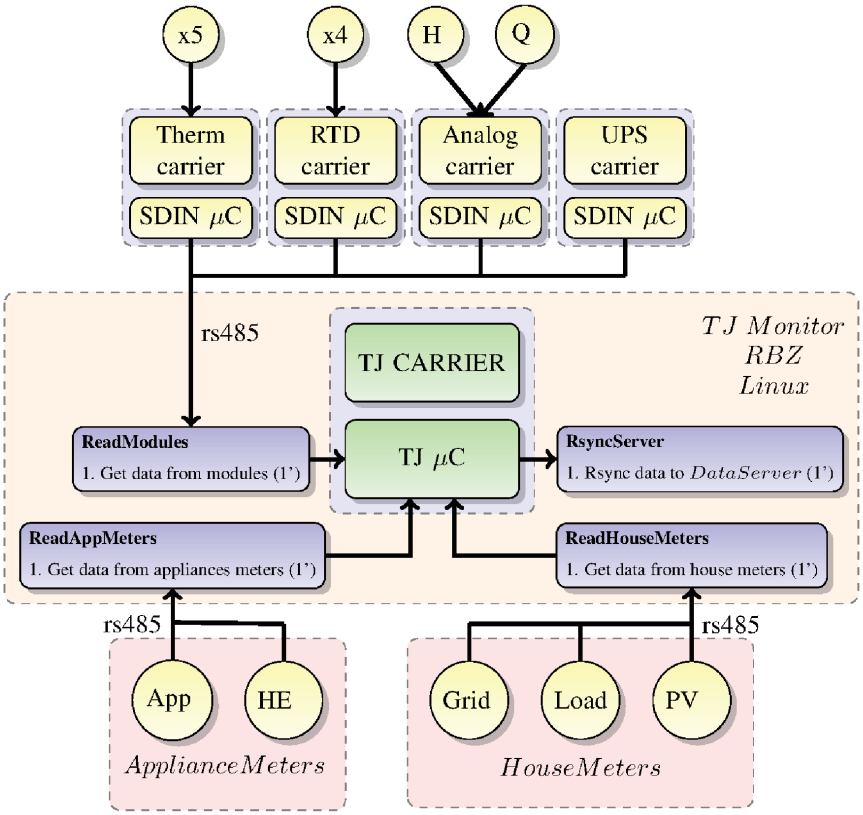

Figura 1.3: Arquitectura de monitorización original. (Robolabo, 2014)

De entre las características técnicas de las plataformas mencionadas, es de especial relevancia para este Trabajo Fin de Máster el método de transferencia de información entre las tarjetas de adquisición y el *TJMON*. La comunicación entre estas plataformas está basada en protocolo serie Modbus-RTU (RS-485) y, tanto la plataforma *TJMON* como las tarjetas de adquisición conectadas, requieren de puertos dedicados a esta interfaz para implementar la comunicación.

1.2.2. Líneas de desarrollo

Razonablemente, de este proyecto se desprenden numerosas líneas de investigación y desarrollo en varios ámbitos, entre las que destacan:

- Gestión activa de la demanda eléctrica.
- Evaluación de técnicas de optimización del consumo eléctrico.
- Generación de modelos de predicción basados en datos.
- Estudios de impacto ambiental a medio-largo plazo.

Para poder desarrollar estas líneas a su máximo potencial, es muy favorable contar con enormes cantidades de datos etiquetados de diversas naturalezas, ámbitos y emplazamientos. Sin embargo, uno de los principales obstáculos en la recogida de estos datos es la dificultad de despliegue en las redes de monitorización cableadas convencionales, siendo necesarios excesivos medios materiales (cable), estructurales (zanjas o canalizaciones) y humanos (mano de obra) para su ejecución. En este contexto, resulta interesante la posibilidad de inclusión en la red *Monitoring Robolabo* de elementos propios de las redes de sensores inalámbricas, fácilmente desplegables y que puedan interactuar de forma transparente con el sistema existente. Este es, por tanto, uno de los principales motivos del desarrollo de este Trabajo Fin de Máster.

1.3. Motivación y objetivos

El propósito de este Trabajo Fin de Máster consiste en el desarrollo de un sistema empotrado autónomo, capaz de monitorizar variables físicas del entorno y de transmitir los datos obtenidos de forma inalámbrica. Además, este sistema debe poder integrarse con la red de monitorización *Monitoring Robolabo*, ya existente, de forma eficiente y transparente para el resto del sistema.

Para lograr este objetivo global, se han propuesto los siguientes objetivos específicos para este proyecto:

- Estudio del estado del arte en sistemas WSN.
- Selección de componentes, diseño y fabricación del prototipo *hardware*.
- Montaje y pruebas eléctricas.
- Desarrollo *software* de una aplicación de red punto-multipunto.

- Pruebas funcionales de la aplicación.
- Integración de los módulos *hardware* y *software*.
- Pruebas y validación del prototipo.

1.4. Planificación del proyecto

Para alcanzar los objetivos planteados en la sección anterior se han llevado a cabo las siguientes tareas, desarrolladas en los capítulos de este documento:

- En el Capítulo 1 se ha presentado un estudio del estado del arte en sistemas WSN, se ha contextualizado el marco de aplicación en una red de monitorización existente y se han descrito la motivación y objetivos de este Trabajo Fin de Máster.
- En el Capítulo 2 se presentará la arquitectura *hardware* implementada, contemplando los requisitos de diseño tenidos en cuenta para la selección de componentes, así como los pasos seguidos en el diseño de la tarjeta electrónica de expansión y el resultado de su fabricación.
- El Capítulo 3 versará sobre la programación de la tarjeta electrónica de control comercial en cumplimiento con los requisitos de la aplicación.
- En el Capítulo 4 se describirán las pruebas eléctricas y funcionales realizadas sobre el prototipo y los resultados de las mismas.
- En el Capítulo 5 se expondrán las conclusiones y las líneas futuras de este Trabajo Fin de Máster.

Capítulo 2

Diseño del prototipo *hardware*

En el Capítulo 1 se han revisado los elementos básicos de un nodo en una red de sensores inalámbrica: capacidad sensorial, distribución espacial, comunicaciones inalámbricas, alimentación por baterías y protocolo de red. Debido a que el presente proyecto sigue un método de diseño basado en plataformas, gran parte de las restricciones de diseño de la tarjeta electrónica a desarrollar vienen impuestas por la tarjeta electrónica comercial de control seleccionada.

A su vez, el tipo de sensores empleados en la aplicación final condiciona en gran medida los circuitos de adaptación y preprocesado de la señal, que deberán estar presentes en la tarjeta. En consecuencia, este capítulo se divide en dos secciones: la selección de componentes (Sección 2.1), donde se realiza una revisión detallada de las necesidades de la aplicación y los componentes electrónicos seleccionados para dar respuesta a dichas necesidades; y el diseño de la tarjeta electrónica de expansión (Sección 2.2), donde se presenta las herramientas CAD (*Computer Aided Design*) y CAM (*Computer Aided Manufacturing*) empleadas y los resultados del diseño y la fabricación de la tarjeta.

Cabe mencionar que algunos de los submódulos empleados en el diseño de la tarjeta electrónica de expansión provienen de otros proyectos propios del Laboratorio de Robótica y Control de la ETSIT-UPM ("Robolabo") y han sido modificados para ajustarse a las necesidades de este proyecto.

2.1. Selección de componentes

2.1.1. Tarjeta electrónica comercial de control

A fin de cumplir con los requisitos de consumo y capacidad de procesamiento de este proyecto, se ha seleccionado la tarjeta electrónica comercial de *STMicrolectronics* NUCLEO-L432KC, mostrada en la Figura 2.1. Esta tarjeta incorpora un microcontrolador STM32L432KC y un depurador/programador integrado vía USB. Además permite la alimentación a través de fuentes externas, cuenta con reguladores de tensión de 3.3V y 5V, y presenta un formato de dimensiones reducidas con 32 pines situados en conectores de expansión. (STMicroelectronics, 2018b)

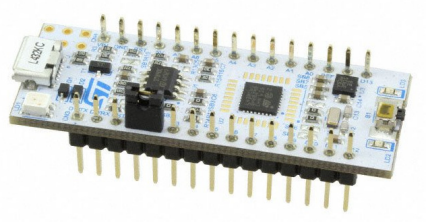

Figura 2.1: Tarjeta NUCLEO-L432KC.

Los dispositivos STM32L432xx son microcontroladores de ultra bajo consumo, basados en el core RISC de 32 bits ARM®Cortex®-M4 (ARM, 2018), capaces de operar a una frecuencia de hasta 80 MHz. El Cortex-M4 incluye una unidad de coma flotante (FPU) de precisión simple que soporta todas las instrucciones de procesamiento de datos y todos los tipos de datos de ARM®. También implementa un conjunto completo de instrucciones de procesamiento digital de señal (DSP) y una unidad de protección de memoria (MPU) que aumenta la seguridad de las aplicaciones.

Estos dispositivos incorporan memorias embebidas de alta velocidad (memorias Flash de hasta 256 KB, hasta 64 KB de SRAM), un interfaz *Quad SPI* de memorias flash y un amplio rango de entradas/salidas y periféricos conectados a dos buses avanzados de periféricos (APB), dos buses avanzados de alto rendimiento (AHB) y una matriz de 32 bits de buses multi-AHB[1].

Todos los dispositivos (STMicroelectronics, 2018e) ofrecen un ADC de alta velocidad (5 Msps) de 12 bits, dos comparadores, un amplificador operacional, dos canales de DAC, un reloj RTC de bajo consumo, un temporizador de 32 bits de propósito general, un temporizador de 16 bits dedicado al control de motores por PWM, cuatro temporizadores de 16 bits de propósito general y dos temporizadores de 16 bits de bajo consumo. Además, incluyen interfaces de comunicaciones estándar y avanzadas, así como un conjunto exhaustivo de modos de ahorro energético que permiten el diseño de aplicaciones para bajo consumo como la desarrollada en este Trabajo Fin de Máster.

[1]Especificación AMBA: `https://www.arm.com/products/system-ip/amba-specifications`

En la Figura 2.2 se presenta un cuadro, proporcionado por *STMicroelectronics*, a modo de resumen de las características de estos microcontroladores.

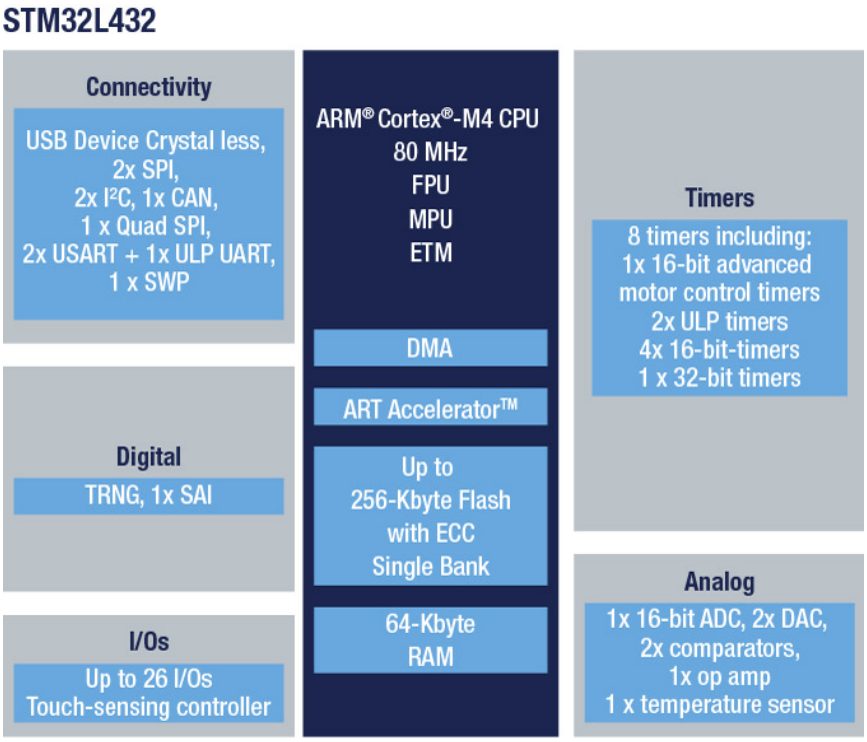

Figura 2.2: Resumen de los microcontroladores STM32L432. (STMicroelectronics, 2018e)

2.1.2. Transceptor de radiofrecuencia

Un transceptor, tal y como se define en (Weik, 1996), es la combinación en un mismo encapsulado de equipos de transmisión y recepción empleando componentes circuitales comunes tanto para transmitir como para recibir.

Con el fin de dotar al nodo de la capacidad de comunicación inalámbrica minimizando el impacto de la misma sobre el consumo, se propone el uso del módulo transceptor de radiofrecuencia fabricado por *NiceRF* "LoRa1278" (Figura 2.3). El transceptor se basa en el encapsulado de *Semtech* "SX1278" (Semtech, 2018), el cual implementa un modem LoRa® de comunicaciones de espectro ensanchado y largo alcance, con alta inmunidad frente a interferencias y bajo consumo en corriente como el que se observa en el diagrama de la Figura 2.4.

Este módulo adopta la técnica de modulación de espectro ensanchado con salto en frecuencia definida en la especificación de LoRa®, logrando sensibilidades de hasta -139dBm en el rango de frecuencias de 433-470MHz, con una tasa binaria en el rango de 0.018-37.5Kbps, manteniendo un consumo de corriente de 13mA en recepción y menor a 200nA en modo de bajo consumo (NiceRF Wireless Technology Co., 2018).

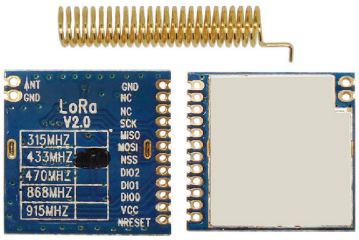

Figura 2.3: G-NiceRF LoRa1278. (NiceRF Wireless Technology Co., 2018)

Asimismo, permite el uso de modulaciones FSK, GFSK y OOK, e incluye protección ESD y un mecanismo de empaquetado de datos con una cola FIFO de 256 bytes y detección de errores por CRC.

En el sistema final, este módulo se comportará como un esclavo conectado a un bus SPI, el cual recibirá de su maestro (el microcontrolador NUCLEO-L432KC) las señales pertinentes para su habilitación y control, y deberá ser alimentado debidamente a través de la tarjeta de expansión, a 3.3V de acuerdo con su ficha técnica. A él deberá conectarse una antena de 50 ohmios de impedancia específica para la comunicación inalámbrica.

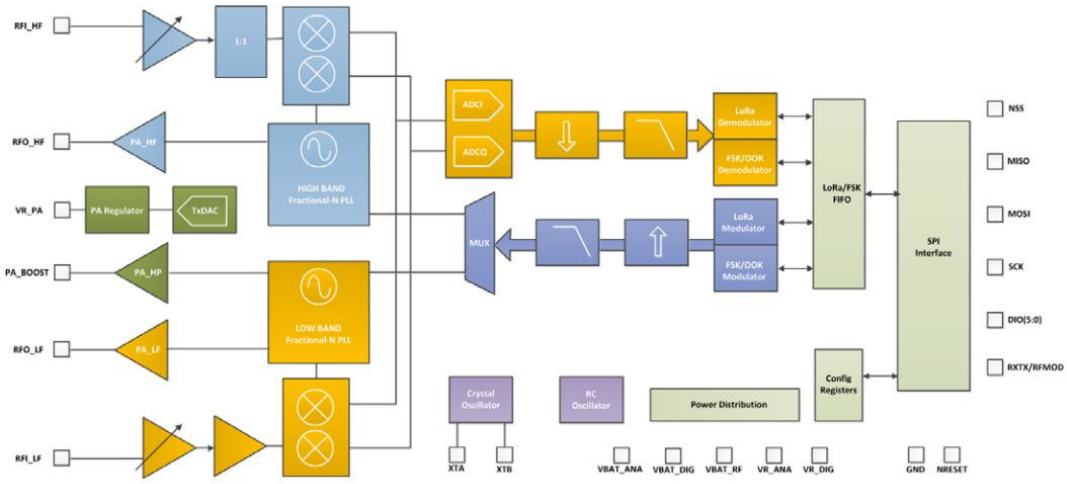

Figura 2.4: Diagrama de bloques del modem Semtech SX1278.
(Semtech, 2018)

2.1.3. Sensores y circuitería de adaptación

A modo de primera aproximación y con el objetivo de alcanzar un compromiso razonable entre costes de diseño y capacidad sensorial del nodo, se propone un diseño que incluye la circuitería necesaria para adquirir información de cuatro entradas conectadas a dos termopares (tipos T, J y K) y dos sensores analógicos con salidas de 4-20mA ó 0-10V.

Para adaptar las entradas de ambos tipos de sensores se han reutilizado los circuitos empleados en las plataformas *SDIN_Therm* y *SDIN_Analog* (Figura 2.5) desarrolladas por el Laboratorio de Robótica y Control ("Robolabo") y que forman parte de la red de monitorización *Monitoring Robolabo*.

Figura 2.5: Plataformas *SDIN_Therm* y *SDIN_Analog*.

Concretamente, para adaptar la señal proveniente de las entradas de termopares se ha incluido un circuito que implementa protección frente a descargas electroestáticas (ESD), filtrado de los modos común (16KHz) y diferencial (1.3KHz) de las señales diferenciales de entrada y conexión a un punto de tensión de entrada predeterminado para fijar el punto de trabajo. De igual manera, para adaptar la señal proveniente de las entradas analógicas se ha incluido un circuito que consiste en una protección frente a sobretensiones y un divisor de tensión cuya tensión de entrada viene condicionada por un interruptor electrónico controlado por software. Este interruptor se implementa mediante un transistor de efecto de campo (MOSFET) cuyo estado determina si la magnitud medida a la entrada es corriente o tensión, de modo que es el usuario quien ha de configurar cada canal analógico en función del tipo de sensor conectado.

Cabe destacar que, para obtener una medida fiable de temperatura, es necesario aplicar una corrección proporcional al valor de la temperatura en la unión con el substrato. En consecuencia, es necesario integrar en la tarjeta de expansión un sensor de temperatura para obtener este valor correctivo, para lo cual se ha seleccionado el encapsulado "ADT7420" del fabricante *Analog Devices*. Este encapsulado (Devices, 2017a) implementa un sensor digital de temperatura de alta precisión, con una resolución máxima de hasta 0.0078ºC, calibración automática y un rango de temperaturas apropiado para la aplicación, con interfaz de comunicación serie por I2C. Además, alimentado a una tensión de 3.3V, se comporta como un dispositivo de bajo consumo con 210μA en modo activo y tan sólo 2μA en modo inactivo, lo cual lo hace idóneo para los requisitos de la aplicación.

2.1.4. Conversor analógico-digital externo

En el diseño de un sistema electrónico analógico mixto es imprescindible la presencia de un elemento que haga las veces de interfaz entre el mundo analógico de los sensores y el mundo digital del microcontrolador. Este papel corresponde indudablemente a un conversor analógico-digital (ADC por sus siglas en inglés), encargado de transformar los niveles de entrada analógicos en un rango de niveles discretos digitalmente representables.

A pesar de disponer de un ADC de 12 bits en el microcontrolador de la tarjeta de control (STMicroelectronics, 2018e), se ha considerado necesario incluir un conversor externo de mayor precisión para cumplir con los requisitos de la aplicación. A tal efecto, el dispositivo seleccionado es el encapsulado "AD7194" del fabricante *Analog Devices*. Este encapsulado implementa un ADC tipo sigma-delta de 24 bits, 8 canales diferenciales ó 16 pseudo-diferenciales, ganancia y filtro de entrada programables y bajo ruido, como se puede observar en el diagrama funcional de la Figura 2.6, extraído de la ficha técnica del dispositivo (Devices, 2017c). El dispositivo se alimenta a una tensión analógica de 3-5.25V y una tensión digital de 2.7-5.25V. Incluye además un reloj interno, que le permite generar una tasa de salida de datos de 4.8KHz.

Es relevante anotar que este dispositivo se controla a través de una interfaz serie SPI, que permite la existencia de varios esclavos conectados al mismo bus, por lo que no supone un inconveniente para su integración en la tarjeta de expansión junto con el módulo radiotransceptor. Únicamente deberá reservarse una línea de selección de esclavo independiente para cada encapsulado y actuar consecuentemente a nivel de software.

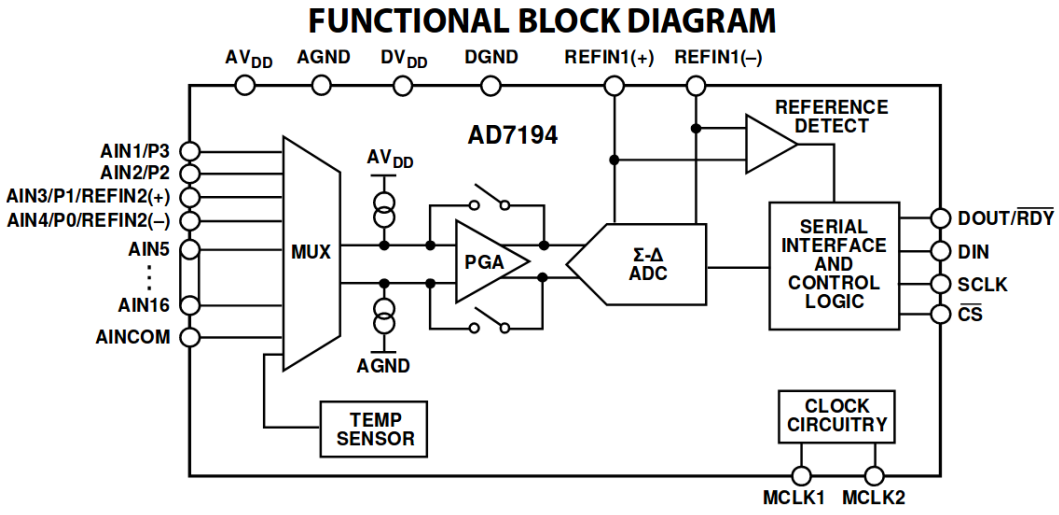

Figura 2.6: Diagrama funcional del conversor "AD7194". (Devices, 2017c)

2.1.5. Alimentación y fuentes de tensión analógicas y digitales

En este proyecto, los nodos se alimentan a través de pilas de 9V conectadas a la entrada de alimentación de la tarjeta de control. Es ésta, de acuerdo a la Tabla 4 de su manual de usuario (STMicroelectronics, 2018f), la encargada de generar los niveles de tensión de 5V y 3.3V necesarios para alimentar los periféricos y demás componentes conectados a través de la tarjeta de expansión.

Sin embargo, el ADC externo requiere tanto alimentación digital como analógica, así como un nivel de tensión de referencia para la etapa sigma-delta del conversor. También es necesario generar el nivel de tensión intermedio para fijar el punto de trabajo en las entradas de termopares. Para cumplir con estos requisitos se ha reutilizado el circuito empleado en la plataforma *SDIN_Therm* que filtra el ruido en la tensión de alimentación de 3.3V mediante un núcleo de ferrita en paralelo con un condensador, definiendo una frecuencia de corte del filtro de 100MHz. Asimismo, este circuito (ver Figura 2.13) se encarga de generar dos niveles de referencia, a 2.5V y 1.25V respectivamente, haciendo uso del encapsulado "AD1582" (Devices, 2017b) del fabricante *Analog Devices* y un divisor de tensión.

2.1.6. Interfaz RS485

Como se ha visto en el Capítulo 1, para hacer posible la integración de estos nodos en la red de monitorización *Monitoring Robolabo* es necesario que el nodo que desempeñe el papel de puerta de enlace en la red inalámbrica tenga capacidad de comunicación por interfaz RS485 Modbus-RTU. Esto permite al nodo interactuar con la plataforma *TJMON* (ver Figura 2.7) de forma totalmente transparente al sistema.

Figura 2.7: Plataforma *TJMON*.

Así, empleando la interfaz de comunicaciones UART de la tarjeta de control, a nivel de *hardware* basta con integrar un circuito que adapte la señal para emplear comunicación diferencial con niveles de 5V. Para ello se ha seleccionado el encapsulado "MAX13487E" del fabricante *Maxim Integrated* (Integrated, 2017).

2.1.7. Elementos mecánicos de interconexión y alimentación

Por último, es imprescindible dotar a la tarjeta de elementos mecánicos que permitan tanto el acoplo a la tarjeta NUCLEO-L432KC como la conexión eléctrica con la misma y entre los sensores y los circuitos de adaptación y protección previos al conversor analógico-digital.

El cumplimiento del primer requisito viene condicionado por el factor de forma de la tarjeta NUCLEO-L432KC, siendo necesaria la colocación de dos conectores hembra de 15 pines y paso 2.54mm, como los fabricados por *Sullins Connector Solutions* (Sullins, 2018). Estos conectores deberán situarse a la distancia apropiada durante el trazado del circuito impreso para permitir la conexión.

Para la interconexión con los sensores externos se propone usar bloques terminales de orificio pasante en ángulo recto, con clavijas macho protegidas. Puesto que será necesario disponer de un mínimo de ocho entradas (cuatro correspondientes a dos canales diferenciales de termopares, dos correspondientes a entradas analógicas y dos correspondientes a la interfaz RS485) y la disposición de los conectores para la tarjeta de control es bastante restrictiva, se ha decidido colocar un único conector de doce posiciones en dos niveles, como el fabricado por *Phoenix Contact* (Contact, 2018), que se colocará en el extremo de la tarjeta opuesto al radiotransceptor.

Finalmente, para poder alimentar el nodo como se ha propuesto a través de una pila de 9V, son necesarios un conector de alimentación y un soporte para la pila. Para dar una mayor resistencia mecánica al conector y evitar desperfectos por desgaste mecánico de la terminación se ha seleccionado un conector de potencia de inserción de tipo barril (Inc, 2018) fabricado por *CUI Inc.* y un soporte de batería de montaje en chasis con terminación en clavija de conexión de barril (Electronics, 2018) fabricado por *SparkFun Electronics*.

2.2. Diseño de la tarjeta electrónica de expansión

En la Sección 2.1 se han determinado los componentes electrónicos necesarios para este proyecto, así como las relaciones y dependencias entre ellos. Entrando ahora en el diseño de la tarjeta electrónica de expansión, resulta útil tener presente una visión funcional del sistema como la que se presenta en la Figura 2.8. En este diagrama se pueden apreciar los princpiales bloques funcionales de la tarjeta de expansión y sus interconexiones: la regulación de tensión de la tarjeta (bloque "*POWERS*"), la conexión con la tarjeta NUCLEO-L432KC (bloque "*HEADERS*"), las conexiones hacia el exterior, incluyendo las entradas de sensores y la antena del radiotransceptor (bloque "*CONNECTORS and ANTENNA*"), el sensor de temperatura integrado (bloque "*THERM SENSOR*"), el ADC externo y los circuitos de adaptación conectados a sus entradas (bloque "*ADC*"), el radiotransceptor (bloque "*Lora1278*") y el circuito de conversión UART-Modbus-RTU (bloque "*SERIAL INTERFACE*).

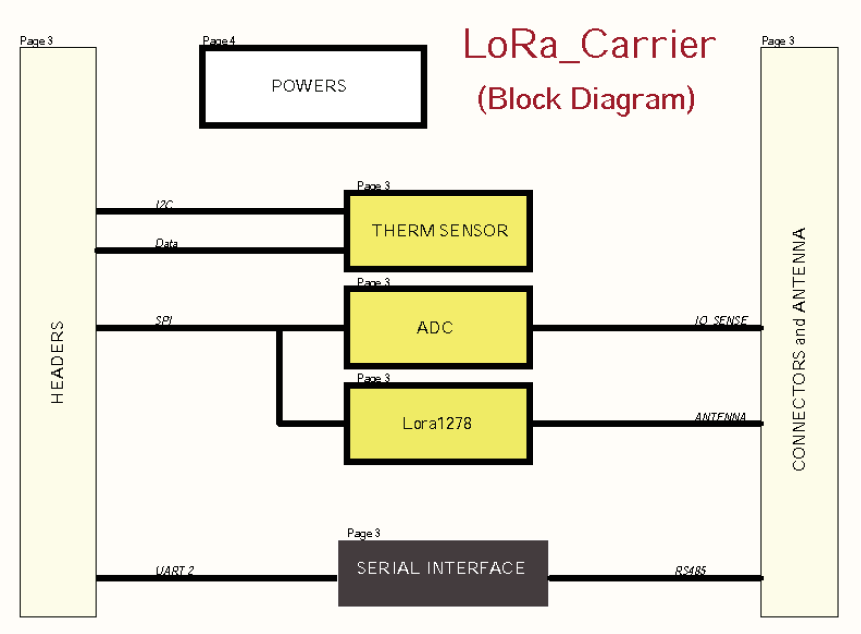

Figura 2.8: Diagrama funcional de la tarjeta de expansión.

2.2.1. Entorno de desarrollo: Altium Designer

El software empleado en el diseño de la tarjeta de expansión ha sido *Altium Designer*[2], una herramienta profesional de diseño asistido por ordenador que permite la captura de esquemáticos, trazado de circuitos impresos, diseño de librerías, simulación y diseño basado en FPGAs. A continuación se presentan las principales características de este entorno de desarrollo en relación con el trabajo realizado en este proyecto.

Editores

[2] `https://www.altium.com/altium-designer/`

Los editores de *Altium Designer* son espacios de trabajo donde se pueden crear y editar documentos dentro de un proyecto. En el desarrollo de este proyecto se han empleado dos editores:

- **Editor de esquemáticos:** permite la captura de esquemáticos del circuito electrónico.
- **Editor de PCB:** permite el trazado físico del circuito impreso que se pretende fabricar.

En la Figura 2.9 se observan los dos editores mencionados dentro del entorno de desarrollo:

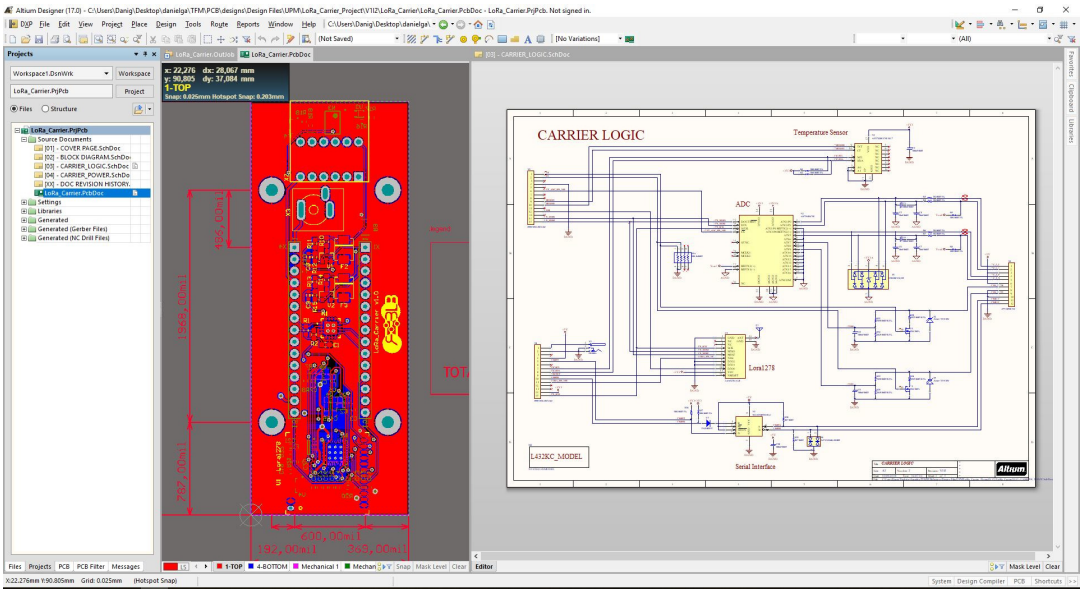

Figura 2.9: Editores utilizados en el diseño de la tarjeta.

Una característica especialmente útil de este entorno es la selección cruzada de componentes. Con esta herramienta, si se han abierto tanto el esquemático como el trazado de un circuito, al seleccionar un componente en uno de los editores se resalta automáticamente en el otro, facilitando enormemente la tarea de ubicar y agrupar grupos funcionales de componentes en un diseño.

Proyectos

Un proyecto en *Altium Designer* consiste en un conjunto de documentos ordenados en carpetas que corresponden a un diseño concreto. Típicamente, en el diseño de un circuito impreso en este entorno de desarrollo existen uno o varios proyectos PCB (ficheros con extensión **.PRJPCB*), correspondientes a las distintas versiones del diseño, que contienen toda la información necesaria para fabricar el circuito impreso correspondiente.

También es habitual la existencia de librerías (ficheros con extensión **.LIBPKG* y **.INTLIB*), conjuntos de archivos que contienen la huella, el símbolo e incluso un

modelo de simulación de los componentes empleados.

En la creación de un proyecto existen además dos tipos de documentos fundamentales: los esquemáticos (ficheros con extensión **.SCH*) y los trazados (ficheros con extensión **.PCB*). Cada uno de estos documentos puede ser creado y editado con los editores mencionados anteriormente y, junto con las librerías, conforman la base de cualquier proyecto de circuito impreso.

Diseño jerárquico

Altium Designer ofrece la posibilidad de dividir el esquemático de un circuito complejo en un conjunto de uno o varios módulos funcionalmente diferenciados, enlazando los distintos bloques y permitiendo un acceso rápido y ordenado a los detalles de cada uno mediante el uso del ratón del ordenador.

Esta funcionalidad puede resultar útil en aquellos proyectos cuya complejidad y extensión es tan elevada que una vista del esquemático al completo satura al diseñador y le impide localizar ágilmente partes concretas del circuito. Así, el esquemático queda reducido a la mínima expresión funcional y en caso de requerir un mayor nivel de detalle en alguna parte del mismo, esta es accesible individualmente para una mayor claridad.

A pesar de no haber hecho uso de esta caracerística en el diseño de la tarjeta de expansión, resulta relevante su mención ya que nos permite introducir el concepto de conectividad. En un esquemático se puede representar de forma lógica un diseño conectando los pines de los componentes entre sí, pero para representar físicamente la conexión entre estos componentes y facilitar el trazado del circuito impreso, *Altium Designer* ofrece unos objetos[3], a los que colectivamente denomina como "identificadores de red", que permiten crear conectividad tanto lógica como física entre los componentes. De entre estos objetos, los que han sido empleados con mayor asiduidad en este proyecto son:

- **Pin:** los pines se colocan en el editor de símbolos de esquemático, para representar los pines físicos del componente.
- **Puerto de alimentación:** crea conectividad con todos los demás puertos con el mismo nombre en todos los esquemáticos del proyecto independientemente de la estructura de diseño. La red eléctrica creada es automáticamente nombrada siguiendo el nombre del puerto.
- **Línea:** una primitiva de diseño eléctrico usada para formar conexiones eléctricas entre dos o más puntos de un esquemático. Una línea es análoga a un cable físico.
- **Etiqueta de red:** un identificador de red usado para crear conectividad a otras etiquetas de red con el mismo nombre en la misma página de esquemático. La red eléctrica etiquetada es automáticamente nombrada como la etiqueta de red. Las etiquetas de red pueden colocarse en pines, líneas y buses.

[3] `https://www.altium.com/documentation/15.1/display/ADES/((Creating+Connectivity))_AD`

En la Figura 2.10 se muestran estos identificadores en el editor de esquemáticos, mientras que en la Figura 2.11 se observa el efecto resultante de su colocación en el editor de PCB:

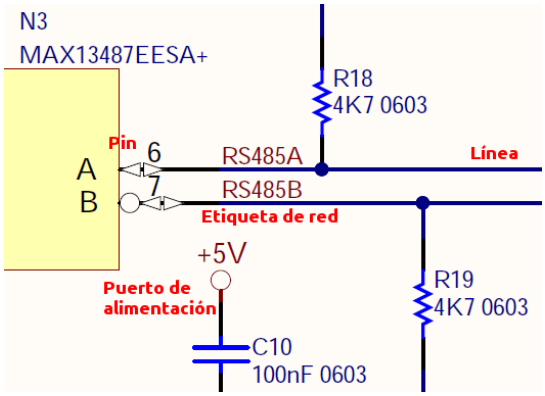

Figura 2.10: Identificadores de red en el editor de esquemáticos.

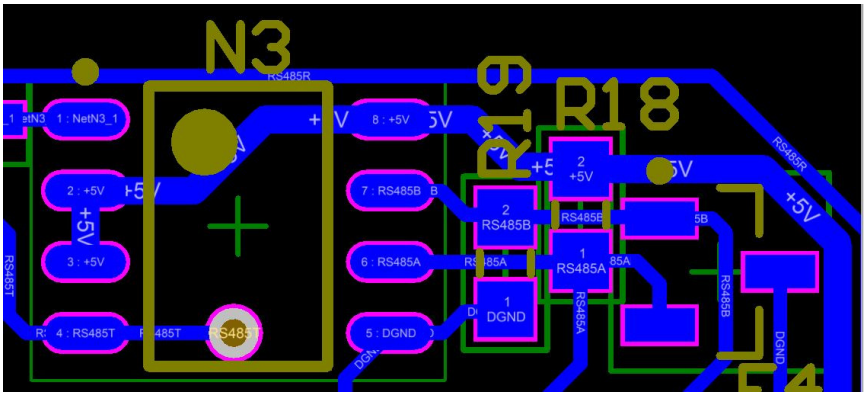

Figura 2.11: Identificadores de red en el editor de PCB.

2.2.2. Captura de esquemáticos

Como se ha adelantado en el apartado anterior, este diseño ha seguido una estructura plana. Es decir, en lugar de emplear un diseño jerárquico, se ha optado por crear documentos esquemáticos separados para las distintas partes del circuito. Concretamente, se han creado dos documentos que representan la alimentación del sistema y su parte funcional, respectivamente. Esta decisión surge de la fuerte dependencia, tanto física como semántica, que existe entre los componentes funcionales del circuito, sumada a la escasa complejidad del mismo que hace ilógica una representación más fraccionada.

En consecuencia, haciendo uso de los puertos de alimentación de *Altium Designer*, se ha creado un documento aislado para la regulación de la alimentación sin perder conectividad en el diseño. El esquemático resultante se corresponde con las

anotaciones realizadas en el Apartado 2.1.5 y se observa en la Figura 2.12:

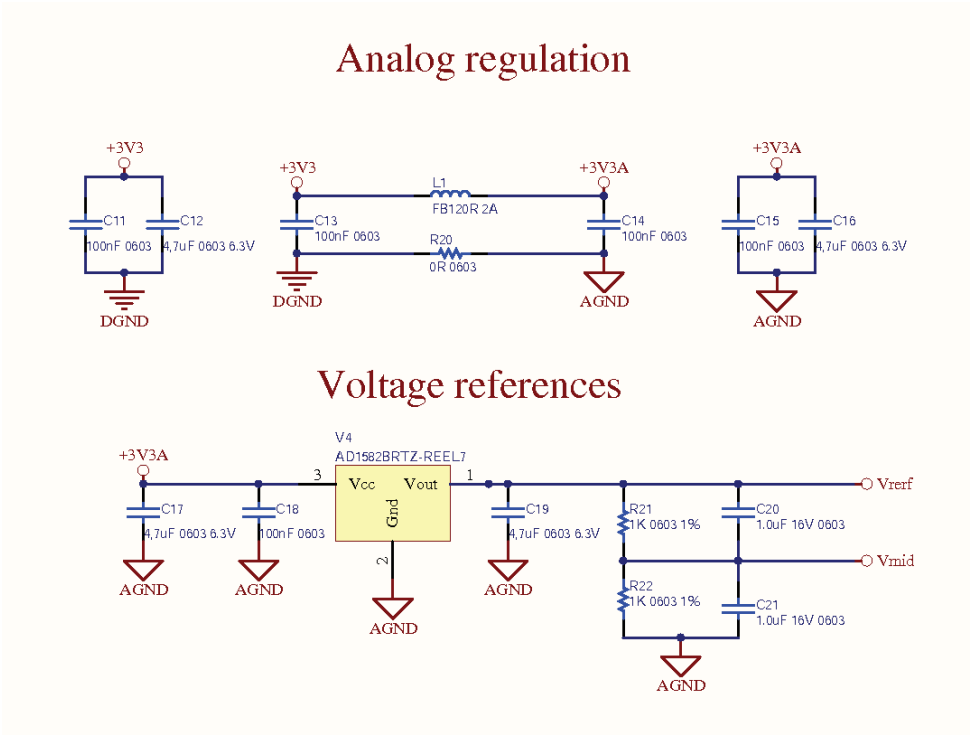

Figura 2.12: Esquemático de la alimentación del sistema.

Análogamente, el esquemático correspondiente a la parte funcional del sistema se observa en la Figura 2.13. Este documento recoge el resto de consideraciones y decisiones expuestas a lo largo de la Sección 2.1, aunando en un único esquemático los componentes y circuitos adicionales correspondientes a: la capacidad sensorial, las comunicaciones inalámbricas y la conexión con la tarjeta electrónica de control. Igualmente, se ha incluido un elemento adicional cuyo valor es meramente gráfico, que representa el modelo en tres dimensiones de la tarjeta electrónica de control y cuya utilidad se demuestra en el Apartado 2.2.3.

Un punto a destacar que no ha sido mencionado con anterioridad es la colocación de condensadores de desacoplo en aquellos puntos del circuito más sensibles, como lo son la alimentación del sistema y la electrónica analógica, cuya función es la de filtrar ruidos e interferencias indeseables en el sistema.

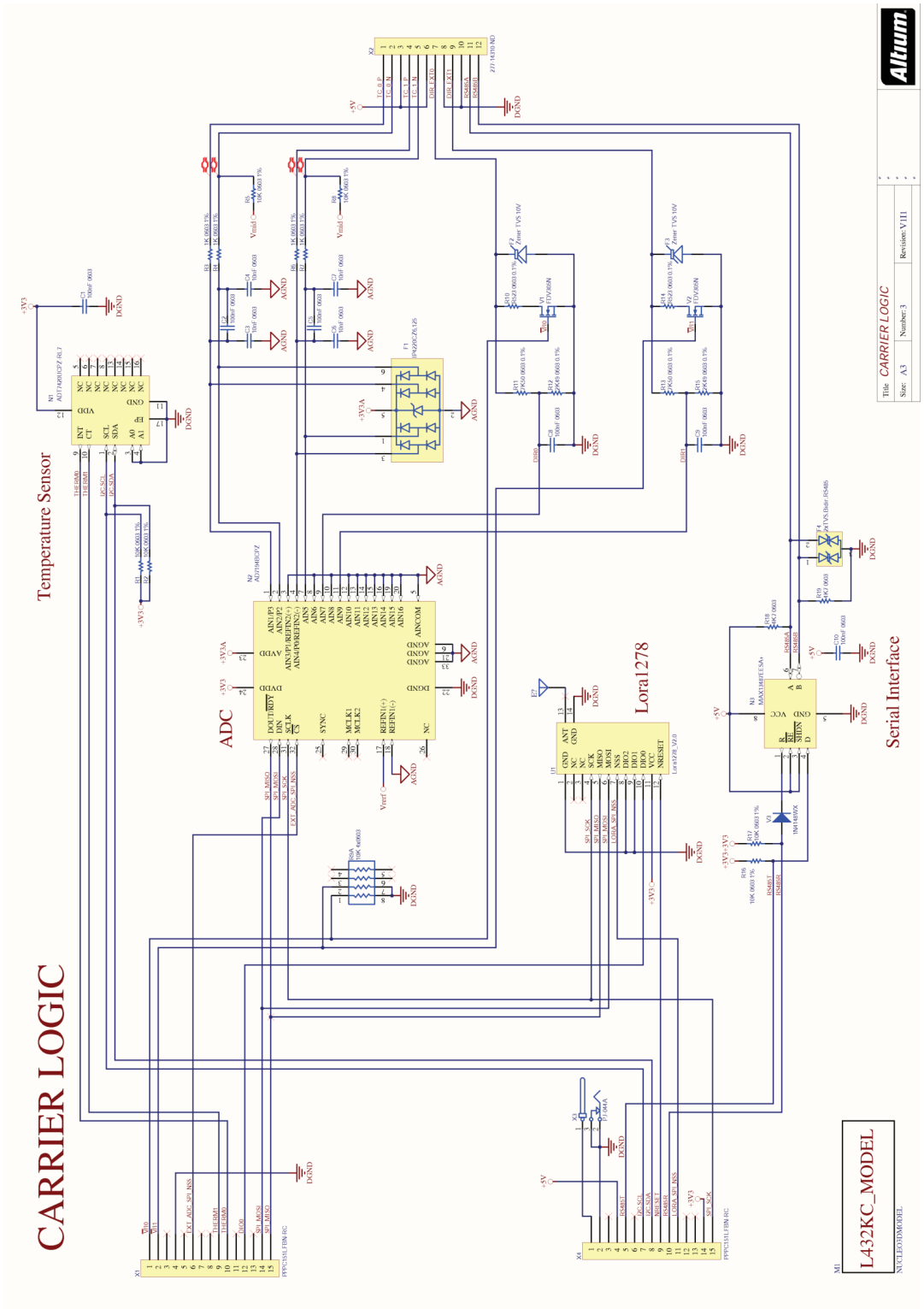

Figura 2.13: Esquemático de la parte funcional del sistema.

2.2.3. Trazado del circuito impreso

Una vez finalizada la captura del esquemático y superada la primera compilación y detección automática de errores (ERC), llega la hora de realizar el trazado físico del circuito impreso. Mientras que el esquemático es una representación lógica y funcional del circuito diseñado, el trazado es una representación física del circuito impreso que se va a fabricar. Éste es un proceso complejo y su realización se torna altamente complicada si no se cuenta con las herramientas de diseño apropiadas. Conceptos como la interferencia entre pistas, el ruido electrónico, la disipación de calor, la distribución de la alimentación o los efectos parásitos de las líneas de transmisión cobran gran importancia durante este proceso y pueden suponer el fracaso de un diseño electrónico (Montrose, 2000).

Por suerte, el entorno de desarrollo *Altium Designer* ofrece la posibilidad de definir reglas de diseño a la hora de realizar el trazado de un circuito impreso. Estas reglas son en realidad directrices primitivas definidas por el diseñador que se comprueban automáticamente dentro del programa a medida que se realiza el trazado, verificando en todo momento su cumplimiento y alertando al diseñador en caso contrario. Además, los fabricantes de circuitos impresos a menudo ofrecen a sus usuarios conjuntos de reglas predefinidas que cubren los aspectos más básicos del diseño para asegurar la compatibilidad con sus capacidades de producción.

En este proyecto se ha empleado un conjunto de reglas heredado de proyectos anteriores realizados en el Laboratorio de Robótica y Control de la ETSIT-UPM ("Robolabo"), adaptando algunas de ellas para este diseño concreto, como es el caso de:

- **Circuitos impresos de dos caras:** al tratarse de un diseño de baja densidad se ha considerado apropiado el empleo de dos caras para la colocación y rutado de los componentes y sus interconexiones. Para ello se han definido para este proyecto reglas relativas al espesor de las capas de cobre (35μm), del dieléctrico que las separa (1.5mm de material FR-4), del ancho de pista mínimo (0.2032mm) y por defecto (0.254mm), así como de los diámetros mínimo y máximo de las vías metalizadas (0.3mm-6.3mm), que atraviesan el substrato para conectar puntos de ambas caras. Los valores numéricos introducidos en estas reglas han sido seleccionados en función de las capacidades de fabricación de la empresa contratada, *Elecrow*[4].
- **Pistas diferenciales:** en el rutado de pistas diferenciales es conveniente asegurar que la impedancia y distancia a recorrer en ambas pistas difiere en la menor medida posible para asegurar así la integridad de la señal que transportan. *Altium Designer* cuenta con una herramienta de rutado de pistas diferenciales que genera automáticamente la ruta de ambos canales simultáneamente de manera que se cumpla la regla de diseño estipulada.
- **Polígonos de alimentación:** al trabajar únicamente en dos capas con un diseño que incluye componentes tanto analógicos como digitales, resulta práctico

[4]Elecrow: `https://www.elecrow.com/pcb-manufacturing.html`

el uso de varios polígonos de alimentación para facilitar la distribución de la misma minimizando las diferencias de tensión entre componentes alejados y reduciendo a su vez la interferencia entre pistas.

- **Mínima distancia anular:** esta regla ha tenido que ser revisada y adaptada a las restricciones del fabricante (0.1524mm) debido a que, durante la creación de la huella del módulo radiotransceptor de acuerdo a su hoja de especificaciones, se ha observado que la distancia anular entre uno de los pads del componente y su agujero de inserción violaba la regla (0.19mm) previamente establecida.

Establecidas las reglas de diseño, se ha realizado el trazado del circuito impreso teniendo en cuenta las restricciones mecánicas impuestas por tres elementos fundamentalmente: los conectores de expansión para la tarjeta de control, el bloque terminal para las entradas y salidas de la tarjeta de expansión y el módulo radiotransceptor cuya antena se ha orientado teniendo en cuenta el factor de forma deseado. Además, se han colocado agujeros de montaje que permitirán un anclaje seguro al chasis una vez terminado el montaje del dispositivo. Todas estas decisiones han sido respaldadas por la funcionalidad de modelado 3D ofrecida por el entorno de desarrollo, que ha permitido analizar de manera más sencilla los espacios disponibles para poder así diseñar una colocación coherente.

En el rutado, se ha optado por distribuir la alimentación a lo largo de sendos planos de cobre equipotenciales, tanto para las fuentes analógicas como las digitales, haciendo uso de vías metalizadas para su conexión con los componentes situados en las capas opuestas. En el caso de las pistas, se ha realizado agrupando espacialmente los componentes de grupos funcionales independientes, tratando así de minimizar las distancias a recorrer por las señales. El resultado final del trazado con los polígonos ocultos y visibles se muestra en las Figuras 2.14 y 2.15, respectivamente.

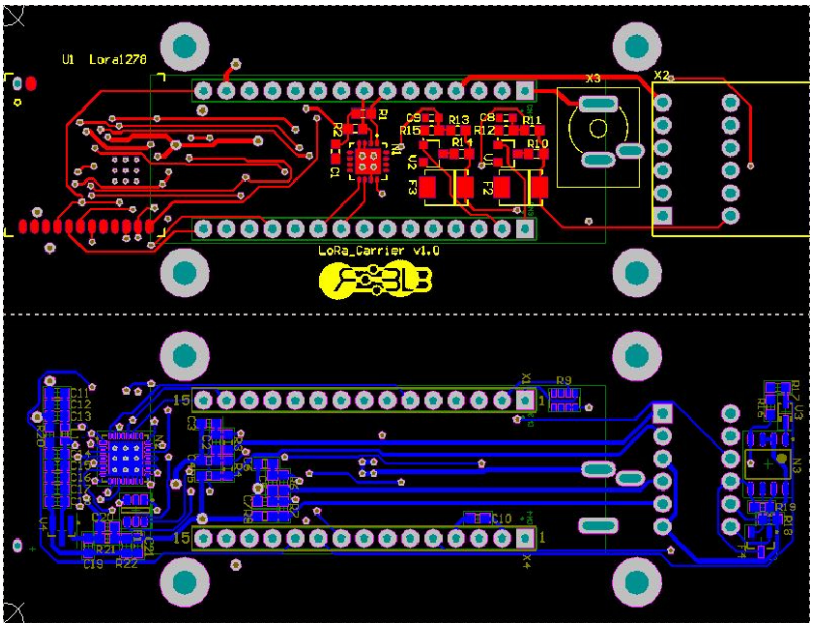

Figura 2.14: Resultado del trazado con polígonos ocultos.

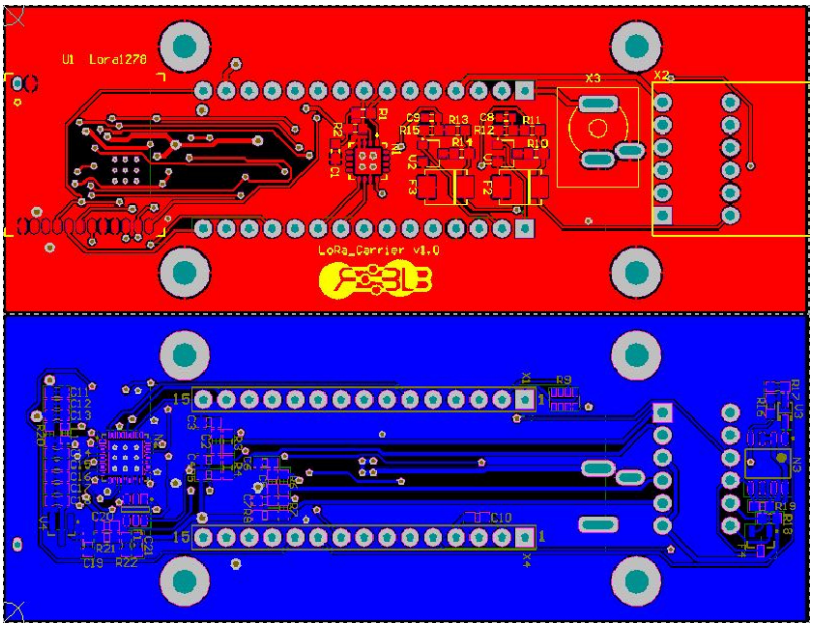

Figura 2.15: Resultado del trazado con polígonos visibles.

2.2.4. Fabricación y montaje

Una vez finalizado el trazado del circuito impreso y superadas las pruebas de detección automática de errores (*ERC* y *DRC*) se debe seguir una serie de pasos previos a la fabricación del diseño:

- Típicamente, el primero de estos pasos consiste en la revisión por parte de otro diseñador del proyecto completo, con el objetivo de asegurar que no existen fallos evitables (polígonos sin regenerar, pistas rutadas descuidadamente, excesos de cobre en zonas muertas...) que el diseñador principal hubiera podido obviar al realizar alguna modificación.

- A continuación, es el turno de generar toda la información precisa para documentar el proyecto correctamente. Esta documentación incluye desde la creación de ficheros PDF que contengan los esquemáticos del diseño y la información referente al ingeniero encargado de su desarrollo, hasta la generación de los ficheros CAM necesarios para la fabricación por parte de terceros del circuito impreso (ficheros Gerber).

- También es usual generar información adicional que pueda ser de ayuda en cualquier otro punto del flujo de trabajo. Aquí podrían incluirse, entre otras: fotografías del modelo 3D del sistema completo, indicaciones de colocación de componentes para facilitar la tarea del montador del circuito o el documento BOM (*Bill Of Materials*) con toda la información necesaria para el proceso de adquisición de los componentes empleados en el diseño.

- Una vez se han finalizado los tres pasos anteriores, puede dar comienzo el proceso de fabricación del circuito impreso, entregando al fabricante toda la documentación que precise para ello.

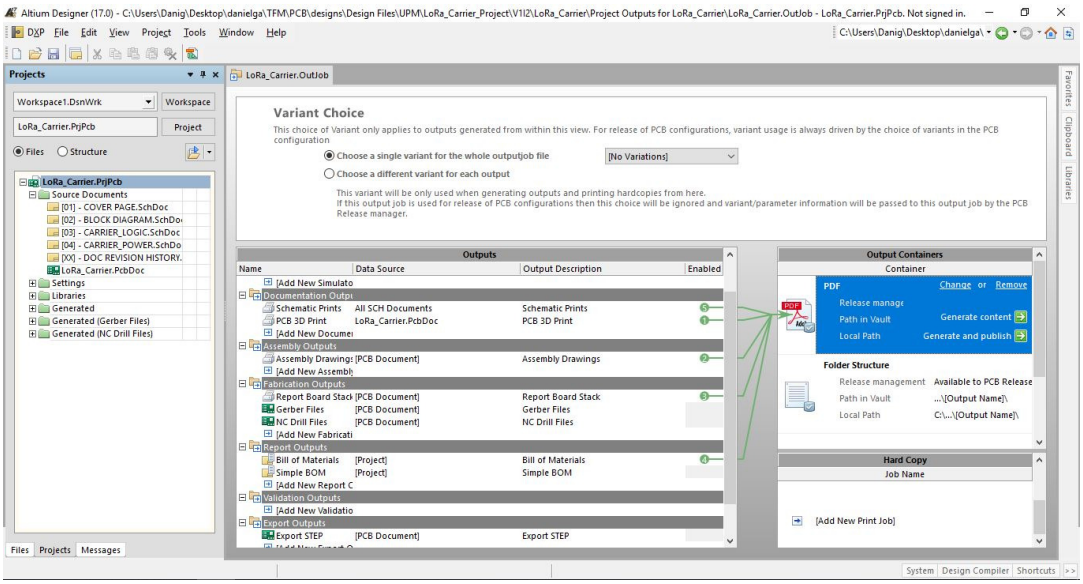

Figura 2.16: Generación automática de documentación del proyecto.

En *Altium Designer*, la generación de la documentación mencionada en los dos pasos intermedios descritos se realiza de forma automática a través de la herramienta de generación de resultados "*Output Jobs*". Esta herramienta permite definir el tipo y formato de documentos a generar con gran variedad de opciones y la posibilidad de incluir variantes del diseño en la documentación. Las Figuras 2.16 y 2.17 muestran esta funcionalidad.

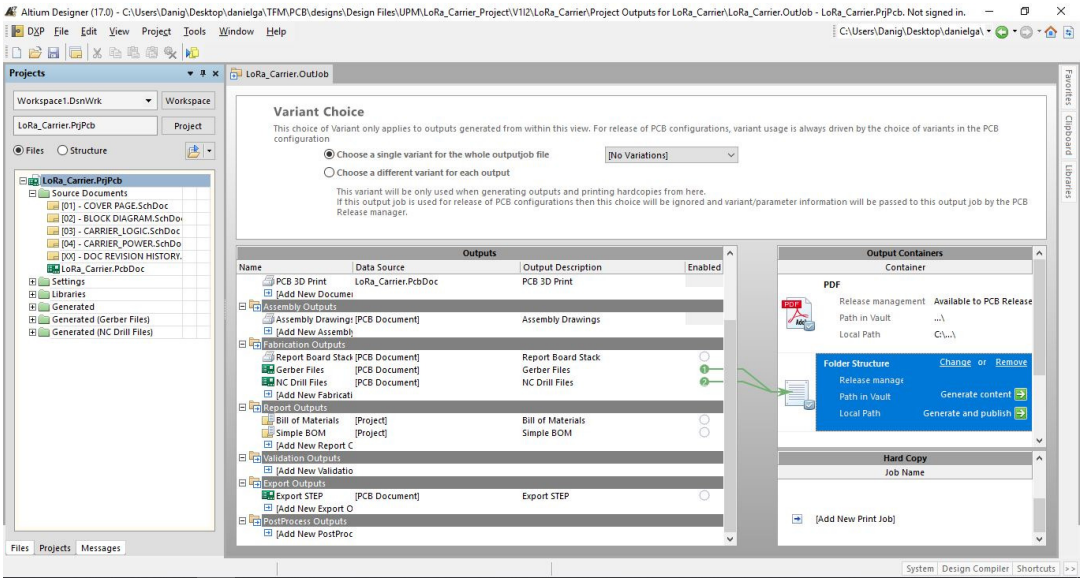

Figura 2.17: Generación automática de ficheros necesarios para la fabricación.

A continuación se muestran los resultados de este proceso de documentación automático entre los que se aprecian: el documento BOM en la Figura 2.18 y los ficheros Gerber (*stencil*, máscara de soldadura, serigrafía y capa de cobre para ambas caras del circuito impreso) de la tarjeta de expansión en la Figura 2.19.

Comment	Description	Designator	Footprint	LibRef	Quantity
100nF 0603	Capacito Ceramic 100nF 10V X7R (-55/125ºC) SMT 0603	C1, C2, C5, C10, C11, C13, C14, C15, C18	C0603(1608)	Cap 100nF 0603	9
10nF 0603	Condensador Ceramico SMT Generico	C3, C4, C6, C7	C0603(1608)	Cap 10nF 0603	4
100nF 0603	Capacitor Ceramic 100nF 50V X7R (-55/125ºC) SMT 0603	C8, C9	C0603(1608) - High_Density	Cap 100nF 0603 50V X7R	2
4,7uF 0603 6.3V	Capacitor Ceramic 4,7uF 6,3V X5R (-55/85ºC) 0603	C12, C16, C17, C19	C0603(1608)	Cap 4,7uF 0603 6.3V X5R	4
1.0uF 16V 0603	Condensador Ceramico SMT X5R	C20, C21	C0603(1608)	Cap 1uF 16V 0603	2
IP4220CZ6,125	USB DUAL ESD PROTECT 6TSOP	F1	SOT26A-6N	IP4220CZ6,125	1
Zener TVS 10V	Zener TVS 10V 600W Unidir SMD	F2, F3	DO-214AA	SMBJ10A	2
2xTVS.Bidir.RS485	2Channel Array Bidir TVS 7V 12V RS485_Port SOT23	F4	SOT95P250X111-3N	CDSOT23-SM712	1
FB120R 2A	FERRITE CHIP 120ohm@100Mhz 2400mA	L1	L0603(1608) - High_Density	Choke 120hm 2A	1
ADT7420UCPZ-RL7	IC SENSR TEMP 16BIT DGTL 16LFCSP	N1	QFN65P400X400X80-16N	ADT7420UCPZ-RL7	1
AD7194BCPZ	IC ADC 24BIT SPI 4.8K 32-LFCSP	N2	QFN50P500X500X80-32N	AD7194BCPZ	1
MAX13487EESA+	Half-Duplex RS-485-/RS-422 Compatible Transceiver with AutoDirection Control	N3	SOIC127P600X175-8N SO8 Maxim Integrated	MAX13487EESA+	1
10K 0603 1%	Resistencia SMT 1% Tolerancia	R1, R2, R5, R8, R16, R17	R0603(1608)	R10K 0603 1%	6
1K 0603 1%	Resistor 1K 1% 0603	R3, R4, R6, R7, R21, R22	R0603(1608)	R1K 0603 1%	6
10K 4x0603	Resistor Array 10K 4x0603	R9	RN0603x4	ResPack 10K 0603x4	1
R523 0603 0.1%	Resistencia SMT 0.1% Tolerancia	R10, R14	R0603(1608)	R523 0603 0.1%	2
7K50 0603 0.1%	Resistencia SMT 0.1% Tolerancia	R11, R13	R0603(1608)	R7K50 0603 0.1%	2
2K49 0603 0.1%	Resistencia SMT 0.1% Tolerancia	R12, R15	R0603(1608)	R2K49 0603 0.1%	2
4K7 0603	Resistor 4K7 0,1W 5% SMT 0603	R18, R19	R0603(1608)	R4K7 0603 5%	2
0R 0603	Resistor 0R 0603 SMT	R20	R0603(1608)	R0R 0603 5%	1
Lora1278_V2.0	433/470MHz 100mW Wireless Transceiver Lora Module	U1	Lora1278	Lora1278_V2.0	1
FDV305N	N-Channel MOSFET, 0.9A, 20V Vgs=2.5V Rds(on)=0.22 Sot23	V1, V2	SOT23	FDV305N	2
1N4148WX	1N4148 0.15A SMD SOD-323	V3	SOD-323	1N4148WX	1
AD1582BRTZ-REEL7	IC VREF SERIES PREC 2.5V SOT23-3	V4	SOT100P237X112-3N	AD1582BRTZ-REEL7	1
PPPC151LFBN-RC	CONN HEADER FMALE 15POS .1" GOLD	X1, X4	PPPC151LFBN-RC	PPPC151LFBN-RC	2
277-14310-ND	TERM BLOCK HDR 12POS 90DEG 2.5MM	X2	PHOENIX_1894846	277-14310-ND	1
CP-0444A-ND	Power Jack Vert 2.0 X 6.	X3	CUI-PJ-044A	PJ-044A	1

Figura 2.18: Documento BOM generado por *Altium Designer*.

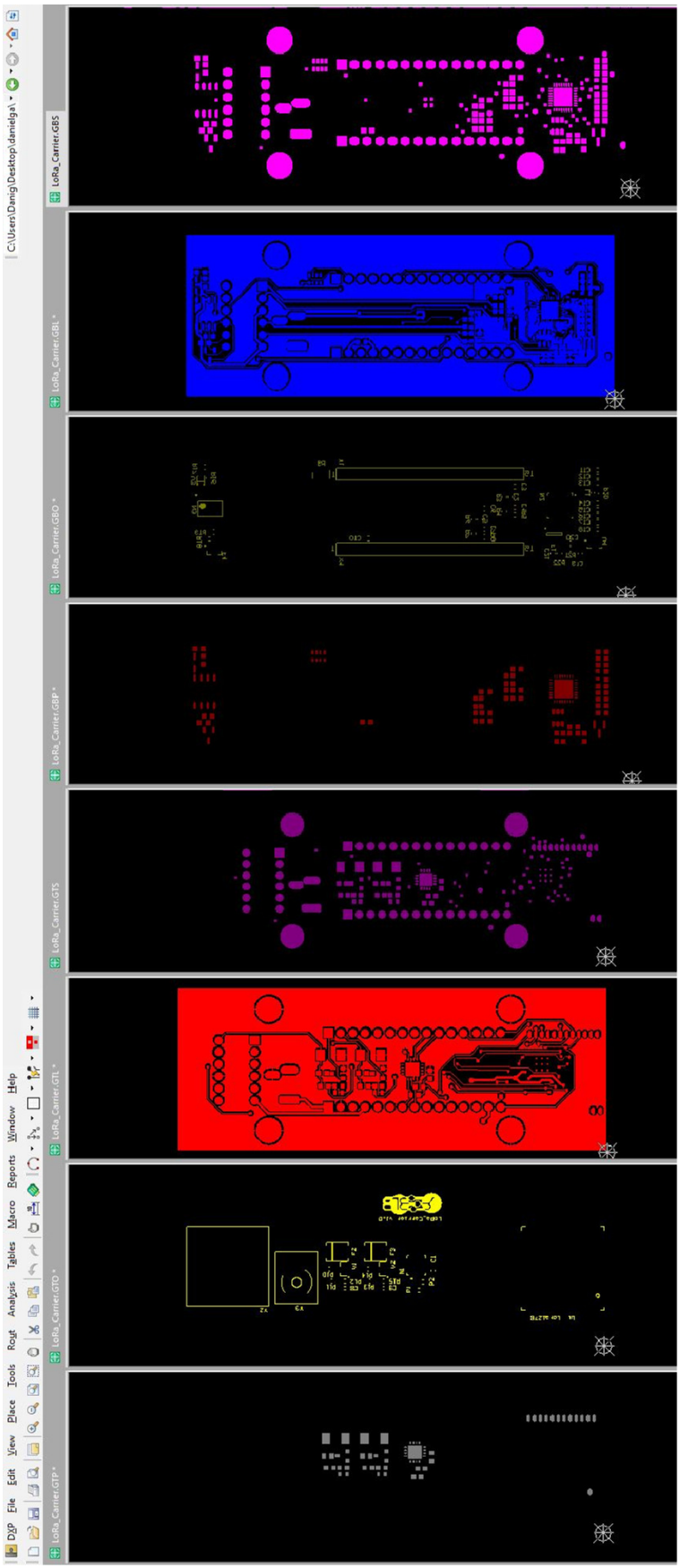

Figura 2.19: Ficheros CAM (Gerbers) generados por *Altium Designer*.

Finalmente, en la Figura 2.20 se observa el modelo 3D de la tarjeta diseñada, acoplada al modelo 3D de la tarjeta NUCLEO-L432KC y en contraposición con el resultado de la fabricación y el montaje, acoplado a la versión real de la misma:

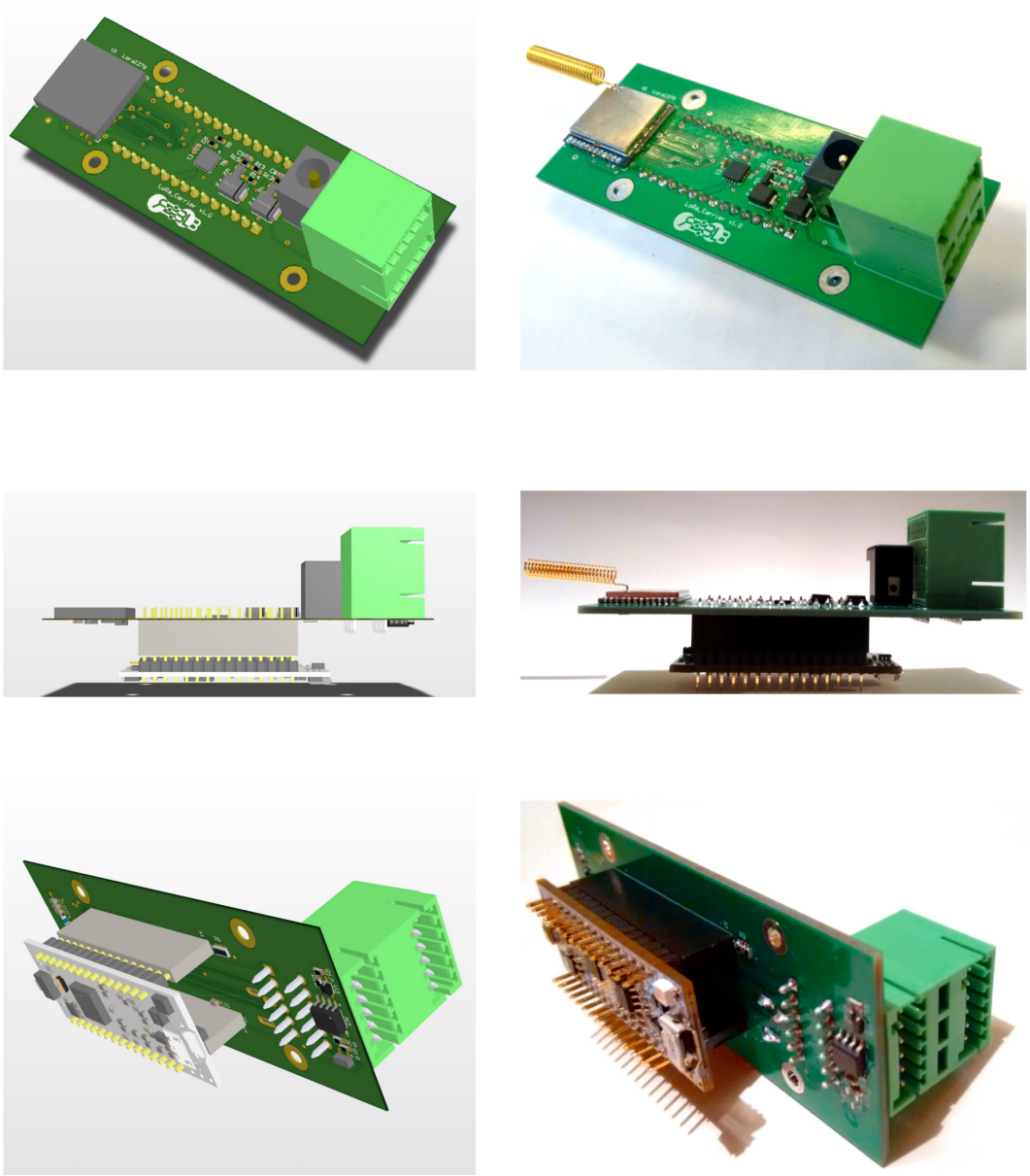

Figura 2.20: Comparativa entre el modelo 3D (izquierda) y el circuito impreso fabricado (derecha).

Capítulo 3

Programación de la tarjeta electrónica de control

Una vez revisado tanto el diseño como la organización del prototipo, el siguiente foco de interés se sitúa sobre la estructura e implementación de la programación *software* de la aplicación. Sin embargo, para entender las decisiones de diseño adoptadas, es necesario introducir al lector a ciertos aspectos básicos de la programación para sistemas empotrados de tiempo real y sus repercusiones en sistemas diseñados para bajo consumo.

Cabe mencionar que, al igual que sucedía en el Capítulo 2 con el diseño de la tarjeta de expansión, algunos de los ficheros empleados en la programación de la tarjeta electrónica de control provienen también de otros proyectos propios del Laboratorio de Robótica y Control de la ETSIT-UPM ("Robolabo") y han sido modificados debidamente para su integración en este sistema.

Así pues, el presente capítulo está organizado de la siguiente manera: en la Sección 3.1 se introduce al lector a los sistemas operativos en tiempo real y, en concreto, al sistema operativo *FreeRTOS* que se emplea en el Trabajo Fin de Máster; posteriormente se presentan las herramientas que se han empleado para la programación y depuración del *software* en la tarjeta de control (Sección 3.2), seguido de una breve introducción al modelado del sistema (Sección 3.3) y de los diversos ficheros que conforman dicha programación (secciones 3.4, 3.5 y 3.6). En la Sección 3.7 se presenta la estrategia de bajo consumo implementada en el proyecto y, por último, en la Sección 3.8 se hace un resumen de la arquitectura *software* del sistema.

3.1. Sistemas operativos de tiempo real (RTOS)

Un RTOS es un tipo de sistema operativo que proporciona una latencia acotada en el servicio de interrupciones y un planificador de procesos que tiene en cuenta restricciones de tiempo real. A un nivel programático, un sistema operativo de tiempo real consiste en un conjunto de líneas de código (referido comúnmente como núcleo o *kernel*) que controla el orden en el que se ejecutan los procesos, cómo es usada la memoria y cómo se comunica información a los dispositivos periféricos y las redes (Lee and Seshia, 2007). Además de éstos, el sistema operativo puede proporcionar otros servicios como la virtualización, alojo, desalojo y protección de memoria, sistema de archivos y servicios de interacción entre procesos como semáforos, mutexes y librerías de paso de mensajes.

3.1.1. *FreeRTOS*

FreeRTOS es una clase de RTOS diseñada para ser suficientemente ligera como para poderse ejecutar en un microcontrolador (aunque su uso no está limitado a aplicaciones en microcontroladores) (Amazon, 2018). *FreeRTOS* proporciona únicamente las funcionalidades primitivas del núcleo de planificación en tiempo real, comunicación entre tareas, temporización y sincronización. Esto implica que la definición más exacta aplicable sería la de *kernel* de tiempo real o ejecutivo de tiempo real. Las funcionalidades adicionales, como la interfaz de consola de comandos o pilas de protocolos de red, pueden incluirse posteriormente con componentes añadidos.

Las ventajas que ofrece el uso de este sistema operativo en este Trabajo Fin de Máster son varias (Barry, 2016), entre las que destacan:

- *FreeRTOS* se encuentra extensamente documentado.
- Cuenta con un planificador de tareas de tiempo real basado en prioridades que sigue una política *Round Robin*, cuyo comportamiento puede ser analizado para valorar la planificabilidad de un sistema.
- Su ligereza permite su ejecución en el microcontrolador seleccionado.
- Es un *software* libre y de código abierto.
- La definición de tareas está estandarizada, lo cual fomenta la portabilidad de la aplicación desarrollada.
- Supone una solución única e independiente de las arquitecturas y herramientas de desarrollo empleadas

Además, a partir de la versión v10.0.0, *FreeRTOS* ofrece soporte al desarrollo de aplicaciones de bajo consumo (Amazon, 2017) a través de la funcionalidad "*tickless idle*" que detiene la interrupción del tick del sistema, colocando al microcontrolador en un estado de bajo consumo siempre y cuando la tarea "*idle*" sea la única disponible para su ejecución. Esta funcionalidad supone una disminución considerable en el impacto que produce la inclusión de un RTOS sobre el consumo del sistema. Esto

es debido a que, típicamente, los sistemas de tiempo real y en concreto *FreeRTOS*, cuentan con una tarea por defecto de prioridad mínima, que entra en ejecución cuando ninguna otra tarea está pendiente de ser ejecutada. Aunque esta tarea no realiza ninguna operación, sí que supone una espera activa para el sistema, en la cual el microprocesador atiende una interrupción periódica del reloj del sistema para alertar al planificador si alguna tarea pasa a estar disponible. En consecuencia, esta situación repercute considerablemente sobre el consumo energético del sistema operativo. La solución implementada a partir de la versión v10.0.0 de *FreeRTOS* alivia esta situación, programando el despertar del sistema para el instante en el que la tarea más cercana pase a estar disponible y eliminándo así la necesidad de generar señal de reloj y atender una interrupción periódica para comprobarlo.

3.2. Desarrollo, programación y depuración del código fuente

El desarrollo del código fuente en este proyecto ha sido posible gracias al uso de numerosas herramientas de código libre que han permitido no sólo la funcionalidad del código, sino también su portabilidad y replicabilidad. El sistema operativo de tiempo real *FreeRTOS* es tan solo una de ellas, por lo que a continuación se presenta un breve resumen de las demás herramientas empleadas y su función.

3.2.1. *ST-LINK* y el paquete *STM32CubeL4*

El código de la aplicación ha sido desarrollado en una máquina con el sistema operativo de base *Linux*, *Ubuntu* (Canonical, 2018), empleando para ello el lenguaje de programación *C* y usando como apoyo la capa de abstracción *hardware* (HAL) proporcionada por el fabricante de la tarjeta electrónica de control en el paquete *software STM32CubeL4* (STMicroelectronics, 2018c).

Una vez generados los ficheros binarios de la aplicación, es necesario el uso de otra herramienta del mismo fabricante para la programación del microcontrolador de la tarjeta de cotrol. Esta herramienta se conoce como *ST-LINK* (STMicroelectronics, 2018a) y se trata de un programador/depurador embebido en la interfaz JTAG de las familias de microcontroladores STM32. Es decir, a través de la propia circuitería de la tarjeta de control y de esta herramienta se programa y depura el código de la aplicación.

Sin embargo, no ha de olvidarse que la máquina objetivo para la que se desarrolla el código tiene una arquitectura diferente a la máquina en la que se ejecuta el compilador. Es por tanto necesario el uso de herramientas de compilación y depuración cruzadas para generar y probar los ficheros binarios anteriormente mencionados.

3.2.2. Proyecto GNU: *GCC* y *GDB*

Para cubrir estas necesidades de compilación y depuración cruzadas se ha optado por hacer uso de dos herramientas de *software* libre, pertenecientes al “Proyecto

GNU"[1]: *GCC* en su versión v5.4.1 y *GDB* en su versión v7.1.1.

GCC es la colección de compiladores del "Proyecto GNU" (Free Software Foundation, 2018a). Esta colección incluye interfaces de compilación para *C, C++, Objective-C, Fortran, Ada* y *Go*, así como librerías para estos lenguajes de programación. Concretamente, en este diseño se ha utilizado la herramienta "arm-none-eabi-gcc", que permite la compilación cruzada para microcontroladores de la familia ARM.

GDB es el depurador del "Proyecto GNU" (Free Software Foundation, 2018b). Esta herramienta permite al desarrollador ver qué está ocurriendo durante la ejecución de un programa o lo que estaba ocurriendo previamente a un error fatal. Soporta los lenguajes de programación *Ada, Assembly, C, C++, D, Fortran, Go, Objective-C, OpenCL, Modula-2, Pascal* y *Rust*.

En concreto, este diseño se ha utilizado la herramienta "arm-none-eabi-gdb", que permite la depuración cruzada para microcontroladores de la familia ARM. *GDB* permite, principalmente:

- Iniciar la ejecución del programa.
- Detener la ejecución del programa ante unas condiciones específicas.
- Examinar lo que ha ocurrido una vez detenida la ejecución.
- Realizar cambios en el programa para experimentar con los efectos de un cambio sobre un determinado error.

3.2.3. *OpenOCD*

Finalmente, para hacer posible la depuración del código desde *Ubuntu* es necesaria una última herramienta. *OpenOCD* (*Open On-Chip Debugger*) es una herramienta de código libre que sirve de interfaz con el puerto JTAG de un circuito de depuración *hardware*. *OpenOCD* (Rath, 2018) proporciona depuración y programación interna para el sistema empotrado objetivo, permitiendo cargar los binarios en las memorias NOR y NAND FLASH del dispositivo.

[1]Proyecto GNU`https://www.gnu.org/gnu/thegnuproject.es.html`

3.3. Modelado del comportamiento de los nodos

3.3.1. Máquinas de estados finitos (FSM) extendidas

Una máquina de estados es un modelo de un sistema con dinámicas discretas que, a cada reacción del sistrema, mapea las entradas con las salidas, donde dicho mapeado puede depender del estado actual del sistema. Una máquina de estados finitos (FSM) es una máquina de estados donde el conjunto de posibles estados es finito (Lee and Seshia, 2007).

Las razones para modelar el comportamiento de un *software* como una máquina de estados finitos son varias, pero entre ellas destacan la idoneidad para modelar comportamientos discretos, la posibilidad de implementación de manera semiautomática y la posibilidad de verificar formalmente el modelo para asegurar que cumple los requisitos.

Sin embargo, cuando el número de estados crece en demasía, como ocurre al incluir variables temporales en el modelo, la notación gráfica usada en máquinas de estados finitos se torna poco manejable. Una máquina de estados extendida soluciona este problema aumentando el modelo de FSM con variables que pueden ser leídas y escritas como parte de una transición entre estados. En la Figura 3.1 se muestra la notación general para máquinas de estados extendidas, donde se pueden apreciar los estados (burbujas) y las transiciones (flechas). En las transiciones se puede observar que la notación empleada es la siguiente:

guarda / acción
modificación de variables

La "guarda" es la condición que debe cumplirse al evaluar las entradas para que se ejecute la transición. En el ejemplo de la Figura 3.1, la guarda de la transición situada en la parte superior se cumple si, en el momento de la evaluación, las señales de entrada (puras) "*up*" y "*down*" están presentes y el valor de la variable "c" es menor que "M". La "acción" es el resultado sobre las salidas de ejecutar la transición. En el caso de la transición mencionada, la señal de salida "*count*" incrementa su valor en una unidad. Por último, los valores de las variables son modificados expresamente en las transiciones. En el caso anterior, si se ejecuta la transición, se incrementa el valor de la variable "c" en una unidad.

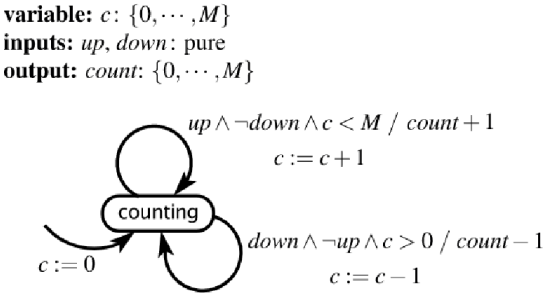

Figura 3.1: Notación gráfica de FSM extendidas (Lee and Seshia, 2007).

3.3.2. Modelo de comportamiento

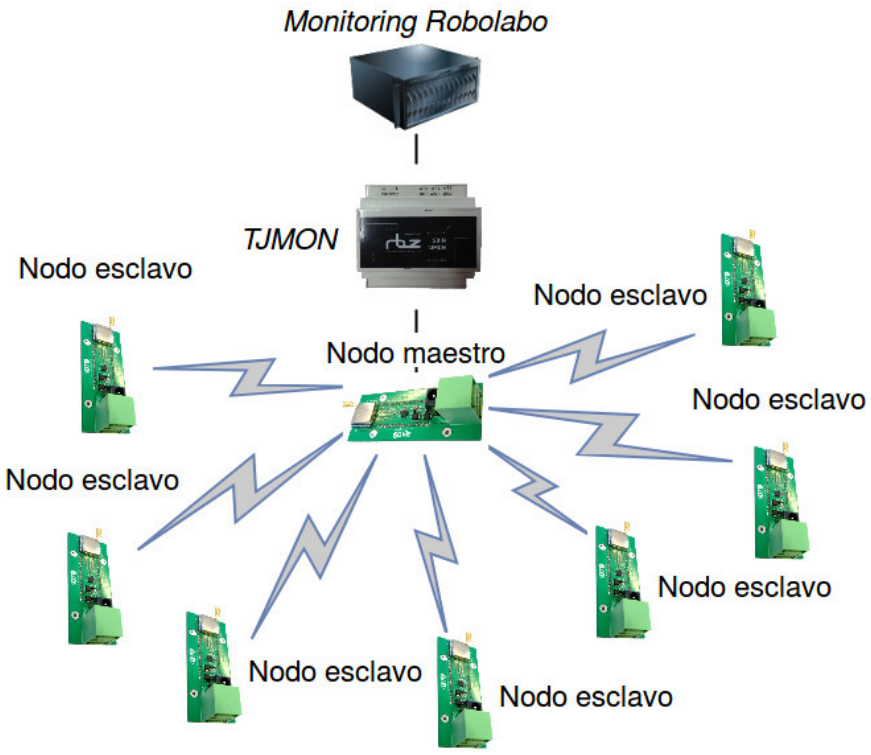

Figura 3.2: Esquema de red inalámbrica punto-multipunto.

En este Trabajo Fin de Máster se ha desarrollado una aplicación de red inalámbrica punto-multipunto con una arquitectura maestro-esclavo, donde existe un único nodo maestro y múltiples nodos esclavos, tal y como se observa en la Figura 3.2. En esta aplicación, el nodo maestro realiza además la función de puerta de enlace entre la red inalámbrica y la red cableada *Monitoring Robolabo*, almacenando los datos enviados por los nodos esclavos (espacialmente distribuidos) y transfiriendo esta información bajo demanda al dispositivo *TJMON* por su interfaz RS-485 Modbus-RTU. Así, siguiendo la metodología de modelado recién introducida, se presentan a continuación las notaciones gráficas de los modelos realizados para describir el comportamiento de los nodos, tanto maestro como esclavos.

El nodo maestro se ha modelado como dos FSM extendidas concurrentes, que describen su comportamiento como maestro de la red inalámbrica punto-multipunto (Figura 3.3) y como esclavo en la red cableada Modbus-RTU (Figura 3.4). Como se puede observar, por un lado el nodo en su papel de maestro de la red inalámbrica, el nodo debe esperar continuamente las transmisiones de los esclavos de la red, almacenando los datos recibidos en tablas que eventualmente transferirá al dispositivo *TJMON*. Además, para asegurar cierta fiabilidad en la red inalámbrica, el nodo maestro debe transmitir un mensaje de confirmación ante cada trama recibida, confirmando el identificador del esclavo emisor, el número de trama y el tipo de

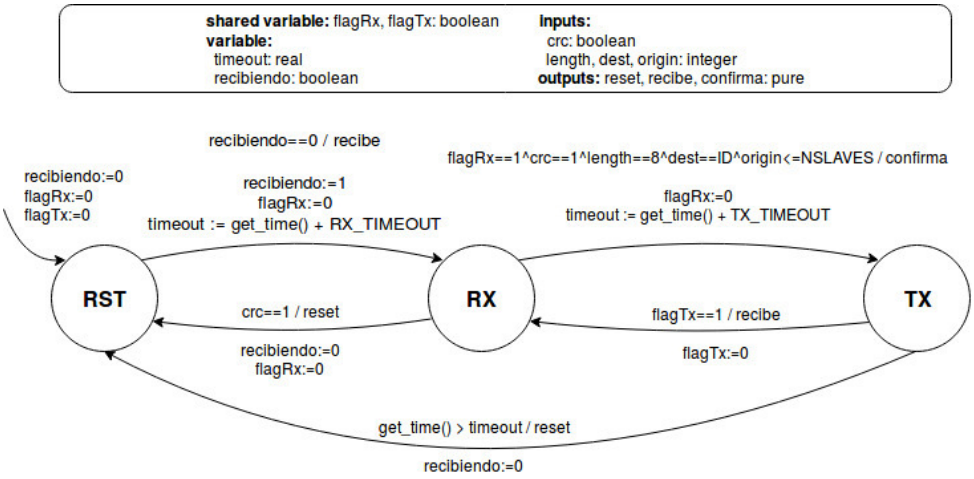

Figura 3.3: FSM que modela el comportamiento inalámbrico del nodo maestro.

dato recibido. En esta máquina de estados son destacables las variables compartidas "*flagRx*" y "*flagTx*", que indican cuándo se ha efectuado una transmisión o recepción por radio. Estas variables, aunque no ha sido reflejado en la Figura 3.3, son asertadas por la FSM que modelaría el comportamiento del controlador del módulo radiotransceptor. Por otro lado, en su papel de esclavo de la red cableada Modbus-RTU debe atender las peticiones del dispositivo maestro de la red (*TJMON*) y, en caso de darse una petición, responder con la información recibida de los nodos esclavos.

Respecto al comportamiento de los nodos esclavos, se han modelado como una FSM jerárquica, donde el estado "*COMM*" modela la transmisión inalámbrica de los datos de la aplicación. Como se puede apreciar en la Figura 3.5, la FSM describe un comportamiento cíclico, a lo largo del cual el nodo permanece en estado de reposo durante un tiempo ("*SLEEP_TIME*") equivalente a un minuto en la aplicación implementada, se activa para obtener las medidas de sus cuatro sensores, transmite la información recogida y vuelve al estado de reposo. Cabe destacar la existencia de una variable compartida, "tramas" que representa el número de nuevas tramas de datos pendientes de ser enviadas. Cada dato sensado se empaqueta en una trama independiente, por lo que esta variable se incrementa dentro del estado "*SENSE*" hasta alcanzar un valor igual al número de sensores.

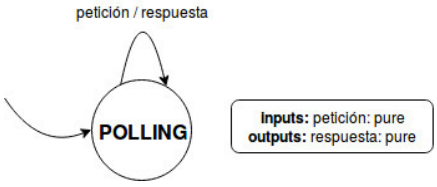

Figura 3.4: FSM que modela el comportamiento en la red Modbus-RTU del nodo maestro.

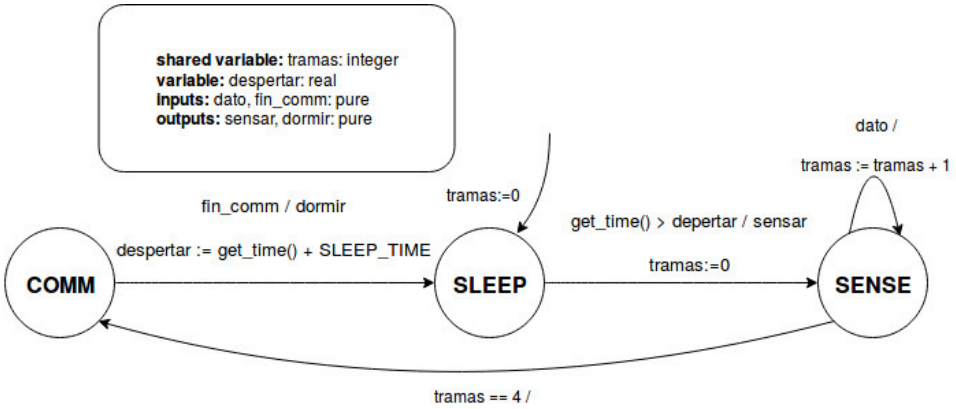

Figura 3.5: FSM jerárquica que modela el comportamiento de los nodos esclavos.

La transmisión inalámbrica se realiza en el estado "*COMM*". En este estado el microcontrolador debe actuar sobre el módulo radiotransceptor, cargando en su cola de transmisión las tramas que contienen la información sensada y colocando al módulo en los distintos estados de transmisión, recepción o reposo en cada fase de la comunicación. Una vez transmitidas todas las tramas pendientes, el nodo retorna a su estado de bajo consumo para ahorrar energía hasta la siguiente medición. Este proceso también ha sido modelado como una FSM extendida, en la cual reaparecen las variables compartidas "*flagRx*" y "*flagTx*", que describen el mismo comportamiento que en el caso del nodo maestro.

Como se puede apreciar en la Figura 3.6, el módulo radiotransceptor parte de un estado inicial de reposo "*RST*" donde el consumo es mínimo. Cuando hay tramas pendientes de transmitir se pasa al estado "*TX*", se carga la primera trama pendiente en la cola de transmisión y se habilita la transmisión a través del controlador del módulo, el cual indicará el momento en el que la transmisión se haga efectiva escribiendo un '1' en la variable compartida "transmitido". Si transcurre un tiempo determinado ("*TIMEOUT_TX*") sin que se produzca dicha escritura, se asume que se ha producido un error en la transmisión de la trama y se reinicia el proceso volviendo al estado "*RST*". En caso contrario, se pasa al estado "*RX*" en el cual se aguarda la trama de confirmación enviada por el maestro de la red, evento que el controlador del módulo señaliza escribiendo un '1' en la variable compartida "recibido". De nuevo, de no producirse dicha escritura en un tiempo determinado ("*TIMEOUT_RX*"), se asume que se ha producido un error en la transmisión y se repite el envío de la trama volviendo al estado "*RST*". Si por el contrario se recibe la confirmación, se decrementa la variable compartida "tramas" y se pasa al estado "*RST*". Cuando el valor de "tramas" se hace cero, se activa la salida "fin_comm" finalizando el proceso de transmisión hasta el siguiente ciclo.

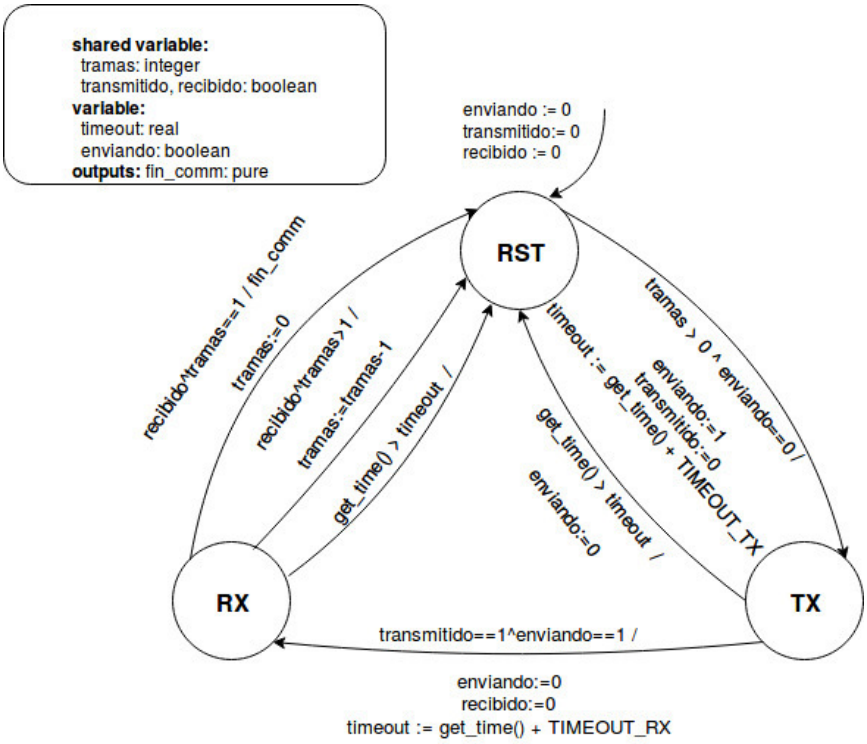

Figura 3.6: Expansión del estado "*COMM*" de la FSM de la Figura 3.5.

3.3.3. Implementación de los modelos

La implementación del modelo de máquina de estados finitos empleada (Moya, 2017) se compone de un par de ficheros (*fsm.c* y *fsm.h*) de código libre escritos en el lenguaje de programación *C* y publicados bajo licencia GNU GPLv3 como parte del repositorio de ejemplos didácticos del Grupo de Investigación de la ETSIT-UPM, *GreenLSI*. Se trata de una implementación orientada a datos, con una tabla de transiciones explícita que hace que el modelo implementado sea más fácil de modificar y más eficiente al reducir los fallos en caché y los saltos.

Basados en esta implementación se han escrito los ficheros principales de la aplicación para maestro y esclavos, en ambas versiones de código bajo el nombre de fichero "*application.c*". En ellos, se realiza la definición, creación y disparo de las FSM presentadas en el apartado anterior, así como la implementación de las funciones de guarda y de acción correspondientes a las transiciones de dichas máquinas de estados. Son estas funciones las que, como se adelantaba al comienzo de este capítulo, requieren de la existiencia de controladores específicos para los distintos módulos del diseño (radiotransceptor, ADC externo, sensor de temperatura integrado) y de librerías que permitan manejar los datos generados por ellos (cálculo de temperatura, lectura de voltajes y corrientes, ahorro energético).

Teniendo esto en cuenta, a continuación se exponen aquellos controladores, librerías y estrategias que han hecho posible la implementación de los modelos de comportamiento de esta aplicación.

3.4. Controlador del módulo *Lora1278*

Como era de esperar, la selección del módulo *Lora1278* en el Capítulo 2 no ha sido aleatoria. En el Apartado 2.1.2 se ha visto que este componente cumple con las necesidades de la aplicación y ofrece una interfaz que facilita su uso al diseñador no familiarizado con la teoría de campos y ondas de telecomunicación. Sin embargo, existe un motivo adicional para la elección de este módulo en particular. Dicho motivo es la existencia de un Trabajo Fin de Máster previo, llevado a cabo en el Laboratorio de Robótica y Control ("Robolabo"), en el cual se desarrolló un controlador de radio para este dispositivo.

Este desarrollo previo (Rodríguez Munca, 2016) permite el control y la gestión del módulo radiotransceptor a través de una serie de definiciones y funciones recogidas en dos pares de ficheros, "*radio.c(pp)/radio.h(pp)*" y "*halLora.c(pp)/halLora.h(pp)*", escritos en los lenguajes de programación *C* y *C++*. Esta duplicidad de lenguajes se debe a que el controlador en cuestión fue implementado para dos plataformas independientes: *Arduino* y *SDIN*. La relevancia de este último dato radica en dos puntos fundamentales:

- El microcontrolador empleado en la plataforma *Arduino* está fabricado por *Atmel* y se basa en la familia de microprocesadores *AVR ATmega*.
- El microcontrolador empleado en la plataforma *SDIN* está fabricado por *STMicroelectronics* y se basa en la familia de microprocesadores *ARM Cortex-M3*.

En ambos casos, bien por estar escrito en otro lenguaje de programación (*Arduino*), o bien por estar basado en una capa de abstracción *hardware* (HAL) diferente (*SDIN*), era necesario migrar el código del controlador a la nueva plataforma. Esta tarea ha sido realizada con éxito en este Trabajo Fin de Máster, resultando en dos pares de ficheros escritos en lenguaje *C* y basados tanto en el controlador original, como en la capa de abstracción harware (HAL) del paquete *STM32CubeL4*. Además, se han incluido del mismo paquete los ficheros "*stm32l4xx_hal_spi.c*" y "*stm32l4xx_hal_spi.h*" y sus dependecias, que permiten al controlador operar sobre la interfaz SPI del microcontrolador.

Este nuevo controlador permite, por tanto, la gestión de los recursos del módulo *Lora1278*, la comunicación con el mismo a través de su interfaz SPI y la inclusión, de forma sencilla, de estas funcionalidades en el esquema de funciones de entrada y salida definido por la implementación del modelo de FSM empleada.

3.5. Controlador del sensor de temperatura integrado

El sensor de temperatura integrado en la tarjeta de expansión ("ADT7420") se comunica a través de la interfaz estandarizada de comunicación serie I2C. En un principio, este hecho debería haber facilitado enormemente la gestión del componente, ya que la mayoría de las plataformas comerciales que incluyen soporte *hardware*

para esta interfaz, ofrecen a su vez las librerías de gestión asociadas a la misma y, efectivamente, así sucede en el paquete *STM32CubeL4*.

A pesar de ello, tras una primera aproximación al uso de esta librería, se observó que el cálculo de los tiempos de espera de los mensajes I2C implementado se realiza empleando esperas activas, algo totalmente opuesto al objetivo de reducción de consumo planteado. Ante esta tesitura, se ha optado por reescribir esta librería, aprovechando su estructura y definiciones básicas pero sustituyendo las esperas activas por esperas gestionadas por el sistema operativo en tiempo real. De esta forma, durante esos tiempos podrán ser ejecutadas otras tareas o se podrá entrar en estado de bajo consumo en caso de no existir ninguna tarea pendiente de ejecución.

Además de los cambios anteriores, se han definido dos nuevas funciones de lectura y escritura de registros por I2C, basadas en las funciones básicas de la librería editada. Estas funciones surgen de la necesidad de satisfacer el protocolo de escritura y lectura de registros implementada en este dispositivo, cuyo funcionamiento se detalla en la hoja de características del sensor (Devices, 2017a).

En resumen, el proyecto cuenta con un controlador para el sensor "ADT7420" que permite al microcontrolador la gestión completa del dispositivo, mediante comunicación a través de su interfaz I2C, incluyendo: reinicio, configuración del modo de medida y lectura y escritura de registros concretos del sensor. Este controlador se compone de los ficheros de código *C* "*stmX_hal_i2c.c*" y "*stmX_hal_i2c.h*".

3.6. Migración del código de la plataforma *SDIN_Therm*

El control de los componentes de la tarjeta de expansión y el procesamiento de los datos provenientes de sus entradas conforman otro punto en el que la reutilización de *software* ha tenido especial relevancia en este proyecto. En realidad esta declaración no supone sorpresa alguna, teniendo en cuenta que en el Apartado 2.1.3 se ha hecho referencia a la reutilización de componentes empleados en la plataforma *SDIN_Therm* para el diseño de la tarjeta de expansión. Así pues, se exponen a continuación los elementos de *firmware* y *software* diseñados para la plataforma *SDIN_Therm* y migrados para su uso en este Trabajo Fin de Máster.

3.6.1. Implementación del protocolo Modbus-RTU

Modbus-RTU es un protocolo serie abierto (RS-485) basado en una arquitectura maestro/esclavo. Este protocolo, ampliamente utilizado a nivel industrial, se emplea para interconectar equipos de campo, como sensores, actuadores y controladores (Organization, 2012).

Este protocolo se emplea dentro de la red *Monitoring Robolabo* en la comunicación entre la plataforma *TJMON* y las distintas plataformas de la familia *SDIN*. Evidentemente, la plataforma *SDIN_Therm* cuenta con una implementación de este protocolo que ha sido migrada para su empleo en este Trabajo Fin de Máster. Sobre la

implementación, está basada en el proyecto de software libre "*FreeModbus Library*" (Walter, 2010) que consiste en una implementación portable de este protocolo en código *C*, en la cual solo es preciso añadir un fichero ("*modbus_main.c*") con las rutinas de atención a las peticiones del maestro en este caso.

Además de esta implementación, cabe mencionar la inclusión de los ficheros "*stm32l4xx_hal_uart.c*" y "*stm32l4xx_hal_uart.h*" del paquete *STM32CubeL4* que permiten a este controlador operar sobre la interfaz UART del microcontrolador.

3.6.2. Controlador del conversor analógico-digital externo

El conversor analógico-digital "AD7194" es uno de los componentes que requieren de una gestión y un control especialmente cuidados, como viene detallado en su ficha técnica (Devices, 2017c), y por lo tanto del desarrollo de un controlador apropiado.

Si bien ya existía un controlador para este dispositivo en el código fuente de la plataforma *SDIN_Therm*, al igual que ocurría con el controlador del radiotransceptor, éste debía ser migrado a la nueva plataforma para poder formar parte del código fuente de este proyecto. Además, se ha ampliado la funcionalidad del controlador existente para permitir la conversión en modo pseudo-diferencial tal y como se describe en el documento de referencia del dispositivo. Ambas tareas han sido llevadas a cabo con éxito en este Trabajo Fin de Máster, resultando en una pareja de ficheros ("*extADC.c*" y "*extADC.h*") escritos en lenguaje *C* y basados tanto en el controlador existente como en la capa de abstracción *hardware* (HAL) del paquete *STM32CubeL4*.

En consecuencia, el proyecto cuenta con un controlador para el conversor "AD7194" que permite al microcontrolador la gestión completa del dispositivo, mediante comunicación a través de su interfaz SPI, incluyendo: configuración del modo de conversión, selección, filtrado y lectura de canales en modos diferencial y pseudo-diferencial, reinicio completo y transición a modo de bajo consumo.

3.6.3. Librería de gestión de termopares

Por último, para procesar las lecturas de termopares y, a partir de los valores entregados por el conversor analógico-digital externo y el sensor de temperatura integrado, calcular la temperatura corregida expresada en grados centígrados, es necesario el uso de una librería adicional de gestión de termopares.

Esta librería, tomada del código fuente de la plataforma *SDIN_Therm* y migrada para su compatibilidad con este proyecto, consiste en una pareja de ficheros escritos en lenguaje *C* ("*thermocouple.c*" y "*thermocouple.h*"). Ofrece funciones de alto nivel que devuelven datos de temperatura en décimas de grado centígrado, para diferentes configuraciones tanto del conversor analógico-digital, como de precisión del sensor de temperatura integrado.

Para ello, esta librería hace uso de los controladores correspondientes presentados anteriormente, así como de una serie de tablas, incluidas en forma de *arrays*

bidimensionales, que recogen todos los posibles valores de tensión devueltos por el conversor analógico-digital en el rango de temperaturas asociado al tipo de termopar empleado (K, J, T).

3.7. Estrategia de bajo consumo

A lo largo de los capítulos 2 y 3 se ha estudiado la reducción del consumo energético desde una perspectiva de bloque, analizando las contribuciones individuales que cada uno de los componentes del sistema (*hardware* o *firmware*) aportan a la cifra de consumo total. Esta Sección, en cambio, versa acerca de la estrategia de bajo consumo tomada con un enfoque global de sistema. Esta estrategia está basada fundamentalmente en dos puntos:

- El microcontrolador seleccionado y su conjunto de modos de bajo consumo y reguladores de tensión independientes.
- La aplicación final del sistema y las opciones de ahorro energético existentes en su diseño.

3.7.1. Bajo consumo en el microcontrolador "STM32L432KC"

Como ya se ha comentado en el Apartado 2.1.1, el microcontrolador "STM32L432KC" ha sido diseñado con un conjunto exhaustivo de modos de ahorro energético e integra varios reguladores de tensión independientes, que permiten el diseño de aplicaciones para bajo consumo mediante la aplicación de técnicas de escalado dinámico en tensión (DVS) (Pillai and Shin, 2001). Estas funcionalidades vienen detalladas en la hoja de características del microcontrolador (STMicroelectronics, 2018d).

En cuanto a las fuentes de alimentación, el microcontrolador "STM32L432KC" cuenta con dos reguladores lineales de tensión: el regulador principal (MR) y el regulador de bajo consumo (LPR).

- El MR se utiliza en los modos "*Run*", "*Sleep*" y "*Stop 0*".
- El LPR se utiliza en los modos "*Low-Power Run*", "*Low-Power Sleep*", "*Stop 1*" y "*Stop 2*". También se emplea para alimentar la memoria SRAM2 de 16KB en el modo "*Standby* con retención de memoria SRAM2".
- Ambos reguladores se apagan en los modos "*Standby*" y "*Shutdown*" dejando sus salidas en alta impedancia y produciendo así un consumo prácticamente nulo.

En cuanto a los modos de bajo consumo, el microcontrolador "STM32L432KC" proporciona siete modos de bajo consumo que permiten lograr el mejor compromiso entre bajo consumo, tiempo de arranque, periféricos disponibles y fuentes de reactivación. Por defecto, el microcontrolador se encuentra en el modo activo "*Run*" tras un reinicio del sistema o de la alimentación. Queda a disposición del usuario seleccionar alguno de los modos de bajo consumo que se describen a continuación:

- **Modo "*Sleep*":** en este modo, solo se detiene la CPU. Todos los periféricos se mantienen operativos y pueden reactivar la CPU a través de interrupciones o eventos.

- **Modo "*Low-power Run*":** en este modo, el LPR pasa a ser la fuente de alimentación de la lógica del circuito (V_{CORE}) para minimizar la corriente de operación del regulador. En esta situación, el código puede ejecutarse desde SRAM o desde Flash y la frecuencia de la CPU se limita a 2MHz. Los periféricos con entrada de reloj independiente pueden utilizar el oscilador "HSI16".

- **Modo "*Low-power Sleep*":** este modo solo es accesible desde el modo "*Low-power Run*". En él, sólo se detiene la CPU y, al producirse una reactivación por evento o interrupción, el sistema retorna al modo "*Low-power Run*".

- **Modos "*Stop 0*", "*Stop 1*" y "*Stop 2*":** los modos "*Stop*" permiten alcanzar el mínimo consumo sin dejar de retener el contenido de la memoria SRAM y los registros. Todos los relojes del dominio "V_{CORE}" se detienen y los relojes derivados del PLL y los osciladores "MSI" y "HSI16" se deshabilitan. Los osciladores "LSE" o "LSI" siguen funcionando y el reloj del sistema a la salida de cualquiera de estos modos puede seleccionarse entre los osciladores "MSI" (hasta 48MHz) o "HSI16". La Figura 3.7 presenta un resumen de las características disponibles en cada uno de estos tres modos.

Mode	Regulator[1]	CPU	Flash	SRAM	Clocks	DMA & Peripherals[2]	Wakeup source	Consumption[3]	Wakeup time
Stop 0	MR Range 1	No	OFF	ON	LSE LSI	BOR, PVD, PVM RTC, IWDG COMPx (x=1,2) DAC1 OPAMPx (x=1) USARTx (x=1,2)[6] LPUART1[6] I2Cx (x=1,3)[7] LPTIMx (x=1,2) *** All other peripherals are frozen.	Reset pin, all I/Os BOR, PVD, PVM RTC, IWDG COMPx (x=1..2) USARTx (x=1,2)[6] LPUART1[6] I2Cx (x=1,3)[7] LPTIMx (x=1,2) USB_FS[8] SWPMI1[9]	108 µA	2.4 µs in SRAM 4.1 µs in Flash
	MR Range 2							108 µA	
Stop 1	LPR	No	Off	ON	LSE LSI	BOR, PVD, PVM RTC, IWDG COMPx (x=1,2) DAC1 OPAMPx (x=1) USARTx (x=1,2)[6] LPUART1[6] I2Cx (x=1,3)[7] LPTIMx (x=1,2) *** All other peripherals are frozen.	Reset pin, all I/Os BOR, PVD, PVM RTC, IWDG COMPx (x=1..2) USARTx (x=1,2)[6] LPUART1[6] I2Cx (x=1,3)[7] LPTIMx (x=1,2) USB_FS[8] SWPMI1[9]	4.34 µA w/o RTC 4.63 µA w RTC	6.3 µs in SRAM 7.8 µs in Flash
Stop 2	LPR	No	Off	ON	LSE LSI	BOR, PVD, PVM RTC, IWDG COMPx (x=1..2) I2C3[7] LPUART1[6] LPTIM1 *** All other peripherals are frozen.	Reset pin, all I/Os BOR, PVD, PVM RTC, IWDG COMPx (x=1..2) I2C3[7] LPUART1[6] LPTIM1	1.3 µA w/o RTC 1.4 µA w/RTC	6.8 µs in SRAM 8.2 µs in Flash

Figura 3.7: Cuadro resumen de los modos de bajo consumo *Stop*. (STMicroelectronics, 2018d)

- **Modo "*Standby*":** este modo sirve para lograr el mínimo consumo de energía con reinicio ante caída de tensión (BOR). Tras entrar al modo "*Standby*", los contenidos de la memoria SRAM1 se pierden. Opcionalmente, se pueden retener los contenidos de la memoria SRAM2. El dispositivo sale del modo "*Standby*" cuando se produce un reinicio externo (pin NRST), un reinicio por *watchdog* (IWDG), un evento de reactivación en el pin WKUP, un evento de RTC (alarma, activación periódica) o cuando se detecta un fallo en el reloj "LSE". El reloj del sistema tras la reactivación es el oscilador "MSI" a 8MHz.

- **Modo "*Shutdown*":** este modo permite alcanzar el mínimo consumo posible. En él, se apaga el regulador interno y toda la lógica del circuito, así como el PLL, los osciladores "HSI16", "MSI" y "LSI". El reinicio ante caída de tensión (BOR) no está disponible en este modo. Todos los contenidos de las memorias SRAM1 y SRAM2 se pierden. El dispositivo sale del modo "*Shutdown*" cuando se produce un reinicio externo (pin NRST), un evento de reactivación en el pin WKUP o un evento de RTC (alarma, activación periódica). El reloj del sistema tras la reactivación es el oscilador "MSI" a 4MHz.

Como se ha indicado, el acceso a estos modos y la selección de las distintas fuentes de alimentación, reloj y reactivación es una labor que queda a cargo del usuario. Para facilitar esta labor, *STMicroelectronics* pone a disposición del desarrollador, a través del paquete *STM32CubeL4*, librerías específicas de gestión de los modos de bajo consumo del microcontrolador y de los periféricos asociados que permiten llevar a cabo las tareas antes mencionadas. De nuevo, estas librerías están escritas en lenguaje *C* y basadas en la capa de abstracción *hardware* (HAL) del fabricante.

3.7.2. Opciones de ahorro energético en un nodo de una red de monitorización

En el diseño de aplicaciones de bajo consumo para sistemas empotrados existen dos posibles estados del sistema: activo e inactivo. Se dice que el sistema está en estado activo cuando se encuentra ejecutando tareas propias de la aplicación. Por el contrario, cuando no existen tareas que ejecutar y el microcontrolador debe preservar energía, se dice que el sistema está en estado inactivo (Sistemas Empotrados Avanzados, 2018). Esta transición entre estados viene determinada por la aplicación y está claramente marcada en sistemas cuya aplicación tiene un comportamiento periódico. Parece intuitivo entonces razonar que la mejor estrategia de bajo consumo posible es la de minimizar el tiempo que el sistema pasa en estado activo y maximizar el tiempo que pasa en estado inactivo. En la Figura 3.8 se observa un ejemplo de una estrategia siguiendo este razonamiento.

La clave de la estrategia de bajo consumo seguida en este Trabajo Fin de Máster se encuentra en la aplicación final del sistema. Como ya se ha introducido en el Capítulo 1, la aplicación final consiste en una red de monitorización inalámbrica punto-multipunto, en la que existe un nodo maestro de la red inalámbrica que periódicamente recibe las medidas tomadas por los nodos esclavos.

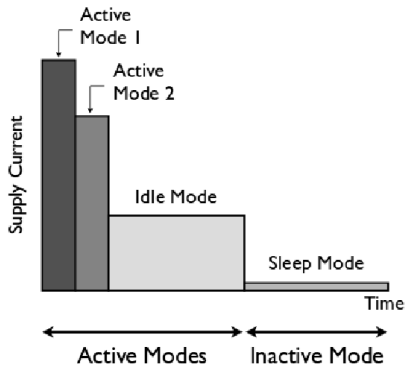

Figura 3.8: Ejemplo de estrategia de bajo consumo. (Sistemas Empotrados Avanzados, 2018)

En esta aplicación, cada nodo esclavo debe ejecutar dos acciones en su modo activo: sensar y transmitir. La duración de estas acciones depende, mayormente, de los tiempos de espera a la confirmación de las tramas enviadas por parte del nodo maestro de la red inalámbrica. Estos tiempos son variables y se ven influenciados por el número de nodos de la red, la distancia de transmisión entre esclavo y maestro, así como por otros factores que pudieran producir errores de transmisión que derivasen en la necesidad de retransmitir tramas. Por otro lado, tras la transmisión de las medidas, el nodo pasa al estado inactivo durante un minuto antes de comenzar el ciclo de nuevo. Teniendo esto en cuenta y a falta de realizar pruebas que lo confirmen, cabe esperar que en una red pequeña (menos de diez dispositivos esclavos) no existan grandes retardos y, en comparación con los sesenta segundos del modo inactivo, el tiempo transcurrido en el modo activo sea despreciable.

Así pues, la estrategia implementada en este Trabajo Fin de Máster ha sido la de minimizar el estado activo en los nodos esclavos, ejecutando las tareas de sensado y comunicación inalámbrica a la máxima frecuencia de reloj disponible (80MHz), y maximizar el ahorro energético en el estado inactivo, haciendo uso de los modos de bajo consumo ofrecidos por el microcontrolador. Concretamente, el paso al modo inactivo se realiza ejecutando la función de salida “dormir” de la FSM descrita en la Figura 3.5. Las acciones realizadas dentro de esta función son:

- Deshabilitar el ADC externo y el módulo radiotransceptor escribiendo un nivel alto en sus pines de selección de esclavo SPI (NSS).
- Suspender las tareas del RTOS para evitar la salida de bajo consumo por la reactivación de alguna tarea.
- Deshabilitar el periférico controlador del bus I2C para evitar una operación incorrecta por un fallo de diseño reflejado en el apartado 2.11.2 de la hoja de erratas del dispositivo (STMicroelectronics, 2016).
- Seleccionar el oscilador “MSI” a 48MHz como reloj del sistema tras la reactivación.

- Habilitar la interrupción del temporizador de bajo consumo "*LPTIM1*", que previamente ha sido programado como fuente de reactivación para generar una interrupción por fin de cuenta con un periodo de sesenta segundos. Además, previamente se ha seleccionado como entrada de reloj del temporizador el oscilador "LSE" que se mantiene encendido en los modos "*Stop*" de bajo consumo.

- Entrar en el modo de bajo consumo "*Stop 1*", apagando el regulador principal y todos los relojes a excepción de "LSE" y "LSI" y reteniendo el contenido de la memoria con el estado del sistema previo a la desactivación.

- Una vez el sistema se reactiva por la interrupción del temporizador de bajo consumo, deshabilitar sus interrupciones, reconfigurar el reloj del sistema para trabajar con los 80MHz proporcionados por el PLL, reconfigurar todos los pines usados en la aplicación y reanudar las tareas del RTOS.

A modo de resumen, la estrategia de bajo consumo seguida en este proyecto para los nodos esclavos de la red inalámbrica se encuentra implementada en la inicialización del sistema y en las funciones de salida de la máquina de estados que describe su comportamiento, ambas cosas contenidas en el fichero fuente "*application.c*". Además, para poder hacer uso de los modos de bajo consumo del microcontrolador y de los periféricos asociados se han incluido las librerías del paquete *STM32CubeL4* implementadas en los ficheros "*stm32l4xx_hal_pwr.c*", "*stm32l4xx_hal_pwr.h*", "*stm32l4xx_hal_lptim.c*" y "*stm32l4xx_hal_lptim.h*" y sus dependencias.

3.8. Estructura del código

Para cerrar el Capítulo 3, en la Figura 3.9 se muestra un diagrama de paquetes en el cual viene representada la estructura del código fuente del proyecto, indicando: los bloques funcionales de código, los nombres de los ficheros que los conforman y las dependencias existentes entre los distintos bloques. Así, se observa cómo el código de usuario está basado tanto en el sistema operativo de tiempo real *FreeRTOS* como en la capa de abstracción *hardware* (HAL) de la tarjeta NUCLEO-L432KC. Esta última, entre otras cosas, permite la implementación de la estrategia de bajo consumo y está incluida en el paquete *STM32CubeL4*. De igual manera, el código de usuario está formado por el código de aplicación, que implementa el comportamiento del nodo modelado como máquina de estados finitos; los controladores y librerías, que dan soporte funcional a las distintas tareas (sensado, procesado y radiotransmisión) que debe realizar el nodo; y los ficheros de configuración del proyecto, que contienen todas las definiciones (tipos de variables, GPIOs asociados a cada periférico, habilitación de la funcionalidad "idle tick" de *FreeRTOS*) necesarias para el correcto funcionamiento del sistema.

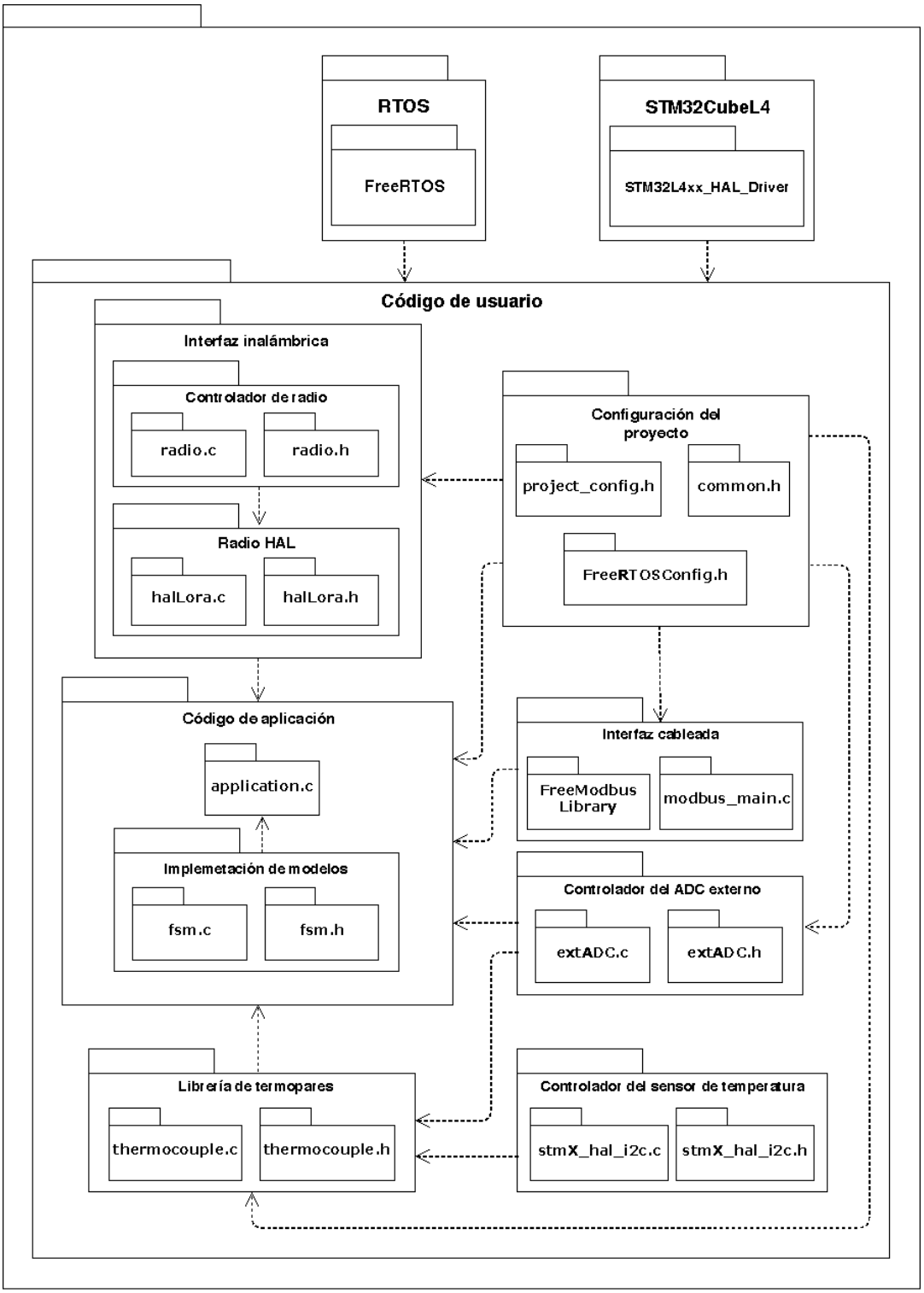

Figura 3.9: Estructura del código fuente del proyecto.

Capítulo 4

Pruebas y validación del sistema completo

Con la finalización de la programación *software* de la aplicación, se da por concluido el diseño del prototipo y da comienzo el último eslabón en la cadena del desarrollo: las pruebas y la validación del sistema. Este proceso es una etapa imprescindible en el desarrollo de cualquier proyecto de ingeniería sobre el diseño y la fabricación de un equipo y, tratándose de un sistema electrónico, el flujo de trabajo incluirá los siguientes pasos intermedios, cada uno de los cuales se detalla en una sección de este capítulo:

- Inspección óptica de la tarjeta electrónica fabricada (Sección 4.1).
- Pruebas eléctricas sobre el prototipo *hardware* (Sección 4.2).
- Pruebas funcionales de la aplicación *software* (Sección 4.3).
- Validación del sistema completo (Sección 4.4).

4.1. Inspección óptica

La primera prueba a realizar en la fabricación de una tarjeta de circuito impreso es la inspección óptica. Esta prueba puede realizarse tanto manualmente como automáticamente (AOI) y permite detectar posibles defectos generados tanto en el diseño del circuito impreso, como en su fabricación o montaje. Si el proceso se realiza de forma automática, esta prueba sirve además para proveer al sistema de montaje automatizado de una realimentación continua a lo largo del proceso.

En este Trabajo Fin de Máster, se ha optado por realizar una inspección óptica manual previa al proceso de montaje (ver Figura 4.1) y también a lo largo del mismo. Esta elección se debe a que la tirada de fabricación ha sido reducida (cinco prototipos) y el proceso de montaje manual. En esta inspección se ha buscado detectar defectos como:

- Rayones.
- Exceso de cobre.
- Circuitos abiertos.
- Cortocircuitos.
- *Pads* no encontrado.
- Errores en serigrafía.
- Vías fuera de su correspondiente *pad*.
- Polaridad del componente invertida.
- Posicionamiento incorrecto del componente.
- Puentes de soldadura.
- Huella del componente incorrecta.

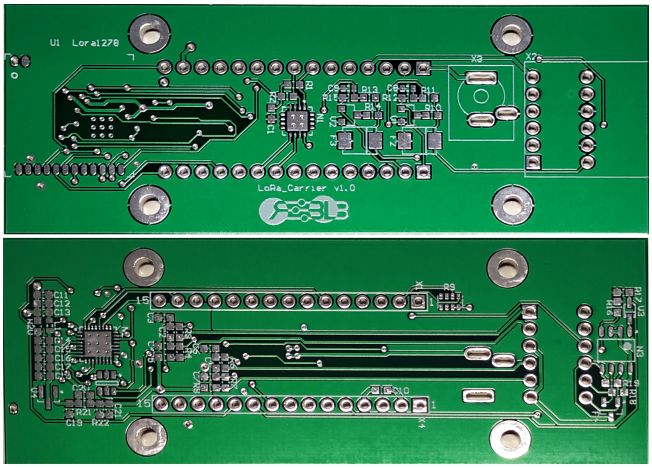

Figura 4.1: Tarjeta electrónica de expansión. Estado previo al montaje.

4.2. Pruebas eléctricas

Una vez descartados los fallos visibles, es necesario realizar una serie de pruebas eléctricas para comprobar la corrección, a nivel de señal eléctrica, del circuito fabricado. Estas pruebas, realizadas con la ayuda del polímetro digital y la fuente de tensión observados en la Figura 4.2, incluyen la comprobación de las siguientes propiedades de la señal eléctrica: circuito abierto, cortocircuito, continuidad de la señal y aislamiento entre señales. A continuación se presenta en mayor detalle el proceso seguido para realizar cada una de estas comprobaciones.

4.2.1. Prueba de circuitos abiertos y cortocircuitos

En esta prueba, cuyo resultado ha sido satisfactorio, se ha empleado la funcionalidad sonora del polímetro para detectar posibles circuitos abiertos en los *pads* de alimentación y masa, tanto analógicos como digitales, de todos los componentes del circuito impreso. También se ha comprobado la posible existencia de cortocircuitos entre planos de potencia y líneas de señal, así como entre los propios planos de alimentación y masa.

Este proceso, ligeramente tedioso si se realiza manualmente, permite detectar errores de diseño o fabricación en las conexiones internas del circuito impreso (pistas y planos), evitando así el posible funcionamiento defectuoso o incluso la destrucción del circuito impreso una vez alimentado.

4.2.2. Prueba de continuidad de la señal

Análogamente a la prueba anterior, se ha empleado el polímetro para comprobar la continuidad de la señal a lo largo del circuito. Para ello, se ha comprobado la existencia de conexión eléctrica en todos aquellos pares de puntos donde se originan y mueren las señales implicadas en el diseño. El resultado de esta prueba también ha sido satisfactorio.

4.2.3. Prueba de aislamiento de canales de entrada

Para esta última prueba se ha empleado, además del polímetro, una fuente de tensión continua. Concretamente, la prueba de aislamiento ha consistido en la introducción de un nivel continuo de tensión a través de uno de los canales de entrada sensorial y la comprobación del nivel de tensión en los canales restantes. Si bien se ha comprobado que el aislamiento entre las entradas es correcto, se ha observado que ante múltiples entradas analógicas, el nivel de tensión de la masa analógica puede subir por encima del nivel de referencia del conversor analógico-digital externo. Esta situación, en tiempo de ejecución, resultaría en una medida incorrecta en los canales de termopares por lo que será necesario revisar este diseño en el futuro.

Figura 4.2: Herramientas empleadas en las pruebas eléctricas.

4.3. Pruebas funcionales

Terminadas las pruebas eléctricas, se han descartado o detectado todos los posibles fallos a nivel eléctrico en el circuito impreso. En consecuencia, los posibles fallos restantes han de ser de naturaleza lógica o temporal, por lo que es necesario realizar una serie de pruebas funcionales sobre los distintos módulos *firmware* y *software* de la aplicación para verificar su corrección. Estas pruebas se detallan a continuación.

4.3.1. Prueba de comunicación entre el *TJMON* y el nodo maestro

Para comprobar el correcto funcionamiento de la implementación del protocolo Modbus-RTU, se ha creado una rutina de prueba en la programación del nodo maestro. Esta rutina está implementada como una tarea de *FreeRTOS* en la cual se habilita y configura la interfaz UART de la tarjeta NUCLEO-L432KC, se inicializa el nodo como un esclavo en la red cableada Modbus-RTU y se atienden peticiones del dispositivo *TJMON* en un bucle infinito, devolviendo un valor ficticio como respuesta. Esta tarea se observa en el siguiente código:

```
xTaskHandle MODBUS_Task_Handle;
...
void MODBUS_Task()
{
  uint8_t dir_esclavo = 0x01;
  /* Habilita UART e inicializa Modbus */
  SERIE_SetModbus();
  eMBInit(MB_RTU, dir_esclavo, 1, 9600, MB_PAR_NONE);
  eMBEnable();
  /* Atiende peticiones */
  while(1)
  {
    eMBPoll();
  }
}
...
xTaskCreate(MODBUS_Task, "MODBUS_Task", MODBUS_TASK_STACK, NULL,
    MODBUS_TASK_PRIORITY, &MODBUS_Task_Handle);
```

4.3.2. Prueba de conversión analógico-digital

Para comprobar el correcto funcionamiento del controlador del conversor analógico-digital externo, se ha creado una rutina de prueba en la programación de los nodos esclavos. Esta rutina está implementada como una tarea de *FreeRTOS* en la cual se habilita y configura la interfaz SPI de la tarjeta NUCLEO-L432KC, se habilita y configura el ADC externo para realizar medidas pseudo-diferenciales y se realizan varias medidas consecutivas (diez en esta prueba) espaciadas un segundo entre ellas, almacenando los valores medidos para su posterior comprobación. Durante esta prueba, se ha conectado la fuente de tensión empleada en las pruebas eléctricas a las entradas analógicas del nodo y se han aplicado distintos niveles de tensión para comprobar la fidelidad de las conversiones. La tarea implementada se observa en el siguiente código:

```
xTaskHandle extADC_Task_Handle;
float medidas[10];
...
void extADC_Task()
{
  /* Define el periodo de muestreo */
  TickType_t ticks_periodo_muestreo = 1 * 1000 / portTICK_PERIOD_MS;
  TickType_t cuentaTicksProximaMedida = xTaskGetTickCount();
  uint8_t i = 0;
  /* Habilita y configura el ADC externo */
  ExtADC_Init();
  ExtADC_Reset();
  /* Recoge diez medidas consecutivas */
  while(i < 10)
  {
    medidas[i++] = ExtADC_ReadVoltageInput(CANAL_ENTRADA_ANALOG_0);
    vTaskDelayUntil(&cuentaTicksProximaMedida, ticks_periodo_muestreo);
  }
}
```

```
...
xTaskCreate(extADC_Task, "extADC_Task", EXTADC_TASK_STACK, NULL,
    EXTADC_TASK_PRIORITY, &extADC_Task_Handle);
```

4.3.3. Prueba del sensor de temperatura integrado

Para comprobar el correcto funcionamiento del controlador del sensor de temperatura integrado, se ha creado una rutina de prueba en la programación de los nodos esclavos. Esta rutina está implementada como una tarea de *FreeRTOS* en la cual se habilita y configura la interfaz I2C de la tarjeta NUCLEO-L432KC, se configura el sensor de temperatura integrado y se realizan varias medidas consecutivas (diez en esta prueba) espaciadas un segundo entre ellas, almacenando los valores medidos para su posterior comprobación. La tarea implementada se observa en el siguiente código:

```
xTaskHandle tempSensor_Task_Handle;
int16_t medidas[10];
...
void tempSensor_Task()
{
  /* Define el periodo de muestreo */
  TickType_t ticks_periodo_muestreo = 1 * 1000 / portTICK_PERIOD_MS;
  TickType_t cuentaTicksProximaMedida = xTaskGetTickCount();
  uint8_t i = 0;
  /* Habilita y configura el sensor de temperatura integrado */
  I2C_Config();
  Therm_Init();
  /* Recoge diez medidas consecutivas */
  while(i < 10)
  {
    medidas[i++] = Therm_ReadTemp();
    vTaskDelayUntil(&cuentaTicksProximaMedida, ticks_periodo_muestreo);
  }
}
...
xTaskCreate(tempSensor_Task, "tempSensor_Task", TEMPSENSOR_TASK_STACK,
    NULL, TEMPSENSOR_TASK_PRIORITY, &tempSensor_Task_Handle);
```

4.4. Validación del sistema completo

Finalmente, una vez asegurado el comportamiento físico y lógico de los componentes del sistema, se ha realizado una serie de pruebas para verificar que el sistema completo implementado se comporta de acorde al modelo y a las especificaciones de diseño establecidas. Estas pruebas, detalladas a continuación, consisten en:

- Comprobar que el funcionamiento en una red punto-multipunto con múltiples nodos esclavos es el esperado.
- Verificar la correcta alimentación del nodo mediante baterías.
- Analizar el consumo medio de los nodos inalámbricos y estimar la duración de las baterías.

4.4.1. Prueba de red inalámbrica punto-multipunto

Entre los objetivos específicos propuestos para la consecución de este Trabajo Fin de Máster, se encuentra el desarrollo de una aplicación de red punto-multipunto. El comportamiento red esperado en esta aplicación ha sido descrito en la Sección 3.3 de este documento y, para esta prueba, se ha comprobado sobre una arquitectura de red compuesta por un nodo maestro y cuatro nodos esclavos. Así, a modo de demostración, se ha desarrollado para realizar esta prueba una variante de la aplicación del nodo maestro. En ella, toda la información correspondiente a las tramas recibidas inalámbricamente es enviada en forma de trazas de ejecución, a través de la interfaz de programación/depuración de la tarjeta NUCLEO-L432KC. Posteriormente, en la máquina de desarrollo, estas trazas se imprimen por pantalla a través del programa emulador de terminal serie, *Minicom*, tal y como se muestra en la Figura 4.3.

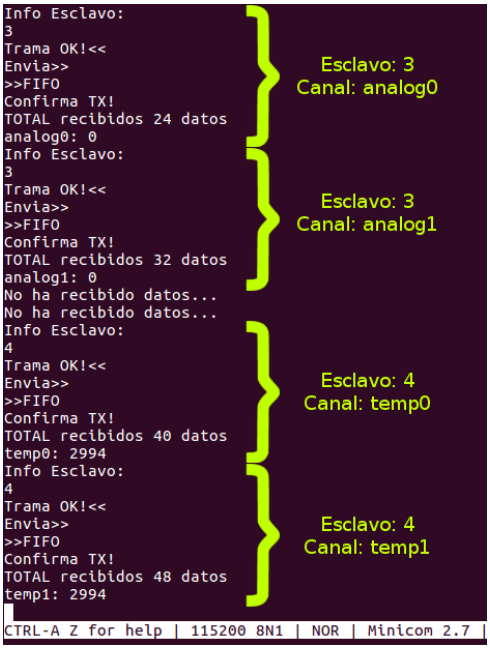

Figura 4.3: Trazas de red inalámbrica en la aplicación de red diseñada.

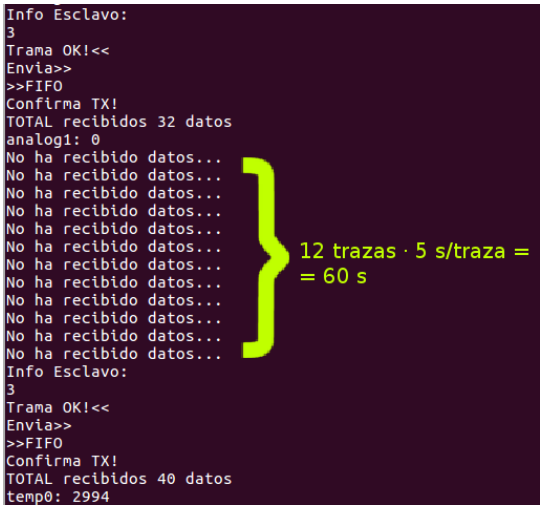

Figura 4.4: Trazas de red inalámbrica mostrando el estado inactivo de un nodo esclavo.

En esta aplicación, si el maestro no recibe ninguna trama en un periodo de cinco segundos, genera una traza con el mensaje "No ha recibido datos...". Esta utilidad, además de servir como referencia para evaluar la holgura temporal entre mensajes de distintos nodos esclavos, permite comprobar la correcta implementación de la estrategia de bajo consumo diseñada. Para ello, basta con conectar un único nodo esclavo a la red y observar el número de trazas de este tipo generadas entre dos activaciones consecutivas. Así, en la Figura 4.4 se demuestra que la implementación del estado inactivo es correcta, puesto que el número de trazas generadas cada cinco segundos entre dos activaciones consecutivas es de doce trazas, es decir, un minuto.

4.4.2. Prueba de alimentación por batería

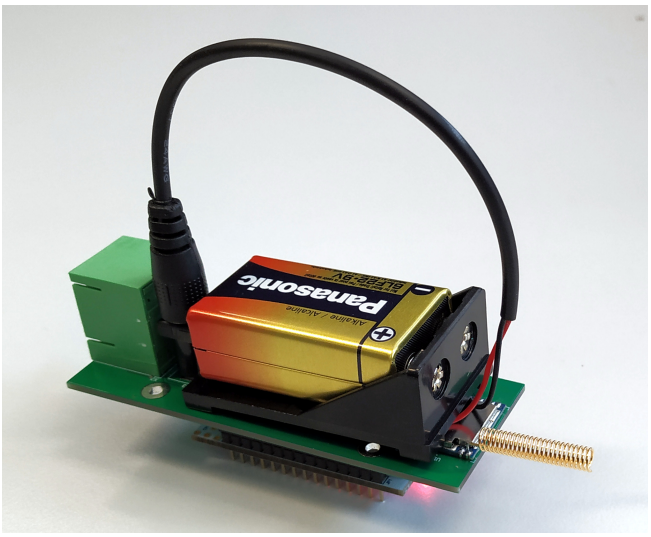

Figura 4.5: Nodo inalámbrico alimentado mediante una pila de 9V.

En segundo lugar, para satisfacer los requisitos de diseño, los nodos esclavos deben poder alimentarse mediante baterías sin que esto modifique su comportamiento. Por ello, en esta prueba se ha mantenido la aplicación de red descrita en el apartado anterior pero, en esta ocasión, los nodos esclavos han sido alimentados a través del conector de alimentación externa de la tarjeta de expansión, utilizando como fuentes pilas alcalinas de 9V. Esta configuración se muestra en la Figura 4.5, habiendo resultado esta prueba satisfactoriamente.

4.4.3. Análisis de consumo y estimación de la duración de la batería

Por último, en el diseño de una aplicación con una estrategia de reducción de consumo, resulta interesante obtener medidas empíricas que corroboren la efectividad de dicha estrategia. En este caso, se busca con esta prueba analizar el impacto del estado inactivo sobre el consumo final de los nodos esclavos. Para ello, en la misma aplicación de red anteriormente descrita, se ha medido con el polímetro la corriente media extraída de la batería por el nodo para los estados activo e inactivo. Cabe destacar que, para obtener una mayor fiabilidad en la medida, se ha realizado esta prueba sobre dos variantes de la aplicación del nodo esclavo: la aplicación final diseñada en este Trabajo Fin de Máster y una versión de la misma en la que el estado inactivo consiste únicamente en una espera activa de un minuto. Los resultados obtenidos en esta prueba se han reflejado en la Tabla 4.1 mostrada a continuación:

	Bajo consumo	*Espera activa*
Duración máxima del estado activo [s]	10	10
Corriente media en el estado activo [mA]	70	70
Corriente media en el estado inactivo [mA]	50	57
Corriente media máxima [mA]	52.857	58.857
Duración estimada de una pila de 550mAh [h]	10.4	9.3

Tabla 4.1: Análisis del consumo en los nodos esclavos.

Capítulo 5

Conclusiones y líneas futuras

5.1. Conclusiones

En este Trabajo Fin de Máster se diseñado e implementado un sistema empotrado autónomo, alimentado por baterías y capaz de actuar como un sensor inalámbrico con comunicaciones de largo alcance y bajo consumo, además de ser integrable de forma transparente en la red de monitorización *Monitoring Robolabo*. Este sistema no sólo cumple su función de acuerdo a las especificaciones de diseño, sino que también cuenta con un campo de aplicación muy extenso, los sistemas WSN, debido a que las tendencias actuales en los ámbitos tecnológico y sociocultural se inclinan hacia la adquisición masiva de datos del entorno de forma autónoma, eficiente y conectada.

Las pruebas realizadas en el Capítulo 4, aún habiendo cumplido su objetivo y arrojado resultados satisfactorios, apuntan a que este sistema puede ser mejorado de diversas maneras. Modificaciones en el diseño electrónico del sistema, en la arquitectura de la red, en los protocolos de comunicación, en la estrategia de reducción de consumo o en la aplicación en sí misma, tienen cabida en este desarrollo y merecen ser estudiadas en mayor profundidad. Sin embargo, teniendo en cuenta la limitación en tiempo de desarrollo y el cumplimiento de los objetivos y requisitos de diseño, se puede concluir que este Trabajo Fin de Máster ha sido exitoso y ha generado beneficios apreciables para las partes implicadas.

En el caso del Laboratorio de Robótica y Control, "Robolabo", este proyecto ha permitido analizar las opciones de crecimiento de la red de monitorización *Monitoring Robolabo* y ha abierto nuevas vías de investigación y desarrollo para futuros proyectos dentro del laboratorio.

En el ámbito personal, el proyecto realizado ha sido gratificante, a la par que formativo. Concretamente, este proyecto me ha permitido desarrollar una visión sistémica y multidisciplinar, permitiendome trabajar desde un enfoque de diseño electrónico orientado a producto, en el que me ha sido necesario tener en consideración conceptos de: diseño y fabricación de circuitos impresos, selección y adquisición de componentes, programación de microcontroladores, comunicaciones inalámbricas, estrategias de bajo consumo para sistemas empotrados, estandarización, compatibilidad entre productos o pruebas y validación de sistemas.

5.2. Líneas futuras

Habiendo cumplido los objetivos propuestos y dada la tesitura en la que se encuentra este proyecto a nivel tecnológico, sociocultural y académico, son numerosas las futuras líneas de desarrollo que se desprenden de este trabajo. A continuación, se exponen algunas propuestas con las que continuar el desarrollo del sistema planteado:

- Se plantea experimentar con arquitecturas de red inalámbrica más sofisticadas, como topologías de malla con paso de mensajes entre los nodos inalámbricos, permitiendo así aumentar el alcance de la red.
- Rediseño y miniaturización de la tarjeta de expansión, prescindiendo de los conectores externos y empleando tecnología MEMS para dar mayor capacidad sensorial a los nodos inalámbricos.
- Se sugiere la sustitución de las baterías alcalinas por baterías recargables de litio, de mayor capacidad y menores dimensiones, incluyendo un circuito de carga y monitorización del nivel de batería en la tarjeta de expansión.
- Se propone el rediseño de la tarjeta de expansión, en esta ocasión orientado a la reducción de costes materiales para su producción a gran escala.
- Se plantea modificar la aplicación del nodo maestro de la red inalámbrica para realizar cálculos sobre las medidas recibidas, obteniendo métricas relevantes en tiempo de ejecución, como: medias, medianas, desviaciones típicas o varianzas. A partir de estas métricas se podría reconfigurar el periodo de muestreo de nodos concretos para adecuarse más a la variabilidad de su entorno.
- Detección y localización de nodos defectuosos en la red, así como la posible reestructuración adaptativa de la misma ante estos fallos.
- Se sugiere la implementación de un algoritmo de envío de tramas que tenga en consideración posibles pérdidas de conectividad en la red. Esto es, que una vez superado un número máximo de intentos de transmisión permita al nodo pasar al estado inactivo hasta la próxima activación, aún existiendo tramas pendientes de envío.

Bibliografía

Akyildiz, I., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422.

Amazon (2017). Reference manual for freertos version 10.0.0 issue 1. `https://www.freertos.org/Documentation/FreeRTOS_Reference_Manual_V10.0.0.pdf`. Online; accesed 29 de agosto de 2018.

Amazon (2018). What is freertos? `https://www.freertos.org/about-RTOS.html`. Online; accesed 29 de agosto de 2018.

ARM (2018). Cortex-m4 processor. `https://www.arm.com/products/processors/cortex-m4-processor.php`. Online; accesed 29 de agosto de 2018.

Augustin, A., Yi, J., Clausen, T., and Townsley, W. M. (2016). A study of lora: Long range and low power networks for the internet of things. *Sensors*, 16(9):1466.

Barry, R. (2016). Mastering the freertos real time kernel - a hands-on tutorial guide. `https://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf`. Online; accesed 29 de agosto de 2018.

Canonical (2018). Ubuntu - the leading operating system for pcs, iot devices, servers and the cloud. `https://www.ubuntu.com/`. Online; accesed 29 de agosto de 2018.

Contact, P. (2018). Header - mcd 0.5/ 6-g1-2.5 - 1894846. `https://www.phoenixcontact.com/pi/products/1894846`. Online; accesed 29 de agosto de 2018.

Devices, A. (2017a). 16-bit digital i2c temperature sensor. `http://www.analog.com/media/en/technical-documentation/data-sheets/ADT7420.pdf`. Online; accesed 29 de agosto de 2018.

Devices, A. (2017b). 2.5v to 5.0v micropower, precision series mode voltage references. `http://www.analog.com/media/en/technical-documentation/data-sheets/ad1582_1583_1584_1585.pdf`. Online; accesed 29 de agosto de 2018.

Devices, A. (2017c). 8-channel, 4.8khz, ultralow noise, 24-bit sigma-delta adc with pga. `http://www.analog.com/media/en/technical-documentation/data-sheets/AD7194.pdf`. Online; accesed 29 de agosto de 2018.

Electronics, S. (2018). 9v baterry holder prt-10512. `https://media.digikey.com/pdf/Data%20Sheets/Sparkfun%20PDFs/PRT-10512_Web.pdf`. Online; accesed 29 de agosto de 2018.

Evans, D. (2011). The internet of things: How the next evolution of the internet is changing everything. https://www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf. Online; accessed 29 de agosto de 2018.

Free Software Foundation, I. (2018a). Gcc, the gnu compiler collection. https://gcc.gnu.org/. Online; accesed 29 de agosto de 2018.

Free Software Foundation, I. (2018b). Gdb, the gnu project debugger. https://www.gnu.org/software/gdb/. Online; accesed 29 de agosto de 2018.

Gardner, J. W., Varadan, V. K., and Awadelkarim, O. O. (2001). *Microsensors MEMS and Smart Devices*. Wiley, England.

Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M. (2013). Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Comp. Syst.*, 29(7):1645–1660.

Herreros Elorza, J. (2016). Diseño e implementación de un analizador de redes eléctricas trifásicas. Technical report, Universidad Politécnica de Madrid.

Inc, C. (2018). Dc power jack - model pj-044a. https://www.cui.com/product/resource/pj-044a.pdf. Online; accesed 29 de agosto de 2018.

Integrated, M. (2017). Half-duplex rs-485-/rs-422-compatible transceiver with autodirection control. https://datasheets.maximintegrated.com/en/ds/MAX13487E-MAX13488E.pdf. Online; accesed 29 de agosto de 2018.

Lee, E. A. and Seshia, S. A. (2007). *Introduction to Embedded Systems - A Cyber-Physical Systems Approach - Second Edition*. MIT Press, USA.

LoRa, A. (2018). What is the lorawan specification. https://lora-alliance.org/about-lorawan. Online; accessed 29 de agosto de 2018.

Montrose, M. (2000). *Printed circuit board design techniques for EMC compliance : a handbook for designers*. IEEE Press, New York.

Moya, J. M. (2017). Greenlsi - unlessons. https://github.com/greenlsi/unlessons/tree/master/fsm. Online; accesed 29 de agosto de 2018.

NiceRF Wireless Technology Co., L. (2018). 433/470mhz 100mw wireless transceiver lora module - lora1278. http://nicerf.com/Upload/ueditor/files/2018-04-17/LORA1278-100mW%20long%20range%20Spread%20Spectrum%20modulation%20wireless%20transceiver%20module%20V2.1-73beaf3d-60ff-471f-b64f-eb086ad35783.pdf. Online; accesed 29 de agosto de 2018.

Organization, M. (2012). Modbus application protocol specification v1.1b3. http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf. Online; accesed 29 de agosto de 2018.

Pillai, P. and Shin, K. G. (2001). Real-time dynamic voltage scaling for low-power embedded operating systems. In *Proceedings of the Eighteenth ACM Symposium on Operating Systems Principles*, SOSP '01, pages 89–102, New York, NY, USA. ACM.

Rath, D. (2018). Open on-chip debugger - free and open on-chip debugging, in-system programming and boundary-scan testing. `http://openocd.org/`. Online; accesed 29 de agosto de 2018.

Robolabo (2014). Tj_monitor - manual de desarrollo. Technical Report 001, Solar Decathlon Europe.

Rodríguez Munca, J. D. (2016). Dispositivo lora de comunicación a largo alcance y bajo consumo energético para aplicaciones del ámbito del desarrollo. Technical report, Universidad Politécnica de Madrid and Universidad Complutense de Madrid.

Semtech (2018). Wireless rf lora transceivers - semtech sx1278. `https://www.semtech.com/products/wireless-rf/lora-transceivers/SX1278`. Online; accesed 29 de agosto de 2018.

Sistemas Empotrados Avanzados, C. D. M. (2018). Embedded systems - designing power efficient embedded systems. `https://moodle.upm.es/titulaciones/oficiales/pluginfile.php/1413844/mod_resource/content/1/low-power.pdf`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2016). Stm32l432kb/kc stm32l442kc errata sheet. `https://www.st.com/resource/en/errata_sheet/dm00218221.pdf`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018a). St-link/v2 in-circuit debugger/programmer for stm8 and stm32. `https://www.st.com/en/development-tools/st-link-v2.html`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018b). Stm32 nucleo-32 development board with stm32l432kc mcu, supports arduino connectivity. `https://www.st.com/en/evaluation-tools/nucleo-l432kc.html`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018c). Stm32cube mcu package for stm32l4 series and stm32l4 plus series (hal, low-layer apis and cmsis (core, dsp, rtos), usb, touchsensing, file system, rtos, graphic - coming with examples running on st boards: Stm32 nucleo, discovery kits and evaluation boards). `https://www.st.com/en/embedded-software/stm32cubel4.html`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018d). Stm32l432kb stm32l432kc. `https://www.st.com/resource/en/datasheet/stm32l432kc.pdf`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018e). Ultra-low-power with fpu arm cortex-m4 mcu 80 mhz with 256 kbytes flash, usb. `https://www.st.com/en/microcontrollers/stm32l432kc.html`. Online; accesed 29 de agosto de 2018.

STMicroelectronics (2018f). Um1956. https://www.st.com/content/ccc/resource/technical/document/user_manual/e3/0e/88/05/e8/74/43/a0/DM00231744.pdf/files/DM00231744.pdf/jcr:content/translations/en.DM00231744.pdf. Online; accesed 29 de agosto de 2018.

Sullins (2018). .100"(2.54mm) contact centers, female headers, straight/right angle/smt. https://s3.amazonaws.com/catalogspreads-pdf/PAGE114-115%20.100%20FEMALE%20HDR.pdf. Online; accesed 29 de agosto de 2018.

Townsend, K., Cufí, C., Akiba, and Davidson, R. (2014). *Getting Started with Bluetooth Low Energy*. O'Reilly, United States of America.

Walter, C. (2010). Freemodbus - a modbus ascii/rtu and tcp implementation. https://www.freemodbus.org/. Online; accesed 29 de agosto de 2018.

Warneke, B. A. and Pister, K. S. J. (2002). Mems for distributed wireless sensor networks. In *9th International Conference on Electronics, Circuits and Systems*, volume 1, pages 291–294. IEEE Xplore.

Weik, M. (1996). *Communications Standard Dictionary. 3a Edición*. Chapman and Hall, Estados Unidos.

Zigbee, A. (2018). Zigbee for developers - zigbee 3.0. https://www.zigbee.org/zigbee-for-developers/zigbee-3-0/. Online; accessed 29 de agosto de 2018.