



UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Edificació
de Barcelona



TFG - Memoria Final

Diseño, desarrollo y programación de una lanza meteorológica

Pau Zaragoza Gallardo

13 de enero de 2026

Especialidad en Ingeniería de Computadores

Director: Antonio Benito Martínez

Resumen

???

Resum

???

Abstract

???

Agradecimientos

???

Índice general

1. Contexto	8
1.1. Introducción	8
1.2. Definición de conceptos	8
1.3. Identificación del problema	9
1.4. Actores implicados	10
2. Justificación	11
3. Alcance	13
3.1. Objetivos	13
3.2. Sub-objetivos	13
3.3. Requerimientos	14
3.3.1. Requerimientos funcionales	14
3.3.2. Requerimientos no funcionales	14
3.4. Posibles obstáculos i riesgos	15
3.5. Competencias técnicas	15
4. Metodología y rigor	16
4.1. Herramientas	17
5. Descripción de las tareas	18
5.1. Introducción	18
5.2. Tareas	18
5.2.1. Gestión del Proyecto	18
5.2.2. Estudio Previo	19
5.2.3. Desarrollo Hardware	19
5.2.4. Desarrollo Firmware	19
5.2.5. Pruebas y Validación	20
5.2.6. Documentación	20
5.3. Recursos	20
5.3.1. Recursos humanos	20
5.3.2. Recursos materiales	21
5.3.3. Recursos digitales	21
6. Estimaciones y Gantt	23
7. Gestión de riesgos: Planes alternativos y obstáculos	25

8. Presupuesto	26
8.1. Identificación de costes	26
8.1.1. Recursos humanos	26
8.1.2. <i>Hardware</i>	27
8.1.3. <i>Software</i>	28
8.1.4. Gastos generales	28
8.2. Estimación de costes	29
8.3. Control de gestión	30
9. Informe de sostenibilidad	31
9.1. Autoevaluación	31
9.2. Dimensión Económica	31
9.3. Dimensión Ambiental	32
9.4. Dimensión Social	32
10. Diseño del sistema	34
10.1. Arquitectura del sistema	34
10.2. Decisiones de diseño	35
10.2.1. Periodo de muestreo	35
10.2.2. Almacenamiento local robusto y simple	35
10.2.3. Tolerancia a datos inválidos	35
10.2.4. Referencia temporal en cada muestra	35
10.2.5. Tensión de trabajo	36
10.2.6. Modularidad del sistema	36
10.2.7. Protocolo de comunicación inalámbrica	36
11. Diseño hardware	37
11.1. Selección de componentes	37
11.1.1. Gestión de potencia (PMIC)	37
11.1.2. Monitoreo de potencia	38
11.1.3. Almacenamiento	39
11.1.4. Sensor de irradiancia	40
11.1.5. Sensor de temperatura y humedad del aire	41
11.1.6. Sensor de temperatura del terreno	42
11.1.7. Sensor de humedad del terreno	42
11.1.8. Microcontrolador	43
11.2. Arquitectura del <i>hardware</i>	45
11.2.1. Interfaces y protocolos	46
11.3. Prueba de concepto	47
11.4. Esquemáticos	50
11.4.1. Potencia y alimentación	50
11.4.2. MCU	58
11.4.3. Conexiones externas	64
11.5. Diseño PCB	67
11.5.1. Apilado físico (<i>stack-up</i>)	67
11.5.2. Requerimientos de JLCPCB	68

11.5.3. Condensadores de desacoplo	69
11.5.4. Islas para los cristales	70
11.5.5. RF	71
11.5.6. Resultado final	74
11.6. Fabricación y montaje	77
11.6.1. Proceso de elección de componentes de JLCPCB	77
11.6.2. BOM	78
11.6.3. Proceso de ensamblaje	79
11.7. Integración en la lanza	79
12. Desarrollo software	80
12.1. Entorno y herramientas	80
12.2. Arquitectura del <i>firmware</i>	80
12.3. <i>Drivers</i>	81
12.3.1. INA3221	81
12.3.2. MB85RS256B	82
12.3.3. TSL2591	83
12.3.4. SHT3x	84
12.3.5. DS18B20	85
12.3.6. SEN0308	86
12.4. Gestión de la energía	87
12.5. Gestión de los datos	88
13. Validación y pruebas	91
13.1. Pruebas de memoria	91
13.2. Pruebas de autonomía	94
13.3. Pruebas de sensores	96
13.4. Pruebas de conexión inalámbrica	98
14. Conclusiones y trabajo futuro	99

Capítulo 1

Contexto

1.1. Introducción

El proyecto "**Diseño, desarrollo y programación de una lanza meteorológica**" se realiza como parte de mi **Trabajo Final de Grado** en la Facultad de Informática de Barcelona (FIB) [1] de la Universidad Politécnica de Cataluña (UPC) [2].

En un contexto de creciente digitalización de la agricultura, las nuevas tecnologías permiten mejorar la productividad y sostenibilidad de los cultivos. La viticultura, un sector clave en la economía española, enfrenta desafíos como la optimización del uso del agua, la detección temprana de plagas y enfermedades, y la mejora de la eficiencia en la recolección de datos del terreno.

Este trabajo se enmarca dentro de un proyecto más amplio orientado a construir un pipeline de monitorización para el sector vitivinícola mediante drones autónomos que capturan imágenes multiespectrales de los viñedos. Dichas imágenes se transfieren a un servidor para su procesamiento con técnicas de visión por computador e inteligencia artificial, con el objetivo de anticipar problemas y optimizar intervenciones. La lanza meteorológica propuesta en este TFG actúa como subsistema de soporte, aportando datos ambientales y del suelo para complementar la información multiespectral y, con ello, mejorar la calidad de las inferencias de la IA.

El proyecto se desarrolla en colaboración con la empresa **Domo21** [3], especializada en soluciones tecnológicas aplicadas, y se alinea con iniciativas de digitalización impulsadas por organismos públicos y privados orientadas a incrementar la competitividad del sector agroalimentario en Cataluña.

1.2. Definición de conceptos

Para enmarcar correctamente el desarrollo del proyecto, es necesario definir ciertos conceptos fundamentales.

- **Layout:** Diseño físico de la PCB que define la colocación de componentes y la geometría de pistas, planos y vías en sus distintas capas. A diferencia del esquemático, el

layout determina el comportamiento real en términos de integridad de señal, ruido y fabricabilidad.

- **Ruteado:** Proceso de trazado de pistas en el *layout* para interconectar eléctricamente los nodos del esquemático.
- **Plano:** Zona extensa de cobre asignada a una red eléctrica (habitualmente GND o alimentación). Se emplea para reducir impedancia, mejorar el retorno de corriente, disminuir el ruido y proporcionar una referencia eléctrica estable.
- **Vía:** Taladro metalizado que conecta eléctricamente distintas capas de una PCB. Se utiliza para cambiar una señal de capa durante el ruteado o para unir planos.
- **Stack-up:** Estructura de apilado de capas de una PCB: capas de cobre, dieléctricos, espesores y materiales. Condiciona el comportamiento eléctrico de la placa, especialmente en líneas de alta frecuencia, el retorno de corriente y el cálculo de impedancias.
- **Impedancia:** Relación entre tensión y corriente en una línea de transmisión, determinada por la geometría de la pista y el *stack-up*. En señales de alta frecuencia es un parámetro crítico, ya que desadaptaciones provocan reflexiones y pérdidas.
- **DRC (*Design Rule Check*):** Conjunto de reglas de verificación automática que comprueba que el *layout* cumple las restricciones mínimas de fabricación y ensamblaje. Se utiliza para detectar errores antes de fabricar la PCB.
- **Bring-up:** Fase inicial de puesta en marcha de una PCB recién fabricada/ensamblada, en la que se verifican aspectos básicos como alimentación, consumo, programación, reloj y comunicaciones fundamentales, antes de realizar pruebas funcionales completas.

1.3. Identificación del problema

Si bien las imágenes multiespectrales obtenidas por drones ofrecen una visión espacialmente rica del viñedo, por sí solas pueden resultar insuficientes para entrenar y validar modelos de IA con el nivel de precisión y robustez que demanda la operación en campo. Entre las limitaciones más habituales se encuentran:

- **Falta de variables de contexto:** Los modelos espectrales pueden confundir efectos de iluminación, atmósfera o microclima con señales fisiológicas de la planta si no se dispone de mediciones locales (temperatura y humedad del aire, viento, irradiancia, etc.) que permitan normalizar y contextualizar los datos.
- **Resolución temporal limitada:** Los vuelos de dron aportan «fotografías» puntuales. Sin sensorización continua, es difícil capturar dinámicas diarias (picos de estrés térmico o hídrico, etc.) relevantes para el diagnóstico.
- **Escasez de *ground truth* de suelo:** La salud del cultivo depende en gran medida del estado hídrico y térmico del suelo. Sin medidas *in situ* (humedad y temperatura del suelo), ciertos patrones espectrales pueden interpretarse de forma ambigua.

En consecuencia, se identifica la necesidad de incorporar datos ambientales y de suelo que actúen como variables de entrada y/o control para los modelos de IA entrenados con imágenes multiespectrales, que posibiliten la validación y calibración de sus inferencias —reduciendo falsos positivos y negativos— y que aporten continuidad temporal entre campañas de vuelo, mejorando la trazabilidad de los procesos y la capacidad de anticipación ante problemas en el cultivo.

La lanza meteorológica propuesta aborda esta brecha proporcionando mediciones locales, persistentes y sincronizadas con las adquisiciones aéreas, incrementando así la robustez del sistema global de monitorización.

1.4. Actores implicados

El éxito del proyecto se apoya en la colaboración entre distintos perfiles y entidades:

- **[PZ] Pau Zaragoza (Yo):** Desempeñaré dos roles principales dentro del proyecto. Por un lado, actuaré como responsable de diseño y definición del sistema, estableciendo la arquitectura global, seleccionando los componentes, diseñando el esquemático y la PCB, y definiendo los criterios de integración y validación del prototipo. Por otro lado, asumiré el papel de desarrollador, implementando el *firmware* necesario, realizando la integración de subsistemas y ejecutando las pruebas pertinentes. Además, seré responsable de la redacción del TFG, documentando el proceso, los resultados obtenidos y las conclusiones del trabajo.
- **[AB] Antonio Benito:** Tutor académico y director del proyecto, encargado de proporcionar orientación técnica y metodológica. Supervisará el desarrollo, revisará los avances y aportará sugerencias para garantizar la calidad y viabilidad del trabajo.
- **[CM] Carlos Morata:** Codirector del trabajo y jefe del departamento dentro de Sicma21.
- **[AV] Álex Veiga:** Apoyará en la parte técnica del desarrollo, brindando asesoramiento y asistencia en la implementación de soluciones.
- **[D21] Domo21:** [3] Facilita el entorno de desarrollo y validación en un contexto real de aplicación y promueve la transferencia de resultados hacia el proyecto de monitorización con drones.
- **[UPC] UPC:** [2] Proporciona el marco académico, metodológico y acceso a recursos de investigación y a la maquinaria especializada del departamento ESAII.

Capítulo 2

Justificación

La decisión de desarrollar una solución a medida se basa en la necesidad de contar con un sistema plenamente alineado con los objetivos y condicionantes del proyecto. Diseñar el dispositivo desde cero permite ajustar el *hardware*, la selección de sensores y el tratamiento de datos a los requisitos reales del viñedo y a la integración con los demás componentes del entorno (dron, procesamiento de imágenes multiespectrales e inteligencia artificial), evitando compromisos y limitaciones propios de productos genéricos.

En el mercado existen sistemas comerciales con funcionalidades parciales o de propósito general que, además de un coste elevado, suelen exigir adaptaciones complejas para su integración efectiva en arquitecturas ya definidas. En este contexto, desarrollar un prototipo propio ofrece ventajas claras: control total sobre la arquitectura *hardware-software*, mayor capacidad de integración con los módulos del proyecto (adquisición en campo, sincronización con vuelos de dron, preprocesado y transmisión) y reducción de dependencias de terceros que puedan condicionar la evolución futura.

Un elemento clave de esta propuesta es la posibilidad de elegir qué sensores y variables recolectar, priorizando aquellas que demuestren mayor relevancia para el rendimiento de los modelos de IA. Este enfoque iterativo (medir, evaluar importancia, ajustar) permite optimizar el conjunto de variables —por ejemplo, seleccionando métricas ambientales o edáficas que mejor expliquen variaciones observadas en los índices espectrales— y, al mismo tiempo, reducir complejidad y consumo energético cuando ciertas mediciones no aporten valor significativo.

La flexibilidad también es un argumento central: la solución se concibe como modular y reconfigurable para adaptarse a diferentes clientes, parcelas y condiciones agroclimáticas. Esta adaptabilidad abarca desde la configuración de sensores y su frecuencia de muestreo hasta el formato de datos, los protocolos de comunicación y las estrategias de alimentación energética, de modo que el sistema pueda ajustarse a los requisitos de cada explotación y escalar en función de nuevas necesidades.

Finalmente, el desarrollo propio contribuye a la sostenibilidad técnica y económica del proyecto: facilita el mantenimiento, la trazabilidad de componentes, la transparencia del diseño y la gobernanza de los datos (incluyendo su calidad y metadatos), al tiempo que reduce el coste total de propiedad frente a la adquisición y adaptación de equipos genéricos de mayor

precio. En conjunto, estos factores justifican la construcción de un prototipo específico como la vía más eficiente para garantizar la integración con el entorno del proyecto, maximizar la utilidad de los datos para la IA y ofrecer una solución realmente ajustada a las particularidades del sector y de cada viñedo.

Capítulo 3

Alcance

El presente trabajo se centra en pensar, diseñar y desarrollar el **primer prototipo** de una lanza meteorológica orientada a complementar la toma de datos en viñedos dentro del proyecto principal. El alcance incluye la definición de variables a medir, la selección de sensores, el diseño de las PCBs, el ensamblado y soldadura de componentes y la ejecución de pruebas en entornos controlados.

3.1. Objetivos

El principal objetivo del proyecto es pensar, diseñar y desarrollar todo el sistema electrónico de la lanza meteorológica capaz de adquirir, almacenar y transmitir datos ambientales y de suelo relevantes para su posterior explotación dentro del proyecto.

3.2. Sub-objetivos

- **PCB principal:** diseño del controlador, gestión de alimentación y puertos de expansión/IO para sensores.
- **Firmware base:** adquisición fiable, metadatos útiles y almacenamiento local; además de la opción de comunicación inalámbrica.
- **Gestión energética:** definición de modos de bajo consumo y protección básica; preparación para batería y paneles solares.
- **Integración mecánica mínima:** fijaciones y protección provisionales suficientes para realizar las pruebas previstas.
- **Pruebas y validación básica:** protocolo de ensayo en entorno experimental y registro de resultados.
- **Documentación:** esquemáticos, *layouts*, BOM, guía de montaje y notas técnicas del *firmware*.

3.3. Requerimientos

3.3.1. Requerimientos funcionales

Los datos ambientales que se pretenden recolectar en este prototipo son:

- **Temperatura y humedad del aire.**
- **Humedad y temperatura del suelo.**
- **Irradiancia solar.**

En un futuro, se tiene pensado ampliar y/o modificar los datos recolectados en función de su utilidad a la hora de entrenar a la IA, pero como es un prototipo, empezaremos por estos que son los básicos.

La lanza deberá ademas cumplir con otros requerimientos funcionales tales como:

- **Almacenamiento local:** La lanza deberá poder almacenar los datos tomados por los sensores localmente durante como mínimo 2 semanas, hasta el envío de estos al servidor.
- **Registro de metadatos:** La lanza deberá guardar junto a los datos tomados, metadatos como la fecha y hora para el posterior análisis.
- **Autonomía de operación:** La lanza deberá ser capaz de operar de forma continua sin intervención humana. Para ello se empleará el uso de paneles solares y baterías para proporcionar al sistema la energía necesaria para su correcto funcionamiento.
- **Comunicación inalámbrica:** La lanza deberá poder comunicarse mediante radiofrecuencia con algún tipo de receptor para enviar los datos tomados.

3.3.2. Requerimientos no funcionales

Entre los requerimientos no funcionales del proyecto destacan:

- **Fiabilidad y precisión:** Los sensores deben ofrecer mediciones precisas para evitar errores en la toma de decisiones.
- **Eficiencia energética:** La lanza deberá ser capaz de manejar la energía de forma eficiente para prolongar al máximo las baterías y asegurar un funcionamiento continuo y correcto.
- **Fácil mantenimiento:** Tanto los componentes como los circuitos deberán ser de fácil mantenimiento y fáciles de encontrar en el mercado en caso de necesitar cambiar algún componente.
- **Escalabilidad y flexibilidad:** El proyecto deberá contemplar y facilitar la modificación y adición de sensores a la lanza.

3.4. Posibles obstáculos i riesgos

Algunos de los principales desafíos que pueden surgir durante el desarrollo del proyecto incluyen:

- **Calibración de sensores:** riesgo de sesgos o derivas que comprometan la calidad de los datos. Requiere definir procedimientos de verificación frente a referencias y registrar factores de calibración como metadatos.
- **Errores de diseño de las PCBs:** Es posible que incluso tras revisar multiples veces tu diseño a la hora de la verdad hayas cometido algún error en tu PCB. Es por eso que se utilizarán metodos rápidos de verificación de placas como placas de cobre fresadas y se intentará realizar todos los circuitos con tiempo para tener margen de error.
- **Suministro de componentes y plazos:** posibles roturas de stock o largos tiempos de entrega. Conviene prever equivalentes compatibles y cerrar compras críticas en fases tempranas.
- **Riesgos de calendario académico:** acotación del alcance y priorización de hitos para asegurar la entrega del TFG incluso ante incidencias técnicas.

3.5. Competencias técnicas

Considerando la naturaleza del proyecto y las tareas que se desarrollarán, las siguientes competencias técnicas se consideran las más adecuadas y directamente relacionadas con el trabajo a realizar:

- **CEC1.1:** Diseñar un sistema basado en microprocesador o microcontrolador. [En profundidad](#).
- **CEC1.2:** Diseñar y/o configurar un circuito integrado utilizando las herramientas de software adecuadas. [En profundidad](#).
- **CEC2.1:** Analizar, evaluar, seleccionar y configurar plataformas hardware para el desarrollo y la ejecución de aplicaciones y servicios informáticos. [En profundidad](#).
- **CEC2.2:** Programar considerando la arquitectura hardware, tanto en ensamblador como en alto nivel. [Bastante](#).
- **CEC2.3:** Desarrollar y analizar software para sistemas basados en microprocesadores y sus interfaces con usuarios y otros dispositivos. [Bastante](#).
- **CEC2.4:** Diseñar e implementar software de sistema y de comunicaciones. [Bastante](#).
- **CEC3.1:** Analizar, evaluar y seleccionar las plataformas hardware y software más adecuadas para el soporte de aplicaciones embebidas y de tiempo real. [Bastante](#).

Capítulo 4

Metodología y rigor

El enfoque metodológico utilizado para el desarrollo de este proyecto se basa en el método Kanban, un sistema ágil de gestión de proyectos que permite un seguimiento visual y flexible de las tareas a medida que avanzan en el proceso de trabajo. Este enfoque se caracteriza por su capacidad para adaptar el flujo de trabajo de manera continua, promoviendo la eficiencia, la mejora continua y la visibilidad de todas las etapas del proyecto.

Este enfoque divide el proyecto en etapas o grupos los cuales tienen tareas asignadas que se colocan en columnas de un tablero dependiendo del estado de la tarea. En este caso utilizaremos estas cuatro columnas para distribuir las tareas:

- **To Do:** En esta columna se almacenarán todas las tareas que aún no han comenzado.
- **In Progress:** Aquí se trasladarán las tareas que están siendo trabajadas activamente.
- **Blocked:** En esta columna se incluirán aquellas tareas que no pueden avanzar debido a algún problema o impedimento.
- **Done:** Una vez que una tarea ha sido finalizada, se moverá a esta columna.

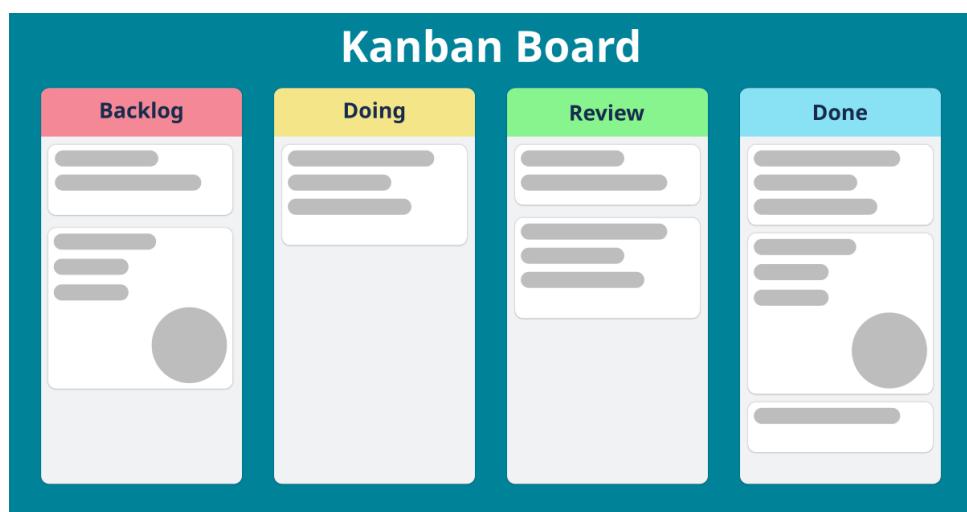


Figura 1: Ejemplo de tablero Kanban. Wikipedia. [4]

A medida que el proyecto avance, reordenaremos y priorizaremos algunas tareas según su importancia y urgencia. Las tareas más críticas o aquellas que dependen de otras se completarán primero para garantizar un flujo de trabajo eficiente y sin retrasos.

4.1. Herramientas

Para llevar a cabo y coordinar correctamente el proyecto utilizaremos principalmente estas herramientas:

- **Correo electrónico:** Para la comunicación constante con el director del TFG, el equipo de la empresa y los clientes.
- **Github:** [5] Para la gestión y control de versiones del código y documentos asociados al proyecto.
- **Google Meet:** Para realizar reuniones periódicas y mantener una comunicación fluida entre todos los involucrados en el proyecto.
- **Trello:** [6] Para gestionar y seguir el progreso de las tareas del proyecto mediante un tablero Kanban.

Capítulo 5

Descripción de las tareas

5.1. Introducción

La realización de este proyecto se realizó en gran medida en horas laborales dentro de la empresa Domo21. La parte de GEP se realizó en el Q2-2024/25. El proyecto en sí empezó el día 15 de septiembre de 2025 y la defensa del trabajo es el 20 de enero de 2026. Eso nos da unos 4 meses y algo para la realización del trabajo. El total de horas estimadas para completar el proyecto es de unas **720 horas**. Dentro de este periodo encontramos la fase de desarrollo, la cual ocupará la mayor parte del tiempo.

5.2. Tareas

5.2.1. Gestión del Proyecto

- **GP1 ⇒ Alcance (10 h):** Redacción de la primera entrega de la asignatura de Gestión de Proyectos, que incluirá el contexto del proyecto, su justificación, el alcance del mismo y la metodología a emplear.
- **GP2 ⇒ Planificación (10 h):** Redacción de la Entrega 2 de la asignatura de GEP, donde se exponen las tareas a hacer y su planificación, el diagrama de Gantt y la gestión de los riesgos que puedan aparecer. **Dependencias:** GP1
- **GP3 ⇒ Gestión económica y sostenibilidad (10 h):** Redacción de la Entrega 3 de la asignatura de GEP, donde se comenta el presupuesto del proyecto y el informe de sostenibilidad. **Dependencias:** GP2
- **GP4 ⇒ Documento final (15 h):** Recopilación, revisión y corrección de las tres entregas anteriores y la posterior redacción de la Entrega final de GEP. **Dependencias:** GP3
- **GP5 ⇒ Reuniones (20 h):** Realización de reuniones periódicas con el director del TFG y la empresa para supervisar el avance y tomar decisiones estratégicas.

5.2.2. Estudio Previo

- **EP1 ⇒ Estudio y selección de componentes (18 h):** Investigación y comparación de alternativas para cada bloque del sistema: microcontrolador, sensores ambientales, regulación de alimentación, etc. Se valoran criterios de precisión, interfaz utilizada, consumo, disponibilidad, coste y facilidad de integración.
- **EP2 ⇒ Diseño de la arquitectura del sistema (12 h):** Definición de la arquitectura general del prototipo: diagrama de bloques, buses de comunicación, estrategia de muestreo, gestión de energía y enfoque de almacenamiento y transmisión de datos orientado a las pruebas.
- **EP3 ⇒ Primeras pruebas con placas de evaluación (20 h):** Pruebas iniciales con placas de evaluación para asegurar que los sensores y el microcontrolador se entienden correctamente. Se revisan lecturas, configuración y estabilidad básica, detectando posibles problemas antes del diseño definitivo. **Dependencias:** EP1, EP2

5.2.3. Desarrollo Hardware

- **HW1 ⇒ Selección final de componentes (16 h):** Elección definitiva de los componentes a utilizar, con elaboración de una BOM preliminar y comprobación de compatibilidades eléctricas y mecánicas. **Dependencias:** EP1
- **HW2 ⇒ Prototipo con placa fresada (16 h):** Integración preliminar de los componentes principales en un prototipo de cobre fresado para validar conexiones a bajo coste. **Dependencias:** HW1, EP3
- **HW3 ⇒ Diseño esquemático PCB principal (22 h):** Desarrollo del esquemático completo de la placa principal: alimentación, MCU, buses, conectores para sensores, conector de programación/depuración, protecciones y puntos de test. **Dependencias:** HW2
- **HW4 ⇒ Diseño layout PCB principal (26 h):** Diseño físico y enrutado de la placa: colocación, reglas de diseño, planos de masa, ubicación de componentes y conectores. **Dependencias:** HW3
- **HW5 ⇒ Fabricación y montaje (10 h):** Preparación y envío de la documentación de fabricación (Gerbers, BOM...) a una empresa especializada, seguimiento del proceso y resolución de incidencias. Recepción de la PCB, inspección visual y finalización del montaje mediante soldadura manual de los componentes no ensamblados por el proveedor. Verificación básica de continuidad y revisión de posibles errores de montaje antes de iniciar las pruebas funcionales. **Dependencias:** HW4

5.2.4. Desarrollo Firmware

- **FW1 ⇒ Drivers (24 h):** Implementación de drivers para los componentes seleccionados y sus interfaces (I2C, SPI, UART...). Incluye inicialización, configuración, lectura de medidas, conversión a unidades físicas y gestión básica de errores. **Dependencias:** EP3, HW5

- **FW2 ⇒ Flujo de programa principal (16 h):** Desarrollo del flujo principal del sistema: arranque, inicialización de periféricos, temporización del muestreo y estructura general de ejecución. **Dependencias:** FW1
- **FW3 ⇒ Estructura de memoria y formato de registro (18 h):** Diseño de cómo se almacenan los datos en memoria: organización del espacio, estructura de bloques y registros, metadatos asociados y mecanismos de integridad. Incluye el tratamiento de casos límite como memoria llena, datos inválidos y recuperación tras reinicios. **Dependencias:** FW1
- **FW4 ⇒ Comunicación inalámbrica (12 h):** Implementación de una interfaz BLE básica para enviar datos. Esta tarea se plantea como funcionalidad no prioritaria y puede limitarse a un perfil mínimo según el tiempo disponible. **Dependencias:** FW2, FW3

5.2.5. Pruebas y Validación

- **PV1 ⇒ Verificación eléctrica y funcional de la PCB (14 h):** Puesta en marcha de la placa: comprobación de tensiones, reguladores y consumo, verificación del acceso de programación/depuración y pruebas básicas de comunicación con los periféricos. Si aparecen problemas, se aplican correcciones de montaje (rework) o ajustes de firmware para dejar una base estable. **Dependencias:** HW5
- **PV2 ⇒ Pruebas de los sensores en entornos controlados (18 h):** Validación de cada sensor en un entorno controlado para observar su respuesta a cambios (por ejemplo temperatura/humedad/iluminación) y confirmar que el comportamiento es coherente con lo esperado. **Dependencias:** PV1, FW2, FW3
- **PV3 ⇒ Pruebas del sistema completo en entornos controlados (12 h):** Pruebas del sistema integrado durante períodos prolongados para observar estabilidad, continuidad de adquisición y gestión de datos. **Dependencias:** PV2

5.2.6. Documentación

- **DOC1 ⇒ Redactar el TFG (90 h):** Elaboración de la memoria del Trabajo de Fin de Grado.

5.3. Recursos

Seguidamente se describen los recursos necesarios, tanto humanos como materiales y digitales, para el correcto desarrollo del proyecto.

5.3.1. Recursos humanos

Los recursos humanos necesarios para el desarrollo del proyecto coinciden con los actores descritos en la Sección 1.4. En este sentido, el trabajo se ha realizado principalmente por el

estudiante, con el apoyo y supervisión del tutor académico, y teniendo como referencia los requisitos y el contexto de aplicación aportados por la entidad destinataria del prototipo.

5.3.2. Recursos materiales

- **[PC] Ordenador:** Agrupa todos los ordenadores que se utilizarán durante el proyecto.
- **[MUL] Multímetro:** Herramienta esencial para la verificación de conexiones eléctricas así como medir voltajes, corrientes o resistencias.
- **[OSC] Osciloscopio:** Herramienta utilizada para la observación gráfica de señales eléctricas en circuitos, muy útil para ver protocolos de comunicación.
- **[FA] Fuente de alimentación:** Dispositivo que proporciona un voltaje y corriente constantes para la alimentación de circuitos electrónicos. Permite simular condiciones de funcionamiento real sin necesidad de baterías.
- **[SOL] Estación de soldadura:** Conjunto de herramientas utilizadas para ensamblar circuitos electrónicos. Incluye principalmente soldador, pistola de calor, pinzas, estaño y flux.
- **[FRE] Fresadora:** Máquina de mecanizado utilizada para fabricar prototipos rápidos de PCBs mediante el fresado de placas de cobre.

5.3.3. Recursos digitales

- **[KCD] KiCad:** Herramienta para el diseño de circuitos electrónicos y PCBs.
<https://www.kicad.org/>
- **[IDE] STM32CubeIDE:** IDE para escribir código.
<https://www.st.com/en/development-tools/stm32cubeide.html>
- **[TER] Tera Term:** Terminal para debugar el firmware.
<https://teratermproject.github.io/index-en.html>
- **[ONS] OnShape:** Herramienta online para diseñar y modelar objetos 3D.
<https://www.onshape.com/>
- **[GIT] GitHub:** Herramienta online para alojar y modificar repositorios utilizando el control de versiones de *Git*.
<https://github.com/>
- **[TRL] Trello:** Herramienta online para gestionar proyectos con el método *Kanban*.
<https://trello.com/>
- **[OVL] OverLeaf:** Herramienta online basada en LaTeX utilizada para la redacción de todos los documentos.
<https://www.overleaf.com/>

- **[DRW] Draw.io:** Herramienta online para hacer diagramas.
<https://www.drawio.com/>
- **[GNT] GanttProject:** Herramienta para crear diagramas de *Gantt*.
<https://www.ganttpoint.biz/>
- **[GMT] Google Meet:** Herramienta online para realizar videollamadas.

Capítulo 6

Estimaciones y Gantt

En la Tabla 1 se muestra un resumen de las tareas que realizaré, así como de sus dependencias y los recursos relacionados con cada tarea.

ID	Nombre	Horas	Dependencias	Recursos
	Gestión del Proyecto	65		
GP1	Alcance	10	-	PZ, AB, CM, PC, OVL
GP2	Planificación	10	GP1	PZ, AB, CM, PC, OVL, GNT
GP3	Gestión económica y sostenibilidad	10	GP2	PZ, AB, CM, PC, OVL
GP4	Documento final	15	GP3	PZ, AB, CM, PC, OVL
GP5	Reuniones	20	-	PZ, AB, CM, D21, PC, GMT
	Estudio Previo	80		
EP1	Estudio y selección de componentes	30	-	PZ, AB, PC
EP2	Diseño de la arquitectura del sistema	20	-	PZ, PC
EP3	Primeras pruebas con placas de evaluación	30	EP1, EP2	PZ, PC
	Desarrollo Hardware	190		
HW1	Selección final de componentes	25	EP1	PZ, PC
HW2	Prototipo con placa fresada	30	HW1, EP3	PZ, AV, PC, KCD, FRE
HW3	Diseño esquemático PCB principal	45	HW2	PZ, AV, PC, KCD
HW4	Diseño layout PCB principal	55	HW3	PZ, AV, PC, KCD
HW5	Fabricación y montaje	35	HW4	PZ, AV, PC, SOL, MUL
	Desarrollo Firmware	165		
FW1	Drivers	60	EP3, HW5	PZ, PC, IDE, TER
FW2	Flujo de programa principal	35	FW1	PZ, PC, IDE, TER
FW3	Estructura de memoria y formato de registro	45	FW1	PZ, PC, IDE, TER
FW4	Comunicación inalámbrica	25	FW2, FW3	PZ, PC, IDE, TER
	Pruebas y Validación	110		
PV1	Verificación eléctrica y funcional de la PCB	30	HW5	PZ, PC, MUL, FA, OSC
PV2	Pruebas de los sensores en entornos controlados	45	PV1, FW2, FW3	PZ, PC, MUL
PV3	Pruebas del sistema completo en entornos controlados	35	PV2	PZ, PC, MUL
	Documentación	110		
DOC1	Redactar el TFG	110	-	PZ, AB, CM, PC, OVL, DRW
	Total	720		

Tabla 1: Tabla resumen de las tareas. Elaboración propia.

En la Figura 2, se representa la planificación de las tareas mediante un diagrama de Gantt. En él, se muestra el espacio de tiempo de cada tarea, así como las dependencias entre ellas.

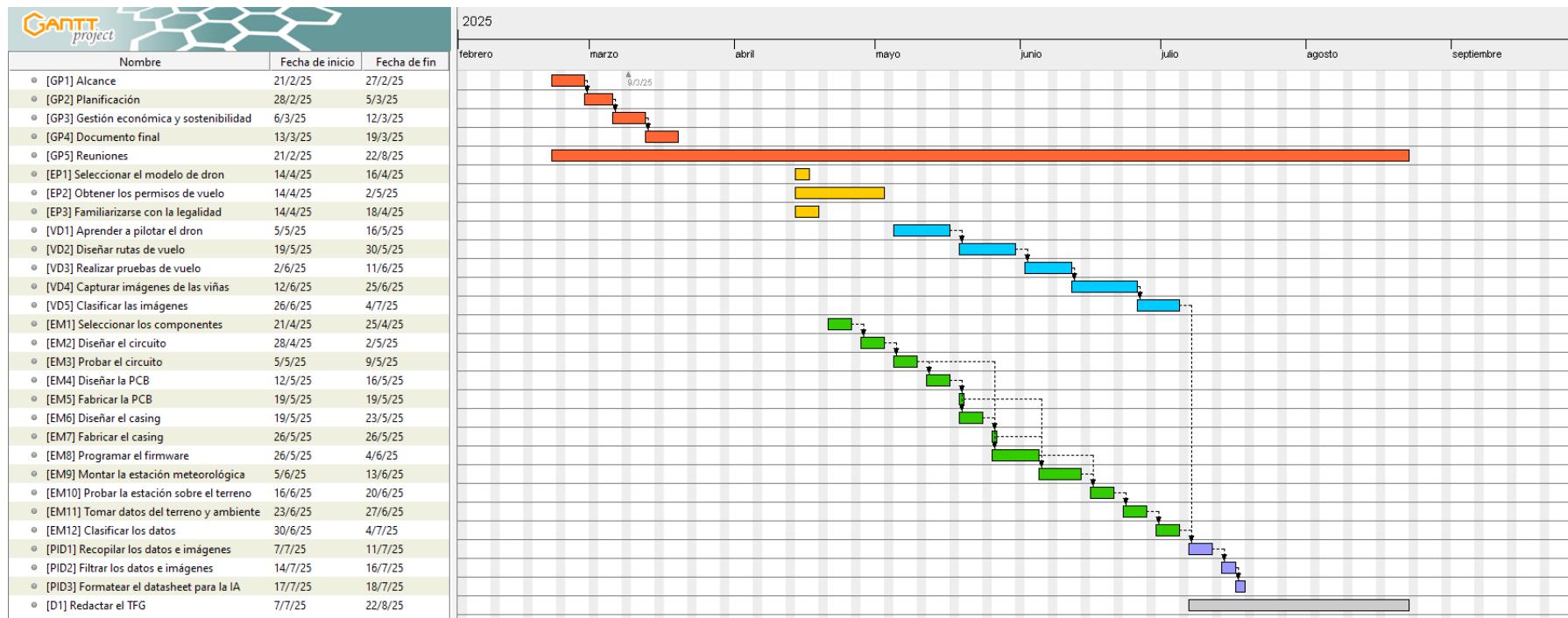


Figura 2: Diagrama de Gantt de las tareas propuestas. Elaboración propia.

Capítulo 7

Gestión de riesgos: Planes alternativos y obstáculos

Es fundamental que estemos preparados para afrontar cualquier imprevisto y, en el peor de los casos, dediquemos más tiempo a una tarea para solucionar los problemas que puedan aparecer. A continuación, se detallan algunos de los posibles contratiempos que podrían afectar el progreso del proyecto.

Uno de los principales riesgos asociados a este proyecto es la posible demora en la fabricación de las placas de circuito impreso (PCBs). Dado que la fabricación y el envío se realizarán en China, estamos expuestos a imprevistos como retrasos en la producción, problemas logísticos o incluso paros laborales. Si esto llegara a suceder, nuestro primer paso será ponernos en contacto inmediato con el proveedor para obtener información detallada sobre la situación. En caso de que el retraso sea significativo, exploraremos la posibilidad de recurrir a otros proveedores para evitar mayores demoras.

Una vez recibidas las PCBs, también existe el riesgo de que surjan problemas relacionados con su diseño. Esto podría incluir fallos en el trazado de los circuitos, errores en las conexiones o incompatibilidades con otros componentes del sistema. Tales problemas podrían retrasar considerablemente el ensamblaje de la estación meteorológica. Para minimizar este riesgo, nos aseguraremos de que el diseño de la PCB sea completamente correcto antes de proceder con la fabricación final. Además, realizaremos pruebas preliminares con placas fresadas para verificar su funcionamiento. En caso de que el problema sea debido a un error de fábrica, nos veremos obligados a reclamar al proveedor para que nos envíe nuevamente las PCBs correctas, lo que podría llevar algún tiempo adicional.

Otro contratiempo común es la aparición de errores en el código (*bugs*) o de casos no contemplados durante el desarrollo del *firmware*. Por ejemplo, situaciones como memoria llena, lecturas puntuales inválidas, reinicios inesperados o fallos intermitentes de comunicación pueden provocar comportamientos erráticos que no siempre aparecen en las primeras pruebas. Aunque suelen ser problemas solucionables, pueden consumir tiempo si no se detectan pronto y de forma sistemática.

Capítulo 8

Presupuesto

8.1. Identificación de costes

En proyectos de esta naturaleza, es fundamental elaborar un presupuesto detallado para estimar los costes asociados. A continuación, se presenta un presupuesto que incluye los costes estimados en recursos humanos, *hardware* y *software*, así como otros gastos generales previstos durante el desarrollo. Además, se han considerado posibles imprevistos en el cálculo del presupuesto.

8.1.1. Recursos humanos

Los costes en recursos humanos hacen referencia a los sueldos de los trabajadores del proyecto, dependiendo de su rol en este. Como en este proyecto participarán ingenieros de diferentes especialidades, me he centrado en los roles que desempeñaré yo mismo.

En la Tabla 2 se muestran el salario por hora de cada rol que tendré y el coste real que asume la empresa de este salario después de sumarle la tributación a la Seguridad Social (calculada como $1,33 \times \text{sueldo}$).

Rol	Salario por hora (€/h)	Salario por hora + SS (€/h)
Ingeniero de Requisitos [IdR]	25,20	33,52
Ingeniero de Desarrollo [IdD]	18,80	25,00

Tabla 2: Tabla con el sueldo estimado de cada rol. Elaboración propia.

En la Tabla 3 se detallan las tareas que realizaré, junto con las horas de dedicación estimadas para cada una, el rol que desempeñaré y el coste total.

ID	Tarea	Horas	Roles	Coste (€)
	Gestión del Proyecto	65		2.178,80
GP1	Alcance	10	IdR	335,20
GP2	Planificación	10	IdR	335,20
GP3	Gestión económica y sostenibilidad	10	IdR	335,20
GP4	Documento final	15	IdR	502,80
GP5	Reuniones	20	IdR	670,40
	Estudio Previo	80		2.000,00
EP1	Estudio y selección de componentes	30	IdD	750,00
EP2	Diseño de la arquitectura del sistema	20	IdD	500,00
EP3	Primeras pruebas con placas de evaluación	30	IdD	750,00
	Desarrollo Hardware	190		4.750,00
HW1	Selección final de componentes	25	IdD	625,00
HW2	Prototipo con placa fresada	30	IdD	750,00
HW3	Diseño esquemático PCB principal	45	IdD	1.125,00
HW4	Diseño layout PCB principal	55	IdD	1.375,00
HW5	Fabricación y montaje	35	IdD	875,00
	Desarrollo Firmware	165		4.125,00
FW1	Drivers	60	IdD	1.500,00
FW2	Flujo de programa principal	35	IdD	875,00
FW3	Estructura de memoria y formato de registro	45	IdD	1.125,00
FW4	Comunicación inalámbrica	25	IdD	625,00
	Pruebas y Validación	110		2.750,00
PV1	Verificación eléctrica y funcional de la PCB	30	IdD	750,00
PV2	Pruebas de los sensores en entornos controlados	45	IdD	1.125,00
PV3	Pruebas del sistema completo en entornos controlados	35	IdD	875,00
	Documentación	110		3.218,60
DOC1	Redactar el TFG	110	IdR, IdD	3.218,60
TOTAL		720		19.022,40

Tabla 3: Tabla con los costes estimados por tarea. Elaboración propia.

8.1.2. Hardware

Para el desarrollo del prototipo se han utilizado herramientas habituales de laboratorio electrónico para el diseño, montaje y verificación de circuitos. Dado que estas herramientas no se adquieren exclusivamente para este proyecto, sino que se reutilizan en otros desarrollos, su coste se imputa mediante amortización.

Para calcular la amortización, se ha utilizado la siguiente fórmula:

$$\text{Amortización} = \frac{\text{Coste} - \text{Valor residual}}{\text{Vida útil}}$$

En nuestro caso, consideraremos un tiempo medio de vida útil de aproximadamente 8 años para cada herramienta de *hardware* y que el valor residual es nulo, lo que simplifica la fórmula a:

$$\text{Amortización} = \frac{\text{Coste}}{8}$$

En la Tabla 4 se muestra el coste estimado de cada herramienta junto con su amortización anual. Para el presupuesto del proyecto se imputará la parte proporcional correspondiente a 4 meses de trabajo (4/12).

Hardware	Coste (€)	Amortización (€/año)
Multímetro	~60	7,50
Osciloscopio	~400	50,00
Fuente de alimentación	~60	7,50
Estación de soldadura	~70	8,75
Ordenador + periféricos	~1.000	125,00
TOTAL		198,75

Tabla 4: *Tabla con los costes estimados del hardware empleado. Elaboración propia.*

Además del equipamiento amortizable, el prototipo requiere materiales y servicios directamente imputables al proyecto (compra de componentes, fabricación de PCB, etc.). El detalle técnico completo se incluye en la Sección 11.6.2; en esta sección únicamente se presenta el resumen económico.

Materiales y servicios del prototipo	Coste (€)
Componentes	206,73
PCB	115,20
Envíos e impuestos	56,88
Consumibles (estaño, flux, cables, protoboards...)	34,68
TOTAL	413,49

Tabla 5: *Resumen de costes de materiales y servicios del prototipo. Elaboración propia.*

8.1.3. Software

Los programas utilizados para el desarrollo de este proyecto son todos gratuitos, por lo que no suponen un coste adicional.

8.1.4. Gastos generales

En este proyecto también existen gastos generales, como electricidad y transporte. Para aproximar el coste de estos gastos, utilizaremos datos generales de precios y tomaremos como periodo de uso el tiempo estimado de desarrollo del proyecto, que es de aproximadamente cuatro meses.

El cálculo del consumo eléctrico se estima en función del uso semanal de cada dispositivo y su consumo energético. Para calcular el consumo utilizaremos la siguiente fórmula:

$$\text{Consumo semanal (kWh)} = \frac{\text{Consumo (W)} \times \text{Uso semanal (h)}}{1000}$$

En la Tabla 6 se presenta un resumen del consumo eléctrico estimado para los principales dispositivos utilizados en el proyecto.

Dispositivo	Consumo (W)	Uso semanal (h)	Consumo semanal (kWh)
Osciloscopio	50	3	0,15
Fuente de alimentación	100	3	0,30
Estación de soldadura	50	3	0,15
Ordenador	200	15	3,00
Luz	150	20	3,00
Otros	50	2	0,10
TOTAL			6,70

Tabla 6: *Tabla con los consumos eléctricos esperados. Elaboración propia.*

Para estimar el coste eléctrico se utiliza un precio medio de 0,185 €/kWh. En cuanto al transporte, para los trayectos entre la oficina y casa utilizaré el transporte público.

Gastos	Coste al mes (€/mes)	Coste 4 meses (€)
Electricidad	5,36	21,44
T-jove	14,67	58,68
TOTAL		80,12

Tabla 7: *Tabla con los costes estimados para los gastos generales. Elaboración propia.*

8.2. Estimación de costes

Para completar la estimación del presupuesto, es necesario incluir el coste de contingencia, es decir, un porcentaje destinado a cubrir posibles desviaciones e imprevistos. En este caso, dada la naturaleza del proyecto, se ha fijado dicho porcentaje en un 15 %, lo que da un total de **2.937,34 €**.

En la Tabla 8 se presentan diversos imprevistos junto con su posible coste adicional para el proyecto. Para estimar el coste relativo, multiplicamos el coste total por la probabilidad de que dicho imprevisto ocurra.

Imprevistos	Probabilidad (%)	Coste (€)	Coste relativo (€)
Problemas con componentes o PCB	25	~80,00	20,00
Envíos urgentes	20	~60,00	12,00
Horas extra	50	~2.000	1.000,00
TOTAL			1.032,00

Tabla 8: *Tabla con los costes estimados para cada imprevisto. Elaboración propia.*

En la Tabla 9 vemos un resumen del presupuesto completo estimado para este proyecto.

	Coste (€)
Recursos humanos	19.022,40
<i>Hardware</i>	479,74
<i>Software</i>	0,00
Gastos generales	80,12
Contingencias (15 %)	2.937,34
Imprevistos	1.032,00
TOTAL	23.551,60

Tabla 9: *Tabla con los costes estimados del proyecto. Elaboración propia.*

8.3. Control de gestión

Para finalizar, vamos a definir una serie de métricas para visualizar posibles desviaciones en el presupuesto.

$$\text{Desviación presupuestaria (\%)} = \frac{\text{Gasto real} - \text{Presupuesto estimado}}{\text{Presupuesto estimado}} \times 100$$

$$\text{Desviación de horas (\%)} = \frac{\text{Horas reales} - \text{Horas estimadas}}{\text{Horas estimadas}} \times 100$$

Capítulo 9

Informe de sostenibilidad

9.1. Autoevaluación

A lo largo de la carrera, hemos tenido varias asignaturas que han puesto énfasis en la sostenibilidad desde distintos enfoques. Esto me ha ayudado a entender la importancia de considerar el impacto ambiental, económico y social en cualquier proyecto.

En el aspecto ambiental, recuerdo una asignatura en la que vimos el problema de los residuos electrónicos y cómo muchos acaban en países con menos recursos, formando enormes “cementerios tecnológicos”. Esto me hizo reflexionar sobre la necesidad de diseñar productos con un ciclo de vida más sostenible, reduciendo el desperdicio y fomentando el reciclaje.

En cuanto a la parte económica, aprendimos sobre macroeconomía y cómo hacer presupuestos básicos en empresas. Aunque no me dedique directamente a la gestión financiera, estos conocimientos son útiles para entender la viabilidad de un proyecto, controlar gastos y optimizar recursos. Saber estimar costes y planificar la amortización de materiales es clave para que cualquier iniciativa sea sostenible a largo plazo.

El ámbito social es, quizás, el que menos hemos trabajado en la carrera. Hemos desarrollado proyectos pensando en mejorar procesos industriales o tecnológicos, pero sin profundizar demasiado en su impacto social. Sin embargo, me doy cuenta de que un proyecto realmente sostenible no solo debe ser viable económicamente y respetuoso con el medio ambiente, sino que también tiene que aportar algo positivo a la sociedad.

En definitiva, estas tres dimensiones están muy conectadas y son esenciales en cualquier desarrollo tecnológico. Haber adquirido esta visión a lo largo de la carrera me será útil en el futuro para diseñar soluciones más equilibradas y responsables.

9.2. Dimensión Económica

Referente a PPP: ¿Has estimado el coste de la realización del proyecto (recursos humanos y materiales)?

Aunque no es mi responsabilidad directa dentro de la empresa calcular los costes y evaluar la viabilidad económica del proyecto, he llevado a cabo un análisis de los sueldos del personal necesario, el coste de los materiales y su amortización a lo largo de su vida útil.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará económicoamente tu solución a las existentes?

Al dirigirnos a un público más específico, los agricultores del sector vitivinícola, podemos adaptar mejor los requerimientos del proyecto, reduciendo así los costes en comparación con otras soluciones más generales ya existentes en el mercado.

9.3. Dimensión Ambiental

Referente a PPP: ¿Has estimado el impacto ambiental que tendrá la realización del proyecto? ¿Te has planteado minimizar el impacto, por ejemplo, reutilizando recursos?

El proyecto tendrá un impacto ambiental muy positivo una vez implementado, ya que contribuirá a optimizar el uso de recursos naturales en la agricultura. Sin embargo, durante su desarrollo habrá impactos negativos inevitables presentes en todos los proyectos de esta índole, como el consumo de energía o el uso de combustible para desplazamientos. Por otra parte, en este tipo de proyectos, la reutilización de recursos no creo que sea una estrategia particularmente aplicable.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará ambientalmente tu solución a las existentes?

Al ofrecer una mayor información sobre el estado de las viñas y del terreno a los viticultores, estos podrán tomar mejores decisiones referentes a estas, mejorando así el impacto ambiental que tendrán sobre el medio ambiente.

9.4. Dimensión Social

Referente a PPP: ¿Qué crees que te va a aportar a nivel personal la realización de este proyecto?

Este proyecto me permitirá adquirir experiencia en la comunicación con clientes y la interpretación de sus requisitos, lo que fortalecerá mis habilidades en la gestión de proyectos. Además, me proporcionará conocimientos técnicos en el sector industrial y en sistemas integrados. También tendrá la oportunidad de aprender a pilotar drones, algo que siempre he querido hacer.

Referente a la Vida Útil: ¿Cómo se resuelve actualmente el problema que quieres abordar (estado del arte)? ¿En qué mejorará socialmente (calidad de vida) tu solución a las existentes?

La identificación de problemas en los cultivos en la actualidad requiere una gran cantidad de trabajo manual, lo que implica un esfuerzo físico considerable por parte de los agricultores. Este proyecto facilitará la detección temprana de enfermedades en las viñas, permitiendo intervenir sólo en las áreas afectadas y reduciendo el trabajo necesario. Asimismo, el monitoreo del terreno permitirá optimizar el uso del agua, mejorando la eficiencia y reduciendo la carga de trabajo de los agricultores.

Referente a la Vida Útil: ¿Existe una necesidad real del proyecto?

Sí, el proyecto ha sido encargado por una asociación agrícola externa, lo que demuestra que existe una demanda concreta en el sector.

Capítulo 10

Diseño del sistema

El diseño del sistema se basa en cuatro pilares: sensórica ambiental, alimentación y autonomía, almacenamiento local y conexión inalámbrica. El objetivo del prototipo es adquirir medidas de forma periódica y fiable, conservar los datos localmente durante un mínimo de dos semanas y permitir posteriormente su volcado a un dispositivo externo. Estas metas se alinean con los requerimientos del proyecto y condicionan el diseño *hardware* y la arquitectura del *firmware* que se describen en los capítulos siguientes.

10.1. Arquitectura del sistema

La Figura 3 muestra un diagrama conceptual del sistema.

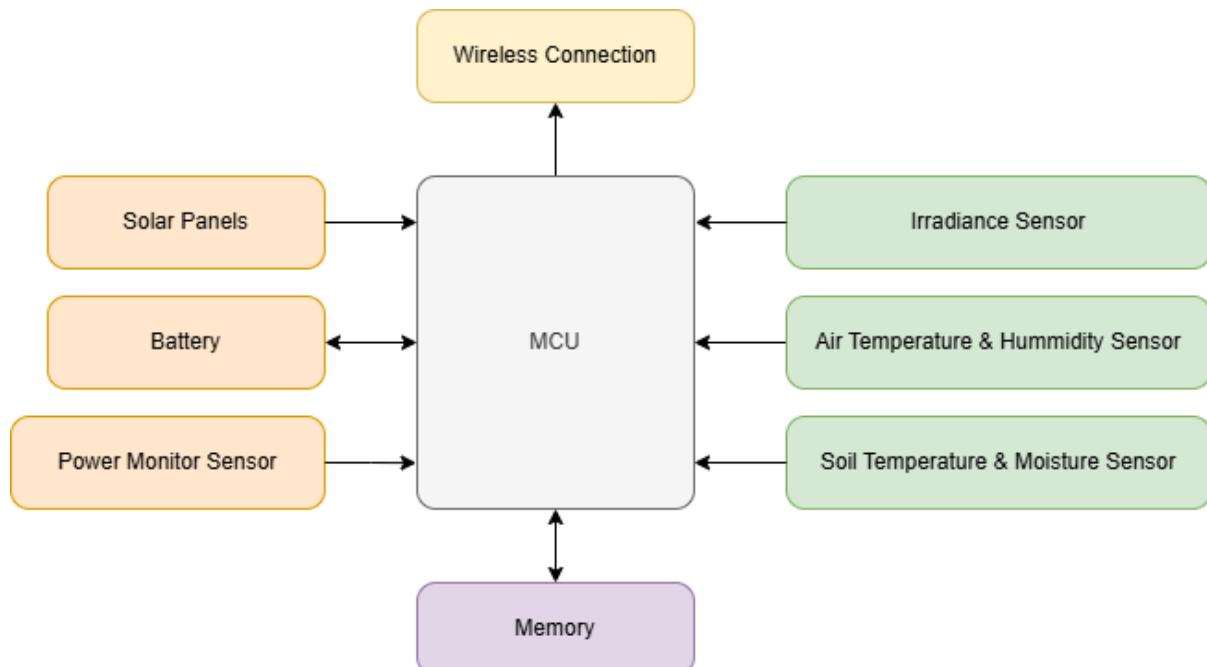


Figura 3: *Diagrama de bloques conceptual del sistema. Elaboración propia.*

10.2. Decisiones de diseño

En este apartado se recogen las decisiones de alto nivel que definen el comportamiento del sistema. Estas elecciones condicionan tanto el diseño *hardware* como la arquitectura del *firmware* que se desarrollan en los capítulos siguientes.

10.2.1. Periodo de muestreo

El periodo de muestreo se ha fijado en 20 minutos debido a que ofrece un equilibrio razonable entre resolución temporal y autonomía del sistema. En una lanza meteorológica no es necesario muestrear continuamente, pero tampoco interesa espaciar demasiado las medidas si luego se quiere analizar la evolución diaria o detectar cambios relativamente rápidos. Con 20 minutos se obtiene una serie temporal suficientemente densa para el estudio posterior, manteniendo el tiempo activo bajo control y permitiendo que la mayor parte del tiempo el nodo permanezca en bajo consumo.

10.2.2. Almacenamiento local robusto y simple

Una decisión clave del diseño es que los datos se almacenen localmente, de forma que la recolección no dependa de tener conectividad en el momento de medir. El objetivo mínimo es poder conservar al menos dos semanas de histórico; con una muestra cada 20 minutos esto supone: $2 \times 7 \times 24 \times 3 = 1008$ muestras en 14 días. Para cumplirlo se emplea una memoria no volátil de 32 kB (256 kbits) organizada en slots de tamaño fijo de 32 bytes. Un slot se reserva para cabecera y metadatos del sistema (identificación, puntero de escritura, número de slots ocupados, etc.), mientras que los 1023 slots restantes se dedican a almacenar muestras.

10.2.3. Tolerancia a datos inválidos

El sistema se ha planteado para que la aparición de datos inválidos en una muestra no implique perder el resto de información de ese instante. En condiciones reales es posible que algún sensor devuelva lecturas erróneas o no responda puntualmente; por ello, cada adquisición registra los datos válidos y marca explícitamente los que no lo sean mediante un vector de validez asociado a la muestra. Esta decisión mejora la robustez del registro y facilita el análisis posterior, ya que permite filtrar únicamente los valores inválidos sin romper la continuidad temporal de la serie.

10.2.4. Referencia temporal en cada muestra

Cada registro incluye una referencia temporal (fecha y hora), lo cual es esencial para el análisis posterior, tanto si se comparan variables entre sí como si se correlacionan con otras externas. Para ello se utiliza un reloj en tiempo real (RTC) que mantiene la hora de manera estable, garantizando que cada muestra queda etiquetada con el instante de adquisición.

10.2.5. Tensión de trabajo

Para simplificar la integración y asegurar compatibilidad entre bloques, el diseño se orienta a que la electrónica del sistema funcione a 3.3 V siempre que sea posible. Mantener un único bus de alimentación reduce conversiones de nivel, facilita la selección de sensores y periféricos y evita complejidad innecesaria. Las excepciones naturales a esta regla son los elementos de potencia como batería y paneles fotovoltaicos, que operan a tensiones distintas y se tratan dentro del subsistema de alimentación.

10.2.6. Modularidad del sistema

El diseño se plantea de forma modular permitiendo ampliar el sistema añadiendo sensores y, si alguno falla o no se adapta bien al despliegue, sustituirlo por otro equivalente sin rediseñar todo el conjunto. Para ello se define un nodo central que concentra el control y al que se conectan los sensores y accesorios (batería, panel fotovoltaico...), de manera que las variaciones se limiten a los módulos afectados.

10.2.7. Protocolo de comunicación inalámbrica

Como canal inalámbrico se propone usar *Bluetooth Low Energy* (BLE) por su bajo consumo y por la facilidad de interacción con dispositivos externos comunes, lo que encaja con el objetivo de realizar un volcado de datos en campo hacia un dron o un portátil. La comunicación se entiende como una función de extracción de información puntual (*dump*), no como transmisión continua.

Dentro del alcance del proyecto, la conectividad BLE se considera una funcionalidad complementaria: el sistema base está diseñado para funcionar correctamente midiendo y almacenando localmente aunque el enlace inalámbrico no se complete.

Capítulo 11

Diseño hardware

11.1. Selección de componentes

11.1.1. Gestión de potencia (PMIC)

Para el subsistema de alimentación se ha seleccionado el **AEM10941** de e-peas, un PMIC orientado a *energy harvesting*. Este tipo de circuito permite alimentar el sistema a partir de una fuente o *source* (en este caso paneles fotovoltaicos) y gestionar simultáneamente la carga de una batería de almacenamiento, maximizando la captación de energía y garantizando una salida regulada estable.



Figura 4: Chip AEM10941 en formato QFN y su respectivo módulo de evaluación. e-peas.

La elección del AEM10941 encaja especialmente bien con el contexto de una lanza meteorológica autónoma porque integra en un único chip: la gestión de entrada de los paneles solares, la carga y control del elemento de almacenamiento, y las salidas reguladas necesarias para alimentar la electrónica. Con ello se reducen los componentes y reguladores necesarios.

En cuanto a su funcionamiento, el AEM10941 dispone de un puerto de entrada para la fuente de *harvesting* donde se conectan los paneles solares. Además, incorpora dos puertos para elementos de almacenamiento (baterías o supercondensadores): uno principal y otro secundario. En el prototipo se utiliza únicamente el puerto principal, ya que la batería de *back-up* se conecta directamente a la MCU (pin VBAT) y se emplea exclusivamente para mantener los

servicios mínimos del microcontrolador, principalmente el RTC y el dominio de respaldo.

El circuito ofrece dos salidas reguladas, HVOUT (*High Voltage Output*) y LVOUT (*Low Voltage Output*). Esto permite alimentar cargas con diferentes necesidades de tensión o separar dominios de alimentación. En este diseño, como toda la electrónica se orienta a operar a 3.3 V, se utiliza únicamente HVOUT como raíl de alimentación y se prescinde de LVOUT, simplificando la distribución de potencia.

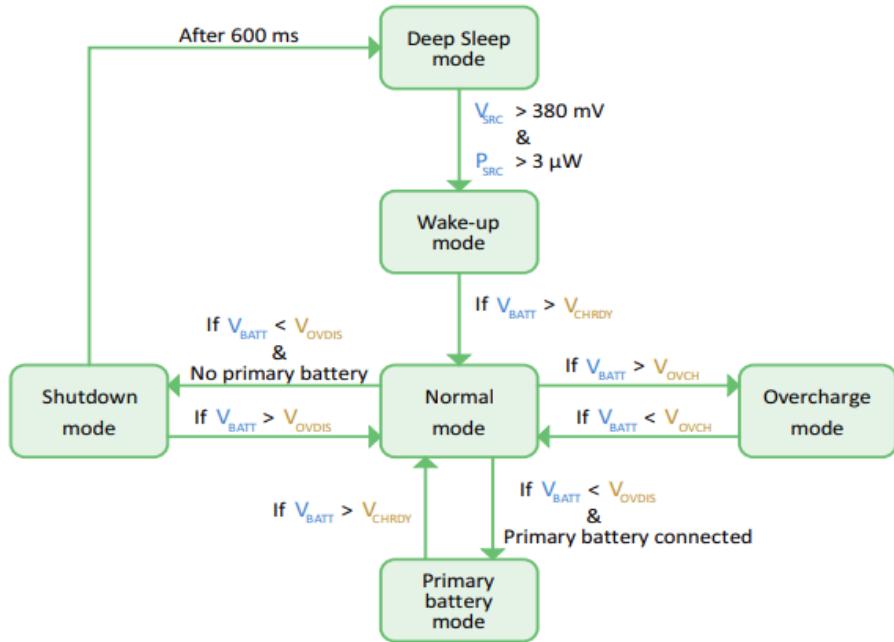


Figura 5: Diagrama de estados del AEM10941. e-peas.

Una ventaja adicional del AEM10941 es la presencia de pines de *status* que indican el modo de funcionamiento: disponibilidad de energía de entrada, estado de carga, habilitación de salidas y condiciones de alimentación insuficiente. Estas señales permiten que la MCU adapte su comportamiento y entre en bajo consumo cuando el sistema se encuentra en un estado energético crítico.

11.1.2. Monitoreo de potencia

Para monitorizar el estado energético del sistema se ha optado por el **INA3221** de Texas Instruments, un chip integrado que incorpora tres canales independientes para mediciones de tensión y corriente mediante resistencias *shunt*. El uso de tres canales permite monitorear simultáneamente los puntos clave del sistema: un canal dedicado a los paneles fotovoltaicos (entrada de energía), otro canal dedicado a la batería (estado de almacenamiento) y un tercer canal dedicado al consumo del sistema (MCU + sensores + periféricos), medido sobre el bus principal de alimentación. Con esta información se puede estimar la potencia instantánea y caracterizar el consumo medio en condiciones reales.

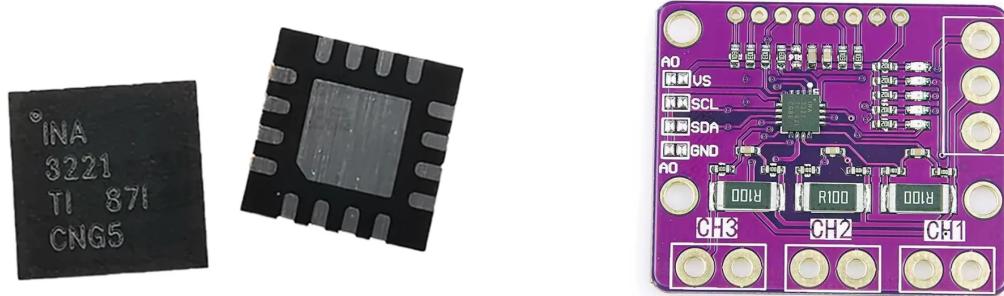


Figura 6: *Chip INA3221 en formato QFN y su respectivo módulo de evaluación. Texas Instruments.*

Además, el INA3221 dispone de pines de alerta configurables que pueden usarse para notificar a la MCU si una magnitud supera un umbral. Esto resulta muy útil para proteger el sistema ante situaciones como un consumo próximo al máximo que el PMIC puede suministrar o desviaciones de tensión que indiquen un estado de batería baja o una condición anómala.

11.1.3. Almacenamiento

Para el almacenamiento local de datos se ha escogido una memoria no volátil de tipo **FRAM** (*Ferroelectric RAM*), concretamente el modelo **MB85RS256B** de 256 kbits (32 kB).



Figura 7: *Chip MB85RS256B en formato SOIC-8. Fujitsu.*

Existen otras alternativas de memorias no volátiles, como la Flash o la EEPROM, pero presentan inconvenientes para este caso de uso. En memorias Flash la escritura suele estar ligada a una gestión por páginas, lo que implica operaciones de borrado previo y reescritura por bloques; esto añade complejidad al *firmware* y no se ajusta bien a un registro periódico de tamaño pequeño. La EEPROM se adapta mejor a escrituras granulares, pero típicamente ofrece una menor velocidad de escritura y una vida útil más limitada cuando se realizan escrituras repetidas durante largos períodos. La FRAM, en cambio, permite escritura directa

sin ciclos de borrado, con baja latencia y una resistencia muy alta a ciclos de escritura, lo que simplifica la implementación del registro y mejora la robustez del sistema a largo plazo.

	FRAM	EEPROM	Flash
Escritura	Directa	Directa	Requiere borrado por páginas
Velocidad de escritura	Alta	Media	Media
Ciclos de escritura típicos	10^{12} a 10^{14}	10^5 a 10^6	10^4 a 10^5
Energía en escritura	Baja	Alta	Alta
Granularidad de actualización	Byte	Byte	Página
Capacidad típica disponible	KB a pocos MB	KB a MB	MB a GB
Complejidad en firmware	Muy baja	Media	Alta
Coste por bit	Alto	Medio	Bajo

Tabla 10: *Comparación entre tecnologías de memoria no volátil. Elaboración propia.*

Como podemos observar en la Tabla 10, la FRAM también presenta algunos defectos, como poca capacidad de almacenamiento disponible en el mercado y un mayor coste por bit frente a sus alternativas. Aun así, para este prototipo no hace falta una capacidad enorme: interesa más que sea fiable y que aguante bien el uso repetitivo de escritura, y por eso se ha priorizado la FRAM frente a otras alternativas.

11.1.4. Sensor de irradiancia

La irradiancia solar (W/m^2) se mide idealmente con un piranómetro; no obstante su coste es elevado y se reserva normalmente para instrumentación de alta precisión. Dado que el objetivo del prototipo no es realizar una medida certificada sino capturar tendencias y obtener una estimación útil, se opta por usar una aproximación basada en un sensor óptico y una conversión experimental.

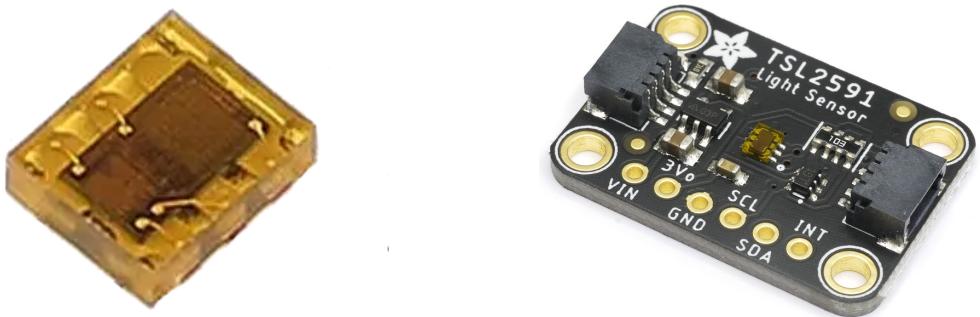


Figura 8: *Chip TSL2591 en formato QFN y su respectivo módulo de evaluación. Adafruit.*

Se ha seleccionado el sensor **TSL2591**, capaz de medir luz visible e infrarroja. A partir de ambas componentes se estima la iluminancia en *lux*. Los *lux* representan el flujo luminoso por unidad de área ponderado por la sensibilidad del ojo humano; por tanto, no equivalen directamente a irradiancia física (W/m^2). Sin embargo, mediante calibración en condiciones representativas es posible ajustar una relación empírica para aproximar irradiancia a partir de lux. Esta solución sacrifica precisión absoluta frente a un piranómetro, pero permite capturar tendencias y niveles aproximados con un coste mucho menor.

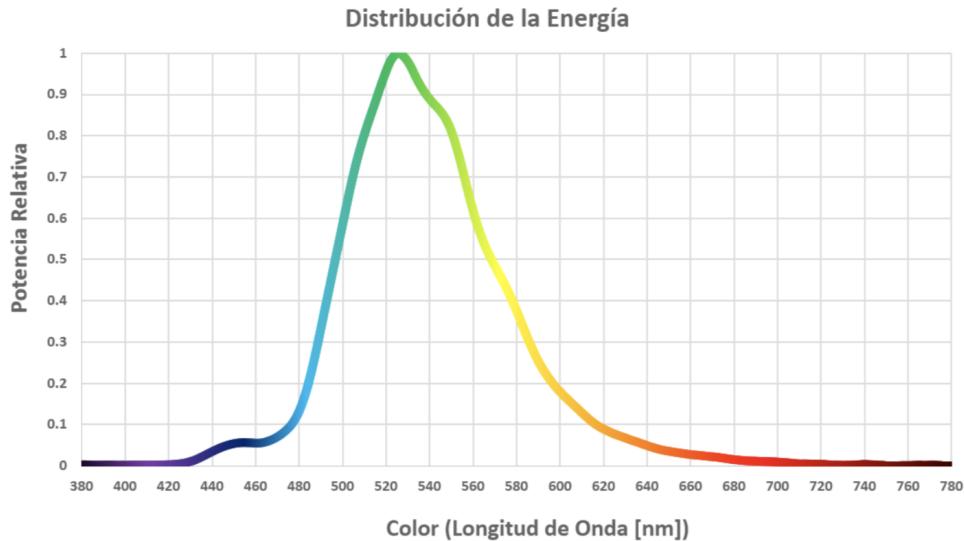


Figura 9: Ponderación espectral fotópica $V(\lambda)$ utilizada para el cálculo de iluminancia (lux) en función de la sensibilidad del ojo humano. EVOLUX. [9]

11.1.5. Sensor de temperatura y humedad del aire

La temperatura y la humedad relativa del aire se miden con un único sensor para reducir complejidad y consumo. Se ha seleccionado el chip **SHT31**, que ofrece un equilibrio adecuado entre coste, estabilidad y prestaciones dentro de su familia.

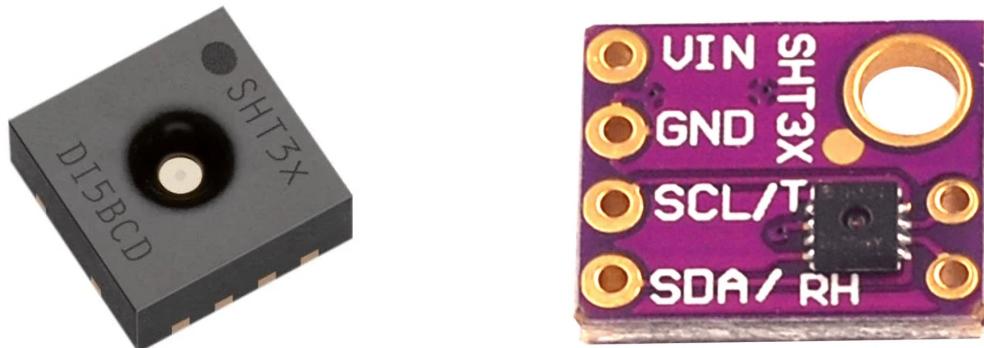


Figura 10: Chip SHT31 en formato QFN y su respectivo módulo de evaluación. Sensirion.

El SHT31, adicionalmente, incorpora un calentador interno (*heater*) que puede activarse puntualmente en condiciones de humedad elevada o presencia de rocío. Esta función ayuda a mitigar problemas de condensación sobre el sensor y mejora la recuperación en situaciones ambientales extremas.

11.1.6. Sensor de temperatura del terreno

Para el suelo se emplean sensores separados de temperatura y humedad, ya que los sensores combinados asequibles suelen tener limitaciones en precisión, estabilidad o encapsulado. La temperatura del terreno se mide con un **DS18B20** en su versión impermeable: el módulo **DFR0198**. Este encapsulado permite el uso directo sobre el terreno sin degradación por humedad y simplifica el despliegue al venir protegido mecánicamente.



Figura 11: Módulo impermeable DFR0198 y sensor DS18B20 en formato TO-92. DFRobot.

11.1.7. Sensor de humedad del terreno

Para medir la humedad del terreno se ha elegido el **SEN0308**, un sensor de tipo capacitivo. Se ha preferido este tipo frente a los sensores resistivos más comunes, porque en la práctica estos últimos suelen dar más problemas cuando se dejan enterrados durante mucho tiempo.



Figura 12: Sensor capacitivo SEN0308. DFRobot.

Un sensor resistivo mide la humedad haciendo pasar una señal eléctrica entre dos partes metálicas en contacto con la tierra. Con el uso, ese contacto eléctrico con el suelo puede

provocar deterioro y corrosión de los electrodos. Esto hace que, tras semanas o meses, el sensor no solo sea menos fiable, sino que también sea difícil comparar medidas entre distintos sensores o campañas.

El SEN0308, en cambio, al ser capacitivo no depende de que el suelo conduzca corriente de la misma manera, sino que detecta cambios eléctricos asociados al contenido de agua y, por eso, suele ser más estable para despliegues prolongados y con poco mantenimiento. Aun así, su lectura no depende únicamente de la humedad: factores como la salinidad y la composición del suelo también influyen en la medida. Por ello conviene realizar una calibración básica en el entorno real para interpretar la señal de manera consistente.

11.1.8. Microcontrolador

Por último, como MCU se ha seleccionado el **STM32WB55RG** de STMicroelectronics [7]. Este microcontrolador pertenece a la familia *STM32*, muy extendida en sistemas embedidos por su buen equilibrio entre prestaciones, consumo y ecosistema de desarrollo. Entre las ventajas generales de la gama *STM32* destacan la amplia documentación y notas de aplicación, la disponibilidad de placas de evaluación, como la **NUCLEO-WB55RG** [8], y el soporte de herramientas profesionales como STM32CubeMX y STM32CubeIDE, que simplifican la configuración de periféricos y aceleran el desarrollo del *firmware*.

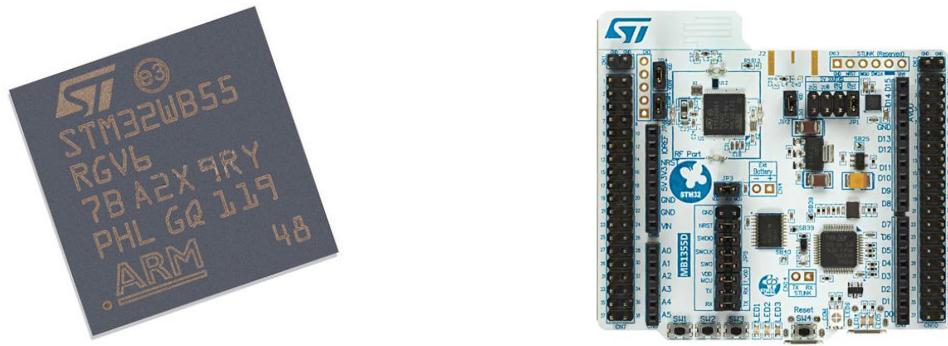


Figura 13: Chip STM32WB55RG en formato QFN y placa de desarrollo NUCLEO-WB55RG. STMicroelectronics.

La elección de este modelo concreto responde además a varios requisitos del prototipo:

- **Muy bajo consumo:** el STM32WB ofrece modos de bajo consumo adecuados para el funcionamiento, permitiendo reducir significativamente el gasto energético entre muestras.
- **BLE integrado:** incorpora radio y conectividad *Bluetooth Low Energy* sin necesidad de módulos externos, reduciendo así complejidad y número de componentes.
- **Periféricos y pines suficientes:** esta variante proporciona un encapsulado de 68 pines y dispone de interfaces habituales (I2C, SPI, UART, GPIO y ADC), lo que permite integrar todos los componentes sin recurrir a soluciones adicionales.

- **Capacidad de proceso:** ofrece potencia de cálculo y memoria suficientes para ejecutar la lógica de adquisición, filtrado y empaquetado de muestras, gestión de memoria y control de estados energéticos del sistema.

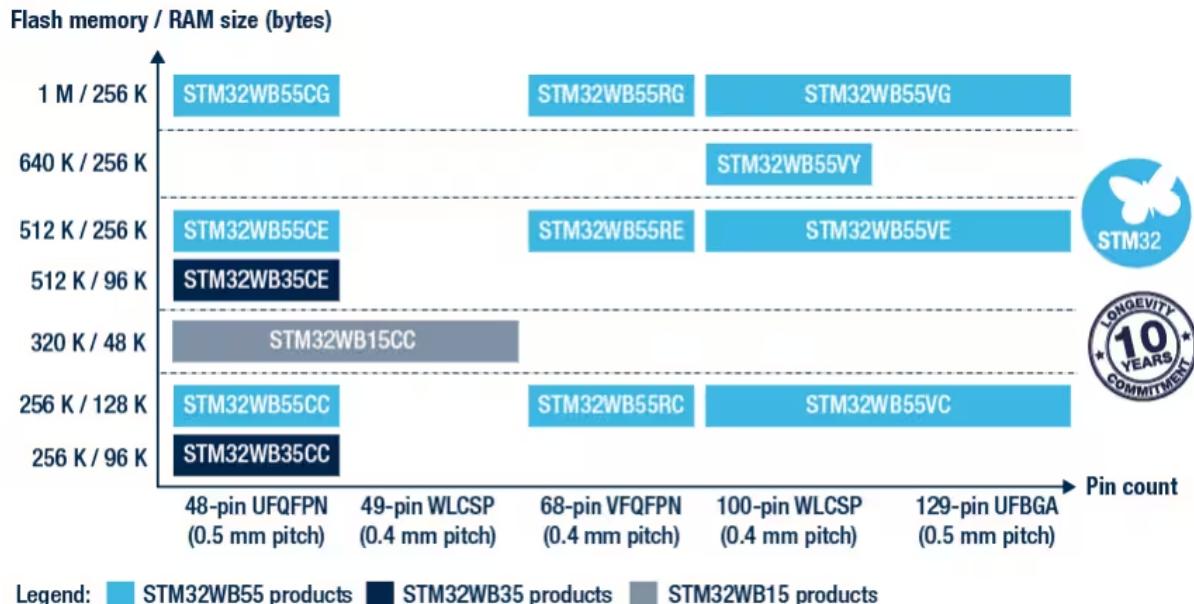


Figura 14: Comparación entre diferentes modelos del STM32WB. STMicroelectronics. [7]

Una característica diferencial del STM32WB55 es su arquitectura *dual-core*. Integra un núcleo ARM Cortex-M4, orientado a ejecutar la aplicación principal, y un núcleo ARM Cortex-M0+ que actúa como coprocesador de comunicaciones.

En particular, el *stack* de *Bluetooth Low Energy* y la gestión de la radio se ejecutan en el Cortex-M0+. Esto permite separar claramente dos dominios: por un lado la aplicación (M4) y por otro las tareas de comunicación inalámbrica (M0+), que requieren temporizaciones estrictas y gestión específica del transceptor. Esta separación facilita que la comunicación BLE no interfiera con la lógica del sistema y, al mismo tiempo, contribuye a un funcionamiento más robusto y eficiente energéticamente, al delegar la pila inalámbrica en el núcleo dedicado.

11.2. Arquitectura del hardware

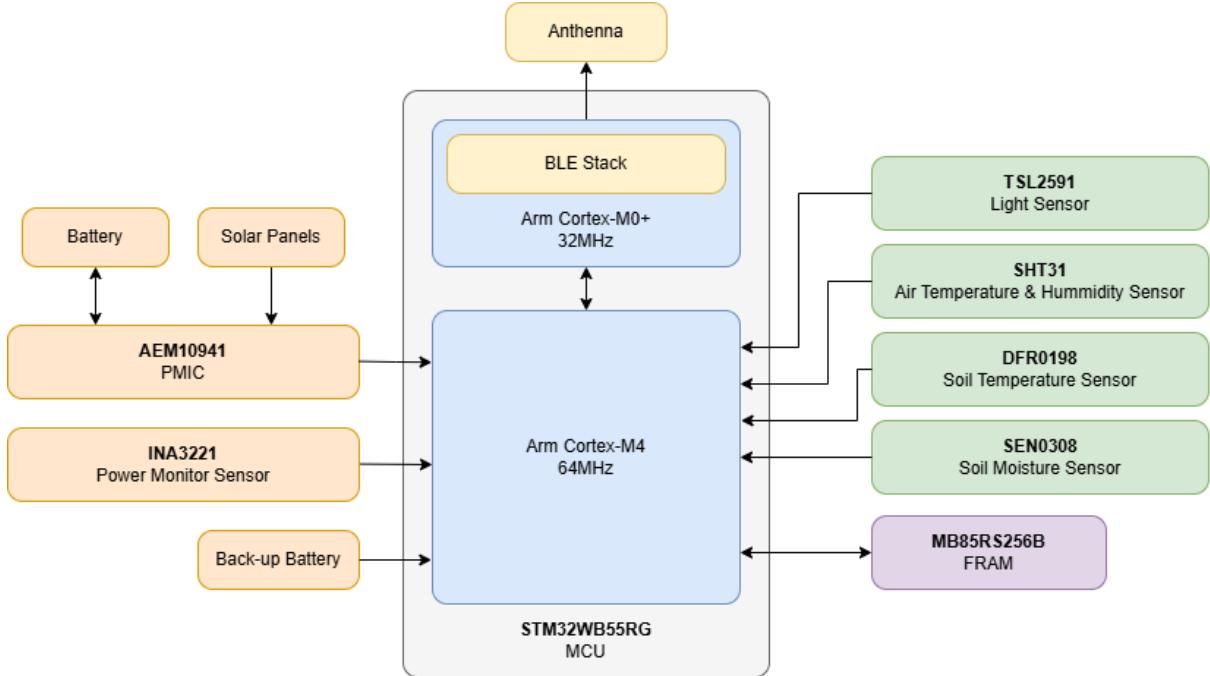


Figura 15: *Diagrama de la arquitectura hardware del sistema. Elaboración propia.*

En la Figura 15 observamos cómo se estructuran los principales componentes y los diferentes bloques funcionales mencionados anteriormente: sensores meteorológicos (verde), control y gestión de potencia (naranja), almacenamiento local (morado) y conexión inalámbrica (amarillo). En el centro se sitúa la MCU (gris), que actúa como nodo principal del sistema.

Dentro del microcontrolador se distinguen sus dos núcleos (azul), el Cortex-M4 y el Cortex-M0+. El primero ejecuta la aplicación principal y se comunica con el resto de periféricos, mientras que el segundo se encarga de la gestión de la pila de *Bluetooth Low Energy* (BLE stack). Ambos núcleos están comunicados internamente, permitiendo que la conectividad inalámbrica no interfiera con la lógica principal del sistema.

11.2.1. Interfaces y protocolos

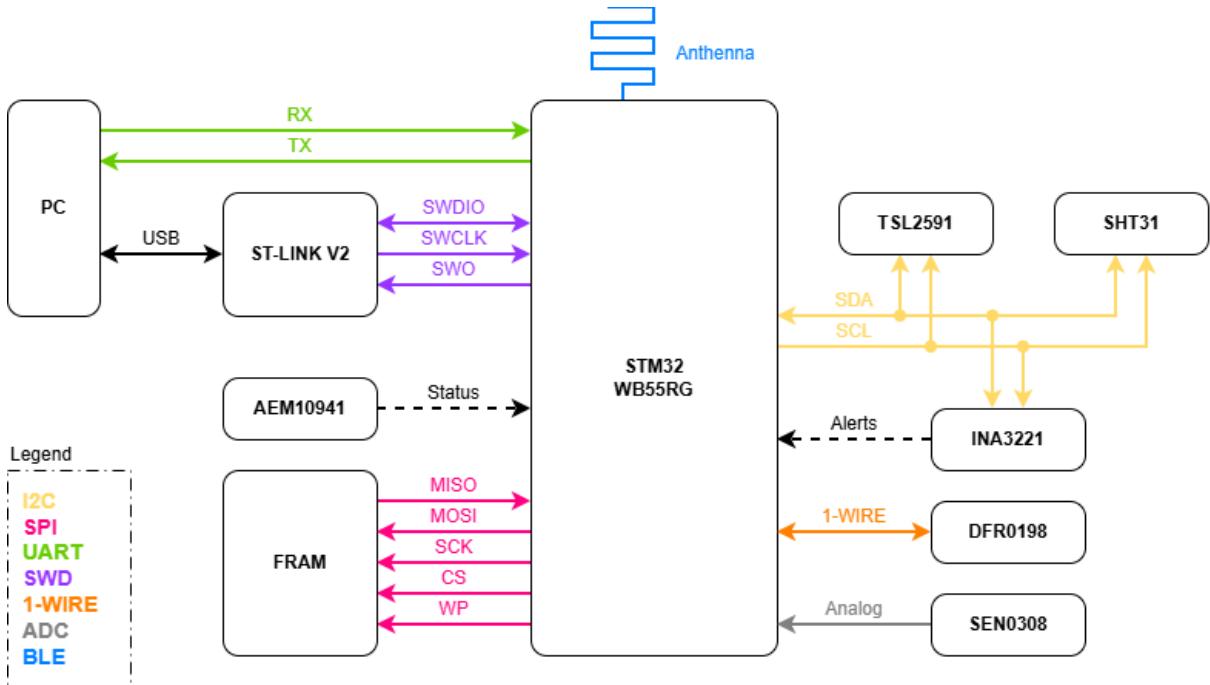


Figura 16: Diagrama de las interfaces y protocolos de comunicación del sistema. Elaboración propia.

La Figura 16 detalla las interfaces y protocolos utilizados para la interconexión entre los diferentes componentes del sistema: I2C, SPI, UART, SWD, 1-Wire, ADC y BLE.

El bus **I2C** se emplea como interfaz principal para sensores y monitorización, ya que permite conectar múltiples periféricos compartiendo las mismas líneas (SDA y SCL) y reduce el número de pines necesarios en la MCU. Por este motivo se ha priorizado su uso siempre que el dispositivo lo soportaba.

La memoria FRAM se comunica mediante **SPI**. Aunque existen alternativas habituales como I2C o buses paralelos, se ha optado por SPI por su equilibrio entre una mayor velocidad que I2C y una integración más sencilla que un bus paralelo (que requeriría más pines y complica el ruteado). En este enlace se utilizan MOSI, MISO y SCK, junto con CS para la selección del periférico y una línea adicional (WP) para poder escribir en registros protegidos.

El sensor DFR0198 (o DS18B20) utiliza el protocolo **1-Wire**, menos común en este tipo de sistemas pero ventajoso al requerir una única línea de datos para comunicación. Por su parte, el sensor SEN0308 proporciona una salida analógica, que se adquiere mediante el **ADC** integrado en el STM32.

Para tareas de desarrollo y validación se incluye una interfaz **UART** hacia el PC, utilizada como consola serie por su sencillez y amplia compatibilidad. La programación del microcontrolador se realiza mediante **SWD** (*Serial Wire Debug*), utilizando un programador externo

ST-LINK V2. Finalmente, la comunicación inalámbrica del sistema se implementa mediante **BLE**.

Además de los buses principales, aparecen señales digitales auxiliares como *Alert* y *Status*, conectadas a entradas de interrupción (**EXTI**) de la MCU. Estas líneas permiten notificar eventos relevantes sin necesidad de sondeo continuo, reduciendo el consumo y simplificando la lógica de control.

11.3. Prueba de concepto

Antes de abordar el diseño de la PCB final se realizó una fase de verificación incremental del sistema mediante una prueba de concepto. El objetivo de esta etapa fue reducir riesgos de integración y validar el funcionamiento de los componentes principales en condiciones controladas, asegurando que tanto la alimentación como la adquisición de datos y la comunicación inalámbrica eran viables con los componentes seleccionados.

La primera aproximación consistió en adquirir placas de evaluación de los elementos *core* del sistema: la NUCLEO-WB55RG como plataforma de desarrollo para el microcontrolador, y módulos para el AEM10941, INA3221 y TSL2591. Con estas placas, junto con baterías, paneles fotovoltaicos, cableado tipo *jumper* y *protoboards*, se montó un circuito de pruebas que replicaba el funcionamiento básico del sistema final: gestión de energía mediante el PMIC, mediciones de consumo con el monitor de potencia y lectura de sensores desde la MCU. Esta configuración permitió comprobar de forma temprana aspectos críticos como la compatibilidad de niveles lógicos, el funcionamiento de los buses de comunicación y el comportamiento general del sistema bajo alimentación real.

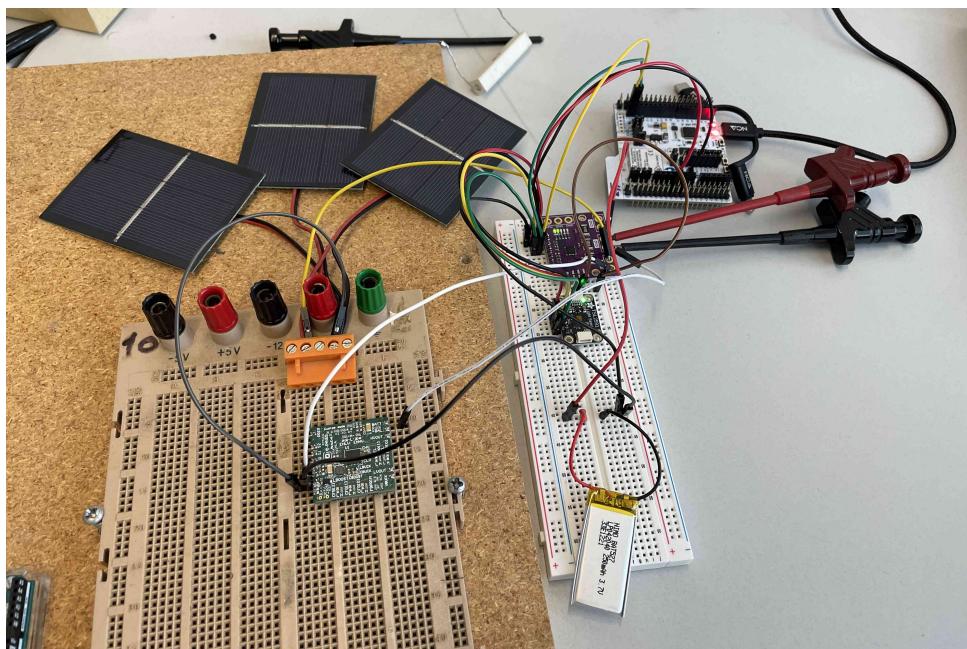


Figura 17: Primera prueba e concepto con placas de evaluación del core del sistema.
Elaboración propia.

Una vez verificado el correcto funcionamiento del núcleo del sistema, el siguiente paso fue desarrollar un primer prototipo de PCB fabricado mediante fresado de cobre. Este método de prototipado consiste en coger una placa aislante recubierta de cobre y emplear una fresadora CNC para eliminar mecánicamente el cobre de la superficie, dejando definidas las pistas, *pads* y contornos del circuito. A diferencia de una PCB industrial (fabricada por procesos fotoquímicos), el fresado permite iteraciones rápidas en laboratorio, pero impone restricciones geométricas y de fabricación que condicionan el diseño.

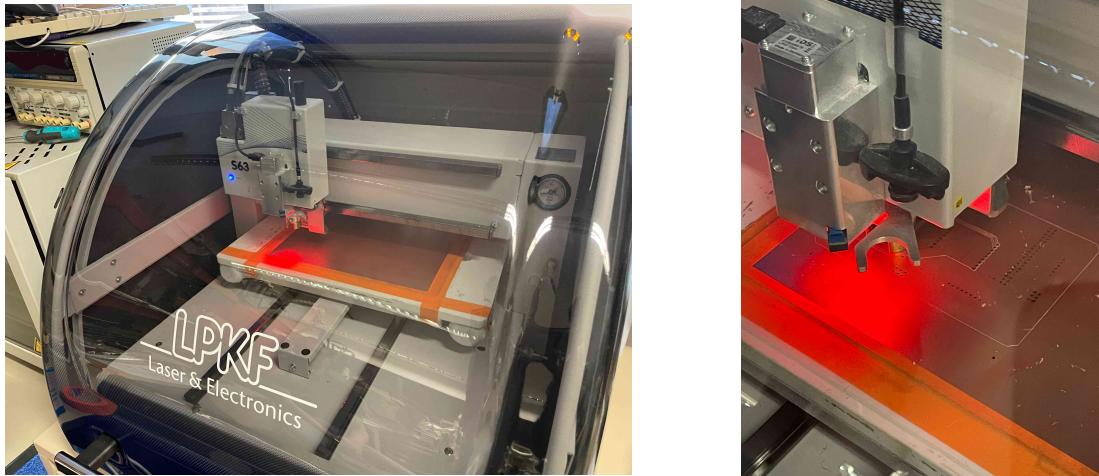


Figura 18: Imágenes de la fresadora de cobre utilizada. Elaboración propia.

En este proyecto en concreto, la PCB fresada se limitó a una de dos capas (*top* y *bottom*) y de dimensiones más conservadoras que las típicas en una fabricación profesional, se trabajó con anchos de pista y separaciones mínimas del orden de 0.5 mm, y con taladros mínimos alrededor de 0.3 mm. Durante esta fase aparecieron algunos errores de diseño como problemas con la unión de planos GND.

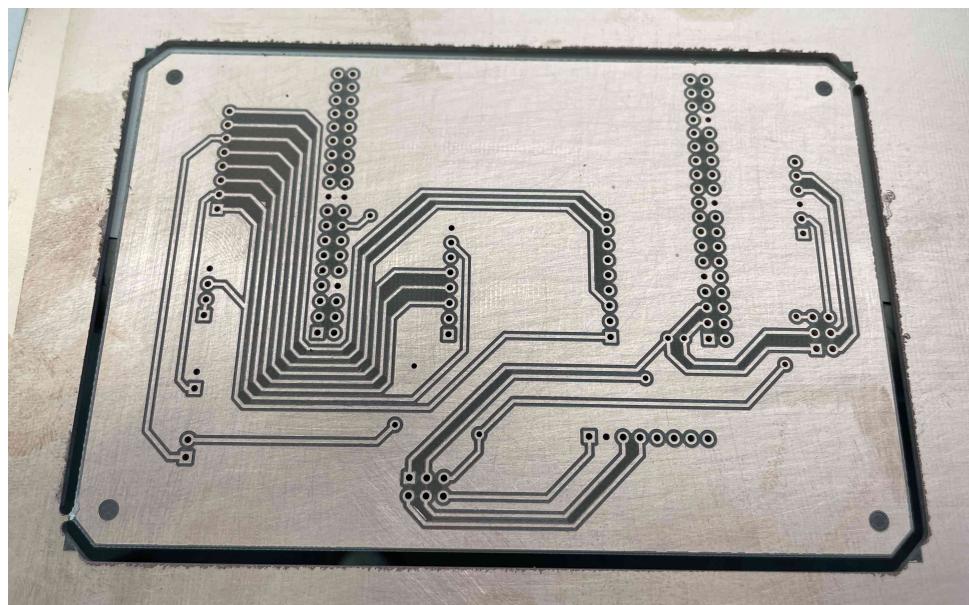


Figura 19: PCB de cobre fresado del prototipo. Elaboración propia.

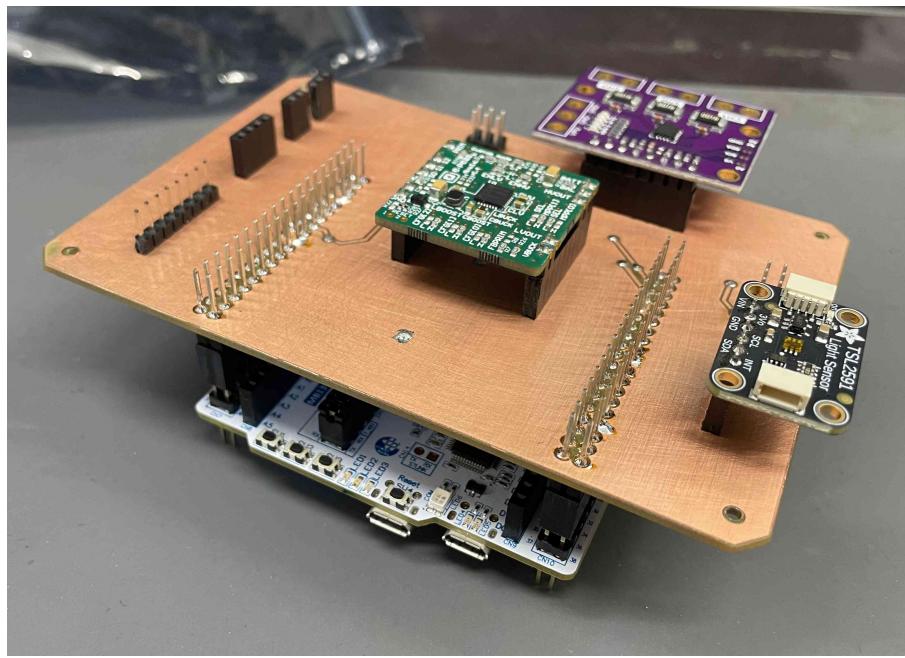


Figura 20: *PCB de cobre fresado después de soldar conectores y conectar los módulos de evaluación. Elaboración propia.*

Tras estabilizar el funcionamiento del prototipo y confirmar el comportamiento del core, se ampliaron las pruebas incorporando el resto de sensores del sistema. Para ello se adquirieron los módulos de evaluación del SHT31, DFR0198 y SEN0308, se conectaron al sistema y se verificó la correcta lectura desde la MCU, así como su integración con la lógica de adquisición.

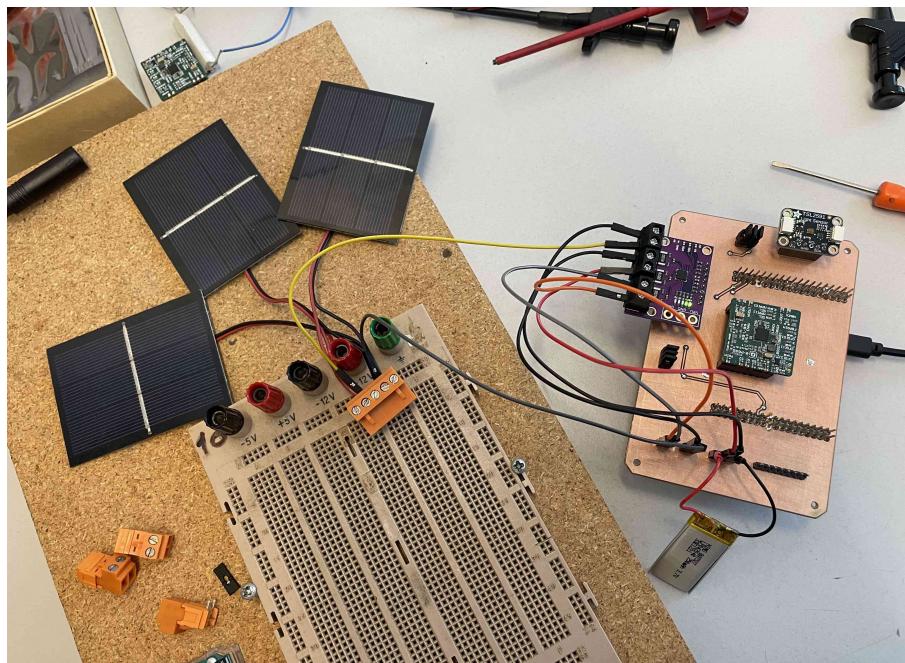


Figura 21: *Prueba de concepto final del core conectada. Elaboración propia.*

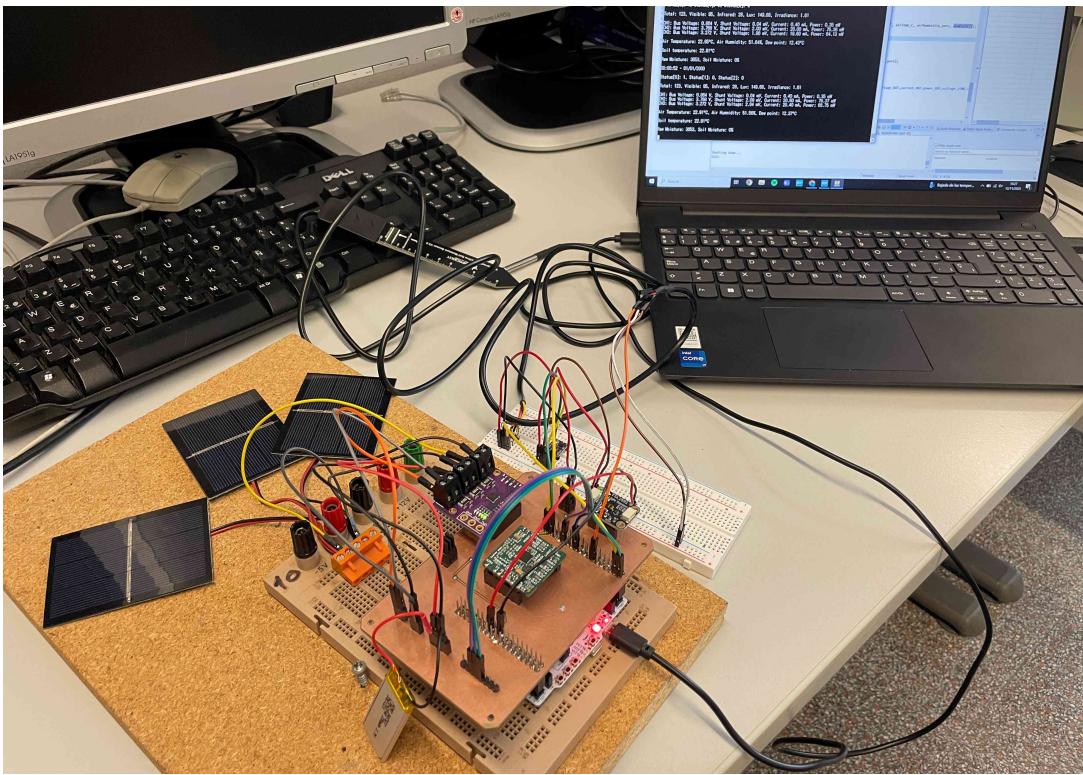


Figura 22: *Prueba de concepto completa conectada y imprimiendo logs por consola.*
Elaboración propia.

Con el conjunto de pruebas satisfactorio, se dio por validada la arquitectura y se procedió al diseño e integración de la PCB final, ya con un nivel de confianza mayor y con criterios de diseño refinados a partir de los aprendizajes obtenidos en la plataforma de testeо.

11.4. Esquemáticos

11.4.1. Potencia y alimentación

AEM10941

La Figura 23 muestra el diagrama de bloques de referencia que nos proporciona el datasheet del fabricante, a partir del cual se ha desarrollado el esquemático del circuito del AEM10941. En ella vemos los pines principales: SRC, BATT y HVOUT, que indican la entrada de los paneles solares, la batería y la salida de 3.3 V regulada, respectivamente.

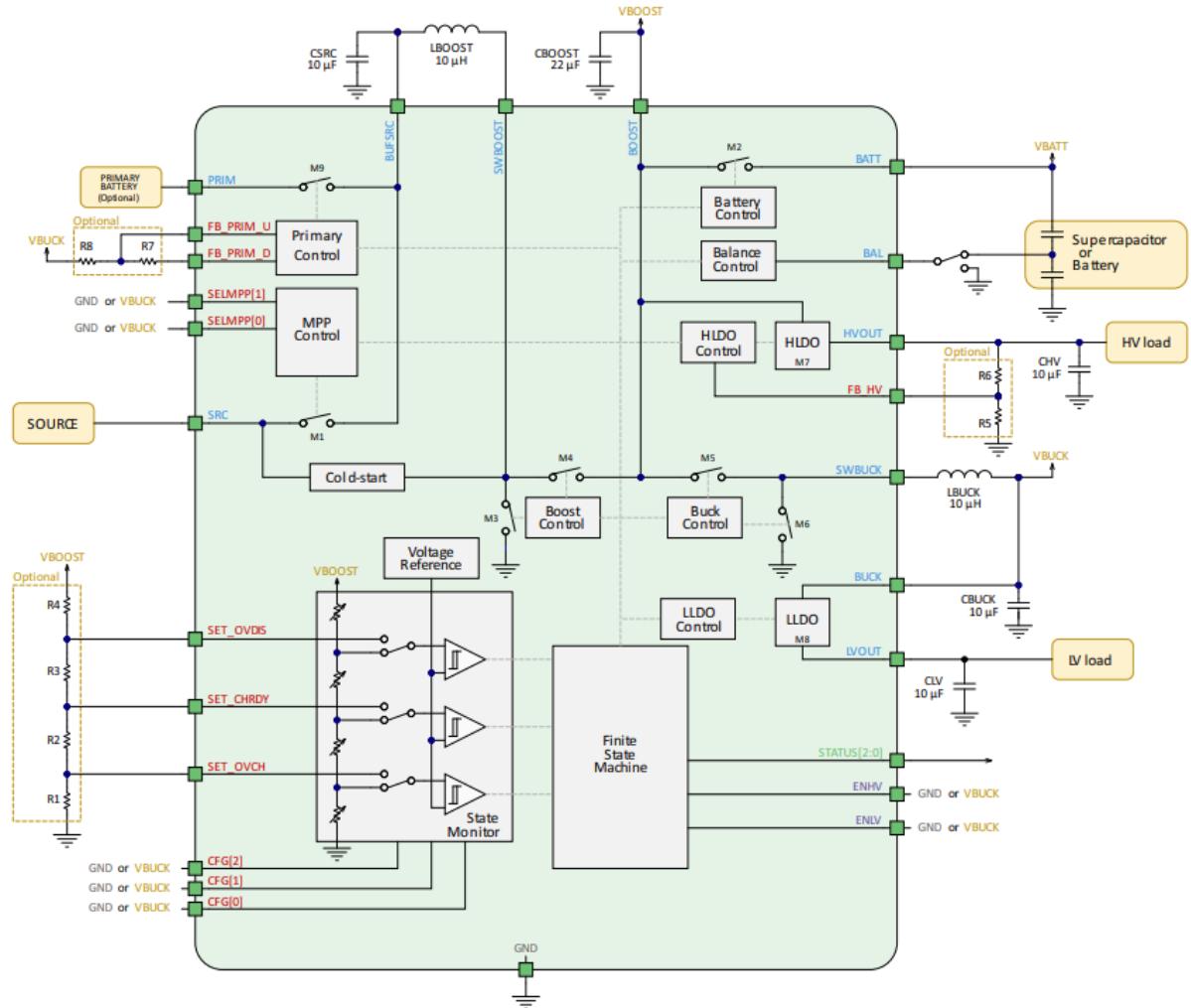


Figura 23: Diagrama de bloques funcionales de referencia del AEM10941. e-peas.

En nuestro sistema solo se utiliza la salida HVOUT y no LVOUT por lo que los pinos de *enable* de los LDOs se fijan a ENHV = HIGH y ENLV = LOW (el voltaje lógico de HIGH es VBUCK). Además la salida LVOUT se deja flotando.

La configuración de la batería y los voltajes de salida de los LDOs se establecen mediante los pinos de CFG[2:0], fijados a en binario, lo que selecciona el perfil de carga correspondiente a una batería Li-ion. En la Figura 24 se observa la tabla de posibles configuraciones. Como no se emplea un modo de configuración personalizado, los pinos SET_OVDIS, SET_CHRDY, SET_OVCH y FB_HV no se utilizan y se dejan flotando.

Configurations pins			Storage element threshold voltages			LDOs output voltages		Typical use
CFG[2]	CFG[1]	CFG[0]	V_{OVCH}	V_{CHRDY}	V_{OVDIS}	V_{HV}	V_{LV}	
H	H	H	4.12 V	3.67 V	3.60 V	3.3 V	1.8 V	Li-ion battery
H	H	L	4.12 V	4.04 V	3.60 V	3.3 V	1.8 V	Solid state battery
H	L	H	4.12 V	3.67 V	3.01 V	2.5 V	1.8 V	Li-ion/NiMH battery
H	L	L	2.70 V	2.30 V	2.20 V	1.8 V	1.2 V	Single-cell supercapacitor
L	H	H	4.50 V	3.67 V	2.80 V	2.5 V	1.8 V	Dual-cell supercapacitor
L	H	L	4.50 V	3.92 V	3.60 V	3.3 V	1.8 V	Dual-cell supercapacitor
L	L	H	3.63 V	3.10 V	2.80 V	2.5 V	1.8 V	LiFePO4 battery
L	L	L	Custom mode - Programmable through R1 to R6.			1.8 V		

Figura 24: Tabla de modos de batería y voltages de salida de los LDOs del AEM10941. e-peas.

Respecto al MPPT, se selecciona SELMPP [1:0] = 00, que fija el seguimiento al 70 % de la tensión en circuito abierto del panel. Esta elección ofrece un compromiso adecuado entre simplicidad y rendimiento, ya que aproxima el punto de máxima potencia de muchos paneles pequeños en condiciones variables sin necesidad de calibración adicional. En la Figura 25 se muestra la tabla de posibles configuraciones del MPPT.

SELMPP[1]	SELMPP[0]	V_{MPP} / V_{OC}
L	L	70%
L	H	75%
H	L	85%
H	H	90%

Figura 25: Tabla de posibles configuraciones del MPPT del AEM10941. e-peas.

Otro punto a tener en cuenta es el pin BATT, que no debe quedar nunca flotante. Dado que se contempla el uso de una batería recambiable, existe la posibilidad de desconexión temporal; por eso se añade un capacitor de 150 μ F en el nodo de batería, tal y como recomienda el fabricante. Asimismo, como no se utiliza batería secundaria en este diseño, los pines PRIM, FB_PRIM_U y FB_PRIM_D se conectan a GND.

Los demás condensadores e inductores son los que recomienda el fabricante por defecto. Las señales STATUS [2:0] se conectan directamente al microcontrolador.

En la Figura 26 se expone el esquemático final del subsistema de control de potencia.

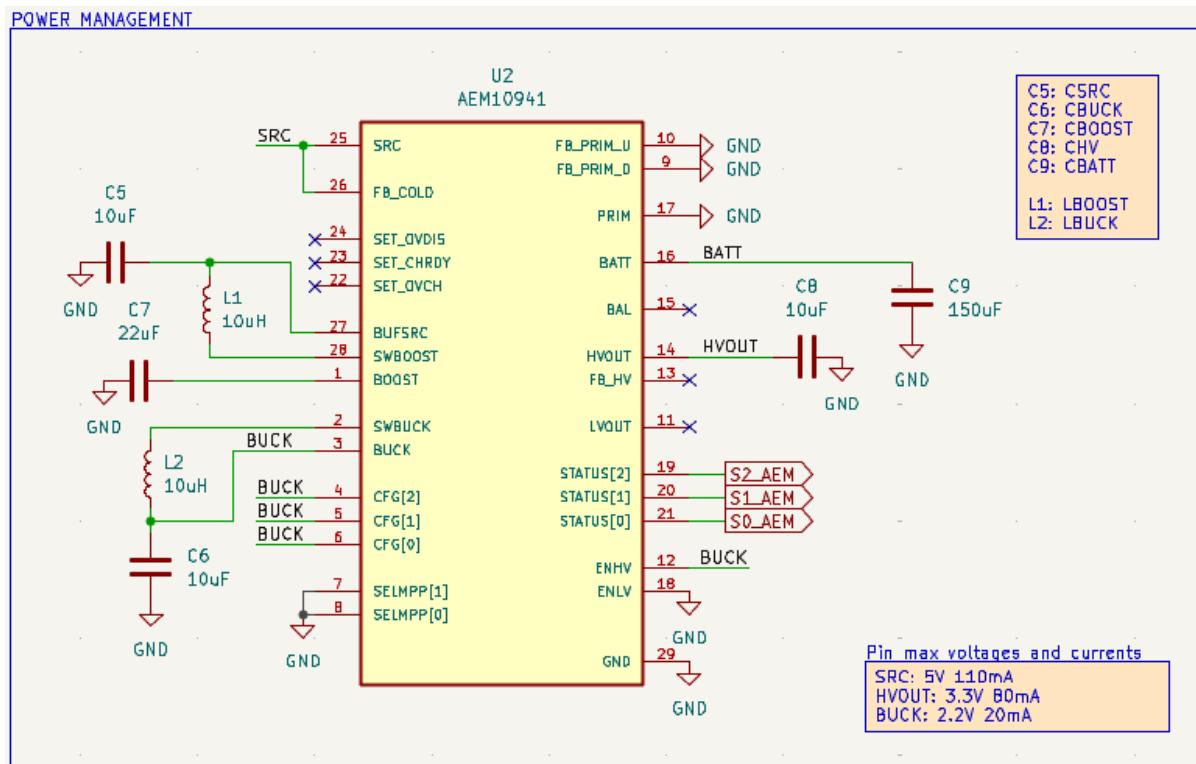


Figura 26: Esquemático final del subsistema de gestión de potencia. Elaboración propia.

INA3221

La Figura 27 muestra el diagrama de conexiones recomendado por el fabricante del INA3221.

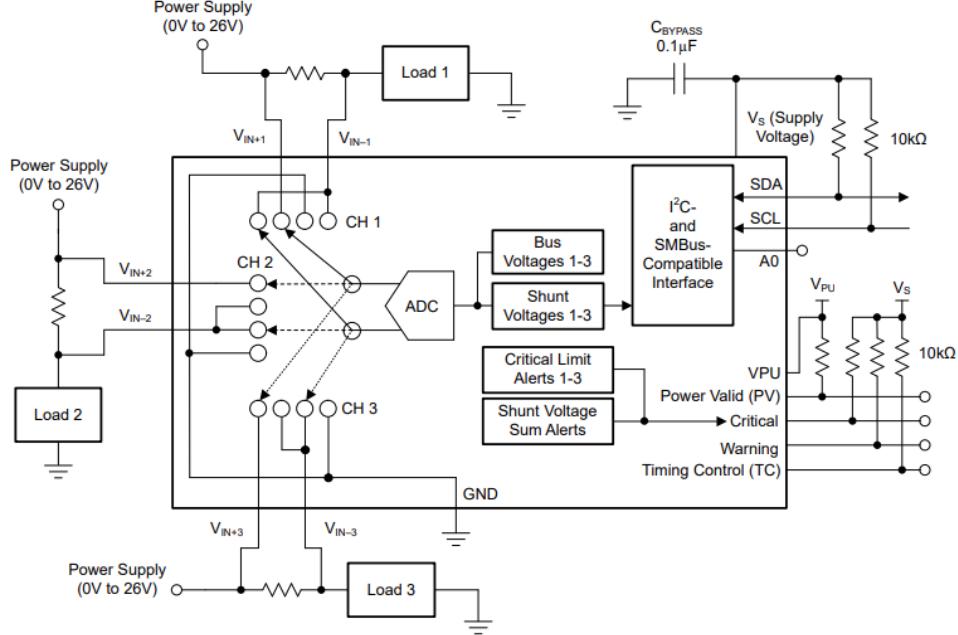


Figura 27: Diagrama de bloques funcional de referencia del INA3221. Texas Instruments.

El INA3221 mide, para cada canal, dos magnitudes: la tensión de bus (diferencia de voltaje entre IN+ y GND), y la tensión de la *shunt* (diferencia de voltaje entre IN+ e IN-). A partir de esta caída diferencial y del valor de la resistencia *shunt* se puede calcular la corriente del canal.

La selección del valor de la resistencia de *shunt* se realiza a partir de la caída máxima deseada y la corriente máxima esperada. Consideramos una caída máxima en la *shunt* de $V_{\text{shunt_max}} = 33 \text{ mV}$ y una corriente máxima de $I_{\text{max}} = 150 \text{ mA}$:

$$R_{\text{shunt}} = \frac{V_{\text{shunt_max}}}{I_{\text{max}}} = \frac{0,033}{0,15} = 0,22 \Omega$$

$$P_{\text{shunt}} = I_{\text{max}}^2 \cdot R_{\text{shunt}} = 0,15^2 \cdot 0,22 = 0,00495 \text{ W}$$

Esta configuración ofrece una precisión aceptable a cambio de una caída de tensión muy pequeña. La disipación de calor también es mínima. En cuanto a la resolución, el INA3221 presenta una cuantización típica de $40 \mu\text{V/bit}$ en el registro de tensión de los canales. Por tanto, la resolución de corriente equivalente quedaría:

$$I_{\text{LSB}} = \frac{40 \mu\text{V/bit}}{R_{\text{shunt}}} = \frac{40 \cdot 10^{-6}}{0,22} = 0,182 \text{ mA/bit}$$

Para reducir ruido de alta frecuencia y mejorar la estabilidad de medida, el fabricante recomienda implementar un filtro de entrada en cada canal que consiste de dos resistencia en serie de 10Ω y un condensador de 100 nF entre IN^+ e IN^- . Esta red actúa como filtro pasa-bajo y ayuda a mitigar picos y transitorios. En la Figura 28 vemos el diagrama de conexión propuesto.

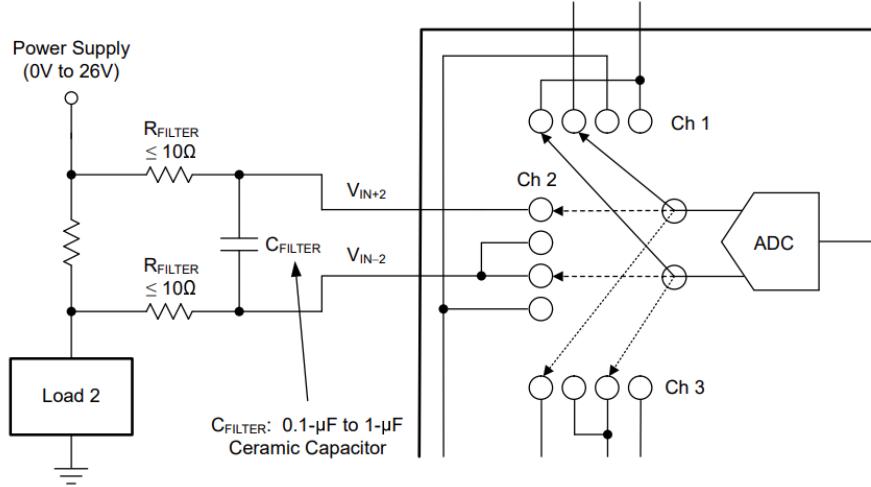


Figura 28: Red de filtrado recomendada para cada canal del INA3221.

El INA3221 incluye además pines de alerta configurables: PV, CRITICAL, WARNING y TC. Este último no lo usaremos, por lo que lo dejamos flotante. Los demás los conectamos a una resistencia de *pull-up* de $10 \text{ k}\Omega$ y al STM32. El INA3221 tiene además la posibilidad de escoger entre 4 direcciones I₂C mediante el pin A0. Como usaremos la dirección por defecto lo conectamos a GND.

En la Figura 29 se expone el esquemático final del subsistema de monitoreo de potencia.

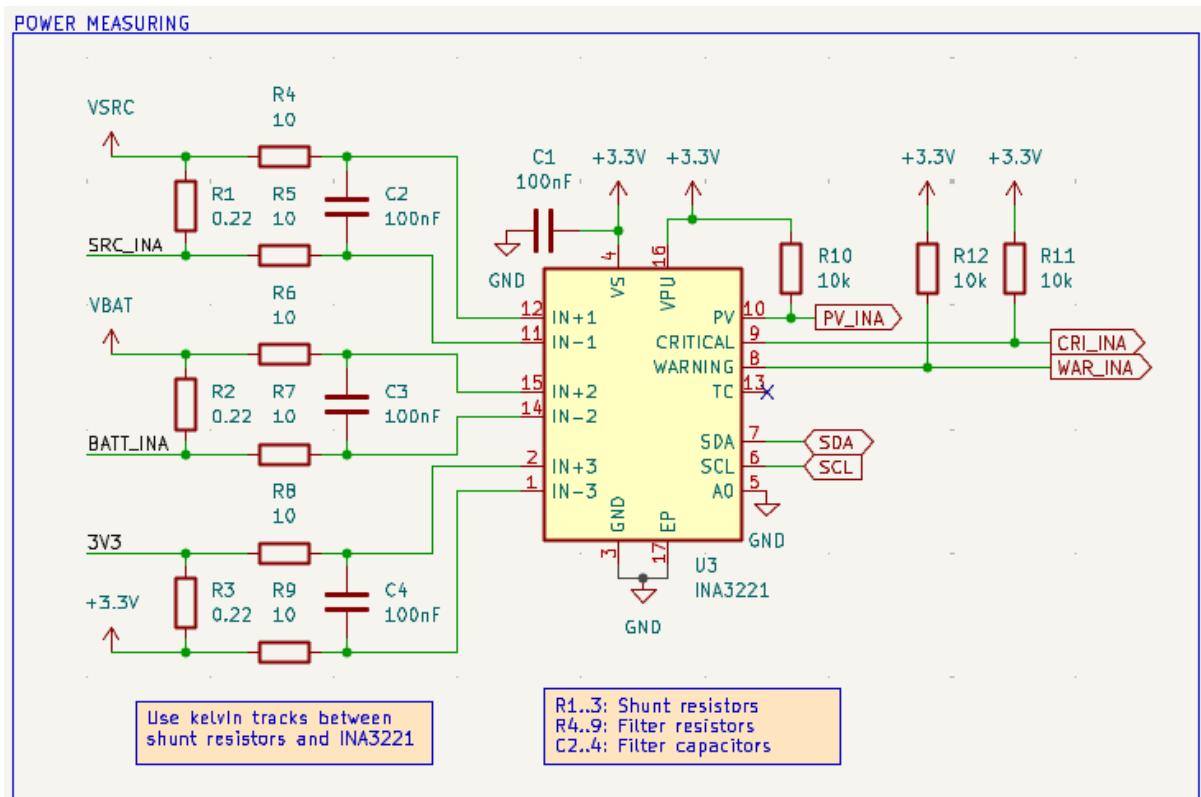


Figura 29: Esquemático final del subsistema de monitoreo de potencia. Elaboración propia.

Bus de 3.3 V

El bus de 3.3 V está diseñado para alimentarse por defecto desde el pin HVOUT del AEM10941, que actúa como fuente principal del sistema durante operación normal. Sin embargo, para facilitar la programación, depuración y pruebas en laboratorio, se incorpora la posibilidad de alimentar dicho bus desde un conector externo (3V3_EXT). Para seleccionar el origen de la alimentación se incluye un *jumper*, que permite commutar entre la fuente generada por el PMIC y la alimentación externa.

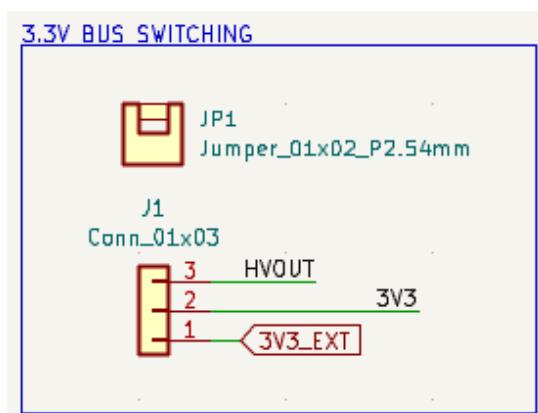


Figura 30: Esquemático de la selección de fuente del bus de 3.3 V. Elaboración propia.

Alimentación de sensores

Con el objetivo de minimizar consumo, la alimentación de los sensores se puede activar y desactivar mediante un interruptor de alta implementado con un MOSFET PMOS. El bus de 3.3 V se conecta al source (S) del PMOS, y la alimentación comutada hacia los sensores al drain (D). La gate (G) se controla desde un pin del STM32.

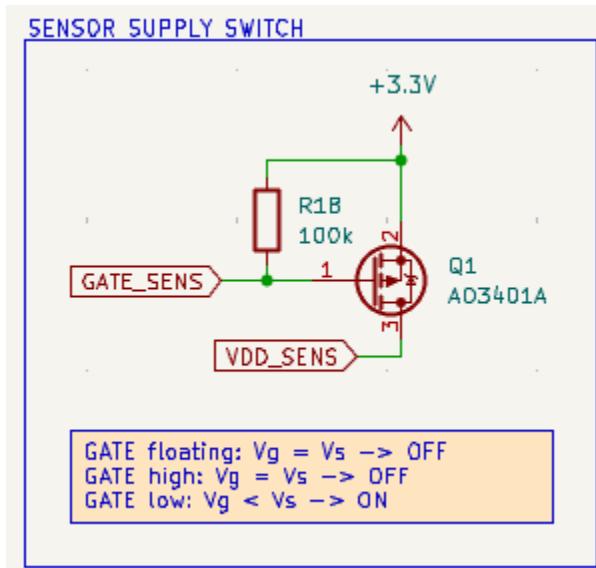


Figura 31: Esquemático del interruptor para alimentación de sensores. Elaboración propia.

Para asegurar un estado seguro por defecto, se añade una resistencia de *pull-up* de $100\text{ k}\Omega$ entre el source y la gate, de modo que, si el microcontrolador no está inicializado, la gate queda al mismo potencial que el source ($V_{GS} = 0$) y el transistor permanece cortado. El valor elevado de esta resistencia reduce la corriente consumida por la red de polarización y facilita al microcontrolador modificar el nivel de gate sin cargas innecesarias.

GATE flotando	$V_{GS} = V_G - V_S = 0$	Interruptor apagado
GATE a HIGH	$V_{GS} = V_G - V_S = 0$	Interruptor apagado
GATE a LOW	$V_{GS} = V_G - V_S < 0$	Interruptor encendido

Tabla 11: Estados lógicos del control de la gate del MOSFET y su efecto sobre la alimentación de sensores. Elaboración propia.

Se cumplen además estos puntos:

- Se utiliza un sólo interruptor para todos los sensores ambientales.
- El interruptor sólo afecta a los sensores, no al STM32 ni al INA3221.
- El bus de 3.3 V entra al source del MOSFET después de pasar por el INA3221.

11.4.2. MCU

STM32WB55RG

Para reducir el riesgo de errores en una parte crítica del diseño, el esquema se ha construido tomando como referencia la placa NUCLEO-WB55RG, replicando las recomendaciones del fabricante. Esta aproximación permite apoyarse en una topología ya validada y minimiza incertidumbres típicas del primer diseño de placa. Las Figuras 32 y 33 muestran el esquemático final de la MCU.

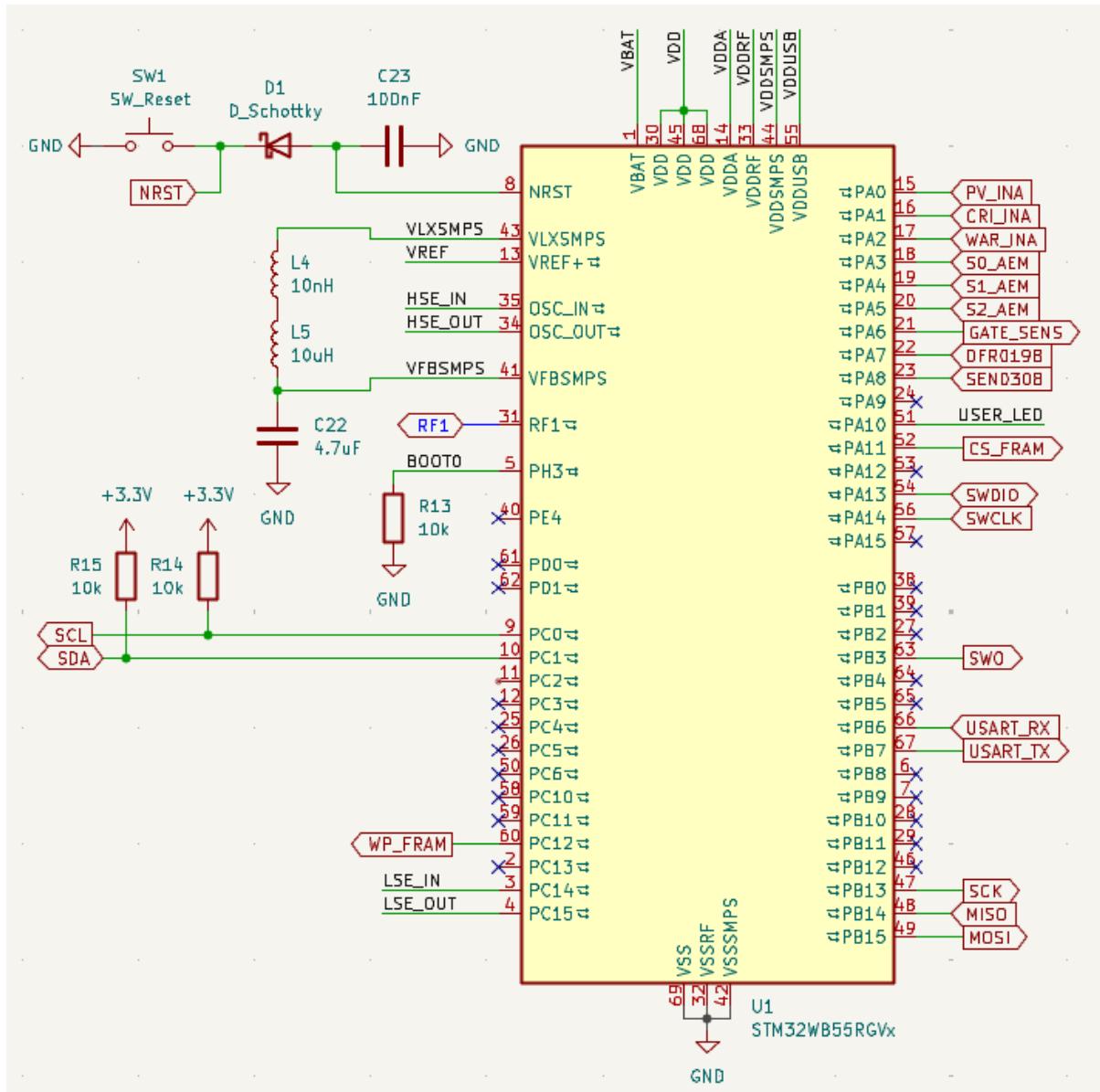


Figura 32: Esquemático del STM32WB55RG y conexiones principales de control y periféricos. Elaboración propia.

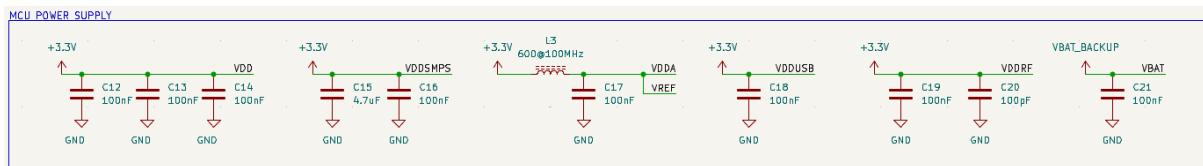


Figura 33: Esquemático de alimentación y desacoplo del STM32WB55RG. Elaboración propia.

Para cada pin de alimentación se ha colocado un condensador de 100 nF lo más próximo posible al encapsulado. Además del desacoplamiento “básico”, el STM32WB incluye dominios con requisitos adicionales, los cuales podemos observar en la Figura 34. En lugar de definir una topología propia, se ha seguido el esquema de referencia de la NUCLEO-WB55RG para estos pines.

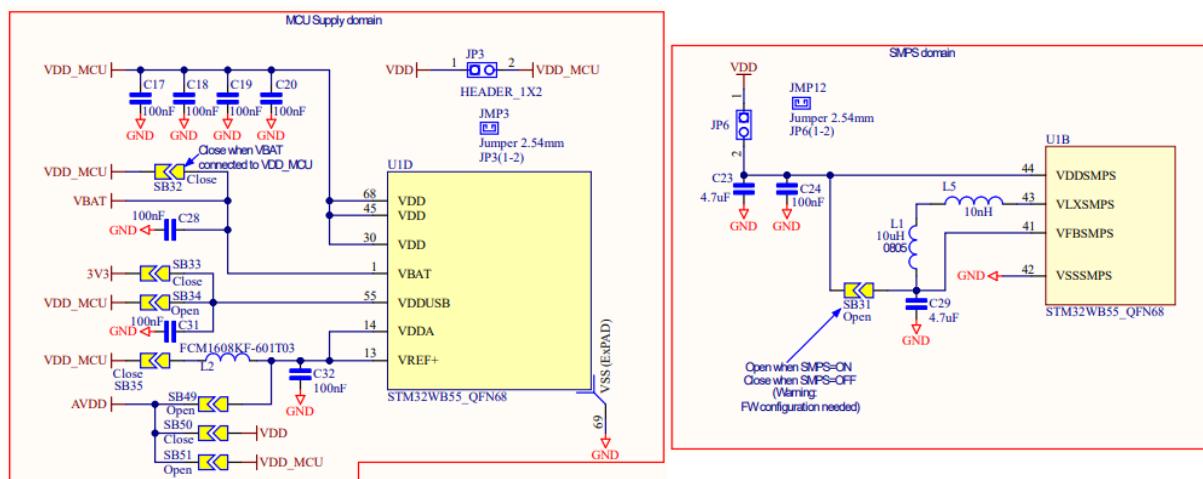


Figura 34: Recomendaciones de conexión de alimentación y SMPS del STM32WB55RG.

La Figura 35 muestra el circuito para las conexiones de los pines de reset (NRST) y de boot (BOOT0) recomendado y del cual se ha inspirado para conectar el propio.

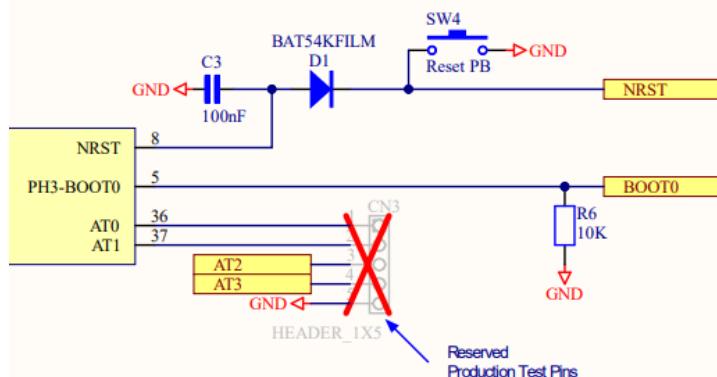


Figura 35: Circuito recomendado para los pines NRST y BOOT0 en el STM32WB55RG.

En la Tabla 12 se muestra el pinout utilizado del STM32WB55RG para este proyecto [15].

Pin STM32	Etiqueta	Función
VDD	VDD	Alimentación
VDDA	VDDA	Alimentación
VDDRF	VDDRF	Alimentación
VDDSMPS	VDDSMPS	Alimentación
VDDUSB	VDDUSB	Alimentación
VBAT	VBAT	Alimentación
VSS	GND	Tierra
VSSRF	GND	Tierra
VSSSMPS	GND	Tierra
NRST	NRST	Reset
VLXSMPS	VLXSMPS	Alimentación
VFBSMPS	VFBSMPS	Alimentación
VREF	VDDA	Referencia de voltaje
RF1	RF1	RF
OSC_IN	HSE_IN	OSC_IN
OSC_OUT	HSE_OUT	OSC_OUT
PC14	LSE_IN	OSC32_IN
PC15	LSE_OUT	OSC32_OUT
PH3	BOOT0	Boot
PA0	PV_INA	GPIO_EXTI0
PA1	CRI_INA	GPIO_EXTI1
PA2	WAR_INA	GPIO_EXTI2
PA3	S0_AEM	GPIO_EXTI3
PA4	S1_AEM	GPIO_EXTI4
PA5	S2_AEM	GPIO_Input
PA6	GATE_SENS	GPIO_Output
PA7	DFR0198	GPIO_Output
PA8	SEN0308	ADC1_IN15
PA10	USER_LED	GPIO_Output
PA11	CS_FRAM	GPIO_Output
PA13	SWDIO	SWD
PA14	SWCLK	SWD
PB3	SWO	SWD
PB6	USART_RX	USART1_TX
PB7	USART_TX	USART1_RX
PB13	SCK	SPI2_SCK
PB14	MISO	SPI2_MISO
PB15	MOSI	SPI2_MOSI
PC0	SCL	I2C3_SCL
PC1	SDA	I2C3_SDA
PC12	WP_FRAM	GPIO_Output

Tabla 12: Tabla con el pinout utilizado del STM32, su etiqueta y su función. Elaboración propia.

Osciladores

El STM32WB55RG permite incorporar dos cristales externos: un oscilador de 32 MHz (HSE) y un oscilador de 32.768 kHz (LSE). El HSE es relevante para el funcionamiento fiable del subsistema RF, mientras que el LSE se recomienda para el RTC por ofrecer mayor precisión que los osciladores internos. Por este motivo se han incorporado ambos en el diseño final.

La selección del cristal no depende únicamente de su frecuencia nominal, sino también de la capacitancia de carga requerida. La capacitancia efectiva del circuito se calcula mediante la siguiente fórmula [10]:

- C_L es la capacitancia de carga especificada por el cristal.
- C_S representa capacitancias parásitas (pistas, pads y entrada del micro).
- C_1 y C_2 son los condensadores externos en cada pin del cristal.

$$C_L = \frac{C_1 \cdot C_2}{C_1 + C_2} + C_S; \quad C_1 = C_2 = 2 \cdot (C_L - C_S)$$

Se fija $C_S = 3$ pF. En el caso del HSE, no se montan condensadores externos porque la carga se puede ajustar mediante configuración interna del microcontrolador [11]. En cambio, para el LSE sí se seleccionan condensadores externos. El cristal LSE escogido tiene $C_L = 9$ pF por lo que el cálculo conduce a:

$$C_1 = C_2 = 2 \cdot (9 - 3) = 12 \text{ pF}$$

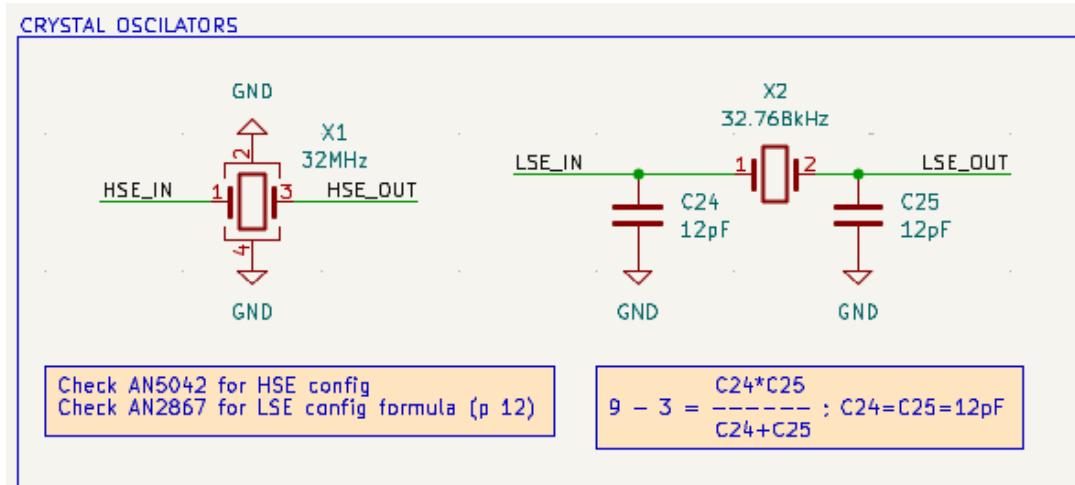


Figura 36: Esquemático de los osciladores HSE (32 MHz) y LSE (32.768 kHz) y sus componentes asociados. Elaboración propia.

RF

Es la parte más sensible del diseño, por lo que se ha seguido estrictamente la documentación de referencia y notas de aplicación de STMicroelectronics [12] [13].

Se utiliza el módulo MLPF-WB55-01E3 (recomendado para STM32WB), junto con una red adicional tipo “H” formada por dos condensadores de 1.2 pF y un inductor de 3.6 nH para adaptar la señal entre el pin RF1 del microcontrolador y la antena. Siguiendo la referencia de la NUCLEO-WB55RG, el primer condensador de la red se omite, manteniendo la topología validada por el fabricante.

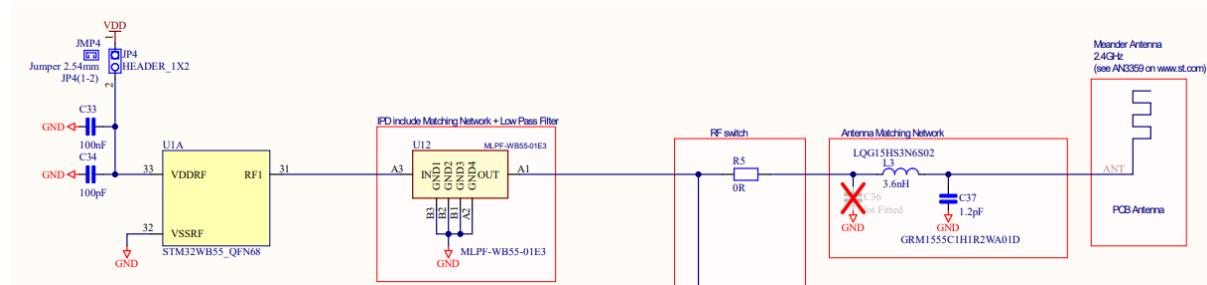


Figura 37: Topología de referencia del subsistema RF para los STM32WB.

Además, se implementan pistas RF con impedancia controlada. La *Application Note AN6335* [13] indica que el tramo entre el STM32 y el MLPF debe aproximarse a $62\ \Omega$ ($62R$), mientras que el tramo desde el filtro hacia la antena debe ser de $50\ \Omega$ ($50R$). Estas impedancias serán claves para el dimensionado de las pistas y la selección del stack-up de la PCB.

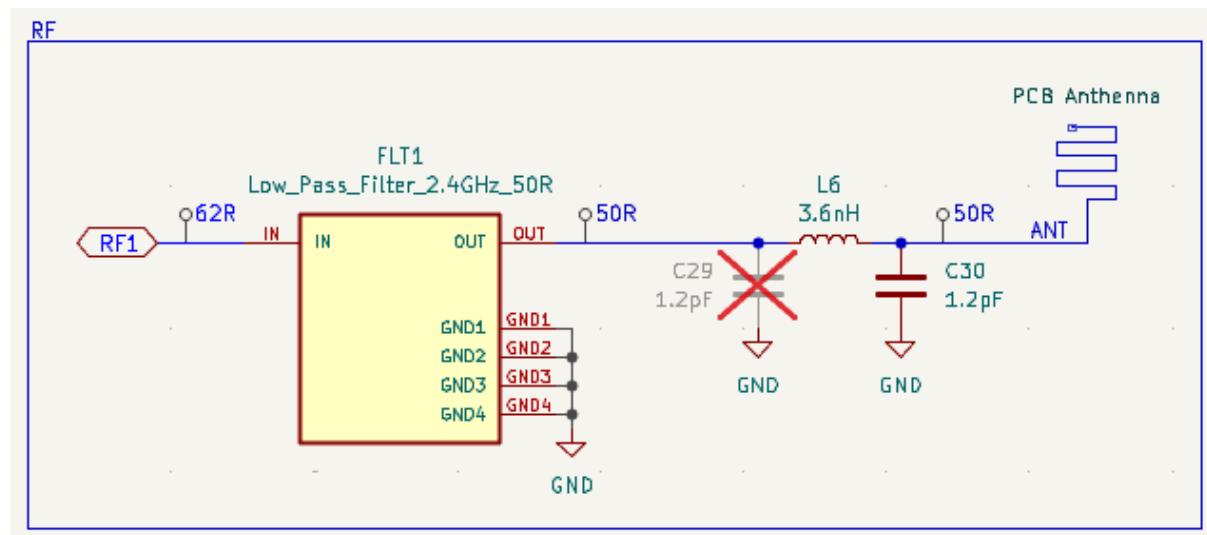


Figura 38: Implementación del filtrado y adaptación RF entre RF1, MLPF y antena.
Elaboración propia.

FRAM

La memoria FRAM se conecta mediante bus SPI común. Adicionalmente incorpora dos pines de control: HOLD, que permite pausar temporalmente la comunicación sin perder el estado, y WP, que habilita la protección de escritura. En este diseño no se utiliza la función HOLD, por lo que se conecta a 3.3 V a través de una resistencia de *pull-up*. El pin WP se controla desde la MCU. Finalmente, se incluye un condensador de 100 nF de desacoplo en la alimentación del integrado.

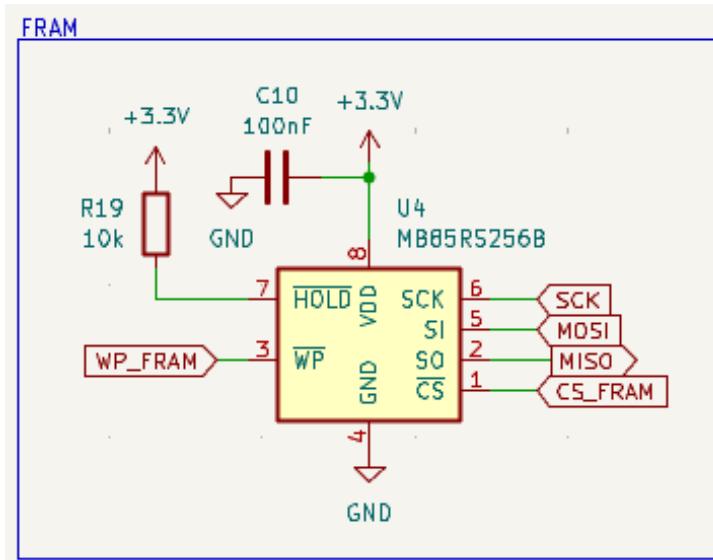


Figura 39: Conexión de la memoria FRAM. Elaboración propia.

LEDs

Se integran dos LEDs en el sistema: uno de POWER, que indica la presencia de alimentación, y otro de USER, controlado por un pin del STM32. El LED de usuario se conecta con el cátodo a GND, de modo que se enciende al fijar el pin a HIGH, y el valor de la resistencia se dimensiona para un consumo aproximado de 1 mA, suficiente para visibilidad sin penalizar el consumo energético.

El cálculo se realiza con la expresión:

$$I = \frac{V_{DD} - V_F}{R}$$

Donde $V_{DD} = 3,3$ V y V_F es la caída de tensión directa del LED, la cual depende del color del mismo.

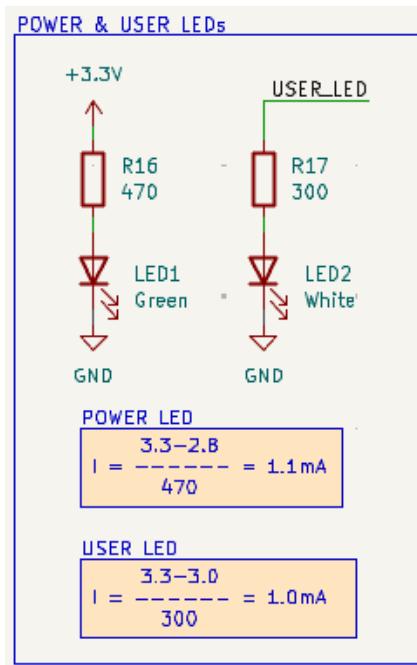


Figura 40: Circuito de los LEDs de POWER y USER. Elaboración propia.

11.4.3. Conexiones externas

SWD

Para la programación del STM32WB55RG se ha optado por la interfaz SWD (Serial Wire Debug), utilizando un programador ST-LINK/V2. El ST-LINK estándar emplea un conector de 20 pines (ver Figura 41), pero para este proyecto dicho conector resulta excesivamente grande para la PCB y, además, la mayoría de sus pines son redundantes o corresponden a GND.

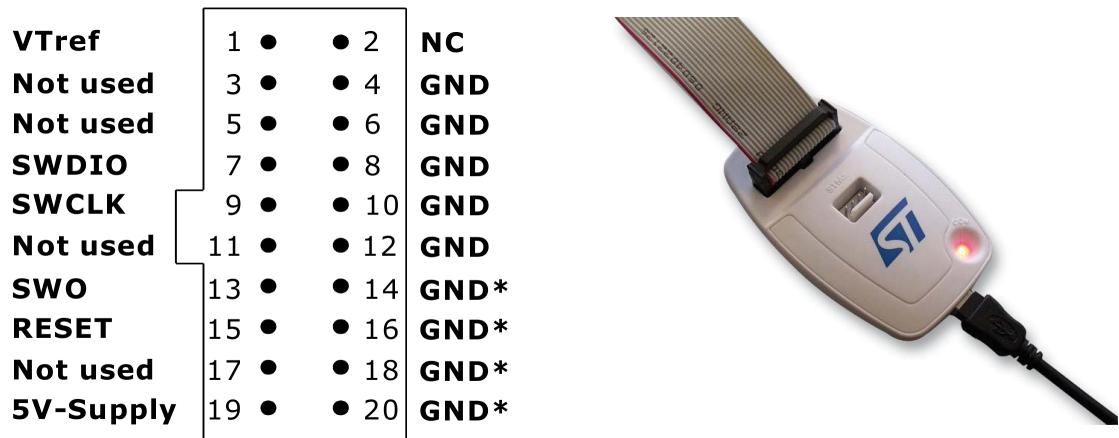


Figura 41: Pinout del conector SWD del ST-LINK y programador ST-LINK/V2 utilizado durante el desarrollo.

Por este motivo, se ha reducido la conexión a las señales mínimas necesarias para programa-

ción y depuración: Vref, SWDIO, SWCLK, GND, NRST y, de forma opcional, SWO. Adicionalmente se enruta una opción de alimentación hacia el jumper del bus de 3.3 V descrito en la sección “Bus de 3.3 V” (Sección 11.4.1), de forma que durante pruebas en laboratorio sea posible alimentar el sistema desde una fuente externa o, si el programador lo permite, desde el propio ST-LINK. En cualquier caso, Vref se utiliza como referencia de niveles lógicos para asegurar compatibilidad eléctrica durante la programación.

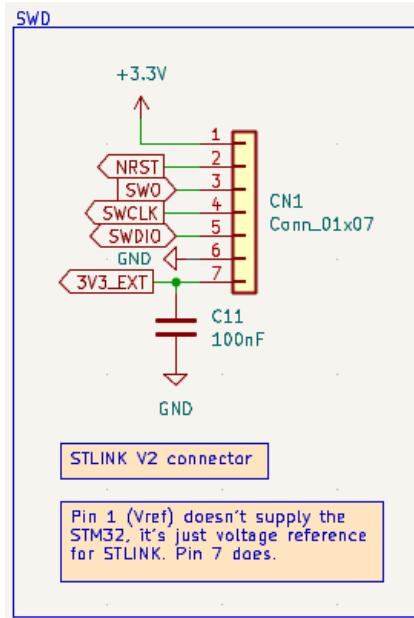


Figura 42: Esquemático de la interfaz SWD. Elaboración propia.

Baterías y paneles solares

La Figura 43 muestra los conectores asociados a la batería y a los paneles solares y sus conexiones. Se han añadido condensadores de desacoplo de 100 nF en las entradas de alimentación por defecto.

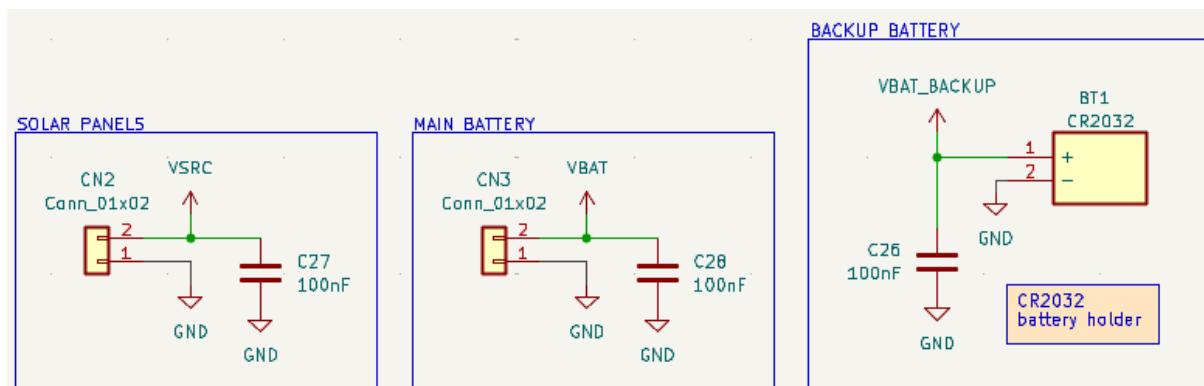


Figura 43: Esquemático de los conectores de la entrada fotovoltaica y las baterías. Elaboración propia.

Sensores

En la Figura 44 se presentan los conectores utilizados para los diferentes sensores del sistema.

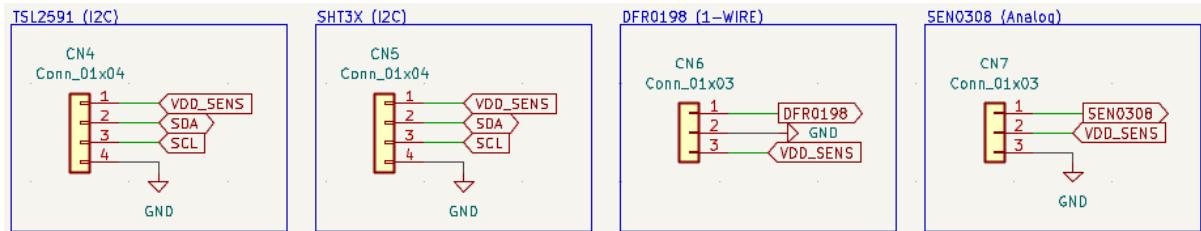


Figura 44: *Esquemático de los conectores de los sensores. Elaboración propia.*

Otros conectores

En la Figura 45 se incluyen conectores auxiliares como la conexión UART para imprimir por consola. También se han añadido un par de puntos de GND accesibles, útiles como referencia de medida para herramientas como el multímetro u osciloscopio, simplificando tareas de verificación en laboratorio.

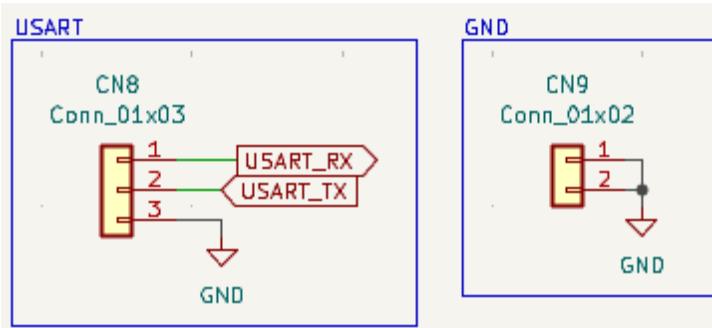


Figura 45: *Esquemático de los conectores de UART y pins GND de verificación. Elaboración propia.*

Adicionalmente, se han incorporado jumpers para desacoplar y depurar ciertas conexiones críticas, como la batería y la entrada fotovoltaica hacia el AEM10941, así como las líneas STATUS del propio PMIC. La finalidad de estos jumpers es poder aislar el AEM10941 en caso de fallo y, si fuese necesario, conectar de forma externa una placa de evaluación del mismo componente para mantener la funcionalidad equivalente durante pruebas.

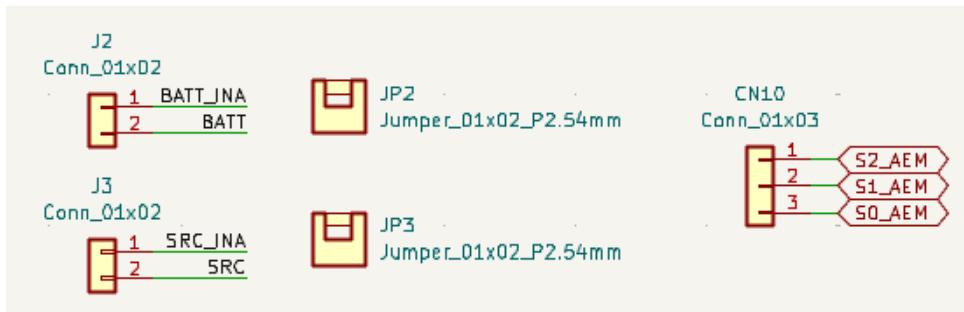


Figura 46: Esquemático de los jumpers de desacople del PMIC. Elaboración propia.

11.5. Diseño PCB

Una vez finalizados los esquemáticos se pasó al desarrollo del layout de la PCB. Para esta etapa se tuvieron muy en cuenta las referencias de la placa NUCLEO-WB55RG. Como reglas de diseño base se utilizó un ancho de pista por defecto para señales de 0.2 mm, y para las vías un diámetro de 0.6 mm con taladro de 0.3 mm. En las pistas de potencia se aumentó el ancho hasta un máximo de 0.8 mm.

La disposición de componentes se realizó agrupando, en la medida de lo posible, los elementos de cada subsistema, lo que facilitó el ruteado y redujo cruces innecesarios. Siempre que existían *layouts* de referencia de los fabricantes, se tomó como guía para minimizar riesgos de integración.

En cuanto a la fabricación, se seleccionó **JLCPCB** como proveedor, por lo que el diseño del *layout* y las reglas de ruteado se ajustaron desde el inicio a sus capacidades y restricciones de proceso.

11.5.1. Apilado físico (*stack-up*)

Para escoger el número de capas y el apilado físico se tuvieron en cuenta varios factores. El primero fue el coste: las placas de 2 capas son las más económicas, seguidas por las de 4 capas. Sin embargo, dado el número de componentes y señales del sistema, se optó por un diseño de 4 capas, que permite un ruteado más limpio y el uso de planos de masa continuos.

Además, el *stack-up* de 4 capas es especialmente ventajoso para el control de impedancia, necesario en el trazado RF y en la conexión hacia la antena. Este aspecto se desarrolla con más detalle en la Sección 11.5.5.

4-Layer Impedance Control Stackup					Outer Copper Weight		Inner Copper Weight			
Thickness	0.8mm	1.0mm	1.2mm	1.6mm	2.0mm	1oz	2oz	0.5oz	1oz	2oz

Figura 47: Grosores de la PCB escogidos. *JLCPCB*.

JLCPCB ofrece distintos espesores y opciones de fabricación para placas de 4 capas. Se escogió la opción estándar de 1.6 mm y 1 oz de cobre en las capas exteriores, tanto por

disponibilidad como por coherencia con el control de impedancia. Dentro de los *stack-ups* ofertados se seleccionó el **JLC04161H-7628**, mostrado en la Figura 48, que define espesores dieléctricos y materiales y permite calcular anchos de pista para impedancias objetivo.

Layer	Material Type	Thickness	
Top Layer	Copper	0.035mm	
Prepreg	7628*1	0.21040mm	
Inner Layer L2	Copper	0.0152mm	
Core>	Core	1.065mm	1.1mm H/H0Z with copper
Inner Layer L3	Copper	0.0152mm	
Prepreg	7628*1	0.21040mm	
Bottom Layer	Copper	0.035mm	

Figura 48: *Stack-up JLC04161H-7628* seleccionado junto con espesores y materiales de cada capa. *JLCPCB*.

Respecto a la distribución por capas, la capa *top* se reservó para la mayor parte de componentes y pistas de señal, con el objetivo de facilitar el ensamblaje. La *inner1* se dedicó a un plano continuo de GND, lo cual mejora el retorno de corriente y ayuda al control de impedancia en líneas críticas. La *inner2* se destinó principalmente a planos de alimentación, con la excepción de la zona RF, donde se mantuvo un plano de GND bajo la línea para asegurar una referencia adecuada. Por último, la capa *bottom* se utilizó para las señales restantes cuando no era posible rutearlas en *top* debido a cortes o congestión de pistas, y para colocar el holder de la pila de backup. La Tabla 13 muestra un resumen de la disposición de capas.

Capa	Contenido
<i>Top</i>	Componentes y señales
<i>Inner1</i>	Plano GND
<i>Inner2</i>	Islas de alimentación
<i>Bottom</i>	Señales

Tabla 13: Resumen del contenido de cada capa. Elaboración propia.

11.5.2. Requerimientos de *JLCPCB*

Además del *stack-up*, fue necesario verificar que *JLCPCB* soportase los anchos de pista, separaciones y dimensiones de vías previstas. Para ello se consultaron las especificaciones del fabricante y se fijaron reglas de diseño (DRC) acordes a los mínimos soportados. En la Figura 49 se muestran los valores mínimos de fabricación.

Cobre	
	Margen mínimo: <input type="text" value="0,1"/> mm
	Ancho mínimo de pista: <input type="text" value="0,1"/> mm
	Ancho mínimo de conexión: <input type="text" value="0,1"/> mm
	Ancho mínimo de anular: <input type="text" value="0,05"/> mm
	Mínimo diámetro de vía: <input type="text" value="0,25"/> mm
	Margen de cobre a agujero: <input type="text" value="0,25"/> mm
	Margen de cobre a borde: <input type="text" value="0,3"/> mm
Orificios	
	Orificio pasante mínimo: <input type="text" value="0,3"/> mm
	Margen de orificio a orificio: <input type="text" value="0,5"/> mm
uVías	
	Diámetro mínimo de uVía: <input type="text" value="0,2"/> mm
	Orificio mínimo de uVía: <input type="text" value="0,1"/> mm
Serigrafía	
	Margen mínimo de elemento: <input type="text" value="0,15"/> mm
	Altura mínima del texto: <input type="text" value="1"/> mm
	Grosor mínimo del texto: <input type="text" value="0,15"/> mm

Figura 49: *Requisitos mínimos de fabricación de JLCPB.*

11.5.3. Condensadores de desacoplo

Se prestó especial atención a la colocación y conexión de los condensadores de desacoplo, ya que tienen un papel fundamental en la estabilidad de los integrados y en la reducción de ruido en los raíles de alimentación. Para ello se siguieron las recomendaciones de diseño de la nota de aplicación SPMA056 [14] que se observan en al Figura 50.

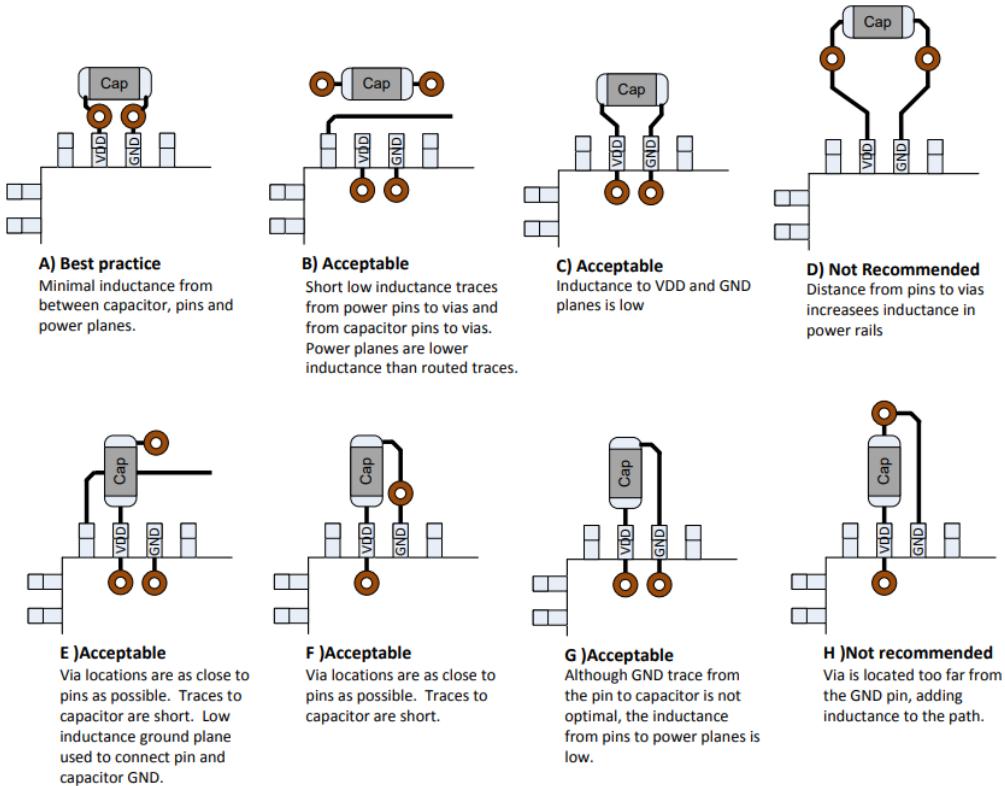


Figura 50: Recomendaciones de colocación y conexión de condensadores de desacoplo. SPMA056.

11.5.4. Islas para los cristales

Otra precaución adoptada para evitar problemas de arranque o inestabilidad fue aislar los osciladores en “islas” dedicadas. Estas islas consisten en una zona local con plano de GND cuidadosamente delimitado, rodeado por un anillo de vías y con un único punto de unión al plano de masa principal.

El objetivo es evitar que corrientes de retorno de señales digitales o de potencia circulen por debajo de las pistas del cristal, reduciendo el acople y mejorando la integridad del oscilador. En la Figura 51 se muestran las islas implementadas para el HSE y el LSE.

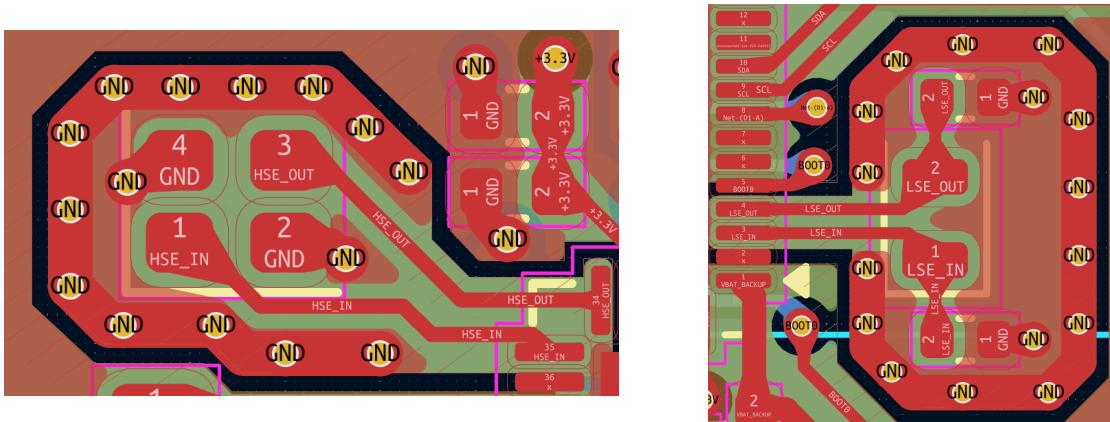


Figura 51: *Implementación de islas de masa y anillo de vías para los osciladores HSE y LSE.*
Elaboración propia.

11.5.5. RF

La última sección crítica del *layout* fue el subsistema RF, tanto la conexión como la geometría de la antena.

Para dimensionar las pistas de impedancia controlada se utilizó la herramienta de JLCPCB para cálculo de impedancia en función del *stack-up* seleccionado [16]. Se optó por una disposición de tipo coplanar con plano de masa, en la que la pista RF discurre encima de una referencia de GND y además está rodeada por cobre de GND lateral en la misma capa (Figura 52). Esta configuración mejora el control de impedancia y confina mejor el campo electromagnético.

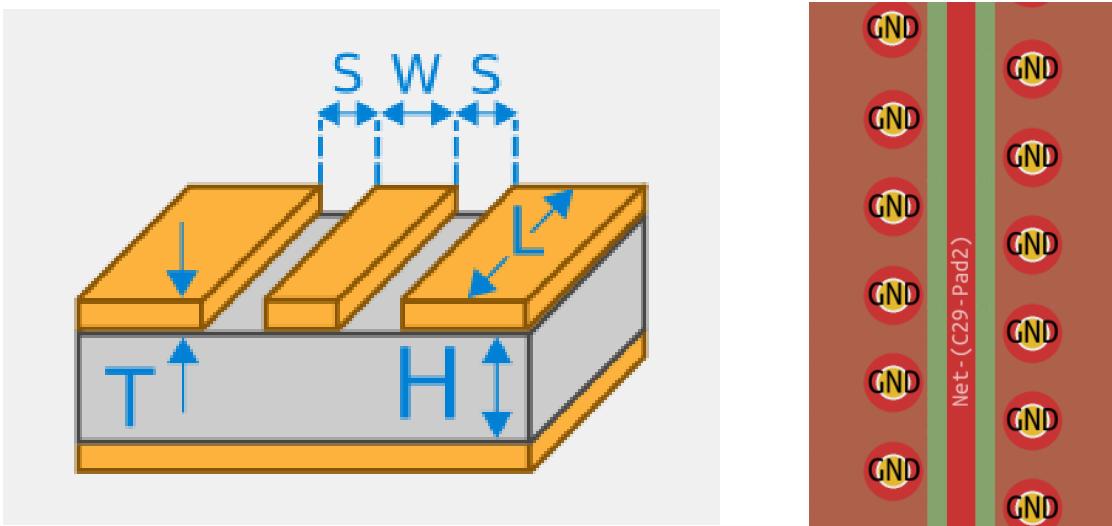


Figura 52: *Estructura de línea coplanar con referencia de GND y su implementación en la PCB para control de impedancia.*

Para el cálculo se requieren el *stack-up*, la impedancia objetivo y la separación entre la pista y el plano GND coplanar (0.16 mm en este diseño). Introduciendo estos parámetros

en la calculadora, se obtiene el ancho óptimo de pista. Para simplificar el diseño y ajustarlo a valores redondeados, se adoptaron los siguientes anchos finales: para $50\ \Omega$ la calculadora devuelve 0.2695 mm y se utilizó 0.28 mm (Figura 53); para $62\ \Omega$ la calculadora devuelve 0.1537 mm y se utilizó 0.16 mm (Figura 54).

The screenshot shows the JLCPCB Impedance Configure tool interface. At the top, there are dropdown menus for 'Layers' (4), 'PCB Thickness' (1.6), 'Inner Copper Weight' (0.5oz), 'Outer Copper Weight' (1oz), and 'Unit' (mm). Below this is a table titled 'Impedance Configure' with columns: Impedance (Ω), Type, Signal Layer, Top Ref, Bottom Ref, Trace Spacing (mm), and Impedance trace to copper (mm). A row is selected for '50' Ω, 'Coplanar Single Ended' type, 'L1' signal layer, 'L2' bottom ref, and '0.16' mm impedance trace to copper. At the bottom of the table, it says '<-3313(Finished thickness1.56mm±10%)' and 'JLC04161H-3313A(Special/Finished thickness1.58mm±10%)' with a 'JLC04161H-7628(Standard/Finished thickness1.59mm±10%)' button.

Impedance (Ω)	Type	Signal Layer	Top Ref	Bottom Ref	Trace Spacing (mm)	Impedance trace to copper (mm)
50	Coplanar Single Ended	L1	/	L2	/	0.16

Figura 53: Cálculo del ancho de pista para $50\ \Omega$ en el stack-up seleccionado.

The screenshot shows the JLCPCB Impedance Configure tool interface. At the top, there are dropdown menus for 'Layers' (4), 'PCB Thickness' (1.6), 'Inner Copper Weight' (0.5oz), 'Outer Copper Weight' (1oz), and 'Unit' (mm). Below this is a table titled 'Impedance Configure' with columns: Impedance (Ω), Type, Signal Layer, Top Ref, Bottom Ref, Trace Spacing (mm), and Impedance trace to copper (mm). A row is selected for '62' Ω, 'Coplanar Single Ended' type, 'L1' signal layer, 'L2' bottom ref, and '0.16' mm impedance trace to copper. At the bottom of the table, it says '<-3313(Finished thickness1.56mm±10%)' and 'JLC04161H-3313A(Special/Finished thickness1.58mm±10%)' with a 'JLC04161H-7628(Standard/Finished thickness1.59mm±10%)' button.

Impedance (Ω)	Type	Signal Layer	Top Ref	Bottom Ref	Trace Spacing (mm)	Impedance trace to copper (mm)
62	Coplanar Single Ended	L1	/	L2	/	0.16

Figura 54: Cálculo del ancho de pista para $62\ \Omega$ en el stack-up seleccionado.

Para el diseño de la antena en PCB se siguieron las recomendaciones de STMicroelectronics recogidas en la *Application Note AN5129* [12]. En la Figura 55 aparecen las dimensiones recomendadas.

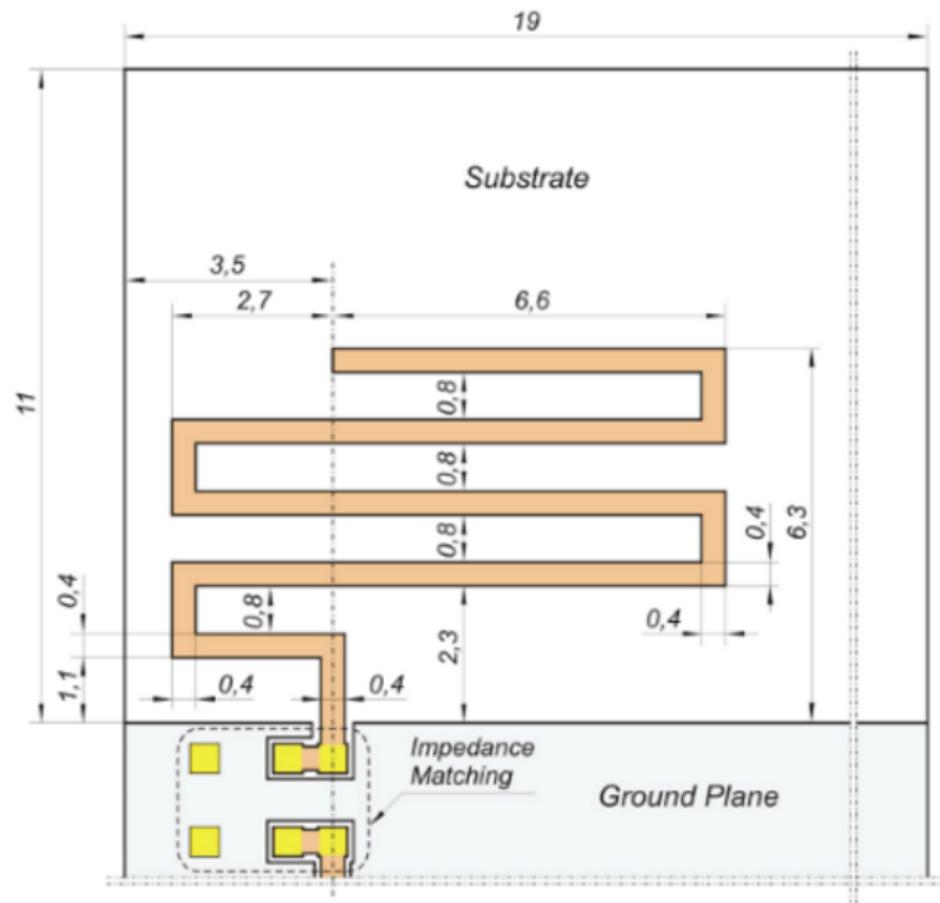


Figura 55: Dimensiones recomendadas para la antena PCB.

11.5.6. Resultado final

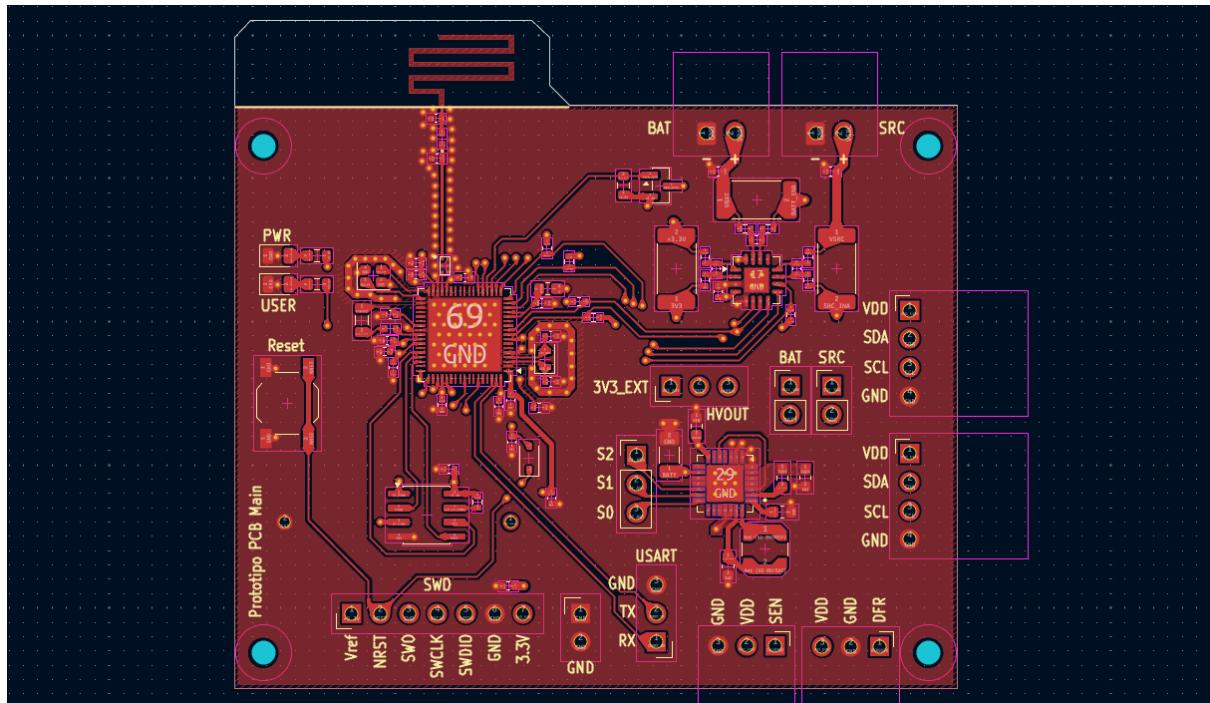


Figura 56: Capa superior del layout de la PCB final (componentes y señales). Elaboración propia.

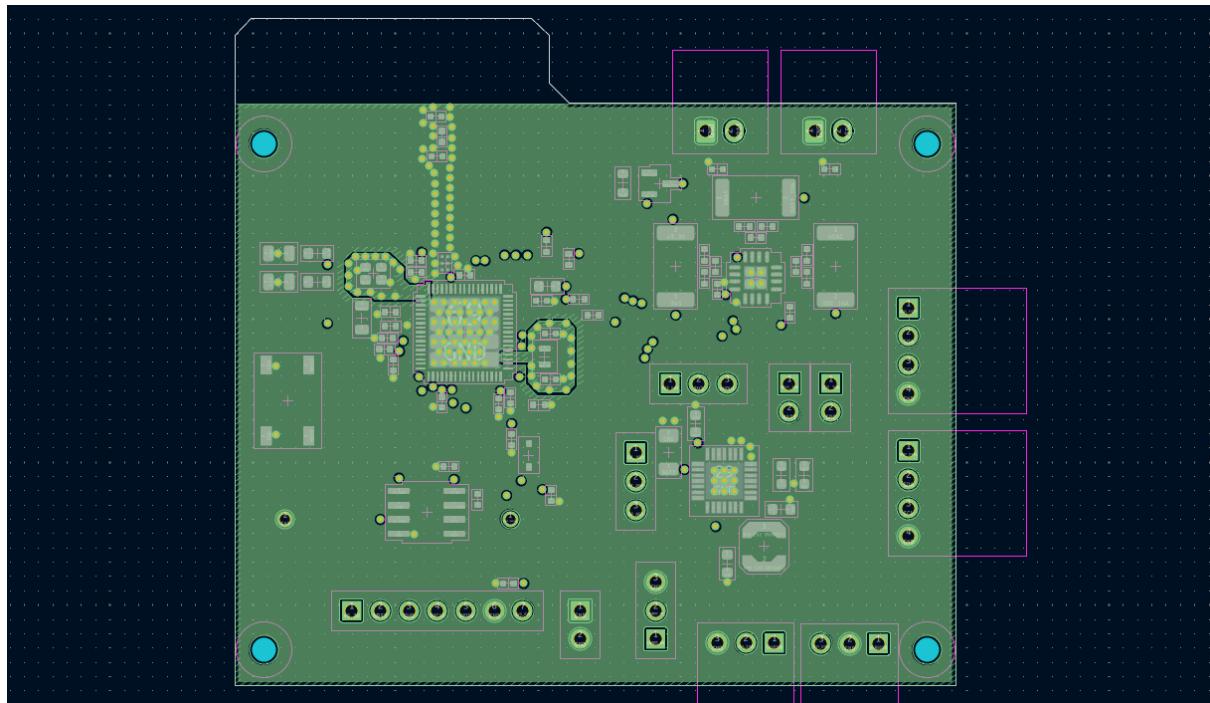


Figura 57: Primera capa interior del layout de la PCB final (plano GND). Elaboración propia.

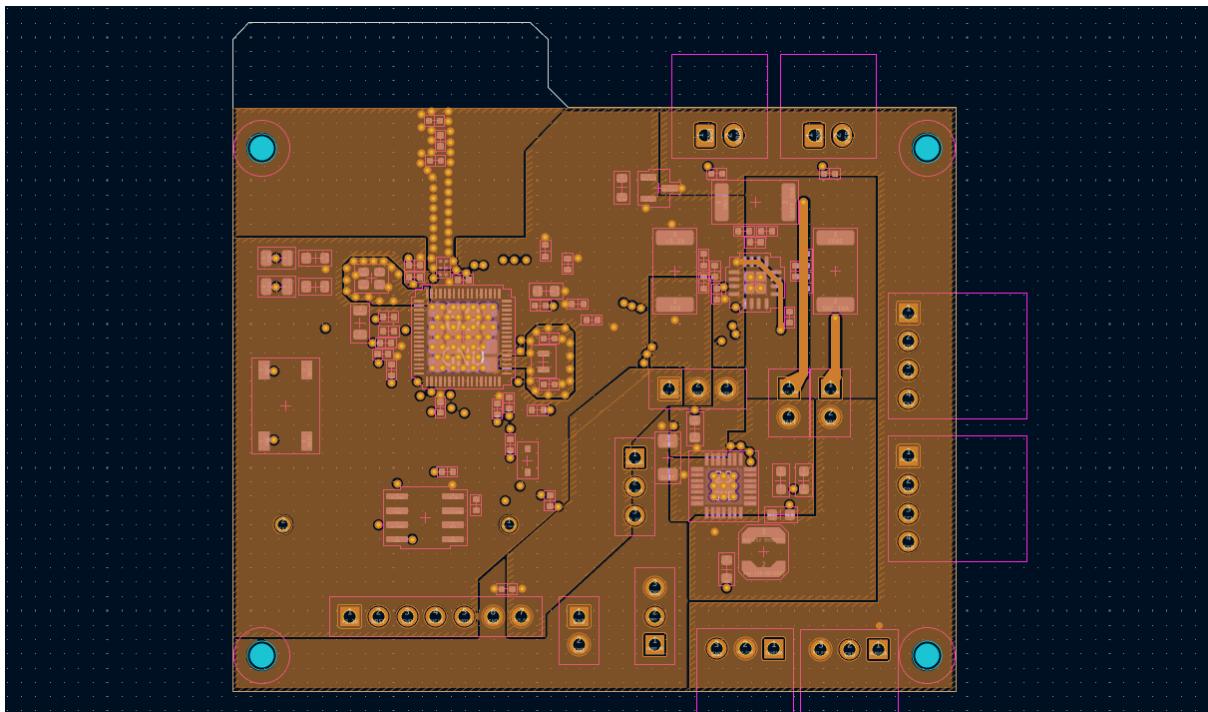


Figura 58: Segunda capa interior del layout de la PCB final (planos de alimentación).
Elaboración propia.

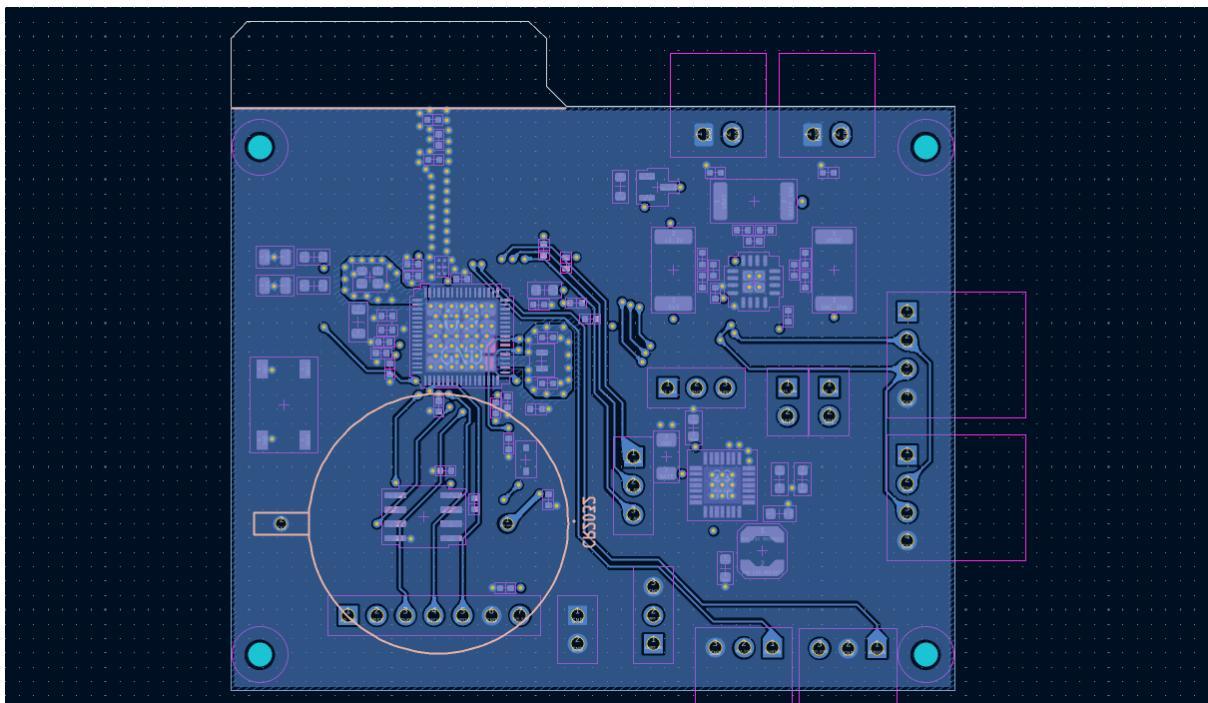


Figura 59: Capa inferior del layout de la PCB final (señales restantes y pila CR2032).
Elaboración propia.

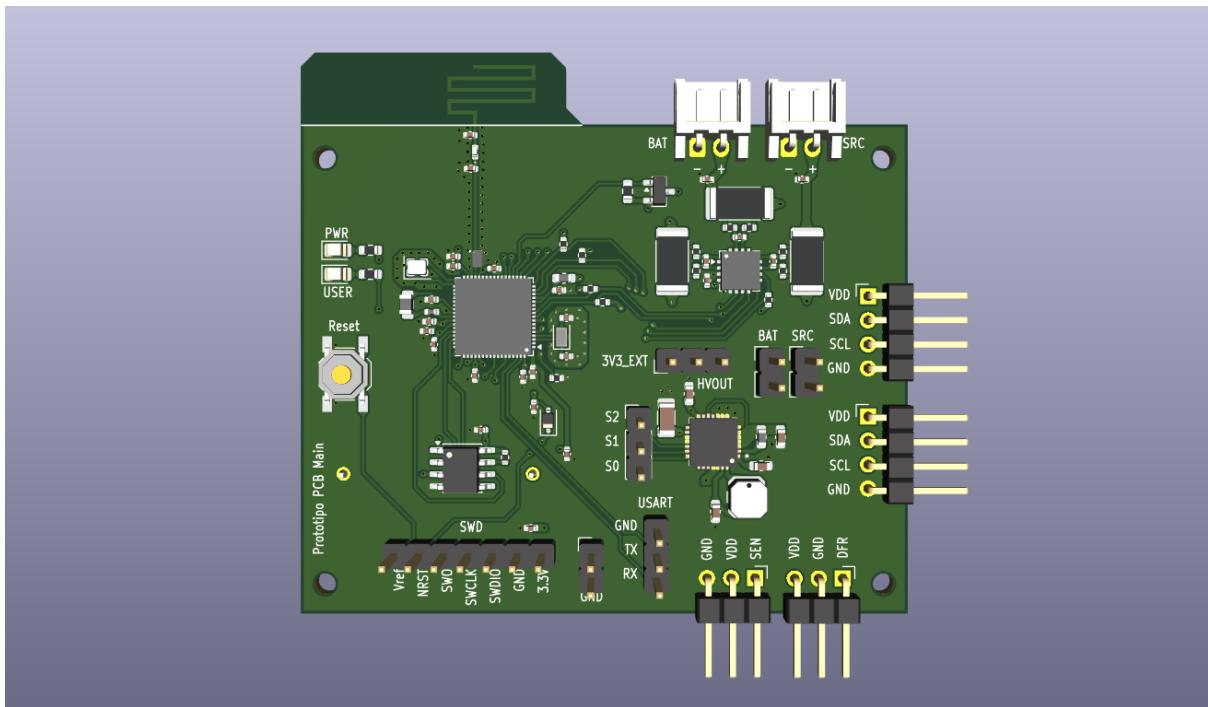


Figura 60: Vista frontal del ensamblaje de la PCB. Elaboración propia.

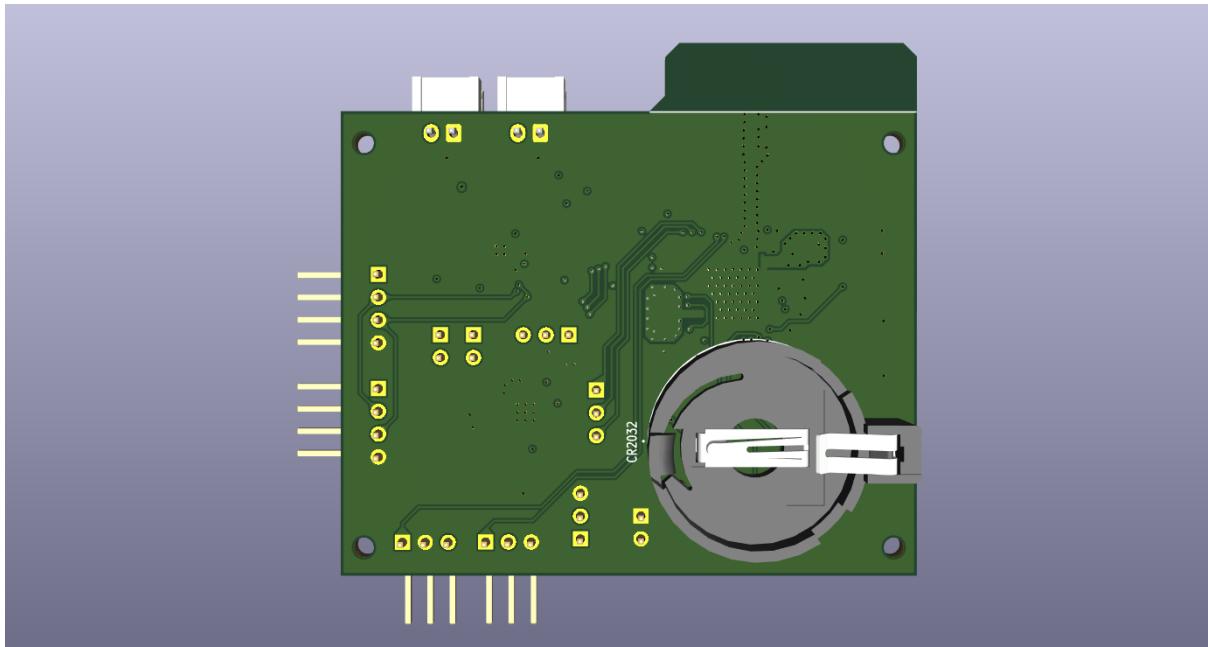


Figura 61: Vista posterior del ensamblaje de la PCB. Elaboración propia.

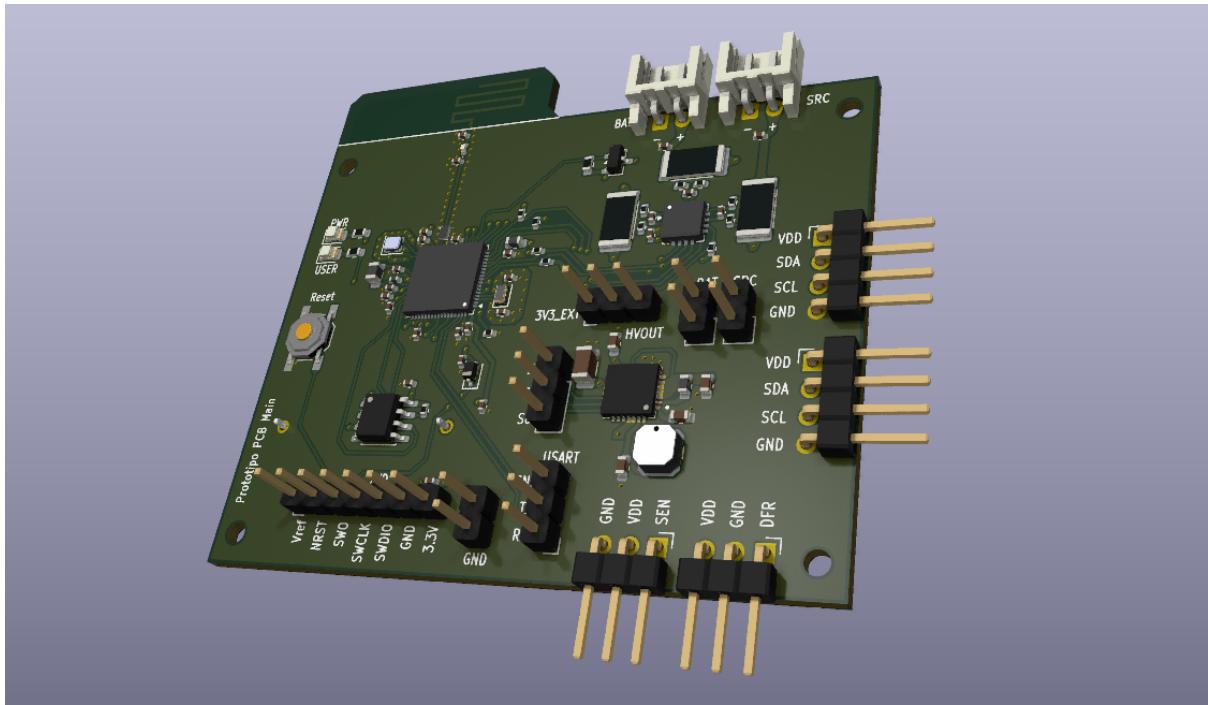


Figura 62: Render 3D del ensamblaje final de la PCB. Elaboración propia.

11.6. Fabricación y montaje

Cuando se finalizó el diseño de la PCB se generaron los archivos de fabricación necesarios para realizar el pedido a JLCPCB. En concreto, los ficheros principales fueron los Gerbers, la lista de materiales (BOM) y el archivo de posición (*Pick & Place*), que permiten tanto la fabricación de la placa como el ensamblaje automático de los componentes.

11.6.1. Proceso de elección de componentes de JLCPCB

JLCPCB ofrece dos modalidades de fabricación: fabricar únicamente la PCB para que el ensamblaje lo realice el usuario, o fabricar la PCB y realizar también el ensamblaje con los componentes seleccionados. Dado que el sistema incorpora encapsulados QFN, difíciles de soldar manualmente con herramientas básicas, y pasivos de tamaño 0402, se optó por la segunda opción, delegando el ensamblaje automático en el fabricante para reducir riesgo de fallo por soldadura.

Para que JLCPCB pueda montar los componentes es necesario asignar a cada referencia un número de parte de su catálogo [17]. Durante el proceso de selección se tuvo en cuenta si los componentes pertenecían a la categoría *Basic* o *Extended*, ya que estos últimos conllevan un coste adicional por componente, lo que puede penalizar el presupuesto si se utilizan muchos componentes en dicha categoría.

En general no hubo problemas para encontrar equivalentes adecuados en el catálogo de JLCPCB. La excepción fue el AEM10941, que en el momento del pedido no estaba disponible para ensamblaje. Por este motivo se decidió comprar dicho integrado por separado y soldarlo

manualmente en la placa tras la recepción, manteniendo el resto del montaje realizado por el fabricante.

11.6.2. BOM

En la Tabla 14 se proporciona un resumen de la BOM resultante del proyecto.

Reference	Value	Component	Description	JLCPCB Part #
C7	22uF	CL10A226MQ8NRNC	22uF 6.3V X5R ±20% 0603 Multi-layer Ceramic Capacitors SMD	C59461
C9	150uF	GRM31CR60J157ME11L	150uF 6.3V X5R ±20% 1206 Multilayer Ceramic Capacitors SMD	C528968
C20	100pF	0402CG101J500NT	100pF 50V C0G ±5% 0402 Multi-layer Ceramic Capacitors SMD	C1546
C30	1.2pF	GJM1555C1H1R2WB01D	1.2pF 50V C0G 0402 Multilayer Ceramic Capacitor SMD	C76899
C1, C2, C3, C4, C10, C11, C12, C13, C14, C16, C17, C18, C19, C21, C23, C26, C27, C28	100nF	CL05B104KO5NNNC	100nF 16V X7R ±10% 0402 Multi-layer Ceramic Capacitors SMD	C1525
C15, C22	4.7uF	CL05A475MP5NRNC	4.7uF 10V X5R ±20% 0402 Multi-layer Ceramic Capacitors SMD	C23733
C24, C25	12pF	0402CG120J500NT	12pF 50V C0G ±5% 0402 Multilayer Ceramic Capacitor SMD	C1547
C5, C6, C8	10uF	CL10A106KP8NNNC	10uF 10V X5R ±10% 0603 Multi-layer Ceramic Capacitors SMD	C19702
D1	D_Schottky	1N5819WS	40V 1A Schottky Diode SOD-323	C191023
FLT1	LPF_2.4GHz	MLPF-WB55-01E3	2.4GHz-2.5GHz 50Ω 0.9dB Low Pass RF Filter SMD-6P, 1.6x1mm	C2651048
L1	10uH	LPS4018-103MRC	10uH 200mΩ 1.3A 40MHz ±20% Magnetic Shielded Inductor 3.9x3.9mm SMD	C6127589
L2	10uH	MLZ1608M100WT000	10uH 1.05Ω 90mA ±20% Inductor 0603 SMD	C76798
L3	600@100MHz	GZ1608D601TF	600Ω@100MHz 450mΩ 200mA Ferrite Beads ±25% 0603 SMD	C1002
L4	10nH	SDCL1005C10NJTD	10nH 400mΩ 300mA Inductor ±5% 0402 SMD	C27147
L5	10uH	SDFL2012S100KTF	10uH 1.15Ω 15mA Inductor ±10% 0805 SMD	C1046
L6	3.6nH	LQG15HS3N6S02D	3.6nH 0.14Ω 750mA Inductor ±5% 0402 SMD	C19216
LED1	Green	KT-0805G	2.6-3.1V 5mA 430mcd 525nm 100mW Emerald Green LED Water Clear 0805 SMD	C2297
LED2	White	KT-0805W	2.6-3.2V 25mA 350mcd 80mW White LED Difuse Lens 0805 SMD	C34499
Q1	AO3401A	AO3401A	P-Channel MOSFET SOT-23	C15127
R16	470	0603WAF4700T5E	470Ω 75V 100mW Thick Film Resistor ±1% ±100ppm/°C 0603 SMD	C23179
R17	300	0603WAF3000T5E	300Ω 75V 100mW Thick Film Resistor ±1% ±100ppm/°C 0603 SMD	C23025
R18	100k	0603WAF1003T5E	100kΩ 75V 100mW Thick Film Resistor ±1% ±100ppm/°C 0603 SMD	C25803
R1, R2, R3	0.22	RLP25FEER220	220mΩ 2W Current Sense Resistor ±1% ±50ppm/°C 2512 SMD	C912987
R10, R11, R12, R13, R14, R15, R19	10k	0402WGF1002TCE	10kΩ 50V 62.5mW Thick Film Resistor ±1% ±100ppm/°C 0402 SMD	C25744
R4, R5, R6, R7, R8, R9	10	0402WGF100JTCE	10Ω 50V 62.5mW Thick Film Resistor ±1% ±200ppm/°C 0402 SMD	C25077
SW1	SW_Reset	TS-1187A-B-A-B	100.000 cycles 12V Gold Tactile Switch 5.1x5.1mm SMD-4P	C318884

U1	STM32WB55RG	STM32WB55RGV6	STMicroelectronics Arm Cortex-M4 MCU, 64 MHz, 1.71-3.6V, 49 GPIO, VFQFPN68	C401505
U2	AEM10941	AEM10941-QFN	Highly Efficient Dual-Output PMIC, QFN-28	C9900027046
U3	INA3221	INA3221AIRGVR	Triple-Channel High-Side Shunt and Bus Voltage Monitor, I2C & SMBUS, QFN-16	C181255
U4	MB85RS256B	MB85RS256BPNF	256Kbit 2.7-3.6V 5mA FRAM memory SPI SOIC-8	C8742
X1	32MHz	Q22FA12800025	8pF 32MHz 60Ω SMD2016-4P Crystal	C187794
X2	32.768kHz	NX2012SA-32.768KHZ	9pF 32.768kHz 80Ω SMD2012-2P Crystal	C891748

Tabla 14: *Lista de materiales (BOM) del diseño principal, incluyendo referencias, valores y número de parte de JLPCB. Elaboración propia.*

11.6.3. Proceso de ensamblaje

Una vez enviados los archivos de fabricación y ensamblaje, se recibieron las placas tras el plazo de entrega correspondiente. Al recibirlas, se realizó una verificación incremental del funcionamiento por subsistemas, con el objetivo de detectar posibles fallos de montaje o de diseño de manera controlada.

En primer lugar se comprobó el encendido del LED de POWER, verificando la correcta alimentación y la presencia del bus de 3.3 V. A continuación se validó la MCU, ya que era uno de los puntos con mayor incertidumbre en la integración: se comprobó la programación mediante SWD y la ejecución básica de un programa. Posteriormente se verificó el bloque de monitorización de potencia, la memoria FRAM y la adquisición de datos de los sensores. Finalmente se comprobó el subsistema con mayor sensibilidad al *layout*: la parte de RF, verificando la comunicación BLE y confirmando que el trazado y la antena funcionaban conforme a lo esperado.

Tras confirmar que el ensamblaje era correcto, se procedió al montaje manual de los elementos no ensamblados por el fabricante. En particular, se soldó el AEM10941 junto con conectores y jumpers. La soldadura del AEM10941 resultó especialmente exigente debido a su encapsulado QFN, que requiere control térmico y correcta cantidad de estaño en los pads. Dado que el resto de componentes ya estaban montados, no era viable un proceso de *reflow* completo en un horno, por lo que se recurrió a una pistola de aire caliente para su soldadura. Tras varias iteraciones de práctica y ajustes de temperatura y flujo, se consiguió una soldadura correcta del integrado y del resto de pines auxiliares.

Figura 63: *PCB final completamente soldada. Elaboración propia.*

11.7. Integración en la lanza

???

Capítulo 12

Desarrollo software

12.1. Entorno y herramientas

Para el desarrollo de firmware en microcontroladores STM32 existen distintas alternativas, sin embargo, para este proyecto se ha optado por el ecosistema oficial de STMicroelectronics por ser la opción más integrada y robusta: facilita la configuración del microcontrolador, genera automáticamente el código base de inicialización y ofrece herramientas consolidadas para programación y depuración.

Las herramientas principales empleadas han sido:

- **STM32CubeMX:** herramienta de configuración y generación de proyecto. Permite seleccionar el microcontrolador, habilitar periféricos (I2C, SPI, ADC, RTC, GPIO...), asignar pines, configurar relojes, opciones de bajo consumo y generar el código de inicialización y la estructura del proyecto.
- **STM32CubeIDE:** entorno de desarrollo integrado utilizado para editar, compilar y depurar el *firmware*. Integra compilador y depurador, y permite programar y depurar la MCU mediante ST-LINK.
- **STM32CubeProgrammer:** herramienta para programar y verificar el microcontrolador. Permite cargar binarios en memoria Flash, comprobar su integridad, leer/escribir memoria y configurar parámetros del dispositivo cuando es necesario.

En el caso del STM32WB55, para poder utilizar la pila BLE fue necesario cargar previamente el stack proporcionado por el fabricante en la memoria Flash. Para ello se empleó STM32CubeProgrammer y, siguiendo las indicaciones de STMicroelectronics, se programó en la dirección 0x080CE000 el archivo «stm32wb5x_BLE_Stack_full_fw.bin» [18].

12.2. Arquitectura del *firmware*

???

12.3. Drivers

Todos los *drivers* de los componentes fueron desarrollados específicamente para este proyecto. Se decidió implementar *drivers* propios en lugar de utilizar librerías de terceros para mantener el máximo control sobre el acceso al *hardware*, reducir dependencias externas y comprender en profundidad el funcionamiento real de cada componente.

La programación de los *drivers* ha seguido estos principios:

- **API sencilla y consistente:** se ha buscado que todos los *drivers* tengan una interfaz similar, de manera que su integración en el firmware sea directa y homogénea.
- **Funcionalidades completas pero solo las necesarias:** se han implementado las operaciones requeridas por el sistema, evitando incluir características poco utilizadas que aumenten la complejidad, el tamaño de código y la superficie de fallo. Aun así, el diseño se ha dejado abierto para poder ampliar funcionalidades si en el futuro fuese necesario.
- **Portabilidad:** los *drivers* no acceden a registros del microcontrolador directamente, sino que se apoyan en la HAL de ST para las comunicaciones. Esto desacopla el código del microcontrolador concreto y facilita reutilizarlo en otros proyectos con cambios mínimos.

Además, la estructura de los *drivers* sigue un patrón común. En general, cada *driver* se compone de:

- Un *struct* que almacena la información imprescindible para su funcionamiento.
- Un conjunto de *defines* y enumeraciones para facilitar el uso del componente (direcciones de registros, máscaras de bits, constantes de conversión, selección de modos...).
- Un conjunto de funciones públicas que exponen las operaciones necesarias para el sistema (inicialización, lectura/escritura...).

12.3.1. INA3221

El *driver* del INA3221 se comunica mediante el bus I2C. El integrado trabaja como esclavo y su dirección I2C por defecto es 0x40. Su funcionamiento se basa en la escritura y lectura de registros internos: por un lado registros de configuración y por otro registros de medida (tensión de bus y tensión en *shunt*) para cada canal.

El *struct* del componente se denomina `INA3221_t` y contiene el *handle* de I2C de la librería HAL, los valores de las resistencias *shunt* y una serie de parámetros de configuración necesarios para la inicialización: modo de promediado, tiempos de conversión de bus y *shunt*, y modo de operación. De este modo, el *driver* queda parametrizado desde software y no depende de constantes globales.

Las funciones públicas expuestas por el *driver* son:

- `HAL_StatusTypeDef INA3221_Init(INA3221_t *dev)`: escribe el registro de configuración del INA3221 con los parámetros almacenados en el *struct*.
- `HAL_StatusTypeDef INA3221_ReadVoltage(INA3221_t *dev, uint8_t channel, float *busVoltage, float *shuntVoltage)`: lee los registros de tensión de bus y tensión en *shunt* del canal seleccionado y devuelve ambas magnitudes ya convertidas a voltios.
- `float INA3221_CalculateCurrent_mA(INA3221_t *dev, uint8_t channel, float shuntVoltage)`: calcula la corriente del canal a partir de la caída en la resistencia *shunt* y el valor de *shunt* configurado.
- `float INA3221_CalculatePower_mW(float busVoltage, float shuntCurrent_mA)`: calcula la potencia aproximada a partir de tensión de bus y corriente.

Internamente, la inicialización (`INA3221_Init`) construye el valor del registro de configuración combinando los campos de promediado, tiempos de conversión y modo de operación, y lo escribe mediante una operación I2C del tipo *memory write*. La lectura sigue una secuencia simple: valida el canal solicitado (1 a 3), calcula la dirección de los registros a leer en función del canal, realiza dos lecturas I2C de 16 bits (registro de *shunt* y registro de bus) y recomponen los bytes leídos y aplica las conversiones a unidades físicas. En la conversión se tiene en cuenta el formato del dato de medida (bits no significativos) y el signo en la lectura de *shunt*, ya que la caída puede ser positiva o negativa.

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* `INA3221_t`.
2. Llamar a `INA3221_Init` para programar el registro de configuración.
3. Llamar a `INA3221_ReadVoltage` para cada canal y obtener `busVoltage` y `shuntVoltage`.
4. Calcular la corriente con `INA3221_CalculateCurrent_mA` usando la tensión de *shunt* medida.
5. Calcular la potencia con `INA3221_CalculatePower_mW` a partir de tensión de bus y corriente.

12.3.2. MB85RS256B

El *driver* de la memoria FRAM MB85RS256B se comunica mediante el bus SPI. El componente funciona como esclavo y su uso se basa en el envío de un comando junto con una dirección de memoria y, según la operación, los datos a transmitir o a recibir. Al tratarse de FRAM, la escritura es directa (no es necesario gestionar borrado por páginas), lo que simplifica el almacenamiento periódico de registros.

El *struct* del componente se denomina `MB85RS256B_t` y contiene el *handle* del periférico SPI de la librería HAL, además de la información necesaria para controlar el pin CS.

Las funciones públicas expuestas por el *driver* son:

- HAL_StatusTypeDef MB85RS256B_Init(MB85RS256B_t *dev): inicializa el *driver*.
- HAL_StatusTypeDef MB85RS256B_Read(MB85RS256B_t *dev, uint16_t addr, uint8_t *data, size_t len): lee len bytes a partir de la dirección addr y los almacena en data.
- HAL_StatusTypeDef MB85RS256B_Write(MB85RS256B_t *dev, uint16_t addr, const uint8_t *data, size_t len): escribe len bytes desde data a partir de la dirección addr.

Internamente, todas las transacciones delimitan el bus mediante el pin CS: se baja CS al inicio y se sube al finalizar. Para una lectura, la función MB85RS256B_Read construye una cabecera de 3 bytes con el comando READ y la dirección (MSB primero), transmite dicha cabecera y a continuación recibe len bytes por SPI. Para una escritura, MB85RS256B_Write habilita previamente la escritura (comando WREN implementado como función interna), transmite la cabecera (WRITE + dirección) y después transmite el bloque de datos. Al finalizar la operación, la escritura se deshabilita de nuevo (comando WRDI) para evitar escrituras accidentales.

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* MB85RS256B_t.
2. Llamar a MB85RS256B_Init para inicializar el *driver*.
3. Llamar a MB85RS256B_Write para almacenar un bloque de datos en una dirección determinada.
4. Llamar a MB85RS256B_Read para recuperar el bloque almacenado.

12.3.3. TSL2591

El *driver* del TSL2591 se comunica mediante el bus I2C. El integrado trabaja como esclavo y su dirección I2C por defecto es 0x29. Su funcionamiento se basa en la escritura y lectura de registros internos: por un lado registros de control y por otro registros de medida que se leen para obtener el nivel de iluminación.

El *struct* del componente se denomina TSL2591_t y contiene el *handle* de I2C de la librería HAL, junto con los parámetros de configuración principales del sensor: ganancia y tiempo de integración. Estos parámetros determinan el rango dinámico y la sensibilidad del sensor, y se emplean durante la inicialización y el cálculo posterior de lux.

Las funciones públicas expuestas por el *driver* son:

- HAL_StatusTypeDef TSL2591_Init(TSL2591_t *dev): inicializa el sensor y configura el registro de control.
- void TSL2591_Enable(TSL2591_t *dev): habilita el sensor escribiendo el registro de enable.

- `void TSL2591_Disable(TSL2591_t *dev)`: deshabilita el sensor.
- `HAL_StatusTypeDef TSL2591_ReadChannels(TSL2591_t *dev, uint16_t *ch0, uint16_t *ch1)`: lee los registros de datos y devuelve los dos canales del sensor: canal completo y canal infrarrojo.
- `float TSL2591_CalculateLux(TSL2591_t *dev, uint16_t full, uint16_t ir)`: calcula el valor de iluminancia en *lux* a partir de las lecturas de los canales, teniendo en cuenta la ganancia y el tiempo de integración configurados.
- `float TSL2591_CalculateIrradiance(float lux)`: obtiene una irradiancia aproximada a partir de lux utilizando un factor de eficacia luminosa definido en el *driver*.

Internamente, el *driver* accede a los registros del sensor utilizando un *command bit* que se añade a la dirección del registro. En la inicialización, primero se habilita el sensor (registro *ENABLE*) y después se configura el registro *CONTROL* combinando los campos de integración y ganancia en un único byte de configuración. La lectura de canales realiza una lectura I2C de 4 bytes consecutivos a partir del registro de datos del canal 0, y reconstruye los dos valores de 16 bits correspondientes al espectro (CH0) y infrarrojo (CH1). A partir de estas lecturas, el cálculo de lux ajusta la medida según la configuración actual.

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* `TSL2591_t`.
2. Llamar a `TSL2591_Init` para habilitar el sensor y programar la ganancia e integración.
3. Llamar a `TSL2591_ReadChannels` para obtener `full` e `ir`.
4. Calcular los lux con `TSL2591_CalculateLux` a partir de `full` e `ir`.
5. Convertir lux a irradiancia aproximada con `TSL2591_CalculateIrradiance`.

12.3.4. SHT3x

El *driver* del SHT3x se comunica mediante el bus I2C. El integrado trabaja como esclavo y su dirección I2C por defecto es 0x44. Su funcionamiento se basa en el envío de *comandos* y la lectura de datos a través del bus: por un lado comandos de control y por otro la lectura de las medidas de temperatura y humedad.

El *struct* del componente se denomina `SHT3X_t` y contiene el *handle* de I2C de la librería HAL junto con los parámetros principales de configuración utilizados por el *driver*: el modo de repetibilidad y el uso de *clock stretching*. Estos parámetros determinan qué comando de medida se envía al sensor y el tiempo de conversión necesario antes de realizar la lectura.

Las funciones públicas expuestas por el *driver* son:

- `HAL_StatusTypeDef SHT3X_Init(SHT3X_t *dev)`: inicializa el componente realizando un *soft reset*.

- `HAL_StatusTypeDef SHT3X_SoftReset(SHT3X_t *dev)`: envía el comando de reinicio del sensor.
- `HAL_StatusTypeDef SHT3X_ReadRaw(SHT3X_t *dev, uint16_t *rawT, uint16_t *rawRH)`: realiza una medida *single-shot* y devuelve los valores crudos de temperatura y humedad.
- `HAL_StatusTypeDef SHT3X_ReadSingleShot(SHT3X_t *dev, float *temp_c, float *rh_perc)`: realiza una medida *single-shot* y devuelve temperatura en °C y humedad relativa en %.
- `float SHT3X_CalculateDewpoint(float temp_c, float rh_perc)`: calcula el punto de rocío a partir de temperatura y humedad.
- `HAL_StatusTypeDef SHT3X_ReadStatus(SHT3X_t *dev, uint16_t *status)`: lee el registro de estado del sensor.
- `HAL_StatusTypeDef SHT3X_ClearStatus(SHT3X_t *dev)`: limpia el registro de estado.
- `HAL_StatusTypeDef SHT3X_Heater(SHT3X_t *dev, SHT3X_Heater_t state)`: habilita o deshabilita el calefactor interno.

Internamente, para realizar una medida *single-shot*, el *driver* selecciona el comando apropiado en función de *repeatability* y *clockStretch*. A continuación envía dicho comando por I2C. Si se utiliza el modo sin *clock stretching*, el *driver* espera el tiempo de conversión correspondiente antes de leer los datos. La lectura devuelve 6 bytes (temperatura + CRC y humedad + CRC), y el *driver* verifica la integridad mediante CRC antes de reconstruir los valores crudos. Finalmente, `SHT3X_ReadSingleShot` convierte los valores a unidades físicas (°C y %) y limita la humedad relativa al rango [0,100].

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* `SHT3X_t`.
2. Llamar a `SHT3X_Init` para reiniciar e inicializar el sensor.
3. Llamar a `SHT3X_ReadSingleShot` para obtener `temp_c` y `rh_perc`.
4. Calcular punto de rocío con `SHT3X_CalculateDewpoint`.

12.3.5. DS18B20

El *driver* del DS18B20 se comunica mediante el protocolo 1-Wire usando un único GPIO configurado como línea bidireccional. A diferencia de I2C/SPI, el 1-Wire requiere una temporización precisa, por lo que el *driver* implementa la capa física mediante generación de pulsos, lectura/escritura de bits y retardos en microsegundos.

El *struct* del componente se denomina `DS18B20_t` y contiene el puerto y pin GPIO usados como bus 1-Wire, además de la resolución seleccionada, que determina principalmente el

tiempo necesario para completar una conversión de temperatura.

Las funciones públicas expuestas por el *driver* son:

- `HAL_StatusTypeDef DS18B20_Init(DS18B20_t *dev)`: inicializa el *driver* y el sistema de retardos de alta resolución utilizado para el 1-Wire.
- `HAL_StatusTypeDef DS18B20_ReadTemperature(DS18B20_t *dev, float *temp_c)`: inicia una conversión, espera a que finalice según la resolución configurada y lee el *scratchpad* para devolver la temperatura en °C.
- `HAL_StatusTypeDef DS18B20_ReadROM(DS18B20_t *dev, uint8_t rom[8])`: lee el *ROM code* de 64 bits del sensor (útil para identificación) y valida la lectura mediante CRC.

Internamente, el *driver* implementa las primitivas 1-Wire necesarias: pulso de *reset* y detección de presencia, escritura y lectura de bits y bytes, y cálculo de CRC8. La función `DS18B20_ReadTemperature` se realiza en dos transacciones: primero envía `SKIP_ROM` y el comando `CONVERT_T` para iniciar la conversión; después espera un tiempo máximo que depende de la resolución y finalmente vuelve a hacer *reset*, envía `READ_SCRATCH` y lee 9 bytes. La integridad se comprueba con CRC y, si es correcto, se reconstruye el valor crudo y se convierte a °C.

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* `DS18B20_t`.
2. Llamar a `DS18B20_Init` para inicializar el *driver*.
3. Llamar a `DS18B20_ReadTemperature` para obtener la temperatura en °C.

12.3.6. SEN0308

El *driver* del SEN0308 se basa en la lectura de una señal analógica proporcional a la humedad del suelo. A nivel de firmware, el sensor se lee mediante el ADC del microcontrolador, utilizando las funciones de la HAL para iniciar la conversión, esperar a que finalice y obtener el valor digital resultante.

El *struct* del componente se denomina `SEN0308_t` y contiene el *handle* del ADC de la librería HAL, junto con parámetros de calibración para convertir la lectura a una escala relativa: `airRaw` (lectura en seco/aire) y `waterRaw` (lectura en agua/suelo saturado). Adicionalmente, el *struct* incluye el puerto y pin asociados al sensor para posibles tareas de control.

Las funciones públicas expuestas por el *driver* son:

- `HAL_StatusTypeDef SEN0308_Init(SEN0308_t *dev)`: inicializa el *driver*.
- `HAL_StatusTypeDef SEN0308_ReadRaw(SEN0308_t *dev, uint16_t *rawMoisture)`: realiza una conversión ADC y devuelve la lectura cruda del sensor.

- `HAL_StatusTypeDef SEN0308_ReadRawAvg(SEN0308_t *dev, uint16_t *rawMoisture, uint8_t numSamples)`: obtiene una lectura promediada para reducir ruido.
- `uint8_t SEN0308_CalculateRelative(SEN0308_t *dev, uint16_t rawMoisture)`: convierte la lectura cruda a un porcentaje relativo (0–100 %) usando los puntos de calibración definidos.

Internamente, `SEN0308_ReadRaw` ejecuta la secuencia típica del ADC con HAL: `HAL_ADC_Start`, `HAL_ADC_PollForConversion` con un *timeout* definido, lectura del valor con `HAL_ADC_GetValue` y parada del ADC con `HAL_ADC_Stop`. La función `SEN0308_ReadRawAvg` repite esta lectura un número configurable de veces, acumula las muestras y devuelve la media (incluyendo un pequeño retardo entre muestras). Finalmente, `SEN0308_CalculateRelative` limita la lectura al rango de calibración (`waterRaw–airRaw`) y aplica un mapeo lineal para obtener el porcentaje relativo, incorporando redondeo para mejorar estabilidad.

El proceso de uso habitual del componente sería:

1. Asignar los valores de configuración del *struct* `SEN0308_t`.
2. Llamar a `SEN0308_Init` para inicializar el *driver*.
3. Llamar a `SEN0308_ReadRawAvg` para obtener la lectura cruda.
4. Convertir la lectura a humedad relativa con `SEN0308_CalculateRelative`.

12.4. Gestión de la energía

Para maximizar la eficiencia energética del sistema se opta por entrar en un modo de bajo consumo entre lecturas, concretamente en STOP2. En este modo el consumo es del orden de pocos microamperios según especificaciones del fabricante, lo que permite minimizar el gasto energético entre adquisiciones.

La contrapartida es que en STOP2 se detienen la CPU y la mayoría de relojes del sistema, por lo que al salir del modo es habitual tener que reconfigurar el reloj del sistema y reinicializar periféricos usados por la aplicación, antes de continuar con el ciclo normal. En STOP2 se conserva el contenido de SRAM y registros, y puede mantenerse activo el RTC.

Sin embargo el STM32WB55RG integra varios subsistemas con comportamiento de potencia independiente: la CPU1 (Cortex-M4), la CPU2 (Cortex-M0+) y el sub-sistema de radio. Cada uno puede estar en su propio estado interno, y el modo de bajo consumo “real” del sistema se alcanza únicamente cuando todos están en STOP2. Si una de las CPUs (o el radio) permanece activa, el sistema no puede caer al modo profundo y el consumo queda dominado por el subsistema que siga en funcionamiento.

Para salir de STOP2 en el núcleo principal existen cuatro vías:

1. **RTC**: despertador periódico cada 20 minutos para leer sensores, almacenar datos y volver a STOP2.

2. **Interrupciones externas (EXTI)**: distintas alertas por pines, tratadas según su origen.
3. **Eventos BLE/radio**: cuando el subsistema inalámbrico detecta una conexión.
4. **UART**: usada para cargar parámetros como fecha y hora y ajustar el RTC.

12.5. Gestión de los datos

En la Figura 64 se observa la estructura a nivel de datos de la FRAM utilizada.

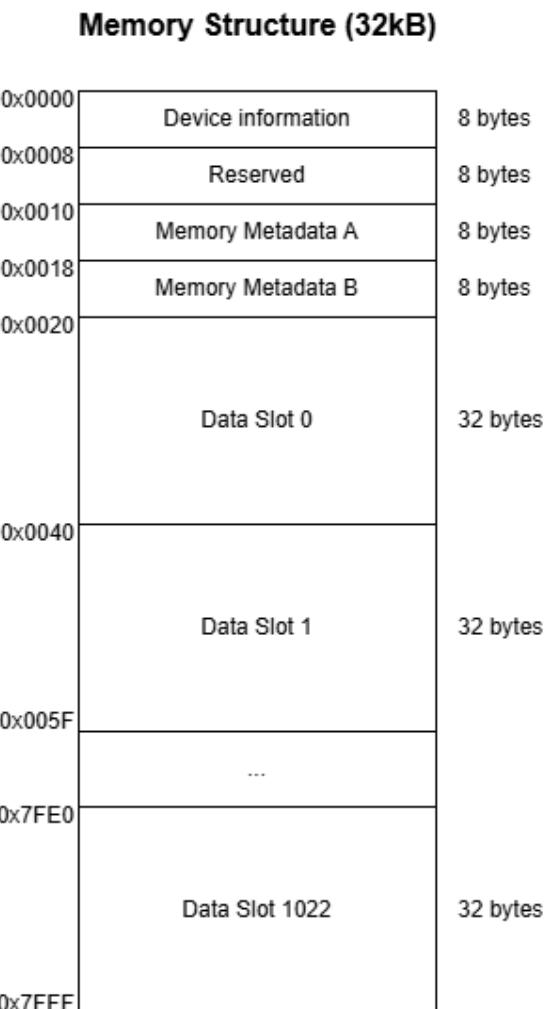


Figura 64: *Estructura de la memoria con contenido, dirección y tamaño de los bloques.*
Elaboración propia.

Para lograr un uso eficiente, robusto y escalable, se decidió dividir la memoria en slots de tamaño fijo, concretamente de 32 bytes. Este enfoque simplifica la gestión (direcciones y tamaños constantes), facilita la detección de errores y permite implementar mecanismos de tolerancia a fallos.

Cabecera

En la Tabla 15 se expone el contenido de la cabecera guardada en el Slot 0 de la memoria.

Dirección	Tamaño (bytes)	Nombre
0x0000	8	Información del dispositivo
0x0008	8	Reservado
0x0010	8	Metadatos de la memoria (copia A)
0x0018	8	Metadatos de la memoria (copia B)

Tabla 15: *Distribución de la cabecera de la memoria. Elaboración propia.*

En el bloque de información del dispositivo se almacenan datos como el identificador del dispositivo (ID) u otros campos útiles para trazabilidad o configuración. Los 8 bytes reservados se utilizan como mecanismo de verificación al arrancar: se escribe un patrón conocido, se lee de vuelta y se comprueba que coincide. Este test permite detectar problemas de comunicación con la FRAM o fallos de escritura/lectura antes de operar con datos reales.

La gestión de la FRAM está diseñada para soportar cortes de alimentación sin perder completamente el estado interno del almacenamiento. Para ello se guardan metadatos tales como el puntero de escritura o un contador de slots ocupados. Estos metadatos se almacenan en dos copias alternas (A y B) de 8 bytes. En cada escritura de un nuevo dato, el firmware actualiza los metadatos, alternando entre A y B (usando un contador de secuencia). La idea es que, si hay una pérdida de alimentación durante la actualización de una de las copias, al reiniciar el sistema se pueda recuperar el estado usando la otra copia válida. Con esta estrategia, el peor caso es perder como máximo un slot.

A continuación se observa el struct de los metadatos guardados en memoria:

```
typedef struct {
    uint16_t write_idx; // 2 bytes
    uint16_t count;    // 2 bytes
    uint16_t crc;      // 2 bytes
    uint8_t seq;       // 1 bytes
    uint8_t commit;    // 1 bytes
} MetaFrame_t; // 8 bytes aligned
```

Para asegurar que los metadatos recuperados son correctos se emplean dos mecanismos de robustez. Por un lado, se calcula y almacena un CRC. El CRC es un código de detección de errores que se obtiene aplicando una operación matemática sobre los bytes guardados. Si durante una escritura se corrompe algún byte, es extremadamente probable que el CRC calculado al leer no coincida con el CRC almacenado. Así, durante el arranque el sistema recomputa el CRC a partir de los campos del *struct* y lo compara con el campo *crc*; si no coinciden, esa copia de metadatos se considera inválida y se descarta.

Por otro lado, se incluye un byte de *commit* que actúa como indicador de escritura completada: durante la actualización de metadatos se escribe primero el contenido con *commit*

= 0 y, una vez finalizada correctamente la operación, se actualiza el `commit = 1`. Así, si se interrumpe la alimentación a mitad del proceso, el `commit` no queda validado y esa copia se descarta, recurriendo a la otra copia (A o B) que sí sea coherente.

En el caso de que ambas copias sean inválidas, el sistema considera que no existe un estado previo recuperable y realiza una inicialización limpia de los metadatos en la copia A.

Datos

El resto de la memoria (del slot 1 al 1023) se destina al almacenamiento de muestras de medida. El paquete almacenado incorpora, además de los valores medidos, mecanismos de robustez similares a los usados en los metadatos: un CRC para detección de corrupción y un byte de commit para detectar escrituras incompletas.

La alineación y el orden de los campos fue un aspecto importante del diseño. Al trabajar con un tamaño fijo de slot, era necesario garantizar que el *frame* completo ocupase exactamente 32 bytes y que el compilador no introdujese rellenos inesperados. Por ello, los campos se ordenaron conscientemente y se añadieron bytes reservados al final para mantener un tamaño constante.

Además se incorpora un vector de bits que indica qué medidas de la muestra son válidas.

```
typedef struct {
    float irradiance_Wm2;           // 4 byte
    float airTemp_C;               // 4 bytes
    float soilTemp_C;              // 4 bytes
    uint8_t airHumidity_perc;      // 1 bytes
    uint8_t soilMoisture_perc;     // 1 byte
    uint16_t batteryVoltage_mV;    // 2 bytes
    uint8_t hours, minutes, seconds; // 3 bytes
    uint8_t day, month, year;      // 3 bytes
    uint16_t validDataVector;      // 2 bytes
} DataSample_t; // 24 bytes aligned

typedef struct {
    DataSample_t data;             // 24 bytes
    uint16_t crc;                 // 2 bytes
    uint8_t commit;                // 1 bytes
    uint8_t _reserved[5];          // 5 bytes
} DataFrame_t; // 32 bytes aligned
```

Para garantizar que el tamaño de los *frames* coincide exactamente con los tamaños esperados, se añadieron las comprobaciones en tiempo de compilación siguientes:

```
_Static_assert(sizeof(DataFrame_t) == 32, "DataFrame_t must be 32 bytes");
_Static_assert(sizeof(MetaFrame_t) == 8, "MetaFrame_t must be 8 bytes");
```

Capítulo 13

Validación y pruebas

Una vez desarrollados el *hardware* y el *firmware* se inició el periodo de verificación experimental del prototipo completo. El objetivo de este capítulo es comprobar el funcionamiento del sistema en condiciones semi-reales de uso y validar los aspectos críticos del diseño.

Las pruebas se han planteado siguiendo un enfoque incremental. En primer lugar se confirmó el correcto funcionamiento de la PCB final (*bring-up*) como paso previo a ensayos prolongados (véase la Sección 11.6.3). A continuación se validó el subsistema de memoria, al tratarse del elemento que garantiza el cumplimiento del requisito de almacenamiento local. Después se realizaron pruebas de autonomía en un escenario desfavorable, con el sistema operando durante 24 horas. Por último, se probaron individualmente los sensores y se verificó de forma independiente el funcionamiento básico de la radio mediante ejemplos del fabricante.

13.1. Pruebas de memoria

Se diseñaron pruebas para validar tanto la lectura/escritura básica como el comportamiento del sistema en escenarios límite. En todos los casos el objetivo es asegurar que el registro de datos es consistente y que el sistema puede operar durante periodos prolongados sin degradación del almacenamiento.

Como metodología común, las pruebas se ejecutaron mediante un bucle periódico en el que el *firmware* realiza: (1) escritura de un *frame* en FRAM, (2) lectura inmediata del mismo *slot* y (3) comprobación de validez del registro leído.

Escritura y lectura básicas

Como prueba inicial se implementó un ensayo en el que el sistema escribe un registro en FRAM y lo lee de vuelta a continuación. Esta prueba permite detectar fallos de comunicación SPI, errores de direccionamiento y problemas de coherencia durante escritura.

Criterio de aceptación: la lectura posterior debe coincidir con lo escrito durante un número elevado de ciclos consecutivos, sin errores ni valores inconsistentes.

En el *log* se observa que, tras cada escritura, la lectura del mismo *slot* es correcta y el registro se marca como válido (Read Valid: 1). Además, durante la inicialización se detecta que ambas copias de metadatos son válidas (Both meta valid), lo cual confirma que el sistema puede recuperar el contexto previamente almacenado.

```
Both meta valid
FRAM inicializada correctamente

Write Slot: 0 (0x0020)
Write Count: 1
Write Meta B
Read Slot: 0 (0x0020)
Read Valid: 1

Write Slot: 1 (0x0040)
Write Count: 2
Write Meta A
Read Slot: 1 (0x0040)
Read Valid: 1

Write Slot: 2 (0x0060)
Write Count: 3
Write Meta B
Read Slot: 2 (0x0060)
Read Valid: 1
```

Prueba de *overflow*

En esta prueba se valida el comportamiento del sistema cuando el área de datos alcanza su capacidad máxima. El prototipo implementa una política de *buffer* circular, de forma que al llenarse la memoria el sistema continúa almacenando nuevas muestras sobrescribiendo progresivamente las más antiguas.

Criterio de aceptación: al alcanzar el número máximo de muestras, el sistema debe seguir operando sin bloquearse y continuar escribiendo registros válidos de forma controlada.

En el *log* se observa que al llegar al valor máximo del contador (Write Count: 1023) el sistema continúa ejecutando el ciclo de escritura/lectura y mantiene lecturas válidas. Esto confirma que el sistema no se detiene al llenarse la FRAM y que el almacenamiento sigue siendo funcional.

```
Write Slot: 304 (0x2620)
Write Count: 1022
Write Meta B
Read Slot: 304 (0x2620)
Read Valid: 1

Write Slot: 305 (0x2640)
Write Count: 1023
Write Meta A
Read Slot: 305 (0x2640)
Read Valid: 1

Write Slot: 306 (0x2660)
Write Count: 1023
Write Meta B
Read Slot: 306 (0x2660)
Read Valid: 1

Write Slot: 307 (0x2680)
```

```
Write Count: 1023
Write Meta A
Read Slot: 307 (0x2680)
Read Valid: 1
```

Prueba de *reset lógico* de FRAM y recuperación tras reinicio

Esta prueba evalúa dos escenarios: (1) un *reset lógico* del subsistema de memoria (reinicialización del área de datos y metadatos) y (2) un reinicio externo del microcontrolador, para comprobar que no se pierde el contexto al arrancar. En el ensayo se fuerza un *reset lógico* cada 10 escrituras.

Criterio de aceptación: tras el *reset lógico* el sistema debe reiniciar el conteo/índice de escritura sin errores, y tras un <Reset externo> se observa la detección de metadatos válidos (Both meta valid) y la continuación del conteo sin pérdida de contexto.

En el *log* se aprecia que, tras el *reset lógico* (Reset FRAM), el sistema vuelve a escribir desde el *slot* 0 con contador reiniciado. Asimismo, tras un <Reset externo> se observa la detección de metadatos válidos (Both meta valid) y la continuación del conteo sin pérdida de contexto.

```
Write Slot: 9 (0x0140)
Write Count: 10
Write Meta A
Read Slot: 9 (0x0140)
Read Valid: 1

Reset FRAM

Write Slot: 0 (0x0020)
Write Count: 1
Write Meta B
Read Slot: 0 (0x0020)
Read Valid: 1

Write Slot: 1 (0x0040)
Write Count: 2
Write Meta A
Read Slot: 1 (0x0040)
Read Valid: 1

<Reset externo>

Both meta valid
FRAM inicializada correctamente

Write Slot: 2 (0x0060)
Write Count: 3
Write Meta B
Read Slot: 2 (0x0060)
Read Valid: 1
```

Prueba de borrado completo

En esta prueba se ejecuta un borrado completo de la FRAM y se verifica que la memoria queda con todos los bytes a 0. Tras el borrado se reinicializa el subsistema y se comprueba que el sistema puede volver a operar con normalidad.

Criterio de aceptación: tras ejecutar el borrado completo, la memoria debe considerarse vacía (sin metadatos válidos), y el sistema debe ser capaz de reinicializarse y comenzar a almacenar desde el inicio sin errores.

Para asegurar que el borrado se ha realizado correctamente, el *firmware* recorre la región borrada y comprueba que todos los bytes leídos son 0x00. Si se detectase cualquier byte distinto de cero, el sistema generaría un mensaje de error y marcaría el borrado como fallido, evitando continuar con la memoria en un estado inconsistente.

En el *log* se observa que antes del borrado el sistema puede detectar metadatos válidos (*Both meta valid*), mientras que tras ejecutar Erase FRAM pasa a *None meta valid*, comportamiento coherente con una memoria limpia. A continuación el sistema se reinicializa y comienza a escribir desde el *slot* 0 con lecturas válidas.

```
<Reset externo>

Both meta valid
FRAM inicializada correctamente

Erase FRAM

None meta valid
FRAM inicializada correctamente

Write Slot: 0 (0x0020)
Write Count: 1
Write Meta B
Read Slot: 0 (0x0020)
Read Valid: 1

Write Slot: 1 (0x0040)
Write Count: 2
Write Meta A
Read Slot: 1 (0x0040)
Read Valid: 1
```

13.2. Pruebas de autonomía

Para evaluar el balance energético del sistema durante un funcionamiento continuo se diseñó una prueba que consistió en dejar el prototipo operando durante 24 horas completas. Para aumentar la representatividad del ensayo, se utilizó el periodo de muestreo objetivo de 20 min y se almacenaron todas las muestras en memoria.

El ensayo se realizó en interior, cerca de una ventana para maximizar la exposición a la luz disponible. Además, la prueba se llevó a cabo en invierno y en un día nublado, por lo que las condiciones de captación son desfavorables frente a un despliegue real en exterior. En consecuencia, los resultados obtenidos pueden interpretarse como una cota conservadora del comportamiento esperado.

Durante el experimento se tomaron 73 muestras, desde las 10:45 hasta las 10:45 del día siguiente, con todos los *slots* marcados como válidos, lo que indica que el sistema mantuvo el ciclo de adquisición y almacenamiento sin interrupciones.

Resultados

La Figura 65 muestra la evolución temporal de la tensión de batería y la irradiancia estimada durante todo el ensayo.

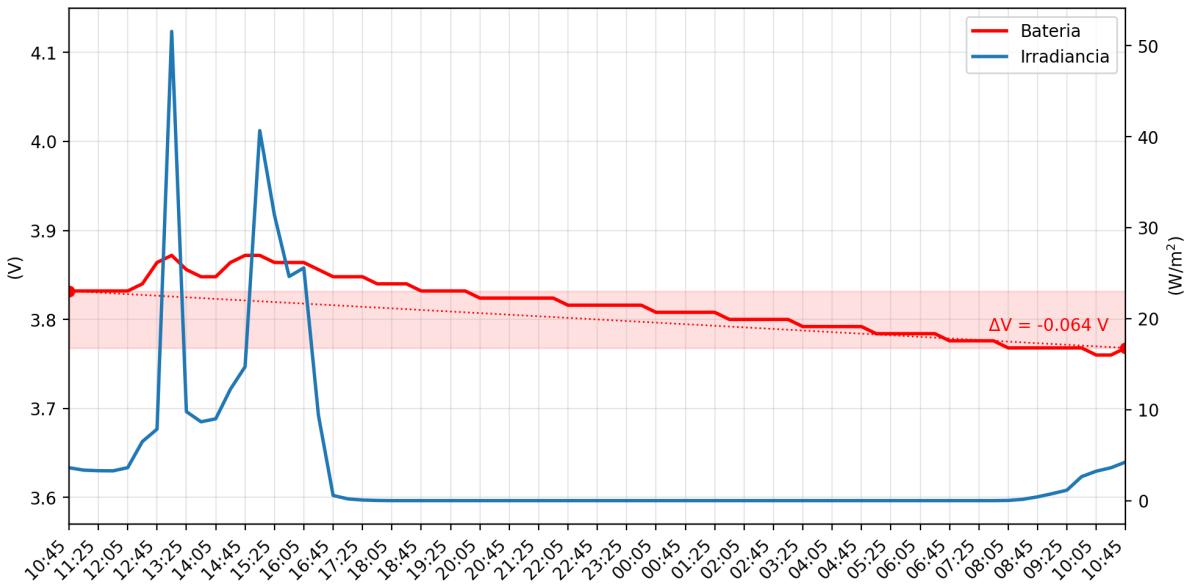


Figura 65: Evolución de la tensión de batería e irradiancia estimada durante 24 h.
Elaboración propia.

En primer lugar, se observa que la tensión final es inferior a la inicial. La tensión de batería al inicio del ensayo fue de $V_{\text{ini}} = 3,832 \text{ V}$ y al final de $V_{\text{fin}} = 3,768 \text{ V}$, lo que corresponde a una caída neta de $\Delta V = -0,064 \text{ V} = -64 \text{ mV}$ en 24 horas.

Esta descarga neta está fuertemente influenciada por las condiciones desfavorables del experimento (interior, invierno y cielo nublado). Aun así, el comportamiento durante los picos de irradiancia muestra que la batería es capaz de recuperar tensión con rapidez: el máximo de irradiancia registrado fue de 51.6 W/m^2 a las 13:05, coincidiendo con un máximo de batería de $V_{\text{max}} = 3,872 \text{ V}$. Si niveles de irradiancia similares se mantuvieran durante un intervalo más prolongado, como ocurriría en una instalación real en exterior, sería esperable una carga diaria mayor y, por tanto, un balance energético más favorable.

Adicionalmente, se aprecia que a partir de las 18:05 la irradiancia pasa a 0 W/m^2 y se mantiene prácticamente nula durante la noche, intervalo en el que la tensión desciende de forma progresiva. Cabe recalcar que los valores de irradiancia del ensayo son reducidos en comparación con magnitudes típicas en exterior incluso en días nublados, los cuales podrían rondar los 100 W/m^2 en estas mismas condiciones.

Aun con estas condiciones, la caída de tensión observada es baja: 64 mV en 24 h. Si se realizase una extrapolación lineal, considerando una batería totalmente cargada en torno a $V_{\text{OVCH}} \approx 4,12 \text{ V}$ y un umbral de descarga $V_{\text{OVDIS}} = 3,6 \text{ V}$, el tiempo para alcanzar dicho umbral sería:

$$t \approx \frac{V_{OVCH} - V_{OVDIS}}{0,064} \approx \frac{4,12 - 3,6}{0,064} \approx 8,1 \text{ dias}$$

En conclusión, aunque el ensayo presenta un balance neto ligeramente negativo en un escenario conservador, el comportamiento observado sugiere que en un entorno real el sistema debería mejorar de forma significativa su balance energético y, por tanto, su autonomía efectiva.

13.3. Pruebas de sensores

A continuación se realizaron una serie de experimentos para evaluar el comportamiento de los sensores integrados en el prototipo.

Consistencia de temperatura y humedad durante 24 h

Con el objetivo de validar la coherencia de las medidas en funcionamiento continuo, se reutilizaron los datos registrados durante el experimento de autonomía. En este ensayo, ambos sensores de temperatura (SHT31 y DS18B20) se encontraban midiendo en aire, por lo que sus lecturas deberían ser comparables. Como se observa en la Figura 66, ambas curvas siguen una tendencia muy similar durante todo el periodo, con diferencias pequeñas, lo que constituye una buena señal del correcto funcionamiento de ambos sensores y de su calibración relativa.

En cuanto a la humedad relativa medida por el SHT31, se mantiene estable dentro de un rango acotado, oscilando entre el 37 % y el 46 % durante las 24 horas. Además, se aprecia un comportamiento coherente con el ciclo día y noche: la humedad relativa tiende a ser menor durante las horas de mayor iluminación y aumenta ligeramente durante la noche.

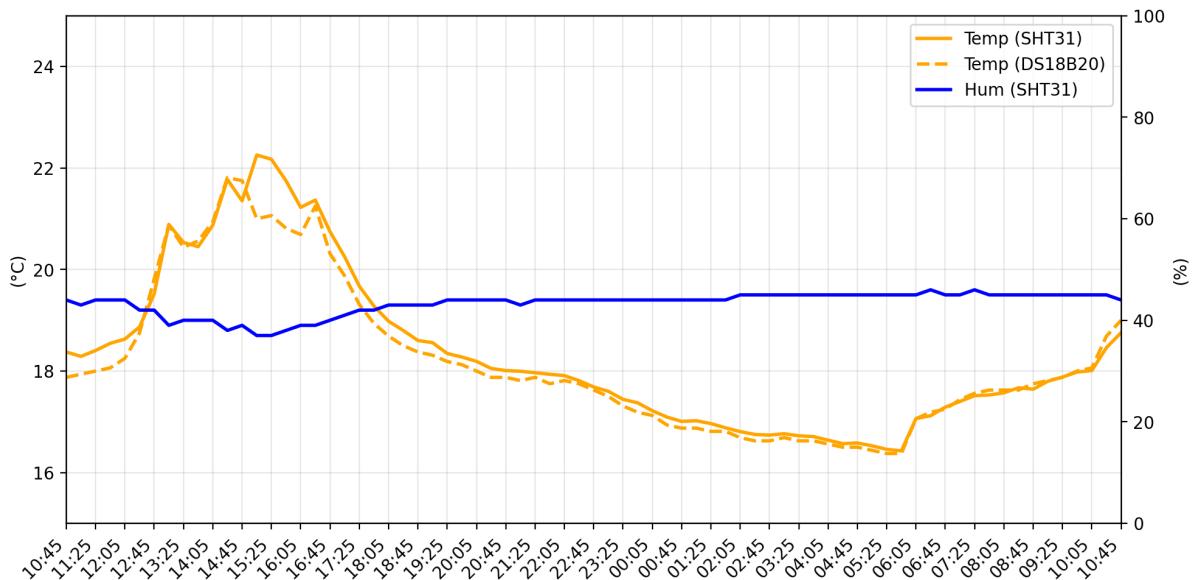


Figura 66: Evolución de temperatura y humedad relativa durante el ensayo de 24 h.
Elaboración propia.

Pruebas de humedad relativa ante cambios bruscos

Además del análisis a 24 h, se realizó una prueba específica para evaluar la respuesta temporal del sensor de humedad relativa (SHT31) ante cambios rápidos del entorno. Para ello se configuró un muestreo rápido, tomando una medida cada 2 s, y se generaron dos incrementos forzados de humedad aplicando vapor de agua cerca del sensor.

En la Figura 67 se observa que el sensor responde de forma clara ante ambos estímulos, alcanzando valores cercanos a saturación y posteriormente recuperándose hacia el nivel base. La prueba se repitió dos veces y se aprecia un comportamiento distinto en la fase de recuperación: en el primer pico la humedad decrece de forma más lenta, mientras que en el segundo la caída es considerablemente más rápida. Esta diferencia se atribuye a las condiciones de ventilación: durante el primer ensayo apenas había circulación de aire, mientras que en el segundo sí existía ventilación, acelerando la renovación del aire y, por tanto, la recuperación de la humedad relativa hacia el valor inicial.

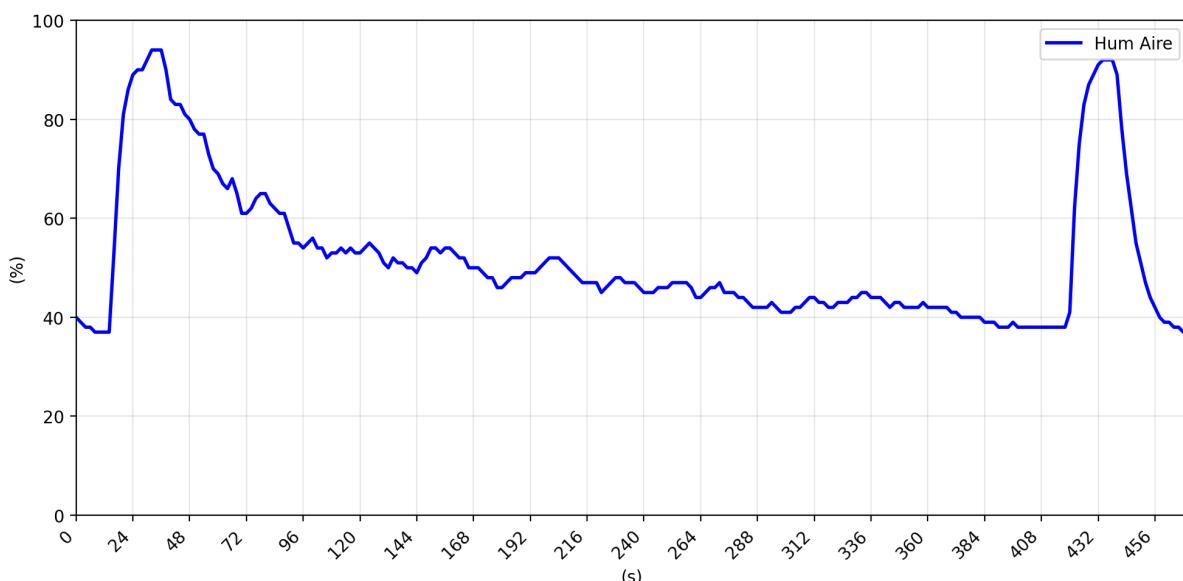


Figura 67: *Respuesta temporal del sensor de humedad relativa ante dos picos forzados de humedad. Elaboración propia.*

Pruebas de humedad en terreno

humedad del suelo: se utiliza una maceta donde se clava el sensor. al principio esta seca y se deposita un poco de agua para comprobar que muestra bien la variacion. la calibracion del sensor se hace midiendo en aire y despues completamente sumergido en agua con lo que nos da el valor de 0 % y 100 % humedo, a partir de aqui se extrae el valor medido.

13.4. Pruebas de conexión inalámbrica

Debido a limitaciones de tiempo, no se integró la funcionalidad BLE completa en el *firmware* final del prototipo. No obstante, se validó el correcto funcionamiento del subsistema inalámbrico del microcontrolador mediante ejemplos de código proporcionados por STMicroelectronics, en concreto se uso el `BLE_p2pServer`. Este *firmware* de demostración implementa un servicio BLE sencillo de tipo *peer-to-peer*, con características para el intercambio básico de datos entre el dispositivo y un *cliente* (en nuestro caso un teléfono móvil), permitiendo comprobar de forma aislada la pila BLE y el funcionamiento de la radio.

Durante la validación se comprobó que el dispositivo:

- Inicia correctamente el *advertising* y es detectable desde un *scanner* BLE.
- Permite establecer conexión de manera estable.
- Permite el intercambio de datos básico mediante el servicio *p2p* (lectura/escritura/notify).

Como resultado, se considera validada la operatividad del *hardware* de radio y la viabilidad de incorporar en el futuro el volcado de datos almacenados en FRAM mediante un servicio BLE, quedando dicha integración fuera del alcance del presente trabajo.

Capítulo 14

Conclusiones y trabajo futuro

Bibliografía

- [1] FIB. *Facultad de Informática de Barcelona.*
<https://www.fib.upc.edu/>
- [2] UPC. *Portal de la Universitat Politècnica de Catalunya.*
<https://www.upc.edu/>
- [3] Domo21.
<https://www.sicma21.com/>
- [4] Wikipedia Commons. *Abstract Kanban Board.*
https://commons.wikimedia.org/wiki/File:Abstract_Kanban_Board.svg
- [5] GitHub.
<https://github.com/>
- [6] Trello.
<https://trello.com/>
- [7] STMicroelectronics. *STM32WB55RG / Product.*
<https://www.st.com/en/microcontrollers-microprocessors/stm32wb55rg.html>
- [8] STMicroelectronics. *NUCLEO-WB55RG / Product.*
<https://www.st.com/en/evaluation-tools/nucleo-wb55rg.html>
- [9] EVOLUX. *Lux vs W/m².*
<https://evolux.cl/lux-vs-wm2>
- [10] Application Note AN2867. *Guidelines for oscillator design on STM8AF/AL/S and STM32 MCUs/MPUs.*
https://www.st.com/resource/en/application_note/an2867-guidelines-for-oscillator-design-on-stm8afals-and-stm32-mcusmpus-stmicroelectronics.pdf
- [11] Application Note AN5042. *How to calibrate the HSE clock for RF applications on STM32 wireless MCUs.*
https://www.st.com/resource/en/application_note/an5042-how-to-calibrate-the-hse-clock-for-rf-applications-on-stm32-wireless-mcus.pdf

- [12] Application Note AN5129. *Guidelines for meander design using low-cost PCB antennae with 2.4 GHz radio for STM32WB/WB0 MCUs.*
https://www.st.com/resource/en/application_note/an5129-guidelines-for-meander-design-using-lowcost-pcb-antennae-with-24-ghz-radio.pdf
- [13] Application Note AN6335. *MLPF-WB and MLPF-NRG PCB design guidelines and reference hardware.*
https://www.st.com/resource/en/application_note/an6335-mlpfwb-and-mlpfnrg-pcb-design-guidelines-and-reference-hardware-stmicroelectronics.pdf
- [14] Application Note SPMA056. *System Design Guidelines for the TM4C129x Family of Tiva™ C Series Microcontrollers.*
<https://www.ti.com/lit/an/spma056/spma056.pdf?ts=1767867045202>
- [15] Arm Mbed. *NUCLEO-WB55RG.*
<https://os.mbed.com/platforms/ST-Nucleo-WB55RG/>
- [16] JLCPCB. *Controlled Impedance Calculator.*
<https://jlcpcb.com/pcb-impedance-calculator>
- [17] JLCPCB. *Assembly Parts Library.*
<https://jlcpcb.com/parts/>
- [18] GitHub. *STMicroelectronics / STM32CubeWB.*
https://github.com/STMicroelectronics/STM32CubeWB/blob/master/Projects/STM32WB_Copro_Wireless_Binaries/STM32WB5x/stm32wb5x_BLE_Stack_full_fw.bin