

# Desarrollo de un sistema de posicionamiento en interiores mediante Bluetooth Low Energy

Alejandro Linde Cerezo  
Máster Universitario en Ingeniería de Telecomunicación  
Universitat Oberta de Catalunya

Consultor: Dr. José López Vicario  
Profesor: Dr. Pere Tuset Peiró

06 / 2024

© Alejandro Linde Cerezo

Reservados todos los derechos. Está prohibido la reproducción total o parcial de esta obra por cualquier medio o procedimiento, comprendidos la impresión, la reprografía, el microfilme, el tratamiento informático o cualquier otro sistema, así como la distribución de ejemplares mediante alquiler y préstamo, sin la autorización escrita del autor o de los límites que autorice la Ley de Propiedad Intelectual.

## FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Desarrollo de un sistema de posicionamiento en interiores mediante Bluetooth Low Energy</i>
Nombre del autor:	<i>Alejandro Linde Cerezo</i>
Nombre del consultor:	<i>José López Vicario</i>
Nombre del PRA:	<i>Pere Tuset Peiró</i>
Fecha de entrega (mm/aaaa):	06/2024
Titulación:	<i>Máster Universitario en Ingeniería de Telecomunicación</i>
Área del trabajo:	<i>Telemática</i>
Idioma del trabajo:	<i>Español</i>
Palabras clave	<i>Bluetooth Low Energy, BLE, IPS, RSSI, fingerprinting, multilateración</i>

### Resumen del Trabajo

Este Trabajo Final de Máster presenta el desarrollo de un sistema de posicionamiento en interiores de bajo presupuesto utilizando la tecnología Bluetooth Low Energy (BLE) y técnicas en el estado del arte. El sistema propone un esquema de procesamiento con un kernel de posicionamiento separado de un filtro de partículas que mejora la precisión de la localización. Las técnicas utilizadas como kernel son la multilateración y el fingerprinting utilizando medidas de RSSI, las cuales se filtran previamente con un filtro de Kalman. Adicionalmente, el sistema ofrece un servicio de posicionamiento semántico basado en un clasificador por Máquina de Soporte Vectorial (SVM).

El sistema se ha desarrollado como un servicio web, lo que facilita su integración, escalabilidad y mantenibilidad. Las pruebas se realizaron desplegando el sistema con balizas BLE basadas en el microcontrolador ESP32, así como dos terminales Android, en dos escenarios diferentes, a saber: en Sala (37 m<sup>2</sup>), con línea de visión directa entre las balizas y el terminal, y en Planta (135 m<sup>2</sup>), sin línea de visión directa.

Los resultados obtenidos muestran que el prototipo del sistema puede operar con un error de localización de aproximadamente 1.5 metros y una precisión promedio al clasificar la estancia del 91% en entornos domésticos. Estas prestaciones validan el diseño en base a los objetivos del proyecto.

En conclusión, este trabajo demuestra cómo una tecnología tan extendida como BLE puede emplearse eficazmente para aplicaciones de posicionamiento en interiores con un bajo coste de desarrollo.

## Abstract

The present Master Thesis describes the development of a low-cost Indoor Positioning System based on the technology Bluetooth Low Energy and state-of-the-art positioning techniques. The system proposes a processing scheme that separates the positioning kernel from a particle filter, which enhances the location precision. The techniques used as positioning kernel are multilateration and fingerprinting based on RSSI measurements, which are filtered previously by a Kalman filter. In addition, the system provides a semantic positioning service based on a Support Vector Machine (SVM) classifier.

The system has been designed as a web service, facilitating its integration, scalability, and maintainability. Testing was carried out by deploying the system with BLE beacons based on the microcontroller ESP32 as well as two Android terminals, across two different scenarios, namely: a Room (37 m<sup>2</sup>), with line-of-sight between the beacons and the terminal, and a Floor (135 m<sup>2</sup>), without line-of-sight.

The yielded results demonstrate that the system prototype works with a location error of approximately 1.5 meters and an average room classification precision of 91% in domestic environments. This performance validates the design according to the project's objectives.

In conclusion, this thesis showcases how such a widely spread technology like BLE can be effectively used for indoor positioning applications with a low development cost.

A Irene, a Elías.

# Índice

1. Introducción.....	11
1.1. Contexto y justificación del trabajo.....	11
1.2. Objetivos del trabajo.....	11
1.3. Enfoque y método seguido.....	12
1.4. Planificación del trabajo.....	13
1.5. Breve resumen de productos obtenidos.....	14
1.6. Impacto en sostenibilidad.....	15
1.7. Breve descripción de los capítulos de la memoria.....	15
2. Estado del arte.....	16
2.1. BLE como tecnología de radiolocalización.....	16
2.1.1. El protocolo <i>iBeacon</i> .....	18
2.2. Técnicas de posicionamiento.....	19
2.2.1. Métricas de distancia.....	19
2.2.1.1. RSSI – Indicador de intensidad de señal recibida.....	19
2.2.1.2. ToF – Tiempo de vuelo.....	20
2.2.1.3. TDoA – Diferencia de tiempo de llegada.....	22
2.2.2. Técnicas basadas en multilateración.....	22
2.2.3. Técnicas basadas en <i>fingerprinting</i> .....	23
2.2.4. Comparación entre la multilateración y el <i>fingerprinting</i> .....	25
2.3. Algoritmos de seguimiento.....	26
2.3.1. Algoritmos basados en el filtro de Kalman.....	26
2.3.2. Algoritmos basados en filtros de partículas.....	27
3. Diseño y desarrollo del sistema.....	31
3.1. Arquitectura del sistema.....	31
3.2. Elementos hardware.....	33
3.3. Elementos software.....	34
3.3.1. Aplicación web.....	34
3.3.2. Aplicación Android.....	34
3.3.3. Aplicación de escritorio.....	36
3.3.4. Programación de las balizas.....	37
3.3.5. Limitaciones técnicas.....	37
3.3.5.1. API de BLE en Android.....	38
3.3.5.2. Librería NimBLE y estándar Bluetooth.....	39
3.4. Algoritmo del sistema de posicionamiento.....	40
3.4.1. Ajuste de los parámetros del algoritmo.....	42
3.4.1.1. Ajuste del filtro de partículas.....	42
3.4.1.2. Ajuste del filtro de Kalman.....	44
3.5. Algoritmo de posicionamiento semántico.....	45
4. Despliegue del sistema en sala.....	46
4.1. Escenario.....	46
4.2. Campaña de adquisición.....	47
4.3. Procesamiento de los datos.....	49
4.3.1. Calibración.....	49
4.3.2. Fingerprints.....	50
4.3.3. Error de localización.....	50
4.3.3.1. Despliegue en sala con menor número de balizas.....	53
4.3.4. Seguimiento del terminal.....	53
5. Despliegue del sistema en planta.....	54

5.1. Escenario .....	54
5.2. Campaña de adquisición .....	55
5.3. Procesamiento de los datos.....	56
5.3.1. Calibración.....	58
5.3.2. Fingerprints.....	59
5.3.3. Error de localización .....	59
5.3.4. Posicionamiento semántico.....	62
5.3.5. Seguimiento del terminal.....	63
6. Resultados y validación del sistema .....	65
7. Conclusiones y trabajo futuro .....	66
8. Bibliografía .....	67
Anexo A. Estudio de viabilidad.....	69
Anexo B. Repositorios.....	72

## Lista de figuras

Figura 1.1 – Estructura de Desglose de Trabajo.....	13
Figura 1.2 – Cronograma del proyecto.....	14
Figura 2.1 – Canales BLE.....	16
Figura 2.2 – Secuencia de tramas en el intervalo de advertisement .....	17
Figura 2.3 – Esquema de técnicas de posicionamiento.....	19
Figura 2.4 – Tiempo de vuelo (ToF) con instantes de ida y vuelta.....	21
Figura 2.5 – Trilateración.....	23
Figura 2.6 – Esquema de fingerprinting.....	24
Figura 2.7 – Algoritmo recursivo del filtro de Kalman.....	27
Figura 2.8 – Funcionamiento del filtro de partículas.....	30
Figura 3.1 – Diagrama conceptual de la arquitectura del sistema.....	31
Figura 3.2 – Diagrama de componente (modelo C4) de la arquitectura del sistema .....	32
Figura 3.3 – Placas ESP32 Dev-Kit C con BLE 4.2 (izquierda) y ESP32-C3 Dev-Kit M con BLE 5.0 (derecha) .....	33
Figura 3.4 – Endpoints de la API REST del servicio web.....	34
Figura 3.5 – Interfaz de usuario de la aplicación Android.....	35
Figura 3.6 – Diagrama de casos de uso de la aplicación Android.....	36
Figura 3.7 – Interfaz de usuario de la herramienta de escritorio .....	36
Figura 3.8 – Diagrama de clases de las balizas BLE .....	37
Figura 3.9 – Diagrama de secuencia del mecanismo de escaneo de advertisement BLE.....	38
Figura 3.10 – Algoritmo de posicionamiento.....	41
Figura 3.11 – Trayectoria estimada del filtro de partículas con datos simulados...	42
Figura 3.12 – Comparación de la estimación del filtro de partículas en diferentes instantes.....	43
Figura 3.13 – Señal RSSI filtrada por Kalman .....	44
Figura 3.14 – Señal RSSI a 1m en los canales BLE (izquierda); tasa de paquetes recibidos (derecha) .....	44
Figura 3.15 – Algoritmo de posicionamiento semántico.....	45
Figura 4.1 – Escenario “Sala” con las ubicaciones candidatas para las balizas. A la derecha, el área cubierta por línea de visión para la constelación seleccionada (B-D-E-H-L).....	46
Figura 4.2 – Imágenes de las balizas durante el experimento.....	48
Figura 4.3 – Distribución de fingerprints en Sala (izquierda) y de blancos estáticos (derecha) .....	48
Figura 4.4 – Mapa de RSSI en Sala con datos reales (izquierda) y simulados (derecha) .....	49
Figura 4.5 – Regresión lineal en Sala del modelo logarítmico de propagación.....	49
Figura 4.6 – Señales RSSI de dos puntos diferentes en Sala .....	50
Figura 4.7 – Distribución del error en Sala para ambos terminales.....	52
Figura 4.8 – Estimaciones en Sala para el Terminal 1 con el mejor caso (izquierda) y el peor (derecha).....	52
Figura 4.9 – Estimaciones en Sala para el Terminal 2 con el mejor caso (izquierda) y el peor (derecha).....	52
Figura 4.10 – Distribución del error en Sala para 3, 4 y 5 balizas .....	53



Figura 4.11 – Mejora de la precisión con el filtro de partículas.....	53
Figura 5.1 – Escenario “Planta” con las ubicaciones de las balizas. A la derecha, las líneas de visión.....	54
Figura 5.2 – Distribución de fingerprints en Planta (izquierda) y de blancos estáticos (derecha).....	55
Figura 5.3 – Mapa de RSSI en Planta con datos reales (izquierda) y simulados (derecha).....	56
Figura 5.4 – Tasa de paquetes recibidos (izquierda) y valores de RSSI promedio (derecha) de Beacon 1.....	57
Figura 5.5 – Regresión lineal en Planta del modelo logarítmico de propagación....	58
Figura 5.6 – Regresión lineal por cada estancia del escenario.....	59
Figura 5.7 – Señales RSSI de dos puntos diferentes en Planta.....	59
Figura 5.8 – Distribución del error en Planta para ambos terminales.....	61
Figura 5.9 – Estimaciones en Planta para el Terminal 1 con el mejor caso (izquierda) y el peor (derecha).....	61
Figura 5.10 – Matriz de confusión del Terminal 1.....	62
Figura 5.11 – Matriz de confusión del Terminal 2.....	63
Figura 5.12 – Seguimiento del terminal en dos trayectorias: del dormitorio 2 a la cocina (izquierda), y del salón al dormitorio 1 (derecha)......	64
Figura 5.13 – Seguimiento del terminal en la trayectoria de la oficina al baño. A la izquierda, con tiempo de integración de 500 ms. A la derecha, con 1 s.....	64
Figura 6.1 –Latencia promedio del servicio de posicionamiento.....	65

## Lista de tablas

Tabla 1.1– Objetivos del proyecto.....	12
Tabla 2.1– Payload del protocolo iBeacon.....	18
Tabla 3.1 – Desviación típica de los valores de RSSI.....	45
Tabla 4.1 – Coordenadas de las ubicaciones en Sala.....	46
Tabla 4.2 – Información de las balizas desplegadas en Sala.....	47
Tabla 4.3 – Calibración para la multilateración en Sala.....	50
Tabla 4.4 – Errores de localización en Sala para el Terminal 1.....	51
Tabla 4.5 – Errores de localización en Sala para el Terminal 2.....	51
Tabla 5.1 – Coordenadas de las ubicaciones en Planta.....	54
Tabla 5.2 – Información de las balizas desplegadas en Planta.....	55
Tabla 5.3 – Valores de RSSI de respaldo por cada baliza en Planta.....	57
Tabla 5.4 – Calibración para la multilateración en Planta.....	58
Tabla 5.5 – Errores de localización en Planta para el Terminal 1.....	60
Tabla 5.6 – Errores de localización en Planta para el Terminal 2.....	60
Tabla 5.7 – Resultados de clasificación de las estancias para ambos terminales.....	62
Tabla 6.1 – Resultados cotejados con los objetivos del proyecto.....	65

# 1. Introducción

## 1.1. Contexto y justificación del trabajo

La proliferación de dispositivos IoT (Internet of Things) demanda cada vez más servicios de localización y navegación en entornos domésticos, sanitarios, comerciales e industriales. Bluetooth Low Energy es un estándar de comunicaciones WPAN (Wireless Personal Area Network) de bajo consumo muy extendido en todo tipo de sensores y terminales móviles, lo que hace atractivo su empleo como infraestructura para sistemas de posicionamiento en interiores. Prueba de ello es la creación en la última década de protocolos como *iBeacon* de Apple [14], o *Eddystone* de Google [18], que proporcionan capacidades de localización sobre Bluetooth Low Energy.

Este trabajo se propone diseñar y desarrollar un sistema de localización en tiempo real que utilice la tecnología Bluetooth Low Energy, así como mecanismos de posicionamiento en el estado del arte. Esto implica el uso de medidas de RSSI (Received Signal Strength Indicator) para la estimación de la posición junto con algoritmos de seguimiento, tales como, por ejemplo, aquellos basados en filtros de partículas.

## 1.2. Objetivos del trabajo

Los objetivos principales del proyecto son los siguientes:

- Desarrollo de un RTLS (Real-Time Location System) que permita el posicionamiento en interiores con un error promedio absoluto (MAE) de 2,5 metros para terminales estáticos.
- Incorporación de métricas de tiempo, tales como ToF (Time of Flight), para mejorar la precisión típica de los sistemas basados en RSSI mediante fusión de datos.

Los objetivos secundarios del proyecto son los siguientes:

- Producción de datasets mediante experimentación en escenarios reales.
- Obtención de una latencia en el servicio de posicionamiento en tiempo real menor a 100 ms (éste es el intervalo de *advertisement* típico con balizas BLE - ver sección 2.1). Entendemos por latencia del servicio el tiempo que transcurre desde que la solicitud de posicionamiento llega al servidor hasta que la respuesta es enviada.
- Posicionamiento semántico de los terminales con un error de clasificación inferior al 20%. El posicionamiento semántico consiste en proporcionar información cualitativa de la ubicación del terminal en tiempo real. La precisión típica con BLE permite determinar en qué habitación o estancia se encuentra el terminal con precisiones superiores al 80%.
- Desarrollo con tecnologías y plataformas estándar, para mantener así la implementabilidad real del sistema masivamente en producción. Algunas

plataformas y librerías objetivo son: API BLE de Android, el software *stack* NimBLE de Apache, interfaces HTTP entre componentes.

Los criterios de cumplimiento de los objetivos, así como el método de verificación, se resumen en la tabla a continuación. Los objetivos que se verifican mediante testeo disponen de un valor umbral de cumplimiento. Los objetivos que se verifican mediante inspección no son cuantificables.

Objetivo	Método de verificación	Umbral
Error del RTLS	Testeo	< 2,5 m
Uso de ToF	Inspección	-
Producción de datasets	Inspección	-
Latencia	Testeo	< 100 ms
Error de clasificación	Testeo	< 20 %
Tecnologías estándar	Inspección	-

Tabla 1.1– Objetivos del proyecto

El error de localización se medirá mediante la métrica MAE (Mean Absolute Error):

$$MAE = \frac{1}{n} \cdot \sum_{i=0}^{n-1} |x_i - \hat{x}_i|$$

Otra métrica muy extendida en sistemas de posicionamiento es RMSE (Root Mean Squared Error), la cual penaliza más los errores de mayor magnitud.

$$RMSE = \sqrt{\sum_{i=0}^{n-1} \frac{(x_i - \hat{x}_i)^2}{n}}$$

En este proyecto consideramos que MAE tiene mayor representatividad, dado que los mayores errores de localización en interiores se suelen corresponder con las ubicaciones menos transitadas, las cuales tienen peor cobertura por diseño.

### 1.3. Enfoque y método seguido

El enfoque de este trabajo es principalmente industrial o profesionalizador. No se trata de un trabajo de investigación, si bien se han llevado a cabo puntualmente investigaciones de cara a mejorar el error de localización en el sistema.

El método seguido ha consistido en dividir el trabajo en tres fases diferenciadas, a saber:

1. Fase de planificación, evaluación de la viabilidad y análisis de las capacidades técnicas, donde se han determinado los objetivos del proyecto.
2. Fase de desarrollo del sistema, con iteraciones completas que permitan desplegar y probar la funcionalidad en el sistema final.

3. Fase de recopilación de resultados y verificación de los objetivos del proyecto.

El desarrollo del sistema ha combinado la metodología en V (también conocida como *waterfall* o desarrollo en cascada) a nivel de sistema, con una metodología ágil a nivel de software basada en iteraciones cortas que cubran desde la concepción de una funcionalidad hasta su despliegue en el hardware objetivo. En este contexto, cada tarea de desarrollo software presentada en el siguiente apartado se ha llevado a cabo con múltiples iteraciones que comprenden las siguientes subtareas: prueba de concepto, diseño, implementación (en algunos casos, con tests unitarios), despliegue y testeo en el sistema final.

## 1.4. Planificación del trabajo

Para la ejecución de este proyecto se ha dispuesto de 14 semanas (desde el 1 de marzo hasta el 10 de junio) con las siguientes entregas:

- PEC 1 – 05/03/2024: propuesta de proyecto.
- PEC 2 – 02/04/2024: presentación del estado del arte, estudio de viabilidad, diseño preliminar y planificación inicial.
- PEC 3 – 07/05/2024: entrega parcial de la memoria del TFM para revisión.
- Final – 10/06/2024: entrega de la memoria del TFM.

El trabajo se ha llevado a cabo con el siguiente desglose de tareas:

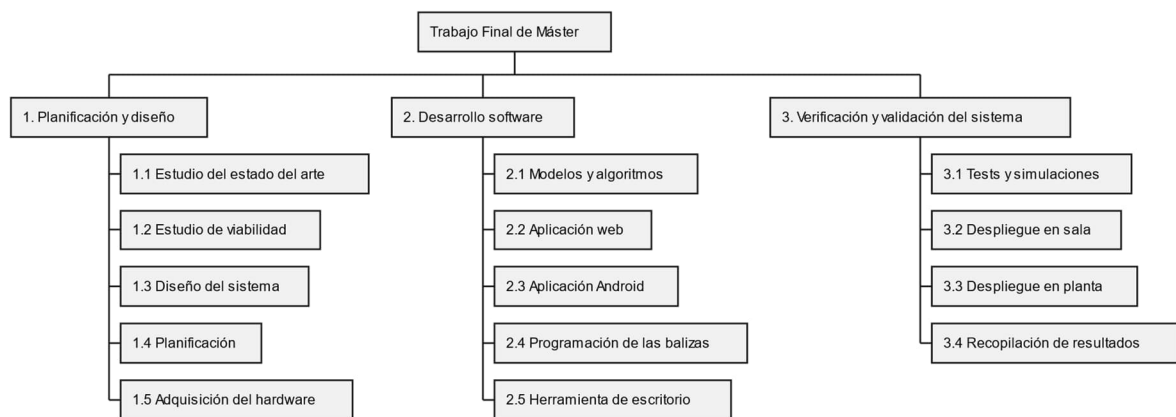


Figura 1.1 – Estructura de Desglose de Trabajo

El cronograma mostrado a continuación representa la asignación de tareas en el tiempo. El tiempo se representa en semanas de trabajo. Se estima una jornada a tiempo parcial (20 h/semana) para la realización del proyecto, lo que supone un total de 280 horas de trabajo.

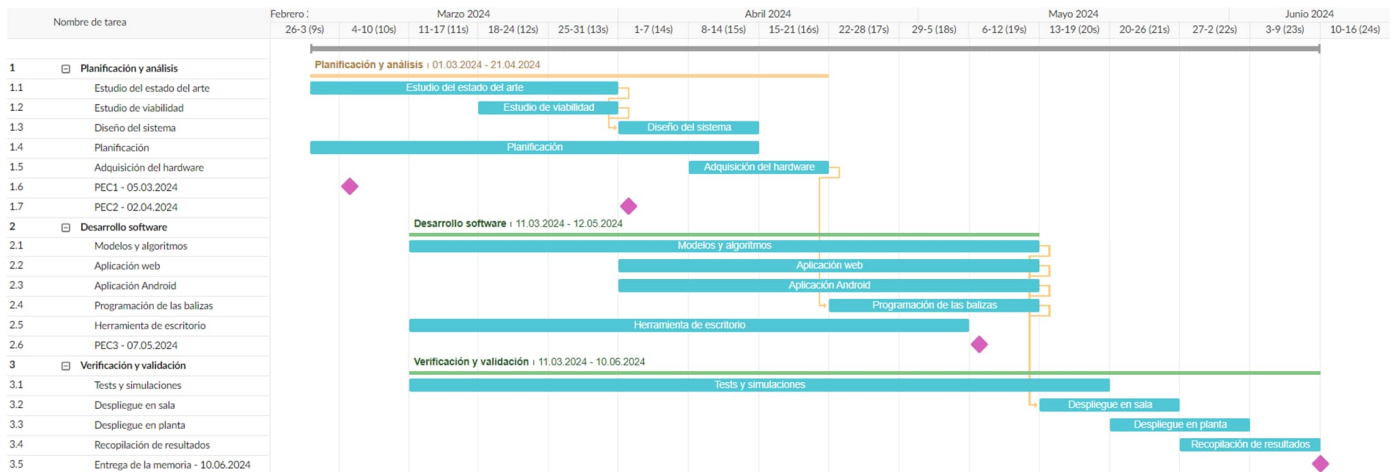


Figura 1.2 – Cronograma del proyecto

## 1.5. Breve resumen de productos obtenidos

Los siguientes elementos son productos del proyecto:

- Memoria del Trabajo Final de Máster.
- Librería software en Python con los siguientes modelos y algoritmos:
  - Algoritmo de multilateración basado en mínimos cuadrados ponderados.
  - Algoritmo de *fingerprinting* basado en k-NN y en una teselación multidimensional por Delaunay.
  - Filtro de partículas.
  - Filtro de Kalman unidimensional.
  - Clasificador por k-NN.
  - Clasificador por SVM.
  - Modelo de propagación *Log-Distance Path Loss*.
  - Modelo de *raytracing* 2D para la simulación de líneas de visión.
- Datasets experimentales con datos reales de RSSI.
- Servicios web de adquisición de datos y posicionamiento con API REST.
- Herramienta de escritorio para la simulación y configuración del sistema.
- Código fuente de las balizas BLE.
- Aplicación Android para adquisición de datos y posicionamiento.

## 1.6. Impacto en sostenibilidad

Este trabajo se encuadra con los siguientes objetivos de desarrollo sostenible:

- ODS 9 – Industry, innovation, and infrastructure: el uso de sistemas de posicionamiento contribuye a la creación de infraestructuras digitales avanzadas (centros comerciales, industriales, ...). Además, permite su explotación a todo tipo de agentes económicos debido a su bajo coste de implementación.
- ODS 11 – Sustainable cities and communities: promueve la mejora de la movilidad en ciudades inteligentes. El consumo energético del sistema es reducido, y puede ayudar a optimizar la climatización y el alumbrado público mediante el control automático de afluencia.

## 1.7. Breve descripción de los capítulos de la memoria

La memoria de este Trabajo Final de Máster se estructura en los siguientes capítulos principales:

- Estado del arte: introducción al uso de Bluetooth Low Energy en sistemas de posicionamiento en interiores y revisión de las técnicas y algoritmos de relevancia para este trabajo.
- Diseño y desarrollo del sistema: descripción de la arquitectura del sistema de posicionamiento desarrollado. Breve documentación sobre los componentes hardware y software del sistema. Presentación de las limitaciones técnicas encontradas para introducir medidas de tiempo. Descripción de los algoritmos de posicionamiento implementados y ajustes de los parámetros previamente al despliegue del sistema.
- Despliegue del sistema en sala: validación del sistema en un escenario con línea de visión directa: descripción del escenario, simulación y ubicación de las balizas. Presentación de las capturas de datos realizadas y de los resultados en sala.
- Despliegue del sistema en planta: validación del sistema en un escenario de mayor tamaño y sin línea de visión directa: descripción del escenario, simulación y ubicación de las balizas. Presentación de las capturas de datos realizadas y de los resultados en planta.
- Resultados y validación del sistema: evaluación de los resultados finales y verificación de los objetivos del proyecto.
- Conclusiones y trabajo futuro: resumen de los logros del trabajo y comentario sobre posibles líneas de trabajo futuras.

## 2. Estado del arte

### 2.1. BLE como tecnología de radiolocalización

La tecnología Bluetooth Low Energy, a la cual nos referiremos como BLE de ahora en adelante, es un subestándar de Bluetooth pensado para sensores y dispositivos que requieren transmisión discontinua pero frecuente de pequeñas cantidades de datos con muy bajo consumo.

BLE utiliza 40 canales de 2 MHz de ancho de banda en la banda ISM a 2,4 GHz. Los canales del 0 al 36 son de datos, mientras que los canales 37, 38 y 39 (portadoras a 2402, 2426, y 2480 MHz, respectivamente) están reservados para *advertisement* (anuncio de los servicios disponibles en un servidor GATT de Bluetooth). Los canales de *advertisement* están ubicados en el espectro para no interferir con canales WiFi, ya que esta tecnología utiliza normalmente mayor potencia de transmisión.

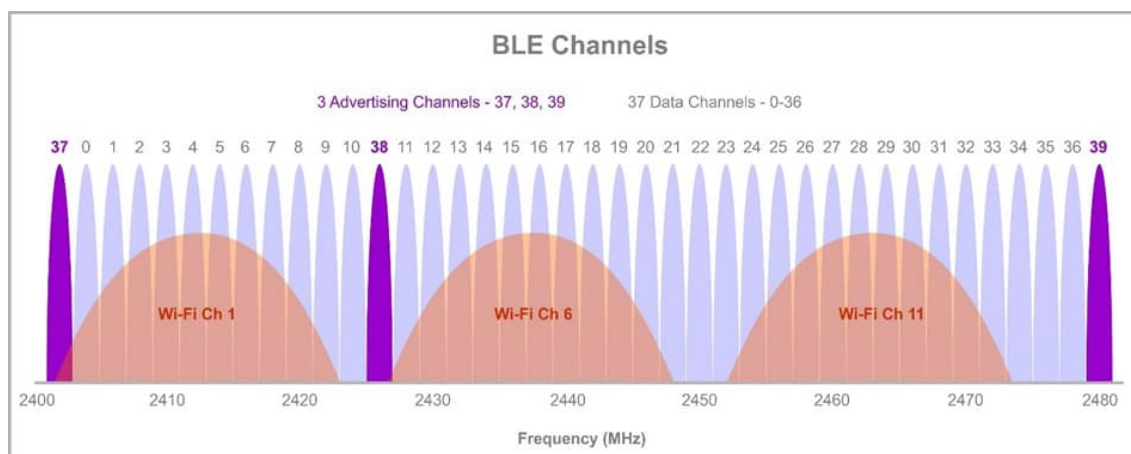


Figura 2.1 – Canales BLE

Si bien BLE es un estándar de comunicación y no de radiolocalización, su proliferación en sensores y dispositivos alimentados por batería (debido tanto a sus prestaciones como al hecho de que puede reutilizar los módulos inalámbricos diseñados para WiFi), lo ha convertido en una opción popular como infraestructura para sistemas de proximidad y de posicionamiento.

Los sistemas de posicionamiento sobre BLE están normalmente basados en el estándar 4.2, pues es la versión de Bluetooth más extendida en el mercado. Estos sistemas no utilizan comúnmente los canales de datos, sino que se centran en implementar una red *broadcast* a través de los canales de *advertisement*. Este es el caso de los dos protocolos más extendidos: *iBeacon* de Apple [14], y *Eddystone* de Google [18]. Bajo este esquema, una serie de pequeños equipos no conectables denominados "balizas" (*beacons* en inglés) llevan a cabo el rol de "dispositivos periféricos" para transmitir de forma continua en la trama de *advertisement* un payload con datos del protocolo de posicionamiento. En este escenario, un terminal que quiera utilizar el servicio de posicionamiento deberá cumplir el rol de "dispositivo central", lo que implica que deberá escanear los canales de



*advertisement* de BLE en busca de tramas con el protocolo deseado. En caso de que el protocolo lo soporte, el terminal podrá enviar una trama *scan request* a la baliza para solicitar datos adicionales (esta posibilidad existe en *Eddystone*, pero no en *iBeacon*). La baliza responderá con dichos datos en una trama *scan response* si está configurada para ello.

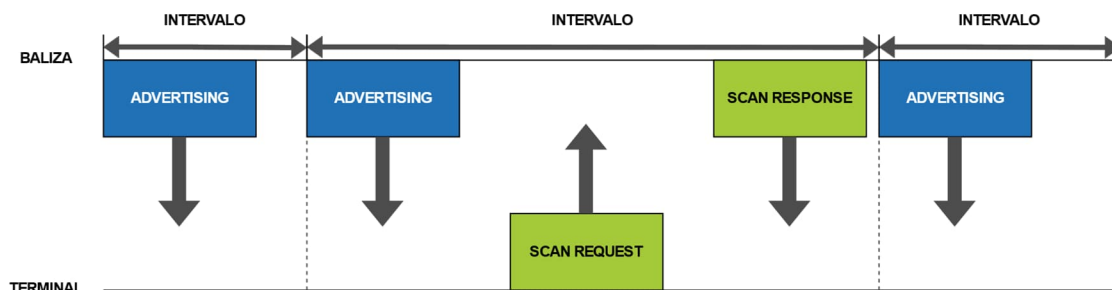


Figura 2.2 – Secuencia de tramas en el intervalo de advertisement

En diciembre de 2016 se publicó el estándar de Bluetooth 5, el cual introduce mejoras en la capa física: duplica la tasa máxima de datos a 2 Mbps y aumenta el alcance hasta varios cientos de metros con línea de visión directa (utilizando potencias de transmisión de hasta 20 dBm). En cuanto a las capacidades de *advertisement*, la especificación BLE 5 introduce las siguientes características:

- *Advertisement* múltiple: envío de múltiples payloads de *advertisement*, por lo que es posible que el mismo dispositivo funcione al mismo tiempo como baliza *iBeacon* y baliza *Eddystone*, por ejemplo.
- *Advertisement* extendido: uso de cualquier canal de datos como canal de advertisement secundario, para transmitir así datos adicionales relacionados con el protocolo (hasta 255 bytes).
- *Advertisement* periódico: envío de la trama de *advertisement* con un intervalo fijo (el intervalo en BLE 4.2 no es constante). Este modo también permite modificar el contenido de la trama de *advertisement* dinámicamente, por lo que se podrían explorar mecanismos basados en ToF si se alcanza la sincronización adecuada en la implementación de las balizas.

Dado que los protocolos de posicionamiento sobre BLE más comunes funcionan con BLE 4.2, puesto que aún hoy en día existe un gran elenco de dispositivos que operan con el estándar antiguo, en este trabajo procuraremos garantizar la compatibilidad con dicha versión. Implementaremos el protocolo *iBeacon* y utilizaremos el modo de compatibilidad para las balizas que implementen el estándar BLE 5 siempre que sea posible.

A pesar de seleccionar *iBeacon* como protocolo por su simplicidad, *Eddystone* o cualquier otro protocolo basado en la métrica de RSSI (Received Signal Strength Indicator) podría emplearse con el sistema propuesto sin impacto alguno en el diseño.

### 2.1.1. El protocolo *iBeacon*

El protocolo *iBeacon* es un protocolo de localización de baja precisión publicado por Apple en 2013 [14] que utiliza BLE como protocolo de la capa de enlace. En concreto, *iBeacon* se basa en medidas de RSSI (sección 2.2.1.1) tomadas en el dispositivo receptor a partir del envío de tramas de *advertisement* desde uno o más dispositivos que actúan como balizas transmisoras. Cada baliza transmisora comparte en la trama el valor de RSSI calibrado a 1 metro de distancia. Esto permite al receptor estimar la distancia en base al modelo de propagación logarítmico. Nótese que, sin información adicional, la estimación ha de asumir el valor para el factor de propagación,  $n$ , de la ecuación (1).

El protocolo *iBeacon* utiliza tramas de *advertisement* del tipo *ADV\_NONCONN\_IND*, esto es, tramas de broadcast que no permiten la conexión ni el escaneo por parte del resto de dispositivos. El payload de la trama se compone de 30 bytes, cuyos campos se muestran en la siguiente tabla:

Byte	Tamaño [Bytes]	Campo	Valor
0	1	Longitud del indicador	0x02
1	1	Tipo de dato / indicador	0x01
2	1	Indicador LE y BR/EDR	0x06
3	1	Longitud de datos (26 bytes)	0x1A
4	1	Tipo – dato del fabricante	0xFF
5	2	Identificación del fabricante	0x4C00
7	1	Subtipo ( <i>iBeacon</i> )	0x02
8	1	Longitud de subtipo	0x15
9	16	UUID de la baliza o aplicación	UUID
25	2	Major	Major
27	2	Minor	Minor
29	1	RSSI a 1 metro	RSSI <sub>0</sub>

Tabla 2.1– Payload del protocolo *iBeacon*

La mayoría de campos son constantes y comunes al protocolo. Los campos que han de adaptarse a cada sistema o aplicación son los listados a continuación:

- UUID: identificador único universal de la baliza (o de la aplicación). En este proyecto utilizamos: 00000000-0000-0000-0000-00000000000x, donde la x se sustituye por el número de la baliza.
- Major: identificador de grupo (p.ej. de una área geográfica). En este proyecto lo mantenemos a 0.
- Minor: identificador de subgrupo (p.ej. de una categoría). En este proyecto lo mantenemos a 0.
- RSSI<sub>0</sub>: calibración específica por cada baliza a 1 metro de distancia.

El protocolo *iBeacon* recomienda un intervalo de transmisión en las balizas de 100 ms, tanto para evitar inundar los canales de *advertisement* como para reducir el consumo de los dispositivos. Sin embargo, se contempla el uso de mayores

frecuencias de *advertisement* para aplicaciones que requieran mejores prestaciones (recordamos que Bluetooth permite un intervalo mínimo de 20 ms).

## 2.2. Técnicas de posicionamiento

Entendemos por técnicas de posicionamiento aquellos algoritmos que proporcionan una estimación de la posición del terminal en base a métricas relacionadas con la infraestructura de radiolocalización. Las métricas más comunes son: RSSI (Received Signal Strength Indicator), ToF (Time of Flight), TDoA (Time Difference of Arrival), y AoA (Angle of Arrival). Las tres primeras métricas permiten obtener una estimación de la distancia del terminal a cada baliza mediante modelos específicos de propagación de la señal. La última proporciona una estimación de la dirección de transmisión de la señal, lo que permite obtener el ángulo entre dos balizas o puntos conocidos. Sin embargo, las medidas de AoA requieren de antenas y circuitería de radiofrecuencia más sofisticados de cara a medir la diferencia de fase de las señales radio, lo cual encarece el coste de los dispositivos.

El esquema mostrado a continuación, el cual no pretende ser exhaustivo, muestra una clasificación de las familias de técnicas de posicionamiento más importantes junto con las métricas utilizadas por las mismas. Se excluyen intencionadamente algunas técnicas menos canónicas como aquellas basadas exclusivamente en métodos bio-inspirados (p.ej. *Migrating Birds Optimization*) y de aprendizaje máquina (p.ej. redes neuronales). En este trabajo utilizamos las técnicas basadas en multilateración y *fingerprinting* mediante el uso de medidas de RSSI.

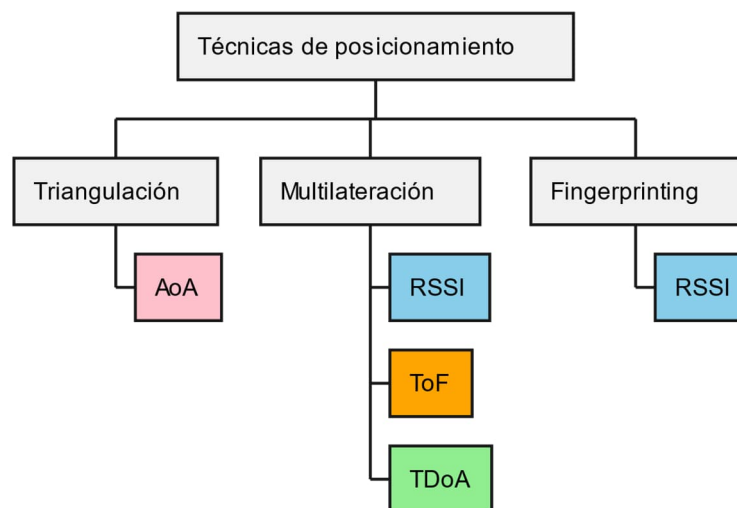


Figura 2.3 – Esquema de técnicas de posicionamiento

### 2.2.1. Métricas de distancia

#### 2.2.1.1. RSSI – Indicador de intensidad de señal recibida

El parámetro RSSI (Received Signal Strength Indicator) es un indicador de la potencia de señal recibida proporcionado por el propio chip de radiofrecuencia de

los dispositivos. En espacio abierto, este indicador decrece monótonamente con la inversa del cuadrado de la distancia con respecto al transmisor. Sin embargo, el valor de RSSI en escenarios *indoor* muestra grandes fluctuaciones debido a efectos como el multi-camino y la presencia de obstáculos, lo que obliga a un tratamiento probabilístico de cara a su estimación.

El modelo de propagación *Log-Distance Path-Loss* es el elegido típicamente para calcular las pérdidas por propagación de las ondas electromagnéticas en interiores. La siguiente ecuación representa el valor de RSSI según dicho modelo:

$$RSSI(d) = RSSI_0 - 10 \cdot n \cdot \log_{10} \left( \frac{d}{d_0} \right) + X_\sigma \quad (1)$$

Donde  $RSSI_0$  representa el valor de RSSI calibrado,  $d_0$  es la distancia de calibración,  $d$  es la distancia entre el transmisor y el receptor,  $n$  es el factor de propagación, y  $X_\sigma$  es una variable aleatoria gaussiana de media cero que introduce principalmente el canal de comunicación debido a efectos de la propagación multicamino como el desvanecimiento. El factor de propagación se encuentra típicamente entre 1 y 5, según la cantidad de obstáculos del escenario. En este proyecto obtendremos este factor empíricamente por cada baliza durante la fase de calibración en cada despliegue del sistema.

En entornos con muros o diferentes pisos en el camino de propagación, se puede utilizar una variante de la ecuación (1) que añade una atenuación por obstáculo [12]:

$$RSSI(d) = RSSI_0 - 10 \cdot n \cdot \log_{10} \left( \frac{d}{d_0} \right) - \sum_{i=0}^{M-1} a_i + X_\sigma \quad (2)$$

En este caso,  $a_i$  hace referencia a la atenuación en decibelios introducida por cada muro. En la práctica, es recomendable obtener dicho valor de atenuación empíricamente, ya que su valor depende del material.

### 2.2.1.2. ToF – Tiempo de vuelo

La métrica de tiempo de vuelo (Time of Flight) hace referencia al tiempo transcurrido desde que el transmisor envía un mensaje hasta que llega al receptor. Existen principalmente dos métodos de estimar el tiempo de vuelo: 1) medir el tiempo de ida y vuelta desde el mismo terminal (RTT, Round-Trip Time) mediante el uso de dos mensajes para calcular posteriormente el ToF, y 2) registrar el tiempo de llegada (Time of Arrival) y obtener una referencia del tiempo de transmisión del propio mensaje (introducida *ex profeso* por el transmisor). En esta segunda opción se necesita obligatoriamente un mecanismo de sincronización o de corrección entre los relojes del transmisor y el receptor.

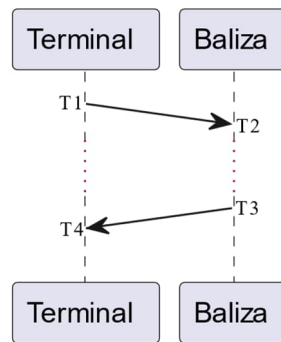


Figura 2.4 – Tiempo de vuelo (ToF) con instantes de ida y vuelta

En el diagrama anterior se puede observar cómo la medida de ToF se puede obtener calculando la diferencia entre T3 y T4, o mediante RTT con la siguiente expresión (asumiendo que la diferencia entre T2, el instante de recepción del primer mensaje en la baliza, y T3, el instante de transmisión del segundo mensaje desde la baliza, no sea negligible):

$$ToF = \frac{(T_4 - T_1) - (T_3 - T_2)}{2} \quad (3)$$

En el caso de Bluetooth, el modo *advertising* introduce un retardo aleatorio de hasta 10 ms antes de transmitir una trama por cualquiera de los canales de *advertising* (a excepción de los modos extendidos de BLE 5). Este retardo sirve para evitar colisiones entre diferentes anunciantes y no puede eludirse en el caso de BLE 4.2 (Bluetooth es un estándar concebido para comunicación, no para posicionamiento). Es por ello que en algunos experimentos se opta por utilizar la ecuación (4) para medir el tiempo de vuelo de la baliza transmisora al terminal. Esta ecuación lineal utiliza, de forma similar al modelo de pérdidas de propagación de RSSI, una constante obtenida por calibración a una distancia conocida. El uso de esta constante permite reducir la dispersión estadística en la medida de ToF en el terminal [6].

$$ToF(d) = ToF_0 + \frac{d - d_0}{c} + X_\sigma \quad (4)$$

En este caso,  $c$  es la velocidad de propagación de la onda electromagnética en el modelo (se recomienda ajustar dicho valor por regresión, y no asumir la velocidad de la luz en el vacío),  $ToF_0$  es el tiempo calibrado a la distancia de referencia  $d_0$ ,  $d$  es nuevamente la distancia, y  $X_\sigma$  es una variable aleatoria que modela un error de incertidumbre que se asume gaussiano, aditivo y de media cero.

No obstante, el principal problema al que nos enfrentamos con el uso de medidas de tiempo en interiores es que la distancia de propagación del orden de metros proporciona tiempos de propagación del orden de nanosegundos, lo cual suele precisar de soluciones específicas a nivel de firmware (en el receptor, el propio retardo de comunicación del controlador inalámbrico a la capa de aplicación a través del *Host Controller Interface* puede ser del orden de milisegundos). A ello se suma la dificultad añadida de que el *stack* software de BLE rara vez ofrece la posibilidad de reaccionar a eventos relacionados con el protocolo desde la capa de

aplicación, por lo que la selección adecuada de la solución tanto a nivel de hardware como del *stack* software se vuelve indispensable para poder incorporar medidas de tiempo. Este trabajo, en concreto, no hace uso de medidas de tiempo debido a una serie de impedimentos técnicos presentados en la [sección 3.3.5](#).

### 2.2.1.3. TDoA – Diferencia de tiempo de llegada

La métrica de TDoA (Time Difference of Arrival) se basa en la diferencia del tiempo de recepción de las señales, las cuales han de ser transmitidas en el mismo instante temporal. Por ello, el modelo geométrico que permite el posicionamiento por TDoA difiere del modelo geométrico utilizado para medidas de ToF (la fusión de ambas medidas suele requerir un sistema de ecuaciones conjunto).

Si bien TDoA no requiere sincronización entre transmisor y receptor, sí que requiere un mecanismo de sincronización entre los transmisores, lo que implicaría comunicación periódica entre las balizas de nuestro sistema. Debido a que las balizas ya hacen uso de su chip de radiofrecuencia para la transmisión de las tramas de *advertisement* de BLE, el uso de TDoA queda descartado en el contexto de este proyecto.

### 2.2.2. Técnicas basadas en multilateración

Las técnicas basadas en multilateración [9] utilizan la distancia de cada baliza al terminal para resolver un sistema de ecuaciones geométricas que permita obtener la localización del terminal. El número mínimo de balizas necesarias para obtener una solución es de tres, en cuyo caso particular hablamos de trilateración.

La estimación de las distancias puede basarse en alguna de las métricas presentadas en la sección anterior. Las ecuaciones geométricas para una distancia estimada mediante RSSI o ToF se muestran a continuación. Consideramos la presencia de tres balizas:  $A$ ,  $B$  y  $C$ , y un sistema de coordenadas bidimensional cartesiano caracterizado por las coordenadas  $x$  e  $y$ . Las distancias al terminal a posicionar se representan mediante  $d$ .

$$\begin{cases} (x - x_A)^2 + (y - y_A)^2 = d_A^2 \\ (x - x_B)^2 + (y - y_B)^2 = d_B^2 \\ (x - x_C)^2 + (y - y_C)^2 = d_C^2 \end{cases} \quad (5)$$

Como puede observarse, cada ecuación se corresponde con la de una esfera. La solución se correspondería, por lo tanto, con el punto de intersección de todas las esferas de no ser porque la estimación de las distancias es ruidosa. Esto proporciona diferentes escenarios en los cuales las circunferencias pueden intersecar en diferentes puntos o incluso no hacerlo en absoluto. En la práctica, se aplica algún método de regresión que estime las coordenadas del terminal mediante minimización o maximización de una función objetivo (p.ej. estimador de máxima verosimilitud).

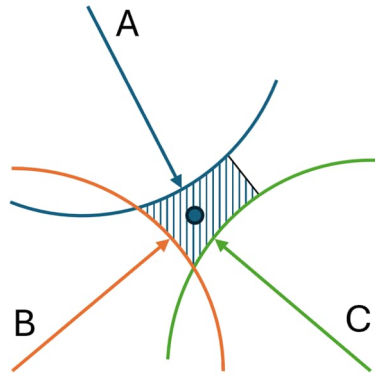


Figura 2.5 – Trilateración

En este trabajo se utiliza el método de mínimos cuadrados ponderados, el cual se enfoca en minimizar iterativamente el módulo del vector de residuos (la suma de los cuadrados de las diferencias entre los valores estimados de distancia y los medidos), pero asumiendo que el error de cada estimación tiende a ser mayor cuanto mayor es la distancia de propagación. Esta asunción de que el error tiende a ser mayor con la distancia a la baliza es utilizada por algunos trabajos para seleccionar un subconjunto de las medidas previamente a resolver el problema geométrico [4]. En este trabajo, sin embargo, utilizaremos todas las medidas de cada baliza disponibles en cada instante de tiempo. Es decir, siempre se realizara multilateración y no “trilateración selectiva”.

El sistema de ecuaciones mostrado en (4) puede modificarse desplazando el origen del sistema de coordenadas a la posición de la baliza A. Con mínimos cuadrados se ha de resolver, para el caso de la trilateración, el siguiente problema de minimización:

$$\min_x \|A \cdot \hat{x} - \vec{b}\| \quad (6)$$

Donde  $A = \begin{bmatrix} 2x_B & 2y_B \\ 2x_C & 2y_C \end{bmatrix}$ ,  $\hat{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ ,  $\vec{b} = \begin{bmatrix} x_B^2 + y_B^2 - d_B^2 + d_A^2 \\ x_C^2 + y_C^2 - d_C^2 + d_A^2 \end{bmatrix}$ .

En el caso de mínimos cuadrados ponderados, los factores de ponderación se aplican antes de la operación módulo sobre el vector de residuos,  $A \cdot \hat{x} - \vec{b}$ .

### 2.2.3. Técnicas basadas en *fingerprinting*

Las técnicas de posicionamiento basadas en *fingerprinting* [11], a diferencia de la multilateración, no precisan de la estimación de la distancia. En cambio, esta familia de técnicas se basa en obtener una “firma” para cada punto de referencia en el escenario que permita posicionar posteriormente a los terminales por similitud. Es decir, cada *fingerprint* se compone de un vector de valores, típicamente de RSSI, y un terminal se localizará en el mapa según la similitud de su vector de RSSI con los *fingerprints* previamente almacenados. El tamaño del vector de *fingerprint* se corresponderá típicamente con el número de balizas en el sistema.

El empleo de *fingerprinting* requiere, por lo tanto, de dos fases diferenciadas: una fase *offline* para obtener los *fingerprints* según un mallado sobre el escenario, y una

fase *online* que ejecute un algoritmo que busque, compare o interpole la posición del terminal en la base de datos de *fingerprints*.

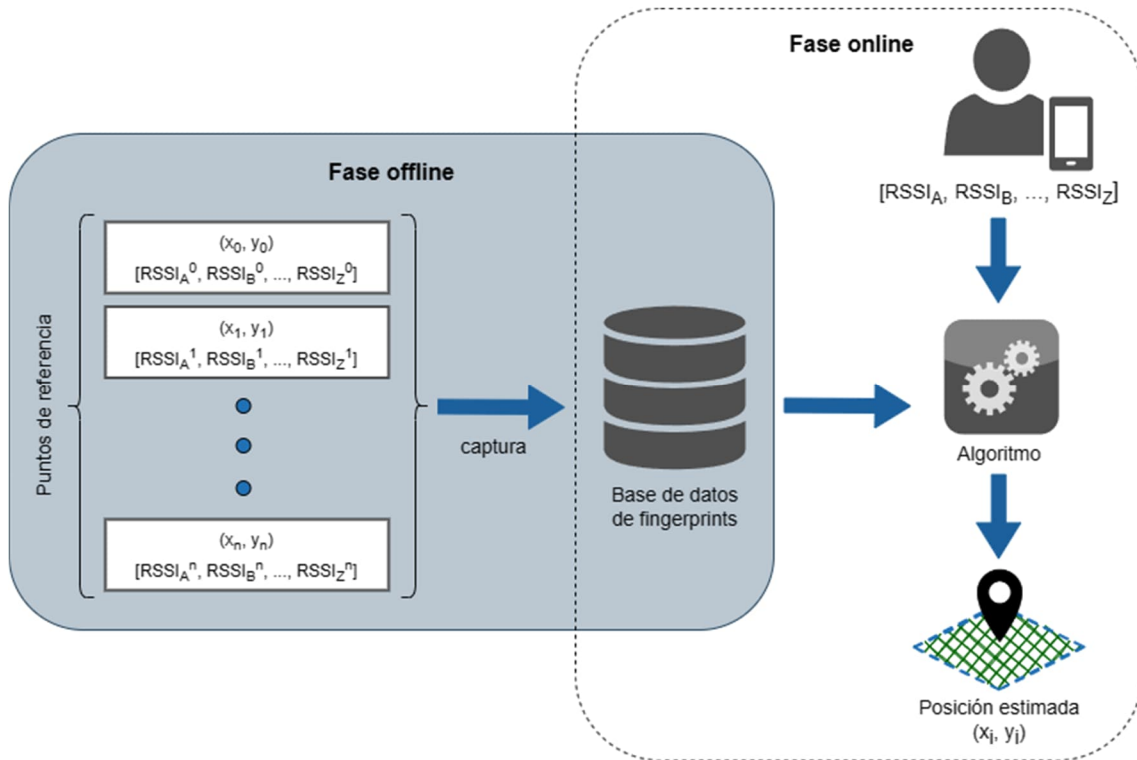


Figura 2.6 – Esquema de fingerprinting

Los algoritmos utilizados en la fase *online* deben resolver un problema de regresión que mapee el vector de valores de RSSI medido con el mallado irregular de fingerprints. Algunas opciones populares son k-NN (k-Nearest Neighbors), las redes neuronales o los bosques aleatorios, En este trabajo se ha utilizado principalmente el algoritmo k-NN ponderado (weighted k-Nearest Neighbors) con los criterios de distancia eucladiana. Este algoritmo calcula la distancia con los  $k$  vecinos más próximos en la malla y realiza un promediado ponderado según la inversa del cuadrado de la distancia:

$$\vec{v} = \frac{1}{\sum_{j=1}^K \frac{1}{d_j^2}} \cdot \sum_{i=1}^K \frac{1}{d_i^2} \cdot \vec{f}_i \quad (7)$$

Donde  $\vec{v}$  es la estimación de la posición a la salida de la fase online del algoritmo,  $\vec{f}_i$  es un vector de *fingerprint* de un punto cercano al vector de medidas y  $K$  es el número de vecinos cercanos a considerar. Nótese que para  $K = 1$ , k-NN funciona como un algoritmo de localización zonal, es decir, la estimación de la posición del terminal será exactamente la del *fingerprint* más cercano.



#### 2.2.4. Comparación entre la multilateración y el *fingerprinting*

A continuación se añade un esquema comparativo de ambas técnicas de posicionamiento:

- Multilateración:
  - Pros:
    - Precisión alta en entornos con línea de visión directa.
    - Calibración sencilla.
    - Bajo coste de despliegue y mantenimiento.
  - Contras:
    - Degradación de la precisión en entornos complejos. Las fluctuaciones en la medida de RSSI provocan errores grandes en la posición.
    - Sensible a la ubicación de las balizas. El diseño del sistema ha de considerar dicha geometría.
- Fingerprinting:
  - Pros:
    - Robustez de la precisión en entornos complejos. Las fluctuaciones en la medida de RSSI no degradan considerablemente la precisión.
    - Robustez ante diferentes geometrías de despliegue de las balizas. El diseño del sistema puede ser más laxo en este aspecto ya que lo importante es el mapa de RSSI.
  - Contras:
    - Alto coste de despliegue y mantenimiento. El mapa de RSSI debe actualizarse periódicamente.

En general, las técnicas basadas en *fingerprinting* son más costosas de implementar que las de multilateración, en tanto en cuanto requieren la creación y mantenimiento de la base de datos de *fingerprints*. Sin embargo, son más robustas en escenarios sin línea de visión directa entre transmisor y receptor, lo que las hace atractivas para mejorar la precisión en entornos complejos.

Tal y como se evidencia en [4], el error de localización que proporciona la misma implementación de una técnica de posicionamiento varía principalmente de acuerdo al número de balizas desplegadas y al escenario, entendiendo por escenario la superficie cubierta por el sistema, así como los obstáculos y otras características de propagación. Para los escenarios utilizados en este trabajo, de ámbito doméstico con una sala de 37 m<sup>2</sup> y una planta de 135 m<sup>2</sup>, el error de localización esperable según la literatura se encuentra entre 1 y 4 metros para ambas técnicas.

## 2.3. Algoritmos de seguimiento

Se considera algoritmos de seguimiento (*tracking* en inglés) a aquellas técnicas que obtienen la localización actual no sólo a partir de la última muestra medida (como así hacen las técnicas de posicionamiento), sino también a partir de la secuencia de posiciones previas. En la práctica, se trata de algoritmos recursivos que permiten la actualización de la posición del terminal no sólo mediante las medidas actuales, sino también en base al último estado calculado en la iteración anterior. Los algoritmos de seguimiento son, por lo tanto, algoritmos con memoria y fuertemente basados en estado.

Los algoritmos de seguimiento utilizarán a menudo fusión de datos con una o más técnicas de posicionamiento para estimar sus parámetros de entrada. Esto ubica a los algoritmos de seguimiento en una suerte de capa intermedia entre el posicionamiento y la navegación. En este trabajo hacemos uso de un subtipo comúnmente denominado "algoritmos de filtrado".

### 2.3.1. Algoritmos basados en el filtro de Kalman

El filtro de Kalman [19] es un algoritmo recursivo que permite estimar el estado de un sistema dinámico a partir de medidas ruidosas. El algoritmo se compone de dos fases, una de predicción y otra de corrección, las cuales se repiten iterativamente. A continuación describimos la versión simplificada del filtro de Kalman utilizada en este trabajo (ver figura x como referencia:

1. Fase de predicción: donde se precalcula el estado de la próxima iteración en base a las ecuaciones del modelo y el estado actual. La fase de predicción contiene los siguientes cálculos:
  - a. Predicción del próximo estado: donde el modelo de transición (matriz  $A_t$ ) transforma el estado actual (vector  $x_t$ ), el modelo de control de entrada (matriz  $B_t$ ) transforma las señales de control (vector  $u_t$ ), si las hubiere, y se modela un ruido del proceso,  $\epsilon_p$ .
  - b. Predicción de la próxima covarianza: a la matriz de covarianza del error,  $P_t$ , se le añade la covarianza del ruido del proceso,  $Q$ .
2. Fase de actualización o corrección: donde se incorporan las medidas actuales para corregir la predicción anterior. La fase de actualización contiene los siguientes cálculos:
  - a. Cálculo de la ganancia de Kalman: se actualiza la ganancia de Kalman (matriz  $K_t$ ), la cual indica la confianza de las medidas observadas, empleando la incertidumbre de las mismas, es decir, la covarianza de las observaciones del modelo (matriz  $R$ ).
  - b. Corrección del estado: se añade la influencia de la nueva medida (vector  $z_t$ ) al estado predicho, proporcionando la estimación del estado actual,  $x_t$ .
  - c. Corrección de la covarianza: se actualiza o corrige la covarianza del error para la próxima iteración. Donde  $\mathbf{I}$  es la matriz identidad.

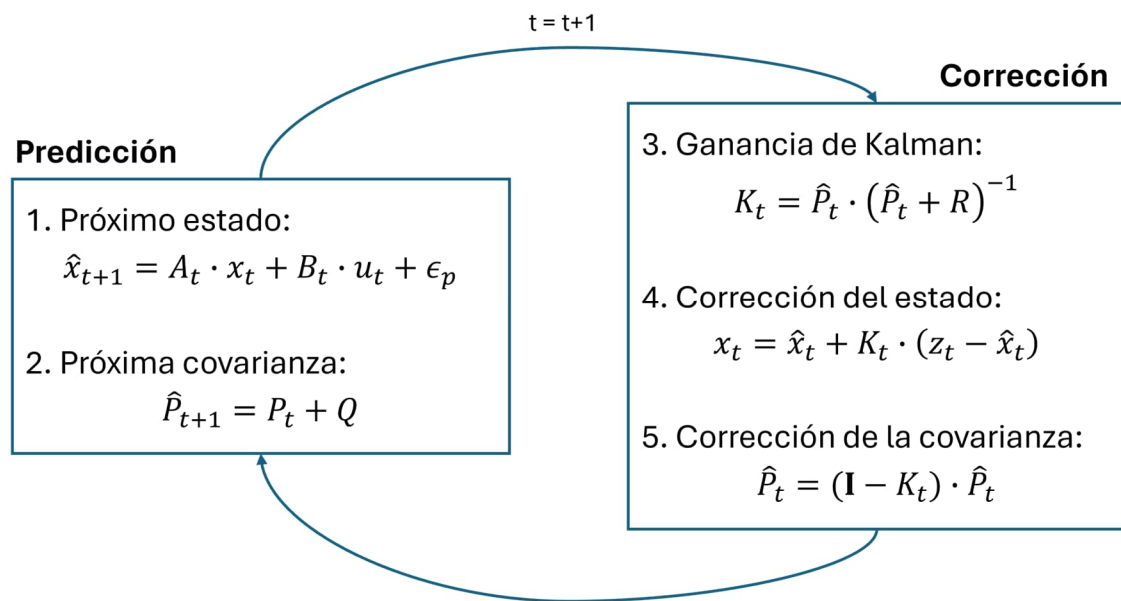


Figura 2.7 – Algoritmo recursivo del filtro de Kalman

Es importante señalar que el filtro de Kalman considera que el ruido es blanco y gaussiano, tanto en las medidas como en el ruido del proceso. Es por ello que este filtro perderá precisión en sistemas con ruido multimodal, es decir, con ruido que no sea modelable mediante una distribución normal.

En el contexto de los sistemas de posicionamiento en interiores, el filtro de Kalman presenta los siguientes casos de uso principales:

- Fusión de datos: fusión de medidas de diferentes sensores (inerciales, etc) para mejorar la precisión de la estimación.
- Control de actitud: estimación de la posición y de otras variables cinemáticas (velocidad, orientación, etc) para control de elementos móviles tales como robots o vehículos.
- Procesamiento de señales: filtrado de señales para eliminar parte del ruido así como extraer información útil.

En este trabajo, utilizaremos un filtro de Kalman unidimensional como procesador de señales para suavizar las medidas de RSSI directamente provenientes del chip de radiofrecuencia del terminal.

### 2.3.2. Algoritmos basados en filtros de partículas

Los filtros de partículas son técnicas basadas en métodos de Monte Carlo que, a diferencia del filtro de Kalman, permiten estimar el estado de un sistema dinámico no lineal a partir de medidas ruidosas multimodales. Los filtros de partículas utilizan una colección de muestras aleatorias, llamadas partículas, que representan la distribución de probabilidad del estado actual del sistema. En el caso de los sistemas de posicionamiento, cada partícula posee unas coordenadas y un peso que representa su verosimilitud en ese instante de tiempo. Al introducir una nueva observación en cada iteración, se actualiza el peso de las partículas y se proporciona una estimación de la posición más probable.

Es precisamente la flexibilidad que proporciona el muestreo en partículas lo que permite modelar sistemas no lineales así como diferentes distribuciones de ruido.

Los filtros de partículas aplicados a sistemas de posicionamiento [1] se componen típicamente de las siguientes fases:

1. Inicialización: se genera la colección de partículas de acuerdo a una distribución estadística. Habitualmente se utilizan la distribución uniforme y la gaussiana, esta última si se conoce una estimación inicial de la posición. Cada partícula posee un peso normalizado y un conjunto de variables de *actitud*, como por ejemplo: posición (en coordenadas cartesianas) y vector de velocidad (a menudo representado mediante el módulo y el ángulo de apuntamiento).
2. Predicción: se propagan las partículas en el espacio de acuerdo al modelo de transición del sistema para así predecir su posición en la próxima iteración. La expresión canónica, desde un punto de vista bayesiano, es la mostrada en (8), pues representa la probabilidad a priori del siguiente estado para cada partícula. A menudo se opta por el conjunto de ecuaciones de movimiento definidas en (9).

$$s_{t+1}^k \sim p(s_{t+1} | s_t^k) \quad (8)$$

Donde  $s_{t+1}$  es el estado siguiente del sistema,  $s_t^k$  es el estado actual de la partícula  $k$ , y  $s_{t+1}^k$  es la predicción del estado siguiente de la partícula  $k$ .

$$\begin{cases} x_{t+1}^k = x_t^k + v_t^k \cdot \Delta t \cdot \cos(\alpha_t^k) + \epsilon_p \\ y_{t+1}^k = y_t^k + v_t^k \cdot \Delta t \cdot \sin(\alpha_t^k) + \epsilon_p \end{cases} \quad (9)$$

Donde, para la partícula  $k$  y el instante actual  $t$ ,  $x_t^k$  e  $y_t^k$  son las coordenadas cartesianas,  $v_t^k$  es el módulo de la velocidad,  $\alpha_t^k$  es la dirección o ángulo de apuntamiento, y  $x_{t+1}^k$  e  $y_{t+1}^k$  son las coordenadas cartesianas predichas para el estado siguiente. Adicionalmente,  $\epsilon_p$  es el ruido gaussiano del proceso, y  $\Delta t$  es el intervalo de tiempo transcurrido desde la anterior actualización y la nueva observación.

3. Actualización: se incorpora la nueva medida y se actualiza el peso de cada partícula según la probabilidad de observar dicha medida desde la predicción de la partícula (probabilidad a posteriori). La ecuación canónica es la mostrada en (10). La utilizada en este trabajo es la expresión de la ecuación (11), la cual representa un ponderado basado en el peso del instante actual y una distribución gaussiana.

$$w_{t+1}^k = p(z_{t+1} | s_{t+1}^k) \quad (10)$$

Donde  $w_{t+1}^k$  es el peso actualizado para la partícula  $k$ ,  $z_{t+1}$  representa la nueva observación, y  $s_{t+1}^k$  es la predicción del siguiente estado para la partícula  $k$  obtenido en la fase 2.

$$w_{t+1}^k = w_t^k \cdot e^{-\frac{1}{2} \left( \frac{d_{t+1}^k}{\sigma_p} \right)^2} \quad (11)$$

Donde  $w_t^k$  es el peso actual de la partícula  $k$ ,  $d_{t+1}^k$  es la distancia entre la predicción de la partícula  $k$  y la nueva observación, y  $\sigma_p$  es la desviación típica del ruido del proceso.

La distancia de la ecuación (11) típicamente se obtiene directamente a partir de aplicar el modelo de propagación en (1) a las medidas de RSSI de cada baliza [1]. Sin embargo, en este trabajo consideramos que ese enfoque limita la estimación de la observación a un esquema de multilateración, por lo que, en su lugar, utilizamos la salida del algoritmo de posicionamiento como referencia para calcular la distancia relativa de cada partícula (ver [sección 3.4](#)). De esta manera, utilizamos el filtro de partículas para suavizar las estimaciones del kernel de posicionamiento, mejorando así el seguimiento de la trayectoria.

Adicionalmente, en la fase de actualización se corrige o actualiza el resto de variables de estado de cada partícula, es decir, la velocidad y la dirección en nuestro caso. Las expresiones utilizadas en este trabajo se encuentran en la [sección 3.4](#).

4. Normalización: se normalizan los pesos de la colección de partículas del filtro:

$$\bar{w}_{t+1}^k = \frac{w_{t+1}^k}{\sum_{i=1}^n w_{t+1}^i} \quad (12)$$

Donde  $n$  es el número total de partículas del sistema,  $w_{t+1}^k$  es el peso actualizado de la partícula  $k$ , y  $\bar{w}_{t+1}^k$  es el peso actualizado normalizado de dicha partícula.

5. Re-muestreo: se generan nuevas partículas a partir del peso de las anteriores. El re-muestreo trata de mantener la diversidad en el espacio de muestras; es por ello una fase importante para evitar la degeneración del sistema. Existen diferentes estrategias de cara al re-muestreo de las partículas: sistémico, estratificado, residual, etc. En este trabajo optamos por implementar un esquema SIR (Sequential Importance Resampling) [20], también denominado re-muestreo multinomial, el cual es sencillo de implementar y mantendrá una diversidad suficiente.
6. Estimación: finalmente, se pondera la posición de las partículas para obtener la estimación de la posición del sistema:

$$\begin{cases} \hat{x}_{t+1} = \frac{1}{n} \sum_{i=1}^n x_{t+1}^i \\ \hat{y}_{t+1} = \frac{1}{n} \sum_{i=1}^n y_{t+1}^i \end{cases} \quad (13)$$

El algoritmo recursivo consiste en la iteración continua de las fases 2 a la 6. La siguiente figura representa la relación entre el estado y el peso de cada partícula con la densidad de una función de probabilidad unidimensional. Puede observarse cómo la fase de remuestreo reproduce las partículas de mayor peso en un mayor

número de partículas, mientras que las partículas de menor peso desaparecen, manteniendo el número de partículas constante.

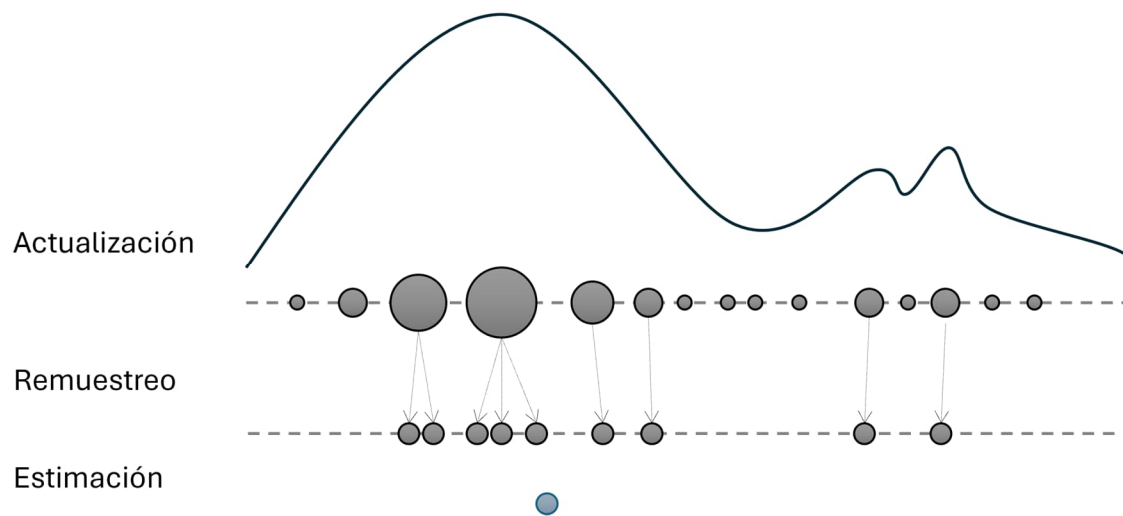


Figura 2.8 – Funcionamiento del filtro de partículas

## 3. Diseño y desarrollo del sistema

### 3.1. Arquitectura del sistema

De cara a cumplir los objetivos definidos en la [sección 1.2](#), este trabajo propone un sistema basado en tecnologías web compuesto por los siguientes elementos:

- Servidor web: con servicios de posicionamiento y mantenimiento.
- Infraestructura BLE: balizas transmisoras compatibles con BLE 4.2.
- Terminales móviles: terminales receptores de clientes o administradores. En este trabajo consideraremos únicamente el caso de terminales Android.
- Estación de trabajo: PC con la aplicación para el despliegue y mantenimiento del sistema.

La siguiente figura muestra un diagrama conceptual de la arquitectura:

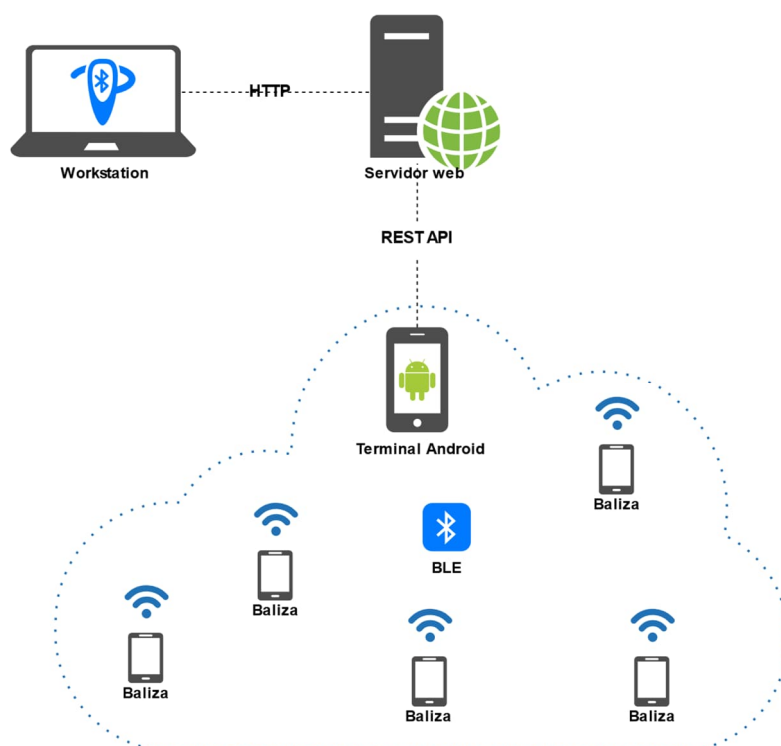


Figura 3.1 – Diagrama conceptual de la arquitectura del sistema

La decisión de diseño por la cual la función de posicionamiento se proporcionará a los terminales por medio de un servicio web se debe a los aspectos no funcionales listados a continuación:

- **Mantenibilidad:** la existencia de un único repositorio software para el producto, y de una única base de datos de posicionamiento (para *fingerprints*, etc) abaratan el coste de mantenimiento del sistema. Si el posicionamiento se realizara en los terminales, se tendrían que mantener versiones de los algoritmos para diferentes plataformas, además de hacer llegar las actualizaciones de la base de datos a cada terminal.

- Portabilidad/compatibilidad: el uso de tecnologías web estándar permite la interoperabilidad con diferentes sistemas operativos y plataformas móviles, así como permite la coexistencia de diferentes versiones del servicio de posicionamiento o de aplicaciones de usuario.
- Escalabilidad: los algoritmos de posicionamiento pueden requerir de alta carga computacional. En este sentido es preferible la escalabilidad horizontal de un servicio web a la escalabilidad vertical necesaria en los terminales. De igual manera, terminales más modestos tendrían acceso al servicio.
- Privacidad: los algoritmos precisan de bases de datos que, en algunos casos, pueden contener metadatos sobre el entorno o, incluso, de otros terminales. Es por ello conveniente mantener dichos datos dentro del dominio del sistema por razones de privacidad.
- Extensibilidad: disponer de la función de posicionamiento como servicio permite extender la funcionalidad del sistema con otros servicios adicionales. Por ejemplo, un servicio de navegación para hospitales o centros comerciales.

El siguiente diagrama de componente, basado en el modelo C4 [17], presenta una arquitectura más detallada del sistema de posicionamiento propuesto.

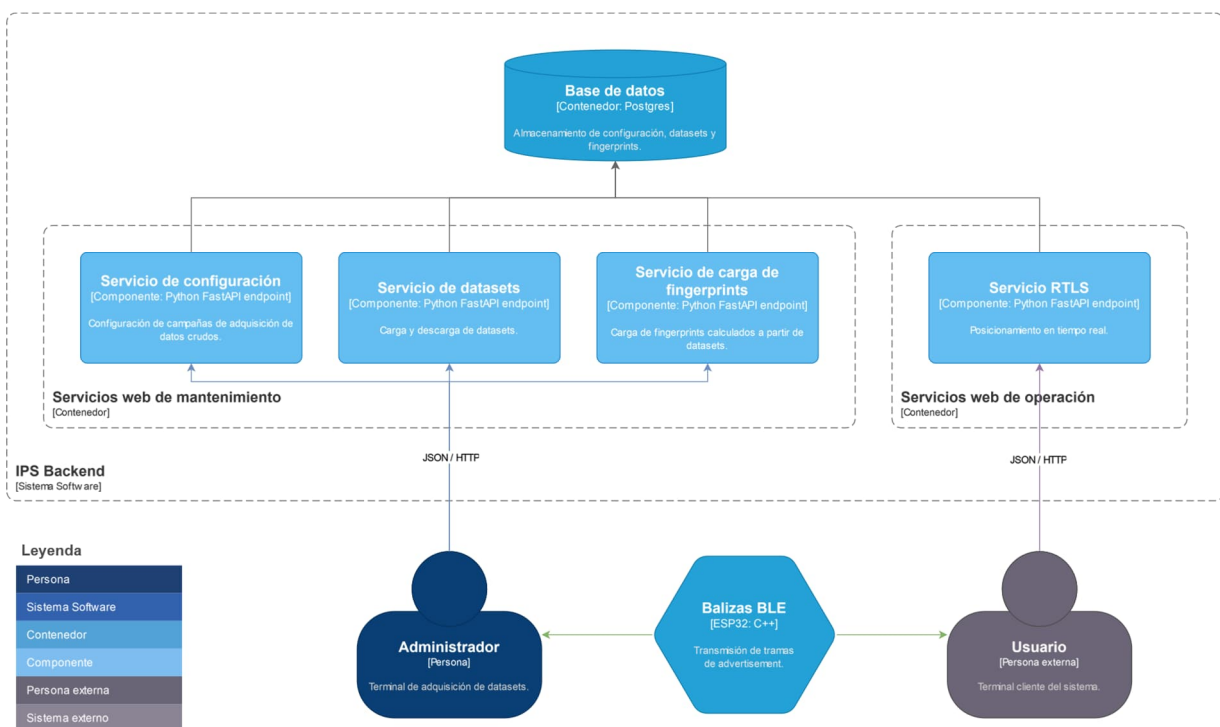


Figura 3.2 – Diagrama de componente (modelo C4) de la arquitectura del sistema

En el diagrama puede apreciarse que el sistema será utilizado por dos roles de usuario diferentes, a saber: el rol de *administrador*, responsable de la preparación y el mantenimiento del sistema, y el rol de *usuario*, el cual es cliente del servicio de posicionamiento mediante una aplicación propia o de terceros. Ambos roles de usuario acceden al sistema mediante terminales *smartphone* convencionales. El servidor web dispone de los siguientes tipos de servicios:



1. Servicios de mantenimiento: servicios pensados para el estudio previo al despliegue y el mantenimiento de la calidad del servicio. Los servicios son: adquisición de muestras, descarga de *datasets*, y carga de configuración y *fingerprints*.
2. Servicio de posicionamiento: servicio de posicionamiento en tiempo real pensado para la fase de explotación del sistema. Puede proporcionar tanto localización basada en coordenadas como posición semántica (áreas o elementos de interés en el mapa).

La parte más importante del proyecto reside en el *backend* o aplicación web. Éste es el componente que contiene la lógica de negocio (*business logic*) del sistema de localización, es decir, los algoritmos de posicionamiento, posicionamiento semántico y seguimiento, así como la base de datos de configuración, *datasets* y *fingerprints*.

## 3.2. Elementos hardware

El prototipo del sistema desarrollado en este trabajo hace uso de los siguientes elementos hardware:

- Ordenador portátil con procesador i5-12450H y 16 GB de RAM: utilizado como servidor web y PC con la aplicación de escritorio para la configuración y tratamiento de los *datasets*.
- Dos terminales smartphone con las versiones de Android 14 (Terminal 1) y 9 (Terminal 2). El Terminal 1 se ha utilizado tanto como terminal de administrador como de usuario. El Terminal 2 se ha utilizado únicamente como terminal de usuario.
- Cinco placas ESP32 Dev-Kit C (dual core): utilizadas como balizas con capacidad BLE 4.2.
- Tres placas ESP32-C3 Dev-Kit M (single core): utilizadas como balizas con capacidad BLE 5.0 en modo *legacy* (formato de trama de acuerdo a BLE 4.2).
- Baterías para las balizas alejadas de una toma de corriente.

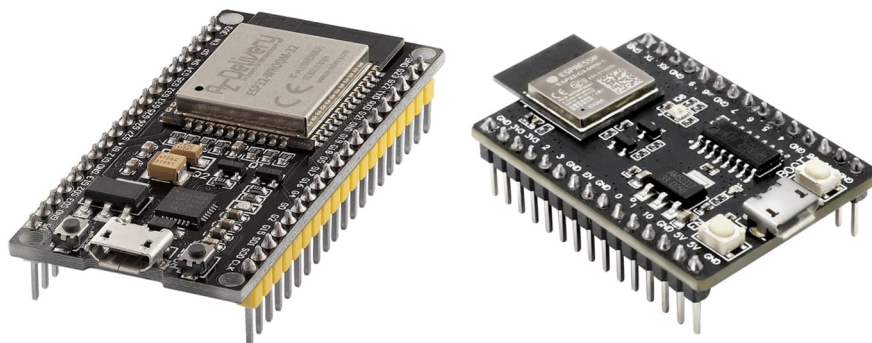
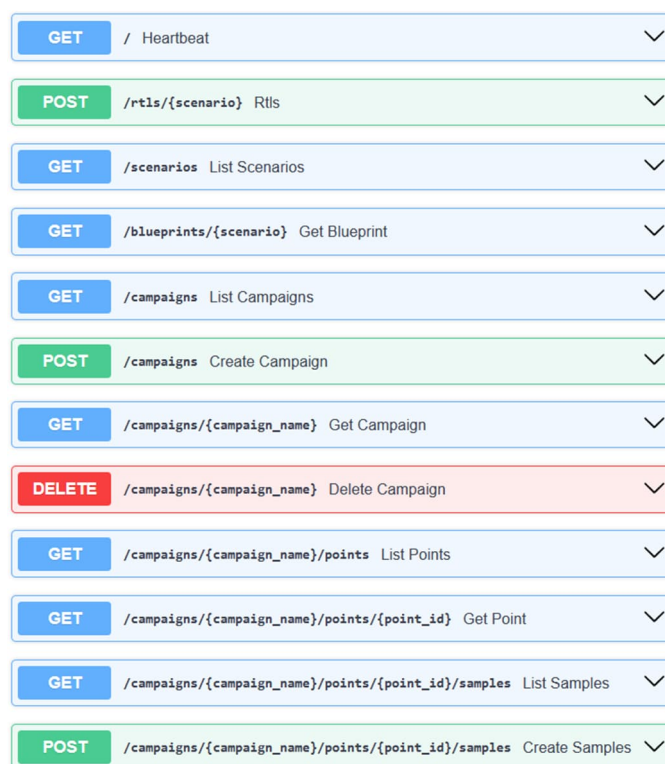


Figura 3.3 – Placas ESP32 Dev-Kit C con BLE 4.2 (izquierda) y ESP32-C3 Dev-Kit M con BLE 5.0 (derecha)

### 3.3. Elementos software

#### 3.3.1. Aplicación web

Como prototipo del *backend* del sistema se ha desarrollado una aplicación web con API REST en Python mediante el framework FastAPI [21]. El servicio se ha desplegado con Docker en un servidor Uvicorn, el cual es un servidor web ASGI (Asynchronous Server Gateway Interface) preparado para gestionar gran cantidad de solicitudes de forma concurrente. La base de datos Postgres se ha desplegado en un contenedor Docker aparte. Tanto el servidor como la base de datos corren en una máquina dentro de una red WiFi privada, a la cual el terminal móvil del administrador tiene acceso. La API REST dispone de la lista de *endpoints* mostrada en la siguiente figura:



GET	/ Heartbeat	▼
POST	/rtls/{scenario} RtlS	▼
GET	/scenarios List Scenarios	▼
GET	/blueprints/{scenario} Get Blueprint	▼
GET	/campaigns List Campaigns	▼
POST	/campaigns Create Campaign	▼
GET	/campaigns/{campaign_name} Get Campaign	▼
DELETE	/campaigns/{campaign_name} Delete Campaign	▼
GET	/campaigns/{campaign_name}/points List Points	▼
GET	/campaigns/{campaign_name}/points/{point_id} Get Point	▼
GET	/campaigns/{campaign_name}/points/{point_id}/samples List Samples	▼
POST	/campaigns/{campaign_name}/points/{point_id}/samples Create Samples	▼

Figura 3.4 – Endpoints de la API REST del servicio web

La ruta “/rtls” da acceso al servicio de posicionamiento en tiempo real para los escenarios “Sala” y “Planta” de este trabajo. La ruta “/blueprints” permite a las aplicaciones descargar el mapa de cada escenario. La ruta “/campaigns” está reservada para administración; ésta permite cargar la configuración de las campañas de adquisición y de subir nuevas muestras desde la aplicación del administrador.

#### 3.3.2. Aplicación Android

Como prototipo de la aplicación de los terminales se ha implementado una *app* en Java para el sistema operativo Android con el IDE Android Studio. Dicha aplicación integra las principales funciones de administración del sistema de

posicionamiento. En un contexto de implantación comercial del sistema, se debería desarrollar una aplicación profesional reservada para el rol de administrador, y otra para el rol de usuario, así como versiones para diferentes sistemas operativos.

La aplicación implementa la funcionalidad descrita a continuación. Las principales características se encuentran agrupadas según el *Fragment* de Android que las implementa. Un *Fragment* es un módulo dentro de la *Activity* de Android que gestiona tanto su propia interfaz de usuario como su ciclo de vida dentro de la aplicación.

- Home: permite configurar la dirección IP y el puerto del servidor web. Proporciona también posicionamiento en tiempo real para el escenario seleccionado.
- Datasets: permite seleccionar un punto de una campaña de adquisición para capturar muestras de RSSI. El tiempo de captura viene determinado por la configuración descargada del servidor.
- Scanner: lista los anunciantes de las tramas de *advertisement* recibidas en el terminal en ese instante. Permite filtrar por el nombre de las balizas del sistema.

La siguiente figura muestra sendas capturas de pantalla de la interfaz de usuario de los fragments de la aplicación:



Figura 3.5 – Interfaz de usuario de la aplicación Android

El siguiente diagrama de casos de uso resume las posibles interacciones por parte del administrador con la aplicación:

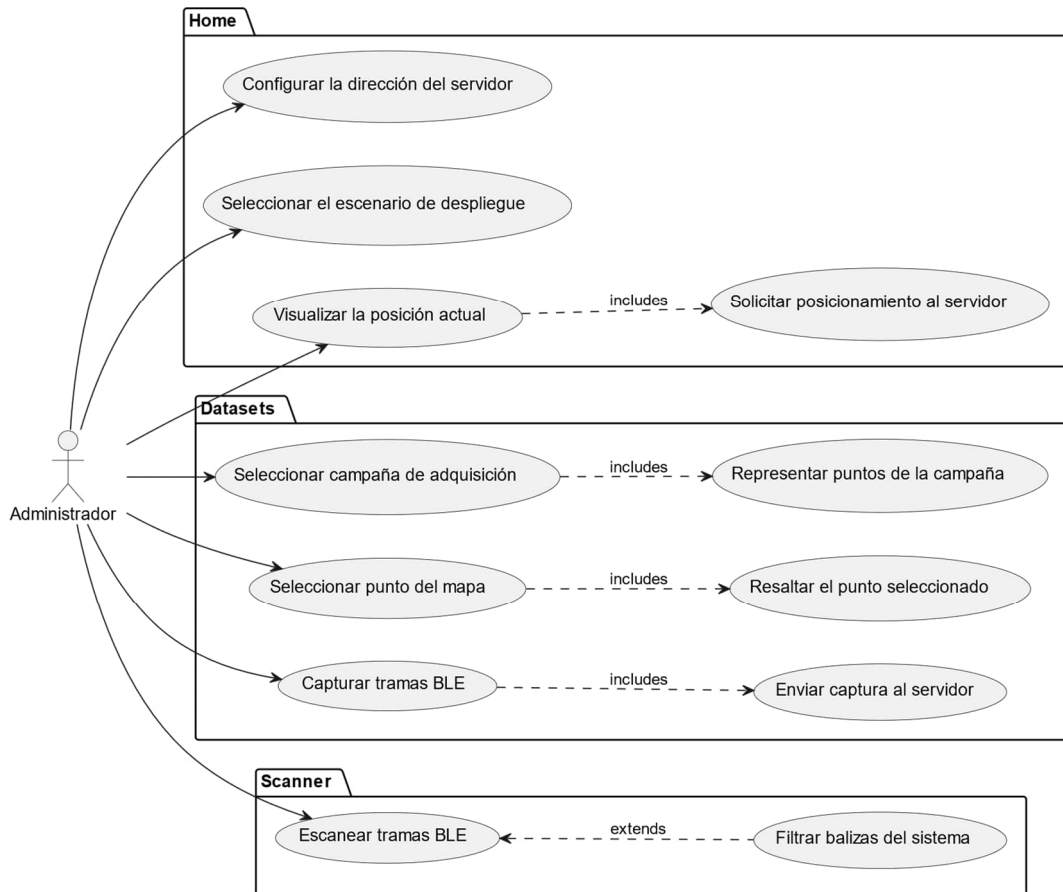


Figura 3.6 – Diagrama de casos de uso de la aplicación Android

### 3.3.3. Aplicación de escritorio

Como herramienta de simulación y configuración de las campañas de adquisición, así como del posterior tratamiento de los *datasets*, se ha desarrollado una aplicación de escritorio en Python mediante la librería Streamlit [22]. Esta herramienta se conecta a los servicios de la aplicación web a través de la misma API REST que utiliza la aplicación Android para descargar los datasets necesarios.

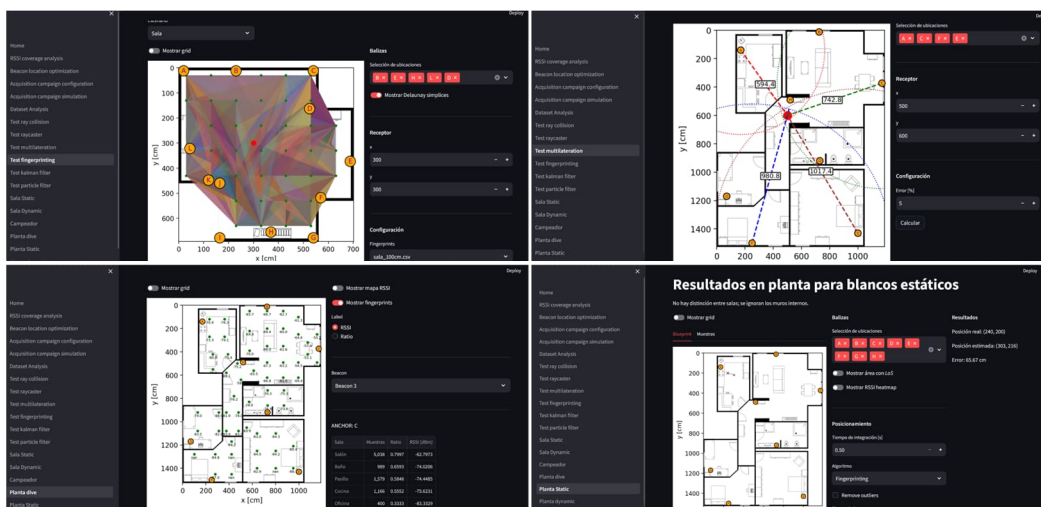


Figura 3.7 – Interfaz de usuario de la herramienta de escritorio

Esta herramienta sirve como prototipo de lo que sería la aplicación de escritorio comercial para la configuración y el mantenimiento del sistema de posicionamiento. Las funciones de la aplicación de escritorio han de ser:

- Generación y carga en la aplicación web de campañas de adquisición de *datasets*.
- Descarga de *datasets* de la aplicación web.
- Análisis estadístico de *datasets*.
- Generación y mantenimiento de bases de datos de *fingerprints*.
- Generación de áreas para posicionamiento semántico.
- Calibración mediante el modelo de pérdidas de propagación.
- Testeo del servicio de posicionamiento de la aplicación web.
- Simulación de balizas y efectos de propagación.

### 3.3.4. Programación de las balizas

Por último, las balizas BLE se han programado en C++ utilizando el plugin PlatformIO del IDE Visual Studio Code, mediante el framework Arduino y la librería NimBLE. El uso de la librería es diferente para las balizas BLE 4.2 y las BLE 5.0, como muestra el siguiente diagrama de clases:

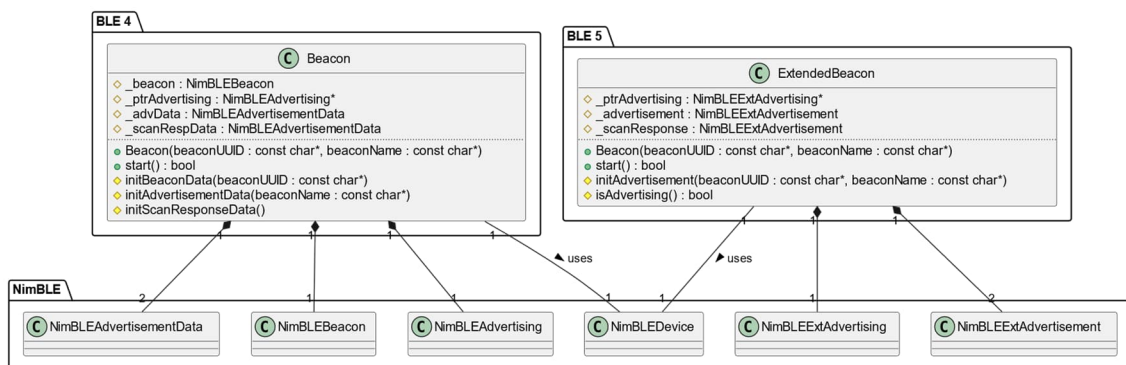


Figura 3.8 – Diagrama de clases de las balizas BLE

Tanto la clase *Beacon* para BLE 4.2, como la clase *ExtendedBeacon* para BLE 5.0, implementan el protocolo *iBeacon*. La clase *ExtendedBeacon* se utiliza para las balizas con el microcontrolador ESP32-C3, el cual hace uso de la parte de la librería NimBLE pensada para el modo extendido de BLE 5. Dicha API permite, a diferencia de la API para el modo clásico, configurar los canales de *advertisement* en los que se quiere transmitir.

### 3.3.5. Limitaciones técnicas

En este apartado se presentan las limitaciones técnicas más importantes encontradas durante el desarrollo del sistema. Estas limitaciones justifican principalmente la imposibilidad de introducir medidas de tiempo (tales como ToF) en el esquema de posicionamiento.

### 3.3.5.1. API de BLE en Android

La API de BLE del SDK (Software Development Kit) de Android presenta dos limitaciones que impiden incorporar medidas de tiempo (p.ej. ToF) en un esquema de radiolocalización basado en tramas de *advertisement*:

- La clase `ScanRecord` de la API de Android proporciona un resumen del escaneo por parte del terminal, en el cual se pone en conjunto los datos de la trama de *advertisement* y los de la trama de *scan response* (dicho resumen se denomina Advertising Report en el estándar BLE). Esto impide reaccionar desde la aplicación de Android a eventos que permitan activar un temporizador con el que medir el tiempo de ida y vuelta (Round-Trip Time) mediante las tramas de *scan request* y *scan response* de BLE (ver el diagrama de secuencia a continuación).
- La clase `ScanResult` de la API de Android, la cual posee un objeto `ScanRecord` como atributo, proporciona el tiempo de recepción en nanosegundos de la última trama de *advertisement* vista en el momento de construir el resultado. Es decir, la API no proporciona el tiempo de recepción del *scan response*. Esto implica que no se puede obtener una marca de tiempo (*timestamp*) en el terminal receptor dentro de un esquema ToF en el que el transmisor envíe una referencia de tiempo como respuesta al *scan request*.

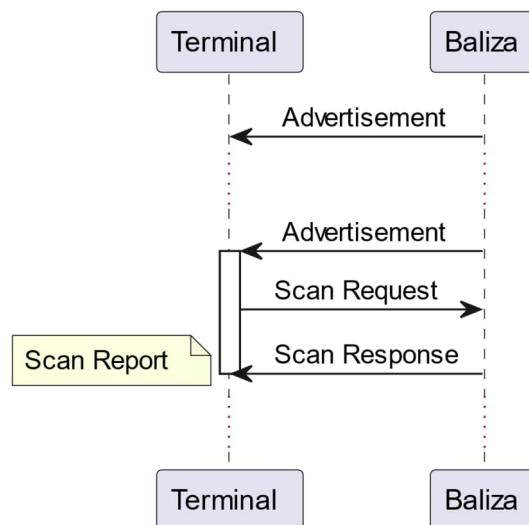


Figura 3.9 – Diagrama de secuencia del mecanismo de escaneo de advertisement BLE

Además, una limitación adicional de la API de Android, bastante problemática de cara a mejorar la precisión del sistema de posicionamiento, es el hecho de que los resultados de escaneo no proporcionan el canal de *advertisement* a través del cual se recibió cada trama. Esto implica que, para cada baliza transmisora, obtendremos una distribución estadística del RSSI que engloba todos los canales de *advertisement* utilizados por la baliza. En otras palabras, la dispersión o varianza de las muestras de RSSI será superior a lo que obtendríamos pudiendo distinguir entre los tres canales de *advertisement* en BLE.

### 3.3.5.2. Librería NimBLE y estándar Bluetooth

La librería NimBLE utilizada en este trabajo es una refactorización en C++ para el framework Arduino del stack NimBLE original de Apache [23]. Dicha librería soporta los microcontroladores ESP32 y ESP32-C3 utilizados para las balizas BLE.

La librería NimBLE presenta una serie de limitaciones que impiden introducir medidas de tiempo en el esquema de *advertisement* de BLE. La mayor parte de estas limitaciones provienen en origen del propio estándar Bluetooth, y es que las tramas de *advertisement* no están pensadas para proporcionar datos dinámicos. Las limitaciones a este respecto son las siguientes:

- La API no expone eventos relacionados con el mecanismo de *scan request* y *scan response*. De tal manera que no es posible incorporar el uso de temporizadores que ayuden a obtener medidas de tiempo (p.ej. ToF).
- La trama de *advertisement* no puede modificarse dinámicamente para incorporar una marca de tiempo (*timestamp*) que ayude a incorporar medidas como ToF para la radiolocalización. El *advertisement* ha de interrumpirse, reconfigurarse y volverse a cargar al controlador inalámbrico mediante el comando HCI\_LE\_Set\_Advertising\_Data del *Host Controller Interface* de BLE, lo cual añade un retardo indefinido.
- La trama de *scan response* no puede modificarse sin interrumpir la transmisión (requiere cargarse mediante el comando HCI\_LE\_Set\_Scan\_Response\_Data). Los *scan response* están pensados para comunicar información estática adicional desde el dispositivo anunciante a petición del cliente.
- La API no permite configurar los canales de *advertisement* para BLE 4.2 (únicamente para BLE 5 mediante el modo extendido), por lo que las balizas BLE 4.2 transmitirán en todos los canales ineludiblemente, obteniendo un muestreo del RSSI más ruidoso (ver las limitaciones de la API de Android), mientras que con las balizas BLE 5 podemos configurar selectivamente en qué canales transmitir (37, 38 o 39) según lo requiera el escenario donde el sistema vaya a ser desplegado.

### 3.4. Algoritmo del sistema de posicionamiento

El sistema de posicionamiento de este trabajo contiene los siguientes componentes funcionales:

1. Filtro de Kalman: realiza un filtrado de la señal de RSSI de cada baliza capturada en el terminal. Al tratarse de un filtro unidimensional, las ecuaciones mostradas en la [sección 2.3.1](#) se simplifican (tanto matrices como vectores son ahora variables o escalares).
2. Cálculo del vector de observaciones: obtiene las señales de RSSI filtradas por Kalman y calcula el vector de observaciones, el cual se compone de  $N$  elementos, uno por cada baliza del sistema. La estrategia empleada en este trabajo para calcular los elementos del vector de observaciones se basa en promediar todas las muestras de la baliza que entren en la llamada "ventana de integración", la cual se define por un intervalo temporal en milisegundos. Asumiendo que en la ventana entran  $m$  muestras:

$$\vec{z} = \begin{bmatrix} RSSI_1 \\ RSSI_2 \\ \dots \\ RSSI_N \end{bmatrix}; \quad RSSI_i = \frac{1}{m} \cdot \sum_{j=0}^{m-1} RSSI_i^j, \quad \forall i \in \{1, 2, \dots, N\} \quad (14)$$

La ventana de integración es configurable según el despliegue. Para el escenario "Sala" se ha utilizado una ventana de 200 ms. Para el escenario "Planta" se ha fijado en 500 ms. El valor de la ventana de integración es dependiente de los requisitos en cuanto a la tasa de actualización de la posición en el sistema. Para la mayoría de aplicaciones, una actualización cada 500 ms es más que suficiente.

3. Kernel de posicionamiento: recibe como entrada el vector de observaciones de RSSI y proporciona a la salida una estimación de la posición en el dominio cartesiano. Las técnicas de posicionamiento utilizadas en este trabajo se presentaron en la [sección 2.2](#).
4. Filtro de partículas: recibe como entrada la estimación de la posición por parte del kernel de posicionamiento y proporciona a la salida una nueva estimación de la posición filtrada, mejorando así la precisión del sistema.

La [sección 2.3.2](#) presenta los filtros de partículas. El modelo de estado utilizado en este trabajo para las partículas es el definido por las siguientes ecuaciones:

$$\left\{ \begin{array}{l} x_{t+1}^k = x_t^k + v_t^k \cdot \Delta t \cdot \cos(\alpha_t^k) + \epsilon_p \\ y_{t+1}^k = y_t^k + v_t^k \cdot \Delta t \cdot \sin(\alpha_t^k) + \epsilon_p \\ \alpha_{t+1}^k = \arctan2(\hat{y} - y_{t+1}^k, \hat{x} - x_{t+1}^k) + \epsilon_a \\ v_{t+1}^k = \frac{1}{2} \cdot \frac{\sqrt{(\hat{x} - x_{t+1}^k)^2 + (\hat{y} - y_{t+1}^k)^2}}{\Delta t} + \epsilon_v \\ w_{t+1}^k = w_t^k \cdot e^{-\frac{1}{2} \left( \frac{\sqrt{(\hat{x} - x_{t+1}^k)^2 + (\hat{y} - y_{t+1}^k)^2}}{\sigma_p} \right)^2} \end{array} \right. \quad (15)$$



Donde las ecuaciones para actualizar las coordenadas  $x$  e  $y$  de la partícula  $k$  son las mismas que en (9), con la diferencia de que  $\Delta t$  es ahora el tiempo de la ventana de integración. La actualización del peso de la partícula es la misma ecuación que en (11), utilizando la distancia euclidiana entre la estimación del filtro de partículas actual y la posición actualizada de la partícula. El ángulo de apuntamiento de la partícula,  $\alpha_{t+1}^k$ , se actualiza a partir del vector con, punto origen, la propia posición de la partícula  $y$ , punto final, la estimación del filtro. El módulo de la velocidad,  $v_{t+1}^k$ , se obtiene de dividir el tiempo de integración a la distancia entre la estimación y la partícula, dividiendo a su vez entre dos (decisión de diseño para reducir el efecto de la incertidumbre en la velocidad). Finalmente, las variables aleatorias gaussianas de media cero:  $\epsilon_p$ ,  $\epsilon_a$ , y  $\epsilon_v$ , son ruidos de proceso para la posición, el ángulo y la velocidad, respectivamente.

El diagrama mostrado a continuación representa el flujo de procesamiento de los datos:

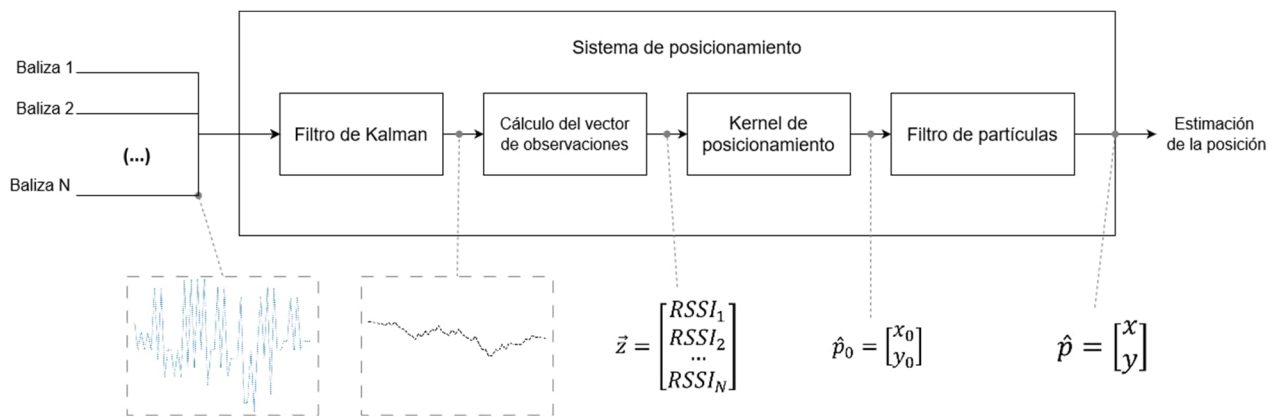


Figura 3.10 – Algoritmo de posicionamiento

A diferencia del uso habitual del filtro de partículas en la literatura (filtro con multilateración: [1], filtro con fingerprinting: [2]), la propuesta de este trabajo es utilizar la salida del kernel de posicionamiento como entrada del filtro de partículas. Es decir, el filtro de partículas trabajará sobre las coordenadas en el dominio cartesiano, en lugar de transformar directamente el vector de observaciones de RSSI a la posición en el plano. Si bien no garantizamos que los resultados obtenidos mediante este esquema sean mejores a aquellos propios de una solución que fusione el kernel de posicionamiento con el filtro de partículas, con este enfoque conseguimos que el filtro de partículas sea independiente del kernel de posicionamiento, permitiendo así reemplazarlo según las circunstancias lo requieran. En este trabajo, utilizaremos kernels basados tanto en multilateración como en *fingerprinting*, pero el esquema funcionaría también con técnicas basadas en aprendizaje máquina, con bioinspiradas o, incluso, con otros filtros de partículas.

Además, aunque no es propio del ámbito de los sistemas de posicionamiento, este trabajo ha estudiado el diseño de un kernel *fingerprinting* basado en una interpolación lineal sobre un mallado multidimensional por Delaunay [24]. Sin embargo, se ha descartado el uso de esta técnica en el sistema debido a que, en la mayoría de ocasiones, el vector de medidas de RSSI observado se encuentra fuera

del conjunto convexo de *simplices*, por lo que la interpolación no es posible. La principal hipótesis del problema es que a mayor dimensionalidad en el mallado, mayor es el número de muestras no interpolables, por lo que queda como línea de trabajo futura el investigar un método que reduzca la dimensionalidad del mallado Delaunay a la vez que permita integrar un número indefinido de balizas.

### 3.4.1. Ajuste de los parámetros del algoritmo

#### 3.4.1.1. Ajuste del filtro de partículas

El filtro de partículas se ha ajustado mediante ensayo y error para un escenario con una incertidumbre de 200 cm de desviación típica tanto en la coordenada  $x$  como en la coordenada  $y$ . Los parámetros utilizados son los siguientes:

- Número de partículas: 500
- Desviación estándar del ruido de proceso de la posición,  $\sigma_p$ : 250 cm
- Desviación estándar del ruido de proceso de la velocidad,  $\sigma_v$ : 20 cm/s
- Desviación estándar del ruido de proceso del ángulo,  $\sigma_a$ :  $\pi/4$  rad

La siguiente figura muestra el resultado de aplicar el filtro de partículas a una trayectoria simulada con una incertidumbre gaussiana en la medida de desviación típica igual a 50 cm por coordenada (izquierda), y una desviación típica a 200 cm (derecha). Puede observarse cómo la trayectoria filtrada de la derecha es inevitablemente más errática.

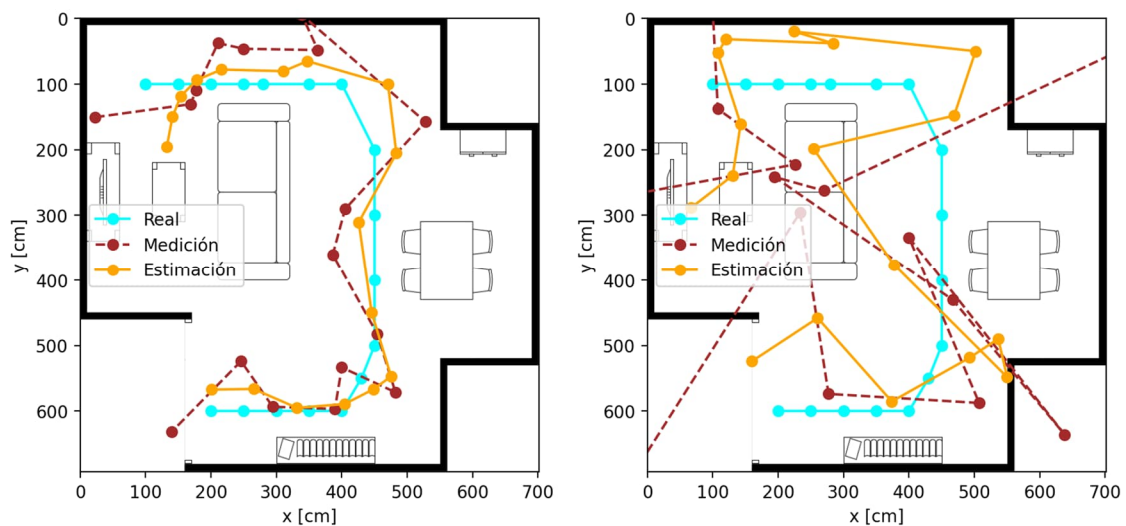


Figura 3.11 – Trayectoria estimada del filtro de partículas con datos simulados

La siguiente figura representa cuatro muestras de la trayectoria para ambos escenarios (nuevamente, con menor incertidumbre a la izquierda, y con mayor a la derecha) así como el estado de todas las partículas del filtro en ese instante. El seguimiento del filtro de partículas es mejor cuanto menor incertidumbre afecta a la entrada, por lo que verificamos así el diseño y su implementación.

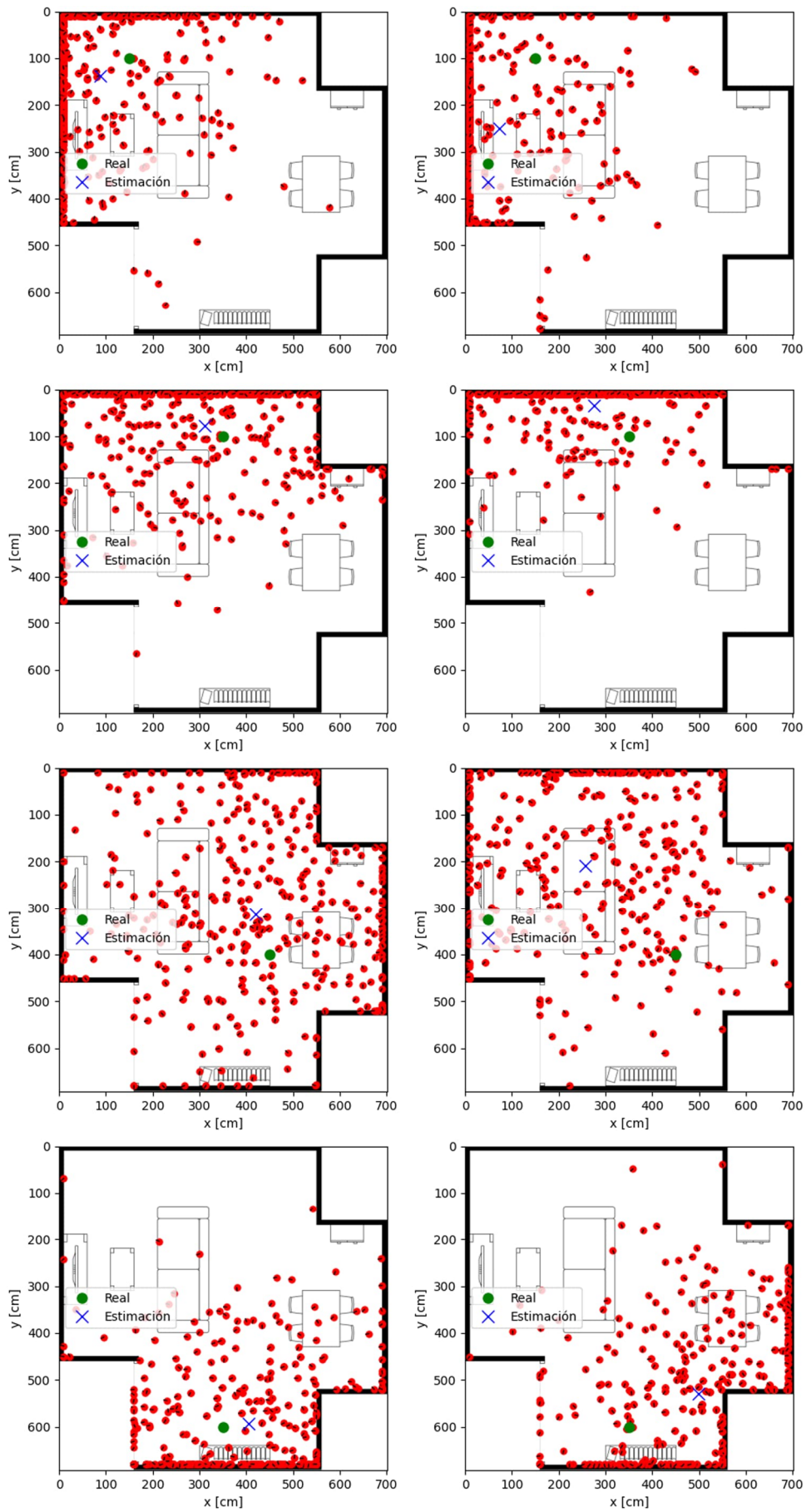


Figura 3.12 – Comparación de la estimación del filtro de partículas en diferentes instantes

### 3.4.1.2. Ajuste del filtro de Kalman

El filtro de Kalman se ha ajustado también mediante ensayo y error. Los parámetros utilizados son los siguientes:

- Modelo de transición,  $A$ : 1
- Modelo de control de entrada,  $B$ : 0
- Covarianza de las observaciones,  $R$ : 1
- Ruido del proceso,  $Q$ : 0.01

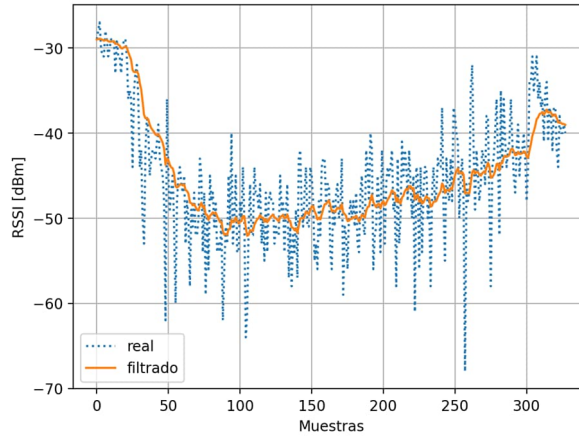


Figura 3.13 – Señal RSSI filtrada por Kalman

Con esta parametrización del filtro de Kalman se ha probado su desempeño con las balizas BLE 5 de cara a tomar la decisión de diseño sobre en qué canales transmitir las tramas de *advertisement* para reducir el ruido o incertidumbre de la señal de RSSI (ver sección 3.3.5.1). El experimento ha consistido en capturar muestras a lo largo de una línea recta con una distancia desde 50 cm hasta 6 m de la baliza y una separación de 50 cm entre puntos de la captura. Los resultados de aplicar el filtro se muestran a continuación:

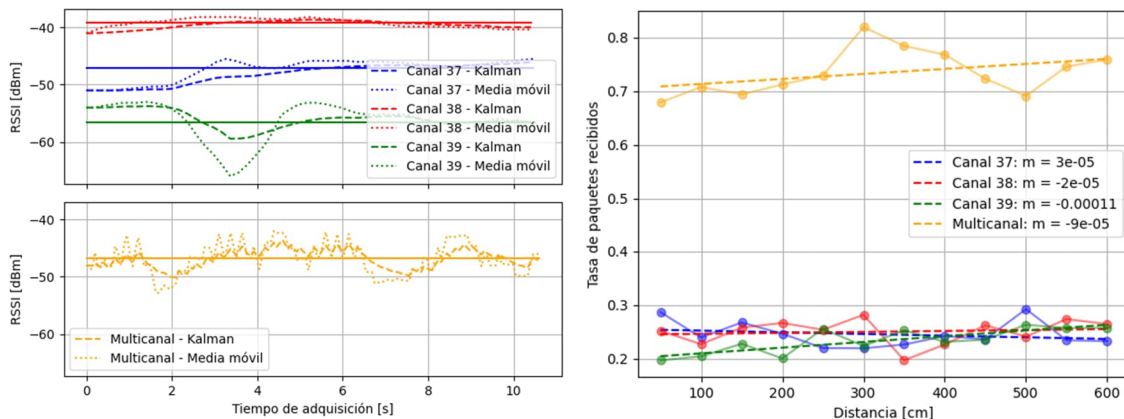


Figura 3.14 – Señal RSSI a 1m en los canales BLE (izquierda); tasa de paquetes recibidos (derecha)

Canal BLE	RSSI [dB]	Media móvil [dB]	Kalman [dB]
37	1.61	1.05	0.81
38	1.87	1.29	1.12
39	2.27	1.55	1.16
Multicanal	4.41	1.52	1.03

Tabla 3.1 – Desviación típica de los valores de RSSI

El filtro de Kalman reduce en general la desviación estándar de las muestras de RSSI. Por ejemplo, para una distancia a 1 metro de la baliza, la desviación de la señal filtrada es de 1 dB tanto para cada canal individual (37, 38 y 39) como para el uso de todos los canales en conjunto. Se puede comprobar que el desempeño es también superior al uso de una media móvil de longitud 10. Con estos resultados, y debido al hecho de que la transmisión de tramas en los tres canales aumenta considerablemente la tasa de paquetes recibidos en el terminal receptor, en este trabajo transmitiremos en los tres canales de *advertisement* en todas las balizas.

### 3.5. Algoritmo de posicionamiento semántico

Adicionalmente a la localización en el plano, el servicio de posicionamiento proporcionará información sobre el área en la que se ubica el terminal. A este servicio complementario lo denominamos “posicionamiento semántico”. En este trabajo, se proporcionará el nombre de la estancia donde se predice la posición del terminal para el escenario “Planta”.

El algoritmo de posicionamiento semántico es, en esencia, un clasificador en el ámbito del aprendizaje máquina. Utilizaremos una Máquina de Soporte Vectorial (Support Vector Machine), la cual será entrenada con los mismos *fingerprints* utilizados para la técnica de *fingerprinting*. Es decir, cada estancia del escenario constituirá una clase y las medidas de RSSI de cada baliza constituirán las características. Consultar la [sección 5.3.4](#) para más información. El diagrama a continuación muestra el flujo de procesamiento del algoritmo. La entrada al clasificador es también un vector de observaciones de las medidas de RSSI similar al presentado en la sección anterior (aunque el tiempo de integración puede ser diferente), por lo que resulta conveniente implementar ambos algoritmos reutilizando los bloques funcionales comunes.

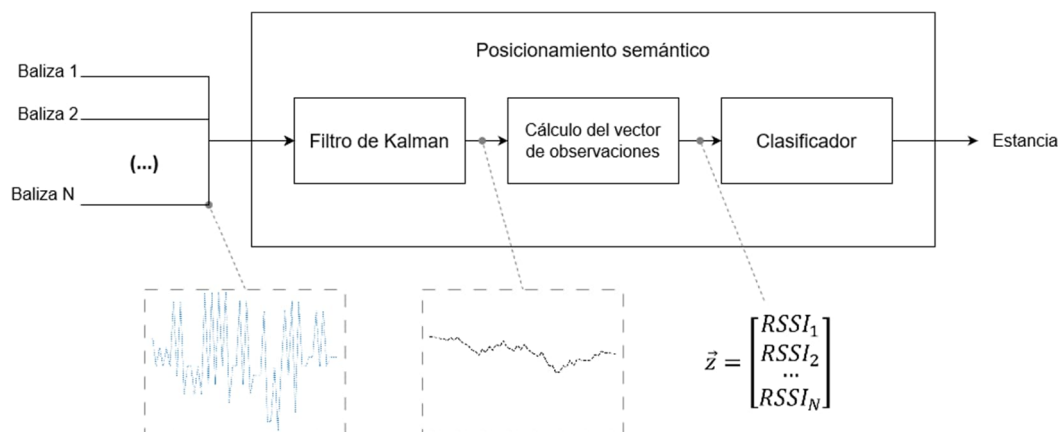


Figura 3.15 – Algoritmo de posicionamiento semántico

## 4. Despliegue del sistema en sala

### 4.1. Escenario

El escenario denominado "Sala" representa la primera prueba empírica del sistema de posicionamiento desarrollado en este trabajo. La elección de este escenario se debe principalmente a la forma irregular de la estancia, ya que se trata de un ático con una disposición de techo y paredes inusual, lo que lo convierte en una base ideal para probar el sistema en un entorno de propagación multicamino complejo.

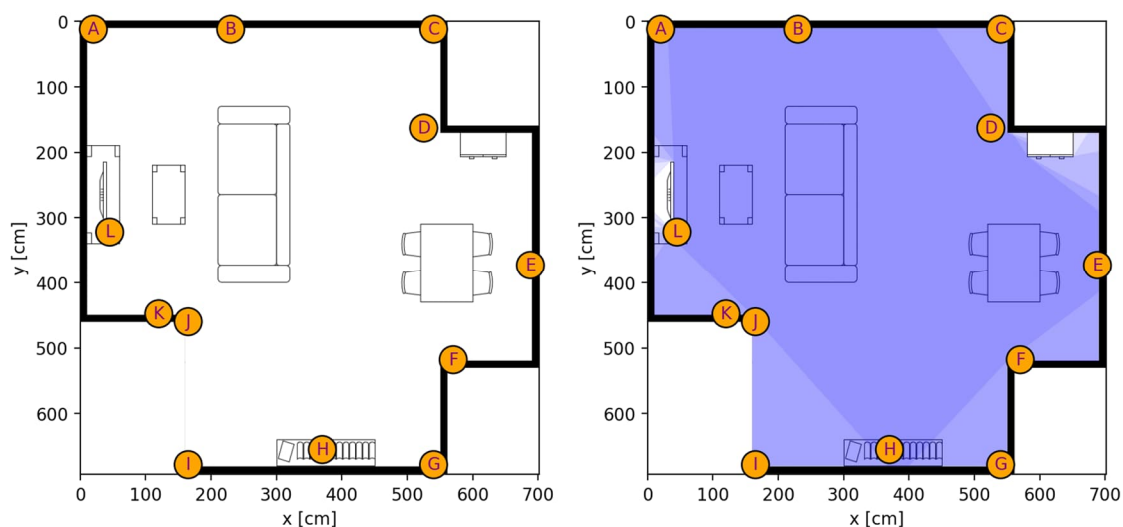


Figura 4.1 – Escenario "Sala" con las ubicaciones candidatas para las balizas. A la derecha, el área cubierta por línea de visión para la constelación seleccionada (B-D-E-H-L).

El escenario "Sala" tiene una superficie de aproximadamente unos 37 m<sup>2</sup> y contiene mobiliario doméstico y muros como obstáculos a la propagación. El número de balizas seleccionadas para el despliegue es de cinco, teniendo como ubicaciones candidatas las definidas en la tabla siguiente:

Ubicación	Coordenadas [cm]	Calidad geométrica
A	(20, 12)	media
B	(230, 12)	alta
C	(540, 12)	media
D	(525, 163)	alta
E	(688, 373)	media
F	(570, 518)	baja
G	(540, 678)	baja
H	(370, 655)	media
I	(165, 678)	baja
J	(165, 460)	baja
K	(120, 448)	baja
L	(45, 322)	media

Tabla 4.1 – Coordenadas de las ubicaciones en Sala

En la tabla se añade una valoración subjetiva sobre la relevancia de la ubicación denominada “calidad geométrica” (baja, media, o alta). Esta observación está fundamentada en el concepto de línea de vista y respaldada por simulación. La geometría del despliegue influye notablemente en los resultados del algoritmo basado en multilateración (éste resuelve, en sí mismo, un problema geométrico) por lo que al menos una de las ubicaciones con calidad geométrica alta debe estar presente en el despliegue (nótese, además, que la “calidad geométrica” de la constelación de balizas puede verse degradada en su conjunto si la selección de ubicaciones es arbitraria).

Tras diferentes simulaciones con multilateración, las ubicaciones más recurrentes que han presentado menor error de localización son las siguientes (ordenadas de mayor a menor recurrencia): E, F, D, B, J, C, H, A, L, G, K, I. En base a estas simulaciones se ha optado por la siguiente constelación de ubicaciones para el despliegue: B-D-E-H-L. En un contexto de despliegue comercial del sistema, deberían validarse los resultados de la simulación empíricamente, así como utilizar métodos de optimización.

## 4.2. Campaña de adquisición

El experimento realizado ha consistido principalmente en una campaña de adquisición de muestras de RSSI con la que obtener los *datasets* necesarios para probar el sistema con terminales estáticos (*datasets* de test), así como para la construcción de la base de datos de *fingerprints* y calibración (*datasets* de entrenamiento o preparación). La captura del dataset de entrenamiento o preparación ha sido realizada con el Terminal 1 (Android 14), mientras que la captura de los datasets de test se ha llevado a cabo tanto con el Terminal 1, como con el Terminal 2 (Android 9), para así poder comparar los resultados con un dispositivo móvil diferente del utilizado para calibrar y generar los *fingerprints*.

El despliegue de las balizas se resume en la tabla siguiente. Se han empleado tres balizas BLE 5.0, configuradas para operar a la potencia máxima de transmisión permitida por el estándar (20 dBm), y dos balizas BLE 4.2, ajustadas a una potencia de 3 dBm. Todas ellas alimentadas por baterías portátiles a 3.7-5 V y configuradas con un intervalo de *advertisement* de aproximadamente 100 ms.

Ubicación	Baliza	Versión de BLE	Potencia de transmisión
L	Beacon 1	4.2	+3 dBm
D	Beacon 2	4.2	+3 dBm
B	Beacon 6	5.0	+20 dBm
E	Beacon 7	5.0	+20 dBm
H	Beacon 8	5.0	+20 dBm

Tabla 4.2 – Información de las balizas desplegadas en Sala



Figura 4.2 – Imágenes de las balizas durante el experimento

La siguiente figura representa los puntos en el escenario donde se han llevado a cabo las capturas. A la izquierda, con un círculo verde, se muestran los puntos utilizados para obtener los *fingerprints*. A la derecha, con un aspa azul, los puntos utilizados para testear la exactitud del sistema.

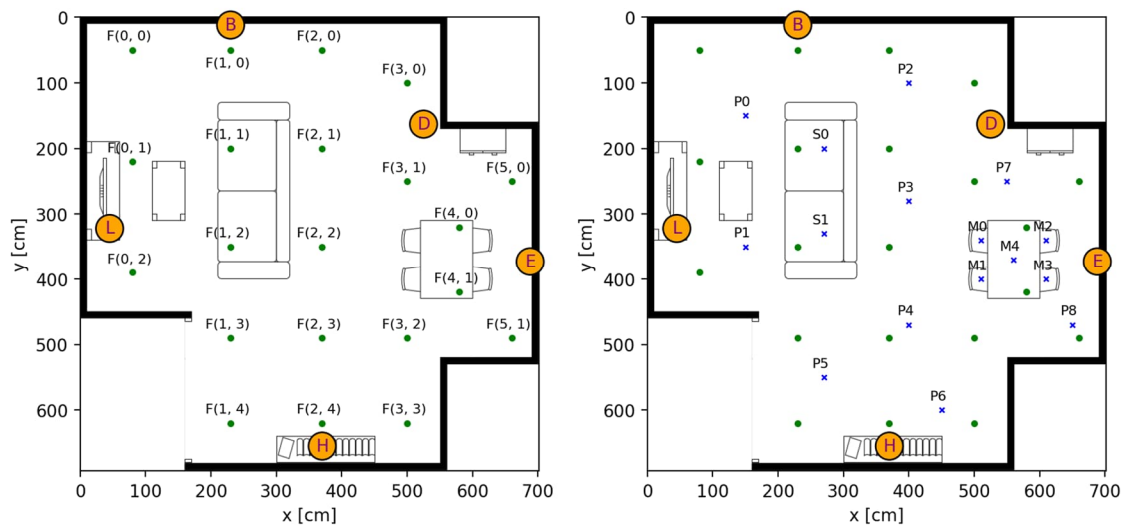


Figura 4.3 – Distribución de *fingerprints* en Sala (izquierda) y de blancos estáticos (derecha)

Las capturas para los puntos de *fingerprint* son de 10 segundos para este escenario (aproximadamente 100 muestras por baliza). Las capturas de test, las cuales denominaremos también “blancos” de ahora en adelante, son de 5 segundos (aproximadamente 50 muestras por baliza). Las capturas se han realizado principalmente con el terminal en la mano derecha, a una altura de  $\pm 50$  cm con respecto a las balizas. La única excepción ha sido para aquellos puntos ubicados encima de la mesa; los *fingerprints*: F(4, 0) y F(4, 1), y los blancos: M0, M1, M2, M3 y M4, donde el terminal se ha dejado apoyado. El propósito de realizar las capturas de esta manera es el de emular un comportamiento lo más cercano posible a la operación real del sistema.

Para obtener una idea de la calidad de la señal en el escenario, se presenta el siguiente mapa de calor de la cobertura BLE. Para confeccionarlo se ha normalizado la suma de los valores de RSSI en cada punto, de tal manera que el color claro representa la mejor cobertura y, el color oscuro, la peor. A la izquierda se muestra el mapa con los valores reales medidos de RSSI. A la derecha se



representa la simulación. Puede observarse cómo el área entre la baliza B y la D presenta valores de RSSI inesperadamente bajos. Ésto puede deberse a efectos de propagación, ya que esa zona tiene el techo abuhardillado de menor altura. Nótese también que la representación no está exenta de artefactos propios de la interpolación.

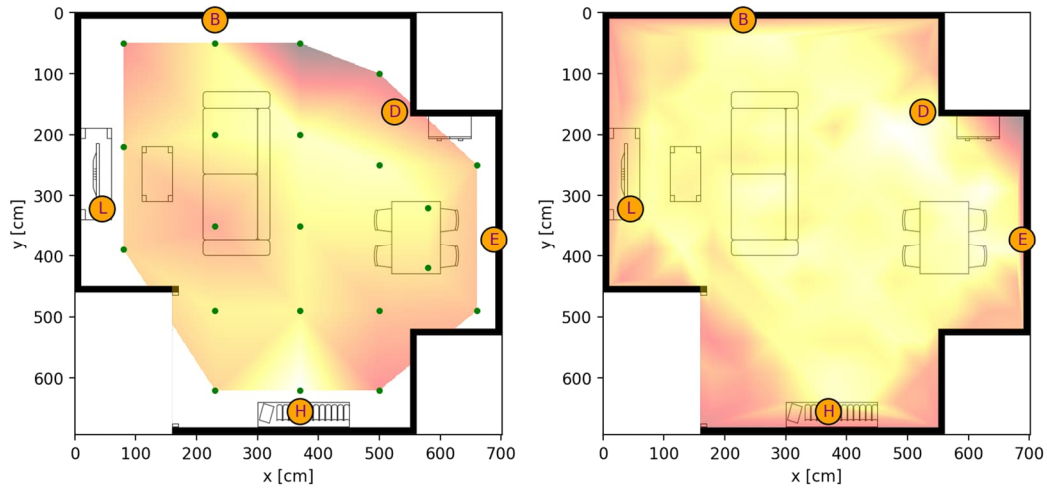


Figura 4.4 – Mapa de RSSI en Sala con datos reales (izquierda) y simulados (derecha)

### 4.3. Procesamiento de los datos

#### 4.3.1. Calibración

La técnica de multilateración requiere un proceso de calibración para cada baliza que permita ajustar los parámetros del modelo de pérdidas por propagación de la ecuación (1). Mediante este modelo se realizará el cálculo de la distancia a cada baliza a partir del vector de observaciones de RSSI.

La calibración de cada baliza se ha realizado mediante una regresión lineal sobre el modelo logarítmico de la ecuación (1) y a partir de las muestras del *dataset* de *fingerprints*, las cuales se han dispuesto según la distancia a la baliza. El valor de RSSI tomado por cada punto es la media aritmética de las muestras de toda la captura. La siguiente figura representa los resultados de la regresión:

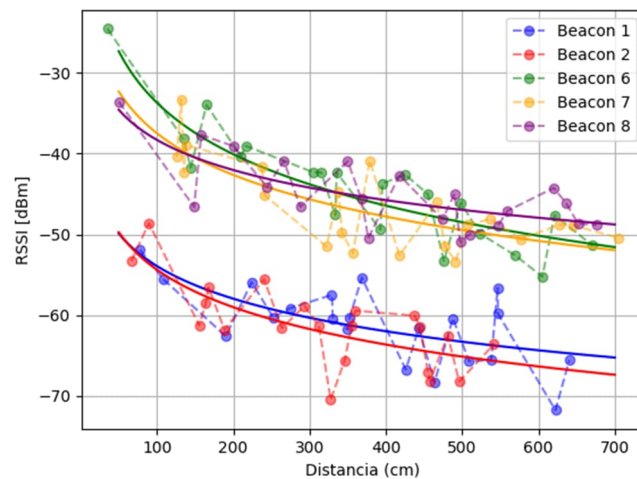


Figura 4.5 – Regresión lineal en Sala del modelo logarítmico de propagación

A continuación se resumen los valores obtenidos para el factor de pérdidas y el valor de RSSI a 1 metro en cada baliza:

Baliza (ubicación)	Factor de pérdidas (n)	RSSI <sub>0</sub> [dBm]
Beacon 1 (L)	1.34	-53.94
Beacon 2 (D)	1.54	-54.39
Beacon 6 (B)	2.12	-33.7
Beacon 7 (E)	1.71	-37.48
Beacon 8 (H)	1.24	-38.31

Tabla 4.3 – Calibración para la multilateración en Sala

### 4.3.2. Fingerprints

Los vectores de *fingerprint* necesarios para la técnica de *fingerprinting* se han obtenido promediando los valores de RSSI de cada baliza de forma similar a como se construye el vector de observaciones con la ecuación (14) descrito en la [sección 3.4](#). En este caso, con la diferencia de que filtramos de antemano las muestras para eliminar los valores atípicos por encima del percentil 90, y por debajo del percentil 10.

Podemos considerar, retrospectivamente, que este proceso tiene margen de mejora para aquellos casos en los que la captura esté sesgada o presente un nivel de incertidumbre significativo. Una opción para minimizar el sesgo sería incrementar el tiempo de adquisición. Como ejemplo, a continuación se representan las señales de RSSI de las balizas en dos capturas de dos puntos diferentes; la captura de la izquierda con un sesgo y desviaciones típicas mayores que los de la derecha.

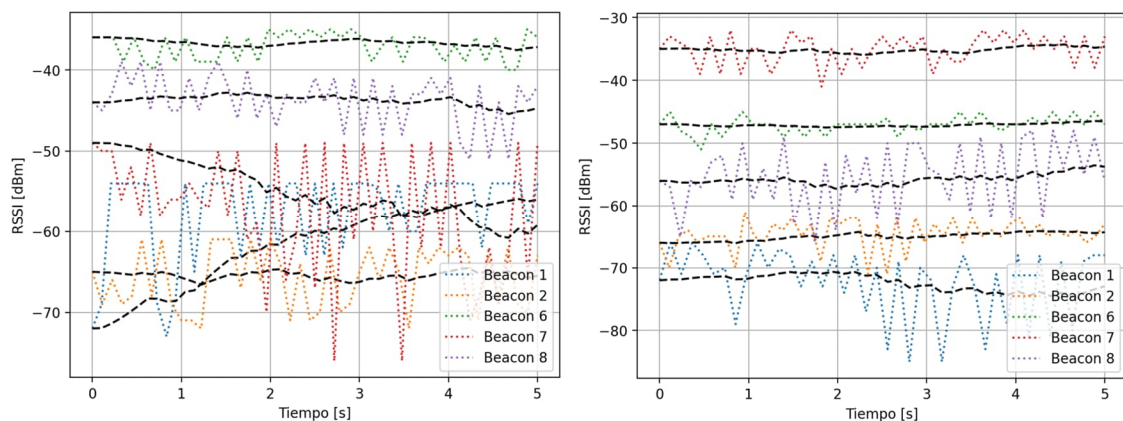


Figura 4.6 – Señales RSSI de dos puntos diferentes en Sala

### 4.3.3. Error de localización

Una vez obtenida la base de datos de *fingerprints* y los valores de calibración para la técnica de multilateración, se ha procesado el *dataset* de prueba con la parametrización presentada en las tablas siguientes. Se han procesado los datos con la técnica de multilateración basada en mínimos cuadrados ponderados, y con la técnica de *fingerprinting* con *k-Nearest Neighbors* (con distancia ponderada y sin

ponderar) para un número de vecinos: 3, 4 y 5. Todos los casos se han probado con y sin el filtro de partículas dentro de la cadena de procesamiento. El tiempo de integración del algoritmo de posicionamiento se ha fijado en 200 ms. La primera tabla muestra los resultados para el Terminal 1. La segunda tabla, los resultados para el Terminal 2.

Técnica	Distancia ponderada	K	Filtro de partículas	MAE (m)	RMSE (m)	ID
Multilateración	sí	-	sí	1.15	1.31	fp_multi
			no	1.20	1.33	multi
Fingerprinting	sí	3	sí	1.26	1.43	fp_w3nn
			no	1.32	1.48	w3nn
		4	sí	1.20	1.38	fp_w4nn
			no	1.23	1.39	w4nn
		5	sí	1.18	1.35	fp_w5nn
			no	1.18	1.34	w5nn
	no	3	sí	1.32	1.47	fp_3nn
			no	1.38	1.51	3nn
		4	sí	1.25	1.43	fp_4nn
			no	1.26	1.41	4nn
5	sí	1.22	1.39	fp_5nn		
	no	1.21	1.37	5nn		

Tabla 4.4 – Errores de localización en Sala para el Terminal 1

Técnica	Distancia ponderada	K	Filtro de partículas	MAE (m)	RMSE (m)	ID
Multilateración	sí	-	sí	1.48	1.82	fp_multi
			no	1.50	1.83	multi
Fingerprinting	sí	3	sí	1.65	1.85	fp_w3nn
			no	1.65	1.85	w3nn
		4	sí	1.57	1.77	fp_w4nn
			no	1.59	1.78	w4nn
		5	sí	1.55	1.75	fp_w5nn
			no	1.55	1.74	w5nn
	no	3	sí	1.72	1.92	fp_3nn
			no	1.73	1.92	3nn
		4	sí	1.60	1.79	fp_4nn
			no	1.61	1.79	4nn
5	sí	1.52	1.72	fp_5nn		
	no	1.53	1.71	5nn		

Tabla 4.5 – Errores de localización en Sala para el Terminal 2

Los mejores resultados se han obtenido con multilateración y filtro de partículas (pues se trata de un escenario con línea de visión). En general, el error promedio absoluto (MAE) para el Terminal 2 es de aproximadamente 30 cm por encima del valor para el Terminal 1. Esta diferencia de error se debe principalmente a los puntos P5 y P6, donde el Terminal 2 sufre una degradación mayor como muestra la siguiente figura (izquierda). A la derecha se representa el histograma con la distribución del error de ambos terminales. La desviación típica del error para el Terminal 2 es mayor debido a la presencia de *outliers*.

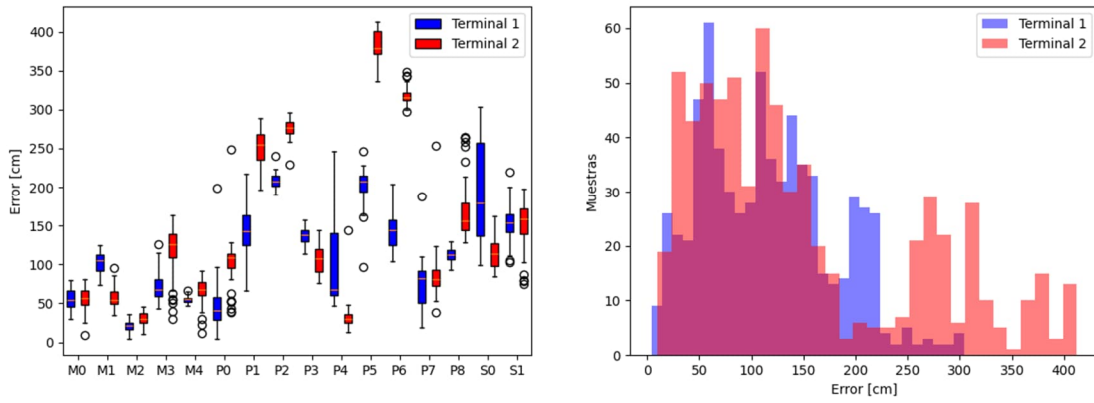


Figura 4.7 – Distribución del error en Sala para ambos terminales

A continuación se muestra la posición real de cada blanco (círculo azul) y su estimación (aspa roja) en el mejor caso registrado (izquierda) y en el peor (derecha), para ambos terminales. El resultado sugiere la posibilidad de aplicar técnicas de corrección del sesgo. Nótese que, para el Terminal 1, el peor punto es P2, el cual se encuentra precisamente en el área de peor cobertura RSSI mostrado anteriormente.

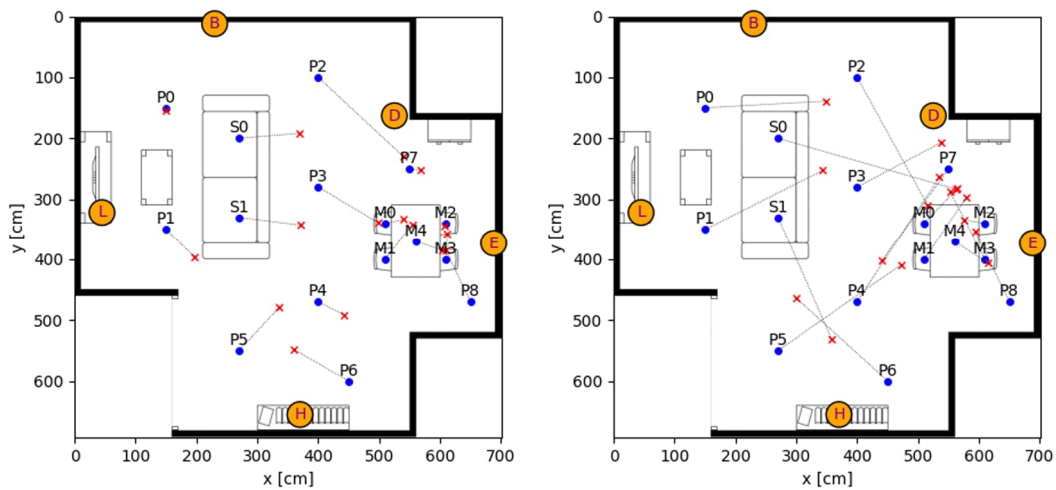


Figura 4.8 – Estimaciones en Sala para el Terminal 1 con el mejor caso (izquierda) y el peor (derecha)

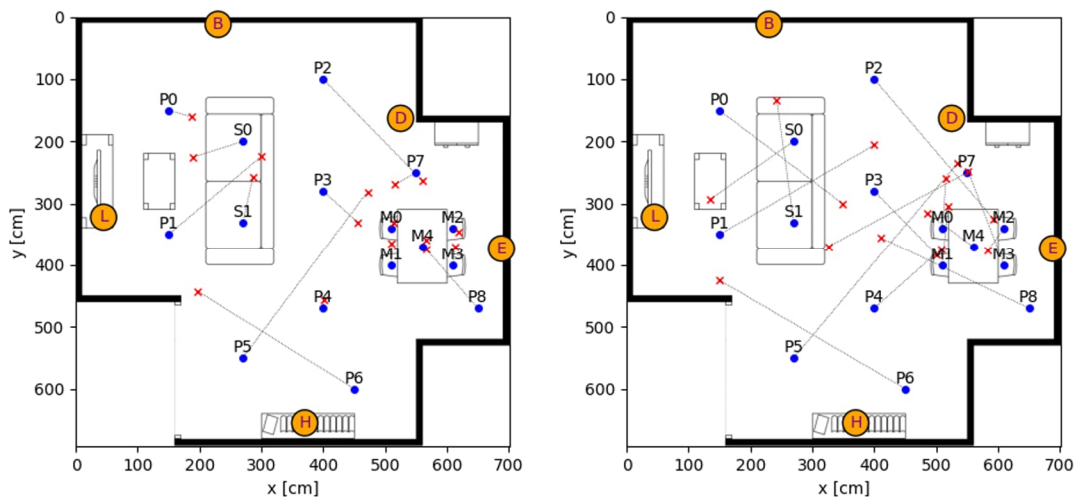


Figura 4.9 – Estimaciones en Sala para el Terminal 2 con el mejor caso (izquierda) y el peor (derecha)

#### 4.3.3.1. Despliegue en sala con menor número de balizas

Adicionalmente, se han procesado los mismos *datasets* para evaluar el rendimiento del sistema con un menor número de balizas, para así abaratar los costes del despliegue y del mantenimiento posterior. Este análisis se ha realizado para 3 y 4 balizas, con las constelaciones B-E-H y B-E-H-L, de acuerdo a las ubicaciones de la tabla x. Es decir, las balizas BLE 5 están presentes en todos los casos de estudio.

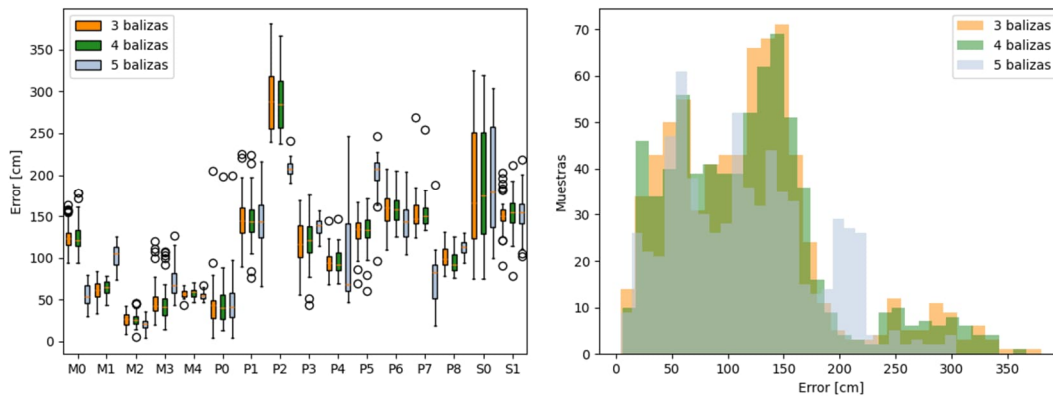


Figura 4.10 – Distribución del error en Sala para 3, 4 y 5 balizas

Los resultados muestran errores parejos al sistema de 5 balizas, con lo que podría reducirse el número de balizas para este escenario y seguir cumpliendo con el objetivo de error de localización.

#### 4.3.4. Seguimiento del terminal

Como se muestra en las tablas anteriores, la incorporación del filtro de partículas en el algoritmo de posicionamiento proporciona una mejora en el error absoluto, aunque esta mejora no es especialmente significativa. Sin embargo, en términos de precisión (que no exactitud), la mejora justifica su uso. En la siguiente figura se muestra el seguimiento del terminal durante la captura de 5 segundos para el blanco P7. A la izquierda se muestra el resultado con multilateración sin el filtro de partículas. A la derecha, con el filtro. El aumento de la precisión tiene importancia de cara a la experiencia de usuario y a otros servicios.

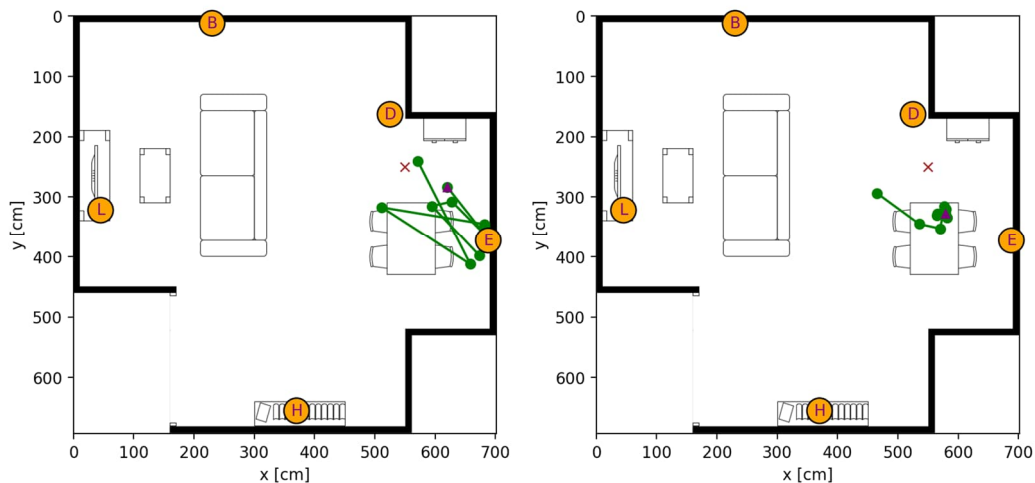


Figura 4.11 – Mejora de la precisión con el filtro de partículas

## 5. Despliegue del sistema en planta

### 5.1. Escenario

El escenario denominado "Planta" representa la segunda prueba empírica del sistema de posicionamiento desarrollado en este trabajo. Este despliegue tiene como objetivo validar el sistema, así como demostrar la aplicabilidad real del prototipo, en este caso, en entornos domésticos.

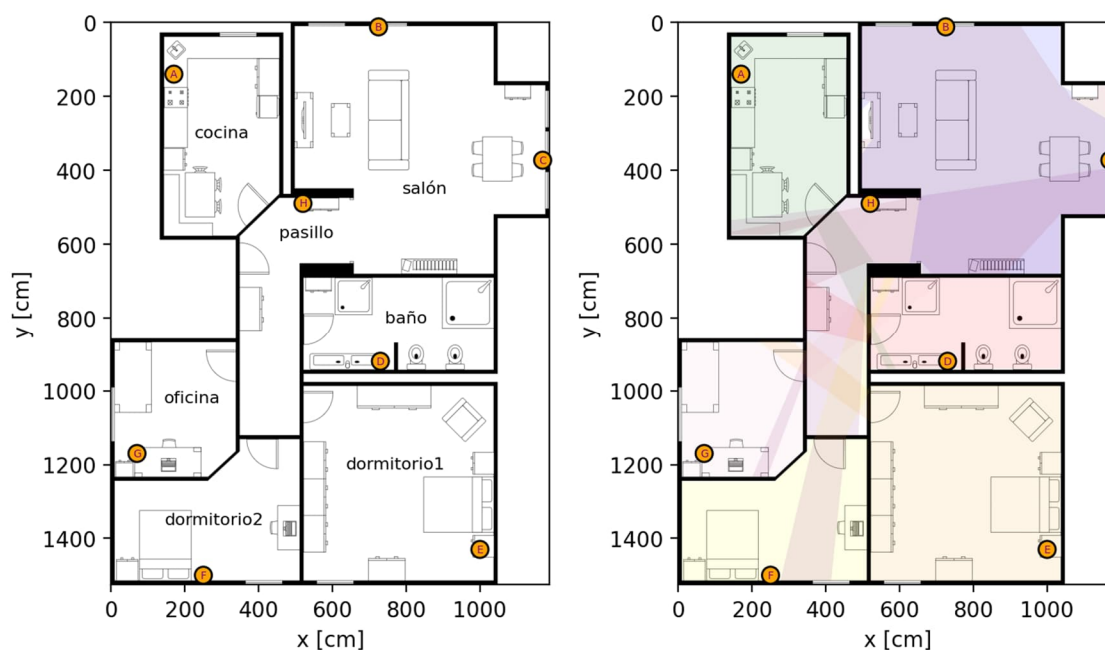


Figura 5.1 – Escenario "Planta" con las ubicaciones de las balizas. A la derecha, las líneas de visión.

El escenario "Planta" tiene una superficie de aproximadamente unos 135 m<sup>2</sup> y dispone de siete salas o estancias con mobiliario doméstico, a saber: cocina, salón pasillo, baño, oficina, dormitorio 1 y dormitorio 2. Como puede observarse, el salón es la misma estancia del escenario "Sala". El número de balizas desplegadas en este escenario es de ocho:

Ubicación	Coordenadas [cm]	Estancia
A	(219, 158)	cocina
B	(725, 12)	salón
C	(1183, 373)	salón
D	(730, 900)	baño
E	(1000, 1400)	dormitorio1
F	(280, 1490)	dormitorio2
G	(130, 1160)	oficina
H	(520, 507)	pasillo

Tabla 5.1 – Coordenadas de las ubicaciones en Planta

A diferencia del despliegue en Sala, debido a la falta de baterías para tres de las ocho balizas, la selección de ubicaciones ha tenido que tener en cuenta la proximidad de las tomas de corriente.

## 5.2. Campaña de adquisición

De manera similar al escenario en Sala, se ha realizado una campaña de adquisición de muestras de RSSI con la que generar los *datasets* de test, y de entrenamiento o preparación (para *fingerprints* y calibración). El Terminal 1 (Android 14) se ha utilizado tanto para la captura de preparación como para la de test. El Terminal 2 (Android 9), sólo para test. El despliegue de las balizas es el detallado en la siguiente tabla. El intervalo de *advertisement* es de aproximadamente 100 ms para todas las balizas. La alimentación es de 3.7-5 V, estando las balizas *Beacon 1*, *Beacon 4* y *Beacon 7* conectadas a la toma de corriente y, el resto, alimentadas por batería.

Ubicación	Baliza	Versión de BLE	Potencia de transmisión
A	Beacon 1	4.2	+3 dBm
B	Beacon 2	4.2	+3 dBm
C	Beacon 3	4.2	+3 dBm
D	Beacon 4	4.2	+3 dBm
G	Beacon 5	4.2	+3 dBm
H	Beacon 6	5.0	+20 dBm
F	Beacon 7	5.0	+20 dBm
E	Beacon 8	5.0	+20 dBm

Tabla 5.2 – Información de las balizas desplegadas en Planta

La siguiente figura representa los puntos del escenario donde se ha capturado las muestras. A la izquierda, con un círculo verde, los puntos de *fingerprint*. A la derecha, con un aspa azul, los blancos estáticos o puntos de test.

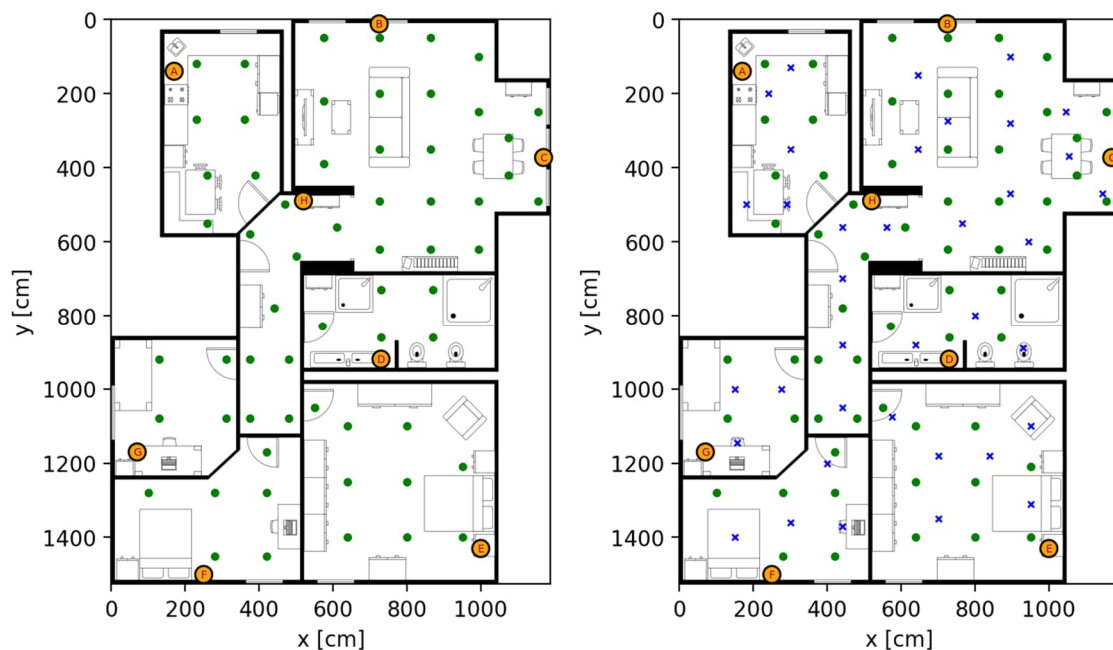


Figura 5.2 – Distribución de fingerprints en Planta (izquierda) y de blancos estáticos (derecha)

En este escenario las capturas en los puntos de *fingerprint* han sido de 30 segundos (aproximadamente 300 muestras por baliza). Las capturas de test han sido de 10 segundos (unas 100 muestras por baliza). De igual manera que en Sala, las capturas se han realizado principalmente con el terminal en la mano derecha, a una altura de  $\pm 50$  cm con respecto a las balizas. La única excepción ha sido para aquellos puntos ubicados encima de superficies tales como mesas.

El mapa de calor mostrado a continuación da una idea de la cobertura BLE en el escenario. Dicho mapa se ha obtenido normalizando la suma de los valores de RSSI de todas las balizas en cada punto, de tal manera que el color claro representa la mejor cobertura y, el color oscuro, la peor. A la izquierda se muestra el mapa con los valores reales medidos de RSSI. A la derecha se representan los valores obtenidos mediante simulación. Como era de esperar, el pasillo muestra la mejor cobertura. Nótese también que, para este escenario, la representación no está exenta de errores de interpolación.

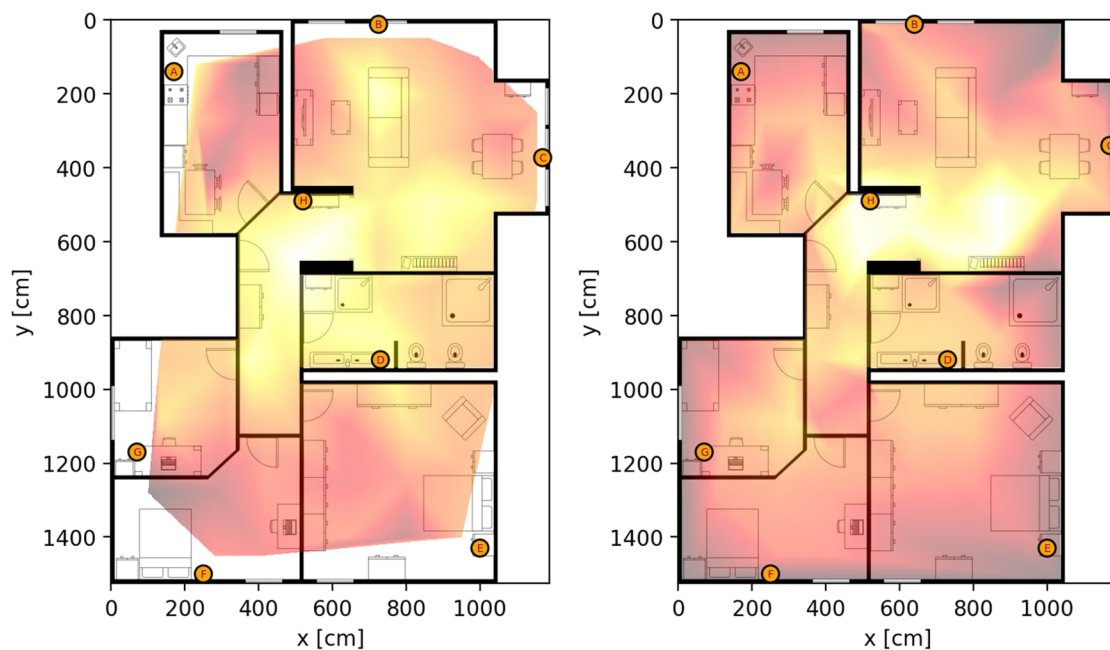


Figura 5.3 – Mapa de RSSI en Planta con datos reales (izquierda) y simulados (derecha)

### 5.3. Procesamiento de los datos

A diferencia del escenario en Sala, el tamaño de este escenario y los obstáculos en la propagación provocan efectos multicamino como el desvanecimiento de la señal. La propagación causa, por lo tanto, no sólo valores bajos de RSSI para aquellos puntos alejados de cada baliza, sino también una degradación de la tasa de paquetes recibidos (tramas de *advertisement*). La siguiente figura muestra a la derecha la tasa de paquetes recibidos de la baliza de la ubicación A (Beacon 1) en los puntos de *fingerprint*. A la izquierda se muestra el valor promedio de RSSI en los mismos puntos. Por ejemplo, la tasa de paquetes recibidos en algunos puntos del dormitorio 2 es menor al 10 %.



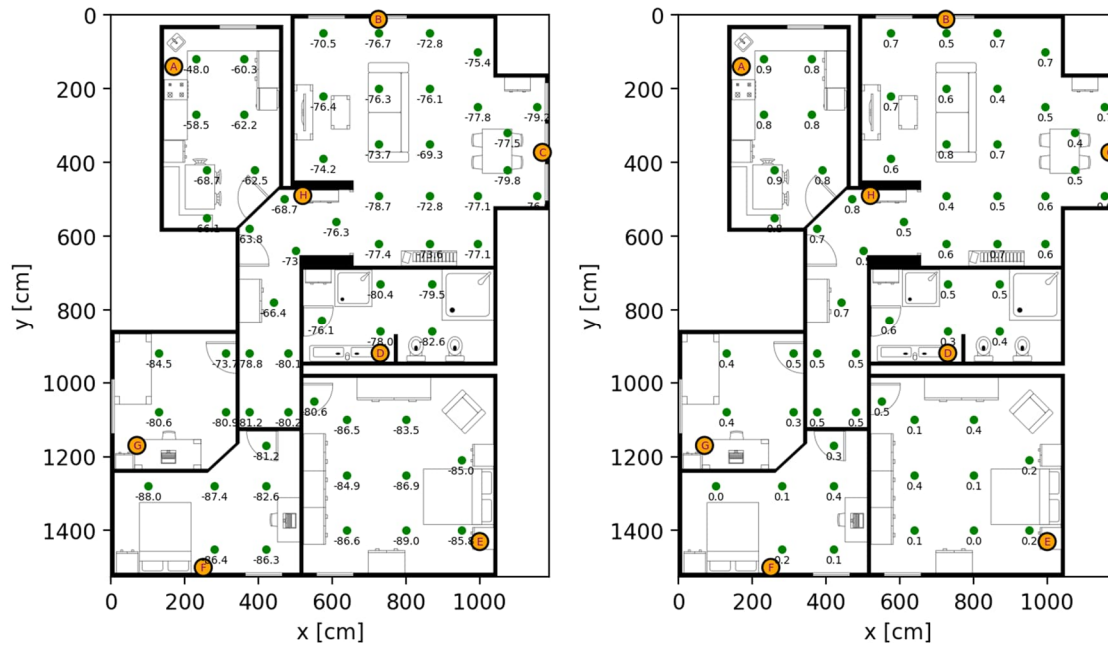


Figura 5.4 – Tasa de paquetes recibidos (izquierda) y valores de RSSI promedio (derecha) de Beacon 1

Este fenómeno implica la posibilidad de no recibir ninguna muestra de RSSI para algunas balizas durante el tiempo de integración, el cual se ha fijado en 500 ms para este escenario. La falta de muestras para algunas balizas no es, en general, un problema para la multilateración. Sí lo es, sin embargo, para el *fingerprinting*. Para solucionarlo, algunos trabajos aplican una máscara o lista negra de las balizas con peor señal según el área en la que se ubique el terminal [7]. De tal manera que el algoritmo de posicionamiento utiliza una constelación de balizas diferente según la estancia y, con ello, diferentes vectores de *fingerprint*. En este trabajo, por el contrario, se ha optado por utilizar un valor de RSSI de respaldo obtenido empíricamente para cada baliza. Dicho valor se añadirá en el algoritmo a falta de muestras de la baliza correspondiente durante el tiempo de integración. De esta manera, nuestro algoritmo se mantiene independiente del servicio de posicionamiento semántico.

Los valores de RSSI de respaldo utilizados son los listados a continuación:

Baliza (ubicación)	Valor de RSSI de respaldo [dBm]
Beacon 1 (A)	-90
Beacon 2 (B)	-90
Beacon 3 (C)	-85
Beacon 4 (D)	-80
Beacon 5 (G)	-85
Beacon 6 (H)	-75
Beacon 7 (F)	-75
Beacon 8 (E)	-75

Tabla 5.3 – Valores de RSSI de respaldo por cada baliza en Planta

### 5.3.1. Calibración

Los parámetros del modelo logarítmico de pérdidas por propagación (1) se han ajustado mediante las regresiones lineales mostradas en la siguiente figura. Las muestras utilizadas se corresponden con las del *dataset* de *fingerprints*. El valor de RSSI tomado para cada punto es la media aritmética de toda la captura.

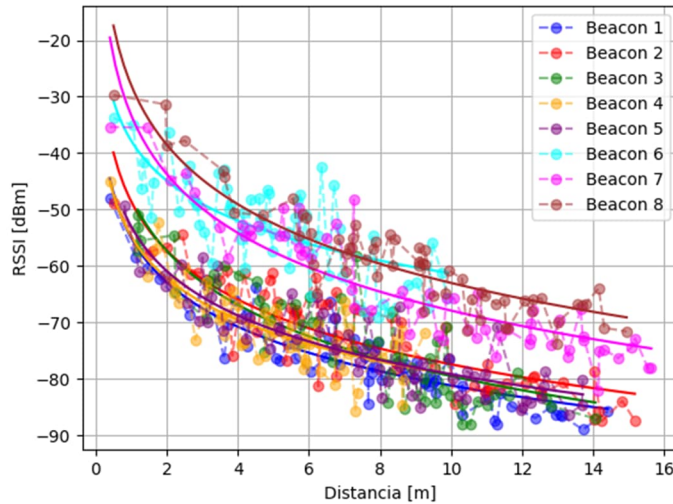


Figura 5.5 – Regresión lineal en Planta del modelo logarítmico de propagación

Los valores obtenidos para el factor de pérdidas y el valor de RSSI a 1 metro en cada baliza son los mostrados en la tabla:

Baliza (ubicación)	Factor de pérdidas (n)	RSSI <sub>0</sub> [dBm]
Beacon 1 (A)	2.62	-55.01
Beacon 2 (B)	2.88	-48.58
Beacon 3 (C)	3.17	-47.78
Beacon 4 (D)	2.45	-54.75
Beacon 5 (G)	2.62	-53.02
Beacon 6 (H)	2.35	-37.8
Beacon 7 (F)	3.46	-33.31
Beacon 8 (E)	3.5	-27.96

Tabla 5.4 – Calibración para la multilateración en Planta

La Figura 5.6 representa los valores de RSSI esperados por cada baliza en cada estancia del escenario.

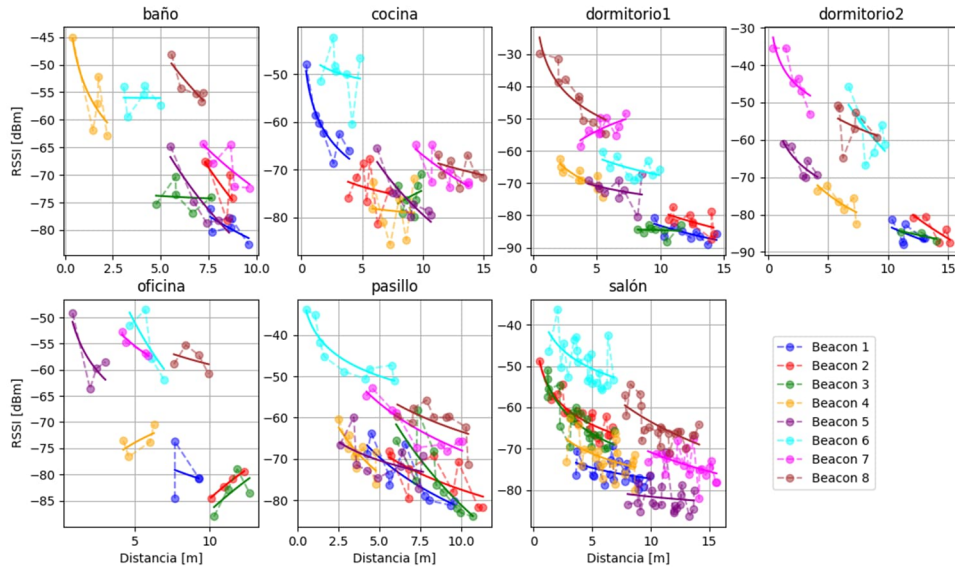


Figura 5.6 – Regresión lineal por cada estancia del escenario

### 5.3.2. Fingerprints

Tal y como se realizó para el escenario en Sala, los vectores de *fingerprint* se han obtenido promediando los valores de RSSI de cada baliza de manera análoga a cómo se construye el vector de observaciones (14). Previamente, filtramos las muestras para eliminar los valores atípicos por encima del percentil 90, y por debajo del percentil 10. Nótese, además, que los vectores se componen de ocho elementos para este escenario.

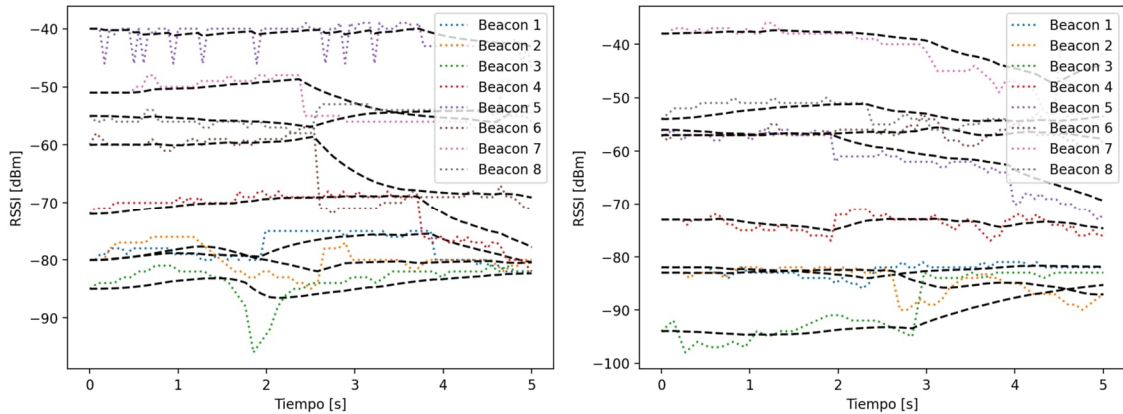


Figura 5.7 – Señales RSSI de dos puntos diferentes en Planta

### 5.3.3. Error de localización

El resultado de procesar los *datasets* de prueba se muestra a continuación. La técnica de multilateración utiliza mínimos cuadrados ponderados. La técnica de fingerprinting utiliza nuevamente k-Nearest Neighbors para un número de vecinos de 3, 4 y 5. Todos los casos se han probado con y sin el filtro de partículas. Es importante señalar que los muros internos del escenario no se han modelado. Es decir, desde el punto de vista del movimiento de las partículas, el escenario en

Planta no es más que una sala de mayor tamaño. La dos tablas siguientes muestran los resultados para el Terminal 1 y 2, respectivamente.

Técnica	Distancia ponderada	K	Filtro de partículas	MAE (m)	RMSE (m)	ID
Multilateración	sí	-	sí	1.32	1.54	fp_multi
			no	1.45	1.68	multi
Fingerprinting	sí	3	sí	1.29	1.56	fp_w3nn
			no	1.42	1.70	w3nn
		4	sí	1.28	1.53	fp_w4nn
			no	1.43	1.65	w4nn
		5	sí	1.29	1.51	fp_w5nn
			no	1.40	1.63	w5nn
	no	3	sí	1.31	1.58	fp_3nn
			no	1.44	1.72	3nn
		4	sí	1.31	1.55	fp_4nn
			no	1.43	1.69	4nn
5	sí	1.33	1.55	fp_5nn		
	no	1.45	1.68	5nn		

Tabla 5.5 – Errores de localización en Planta para el Terminal 1

Técnica	Distancia ponderada	K	Filtro de partículas	MAE (m)	RMSE (m)	ID
Multilateración	sí	-	sí	1.76	2.08	fp_multi
			no	1.78	2.12	multi
Fingerprinting	sí	3	sí	1.73	2.11	fp_w3nn
			no	1.76	2.15	w3nn
		4	sí	1.69	2.03	fp_w4nn
			no	1.72	2.08	w4nn
		5	sí	1.71	2.07	fp_w5nn
			no	1.75	2.12	w5nn
	no	3	sí	1.76	2.12	fp_3nn
			no	1.79	2.17	3nn
		4	sí	1.70	2.04	fp_4nn
			no	1.75	2.10	4nn
5	sí	1.75	2.10	fp_5nn		
	no	1.80	2.15	5nn		

Tabla 5.6 – Errores de localización en Planta para el Terminal 2

Los mejores resultados para ambos terminales se obtienen en este escenario, como era de esperar, con *fingerprinting*. Los resultados con multilateración, sin embargo, se encuentran en el mismo orden de magnitud, por lo que podemos determinar que es también una técnica viable en entornos domésticos sin línea de visión directa. El error promedio absoluto para el segundo terminal es aproximadamente 40 cm mayor; los resultados son comparativamente peores en el dormitorio 1, en el pasillo y en el salón. La siguiente figura compara el error de ambos terminales por estancia y muestra la distribución del error con la distancia.

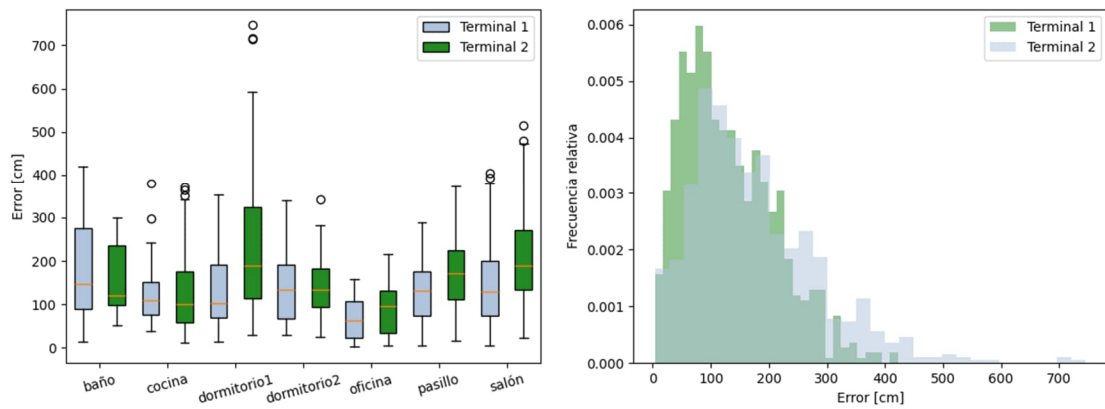


Figura 5.8 – Distribución del error en Planta para ambos terminales

Mostramos debajo la posición real de cada blanco (círculo azul) y su estimación (aspa roja si la estancia es incorrecta, aspa verde si la estancia es correcta) en el mejor y peor caso registrado para el Terminal 1 (izquierda y derecha de la figura, respectivamente). Obviamos la misma representación para el Terminal 2 al ser prácticamente idéntica a la mostrada.

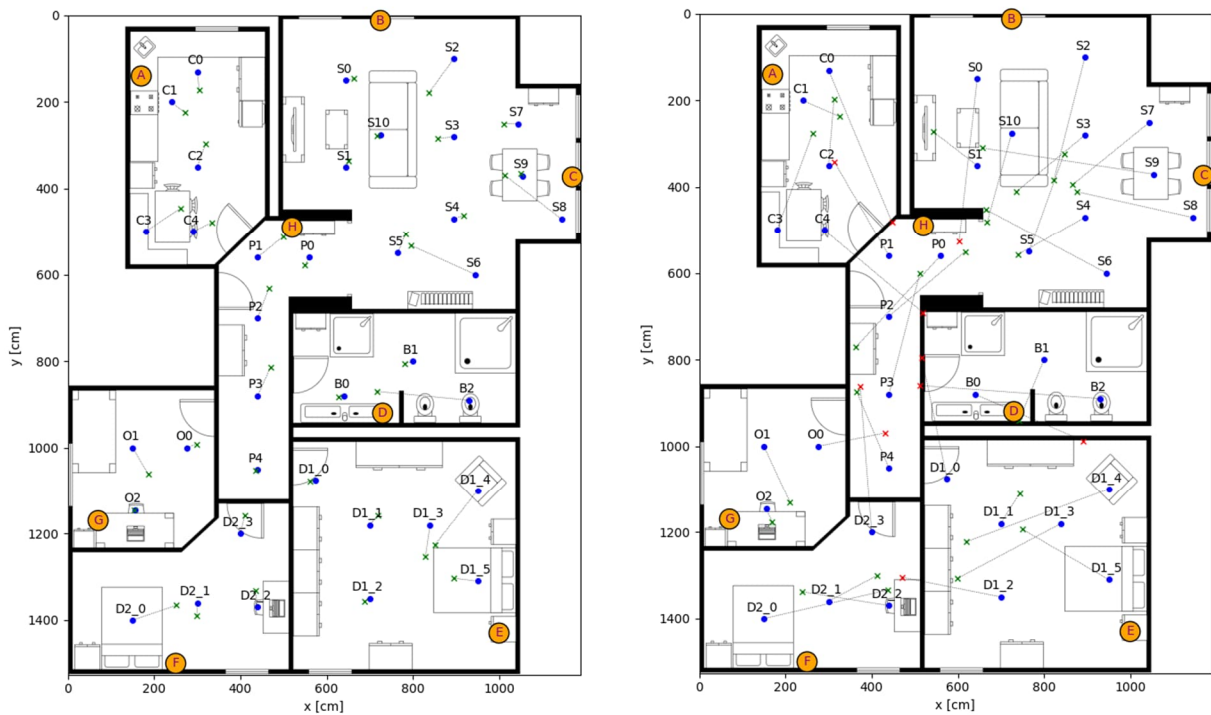


Figura 5.9 – Estimaciones en Planta para el Terminal 1 con el mejor caso (izquierda) y el peor (derecha)

### 5.3.4. Posicionamiento semántico

Para el posicionamiento semántico se ha entrenado el clasificador SVM con el *dataset* de *fingerprints*. Adicionalmente, para cada punto de *fingerprint*, se ha incluido en el entrenamiento dos vectores adicionales: el vector de RSSI con los valores de la media más la desviación estándar, y el vector de RSSI con los valores de la media menos la desviación estándar.

El kernel utilizado en el clasificador es lineal. El parámetro C de compensación se ha fijado en 0.001. A diferencia del tiempo de integración de 500 ms utilizado para el posicionamiento en el plano, el tiempo de integración para el algoritmo de posicionamiento semántico se ha fijado en 1 segundo. Este medio segundo adicional permite mejorar la precisión alrededor de 3 puntos porcentuales.

La tabla siguiente resume el desempeño del algoritmo para cada terminal con el *dataset* de blancos estáticos, el cual se ha usado como set de datos de validación:

Estancia	Terminal 1		Terminal 2	
	Muestras	Precisión [%]	Muestras	Precisión [%]
cocina	51	98	51	98
salón	110	98	111	100
pasillo	50	80	50	66
oficina	31	87	31	97
baño	30	83	30	97
dormitorio1	60	98	60	93
dormitorio2	40	85	40	75
Total	372	92	373	91

Tabla 5.7 – Resultados de clasificación de las estancias para ambos terminales

Las siguientes matrices de confusión representan los resultados de la clasificación para cada estancia en los dos terminales. A la izquierda se muestra por número de muestras del set de validación. A la derecha, el ratio de precisión.

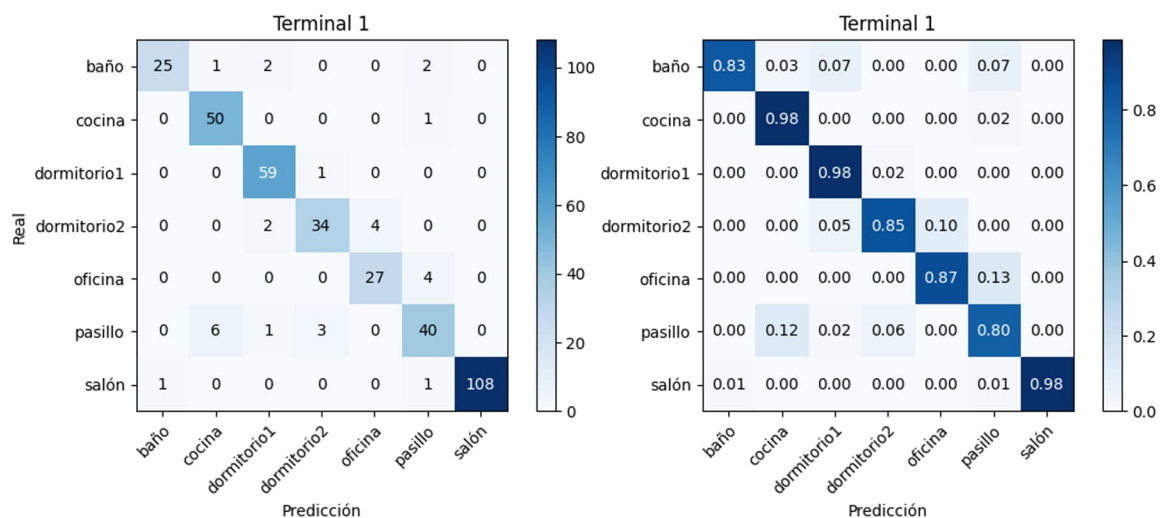


Figura 5.10 – Matriz de confusión del Terminal 1

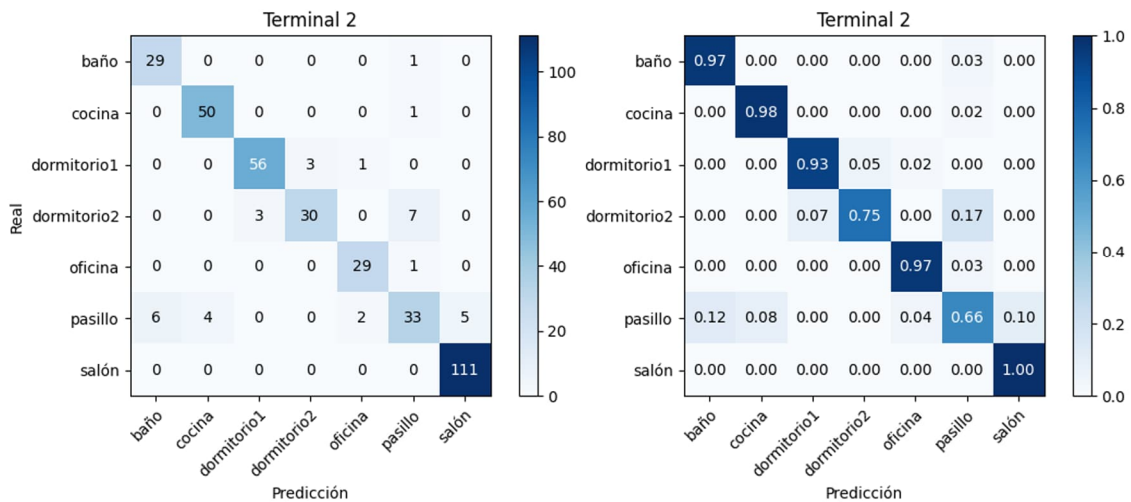


Figura 5.11 – Matriz de confusión del Terminal 2

Puede observarse que, para el Terminal 1, la tasa de error es siempre menor a 0.2 en todas las estancias, siendo la más problemática el pasillo, donde hasta un 12% de las muestras se clasifican erróneamente como pertenecientes a la cocina.

El Terminal 2, si bien cumple el objetivo de precisión global, tiene un mal desempeño para el dormitorio 2 y el pasillo. En el caso del dormitorio 2, el 17% de las muestras se clasifican erróneamente como pertenecientes al pasillo. En el caso del pasillo, el 12% se clasifican como parte del baño, el 10% como parte del salón, y el 8% como pertenecientes a la cocina.

### 5.3.5. Seguimiento del terminal

Hasta ahora todas las pruebas se han realizado para blancos estáticos, es decir, manteniendo el terminal ininterrumpidamente en el mismo punto para cada captura. Para el escenario en Planta se han realizado tres capturas adicionales, esta vez con el terminal en movimiento. El propósito es evaluar el rendimiento del algoritmo de posicionamiento durante el proceso en el que el usuario del servicio cambia su posición.

Las capturas se han realizado por tramos, utilizando puntos de *fingerprint* como *waypoints* para así tener referencias de la trayectoria seguida por el terminal sobre el terreno. Las trayectorias de prueba seguidas son las listadas a continuación:

1. Trayectoria desde el dormitorio 2 hasta la cocina (recorriendo el pasillo) durante aproximadamente 17 segundos.
2. Trayectoria desde el salón hasta el dormitorio 1 (recorriendo el pasillo) durante aproximadamente 20 segundos.
3. Trayectoria desde la oficina hasta el baño (atravesando el pasillo) durante aproximadamente 10 segundos.

Los resultados del procesado se representan a continuación para el caso de un tiempo de integración de 500 ms y la técnica de *fingerprinting* ponderado con 4 vecinos y filtro de partículas (fp\_w4nn). Como puede observarse, el seguimiento de la trayectoria es errático. Sin embargo, el error promedio del posicionamiento se

encuentra en el mismo orden de magnitud que el obtenido para las capturas estáticas. Para la tercera trayectoria, además, se muestra el resultado aplicando un tiempo de integración de 1 segundo y un ciclo de refresco de 2 segundos (se actualiza la posición en la interfaz cada 2 segundos).

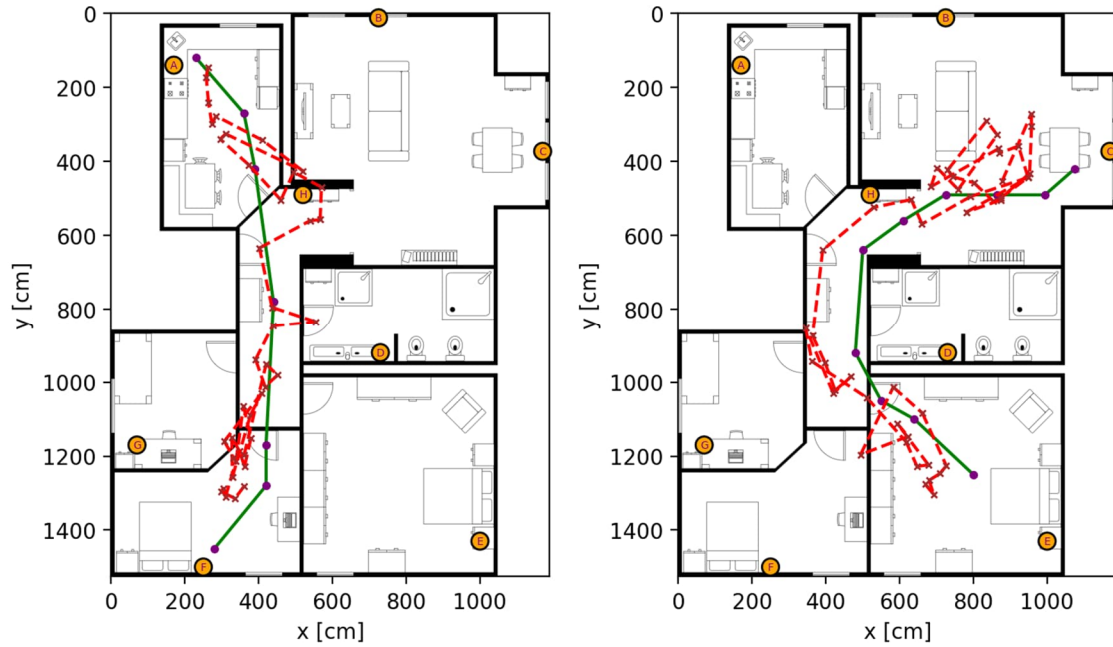


Figura 5.12 – Seguimiento del terminal en dos trayectorias: del dormitorio 2 a la cocina (izquierda), y del salón al dormitorio 1 (derecha).

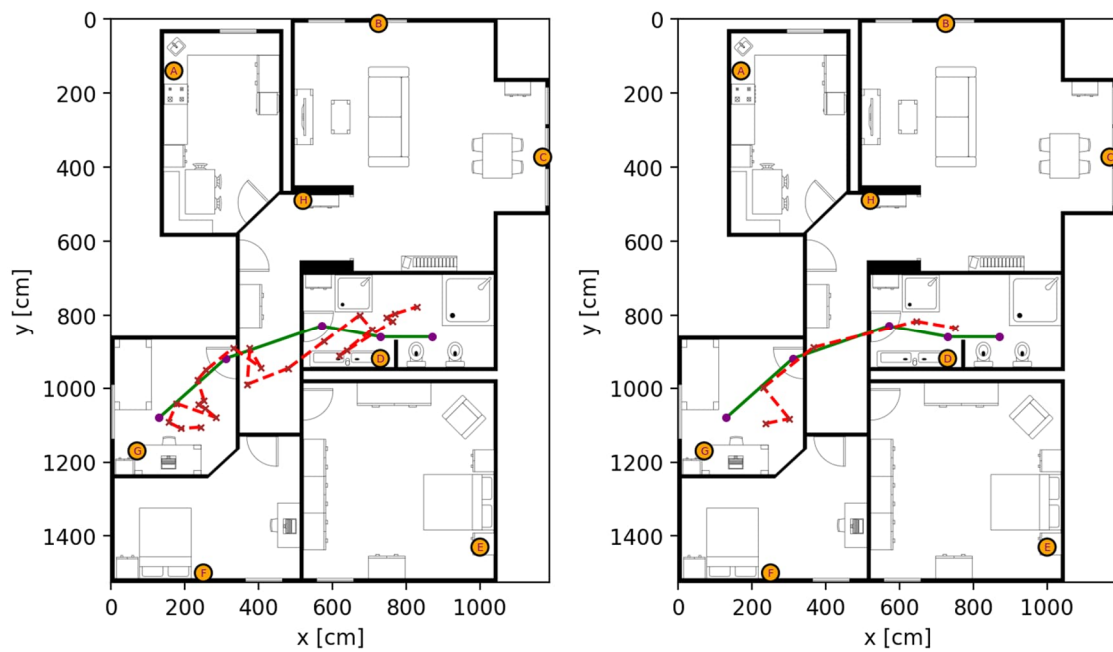


Figura 5.13 – Seguimiento del terminal en la trayectoria de la oficina al baño. A la izquierda, con tiempo de integración de 500 ms. A la derecha, con 1 s.



## 6. Resultados y validación del sistema

Recopilamos en este capítulo los resultados de la experimentación con el prototipo del sistema desarrollado para así cotejarlos con los objetivos de diseño planteados al inicio del proyecto.

Objetivo	Método de verificación	Umbral	Resultado
Error del RTLS	Testeo	< 2.5 m	~ 1.5 m
Uso de ToF	Inspección	-	NO
Producción de datasets	Inspección	-	SÍ
Latencia	Testeo	< 100 ms	~ 40 ms
Error de clasificación	Testeo	< 20 %	~ 9 %
Tecnologías estándar	Inspección	-	SÍ

Tabla 6.1 – Resultados cotejados con los objetivos del proyecto

Podemos comprobar que todos los objetivos se han cumplido exceptuando el de inclusión de medidas de ToF en el esquema de posicionamiento, lo cual se ha justificado en la [sección 3.3.5](#). Para los resultados de latencia se ha medido el tiempo que tarda el servidor web en atender una solicitud de posicionamiento proveniente del terminal. Los resultados del tiempo de procesado del esquema de posicionamiento se representan a continuación:

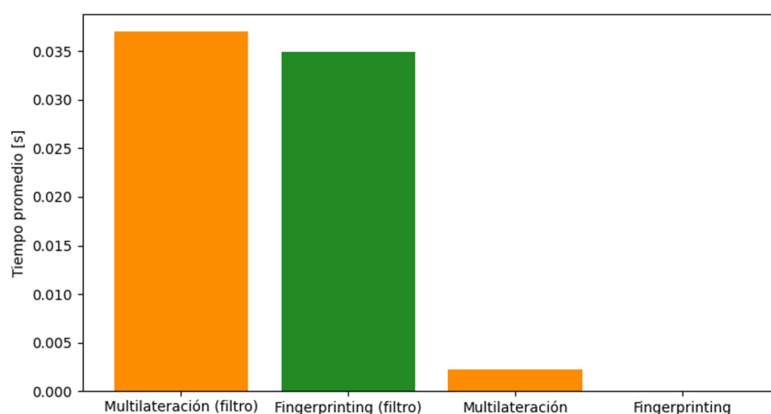


Figura 6.1 – Latencia promedio del servicio de posicionamiento

Los esquemas que utilizan el filtro de partículas son, por ello, varios órdenes de magnitud más costosos de computar que el resto, necesitando un tiempo promedio de hasta 37 ms por solicitud para una colección de 500 partículas. En el prototipo se ha mantenido el estado del filtro de partículas en memoria. En un entorno en producción, el estado debería persistirse por algún servicio distribuido (p.ej. Redis). Esto añadiría un retardo adicional a considerar en el procesamiento.

Con todo ello, podemos validar el diseño del sistema asumiendo la representatividad de los resultados obtenidos con el prototipo implementado.

## 7. Conclusiones y trabajo futuro

Este Trabajo Final de Máster tenía como objetivo primordial el desarrollo de un sistema de posicionamiento en tiempo real con un error de localización inferior a 2.5 m utilizando la tecnología Bluetooth Low Energy así como técnicas en el estado del arte. Dicho objetivo se ha cumplido con un amplio margen mediante el desarrollo, despliegue y testeo de un prototipo del sistema en dos escenarios domésticos, los denominados Sala y Planta. El error promedio absoluto (MAE) obtenido en Sala se encuentra aproximadamente entre 1.15 y 1.50 m, mientras que el error en Planta se encuentra entre 1.30 y 1.70 m.

Otro objetivo principal del proyecto era el incluir medidas de tiempo como ToF en el esquema de posicionamiento. Este objetivo no se ha cumplido debido a limitaciones técnicas del *stack* software elegido. En concreto, la API de Android no proporciona control sobre el mecanismo de *Scan Request* y *Scan Response* del modo de *advertisement* de BLE.

De cara a los objetivos secundarios, todos ellos se han conseguido llevar a cabo, obteniendo los *datasets* planteados y consiguiendo un sistema con baja latencia, posicionamiento semántico preciso y desarrollado con tecnologías estándar.

Con esto, hemos validado el diseño propuesto, cuyo punto fuerte es la separación del kernel de posicionamiento (multilateración, *fingerprinting*, etc) del filtro de partículas, el cual mejora la precisión de la estimación final. Además, consideramos que hemos establecido las bases de una arquitectura basada en servicios web altamente extensible, mantenible y actualizable.

Se contemplan las siguientes líneas de trabajo futuras:

1. Uso de medidas de ToF (Time of Flight): cumplir con el objetivo de fusionar medidas de tiempo y RSSI en el esquema de posicionamiento.
2. Métodos de corrección del sesgo del sistema: implementar métodos de mejora de la exactitud del sistema en base a la compensación de las desviaciones de las predicciones.
3. Fusión de datos con el sensor inercial de los terminales móviles: lo que permitiría mejorar la precisión del sistema así como el seguimiento de los terminales.
4. Métodos de autocalibración o calibración guiada de nuevos terminales: desarrollar procedimientos automáticos o semi-automáticos de calibración de nuevos dispositivos usuarios del sistema.
5. Métodos de optimización de la ubicación de las balizas y parámetros del sistema: sería conveniente llevar a cabo un trabajo de optimización previo al despliegue mediante, por ejemplo, el uso de algoritmos genéticos sobre el bucle de simulación. De esta manera se seleccionarían tanto las ubicaciones de las balizas como los parámetros del algoritmo que maximicen las prestaciones del sistema.

## 8. Bibliografía

- [1] Y. Shen, B. Hwang and J. P. Jeong, "Particle Filtering-Based Indoor Positioning System for Beacon Tag Tracking," in *IEEE Access*, vol. 8, pp. 226445-226460, 2020, doi: 10.1109/ACCESS.2020.3045610.
- [2] Luca, Dierna Giovanni. "Towards accurate indoor localization using iBeacons, fingerprinting and particle filtering." (2016).
- [3] Safwat, Rokaya & Shaaban, Eman & Altabbakh, Shahinaz & Emara, Karim. (2023). "Fingerprint-based indoor positioning system using BLE: real deployment study". *Bulletin of Electrical Engineering and Informatics*. 12. 240-249. 10.11591/eei.v12i1.3798.
- [4] Paterna, Vicente Cantón, Anna Calveras Augé, Josep Paradells Aspas and María A Bullones. "A Bluetooth Low Energy Indoor Positioning System with Channel Diversity, Weighted Trilateration and Kalman Filtering." *Sensors (Basel, Switzerland)* 17 (2017): n. pag.
- [5] Davide Giovanelli, Elisabetta Farella. "RSSI or Time-of-flight for Bluetooth Low Energy based localization? An experimental evaluation". 11th IFIP Wireless and Mobile Networking Conference (WMNC 2018), Sep 2018, Prague, Czech Republic. pp.32-39. hal-01995171.
- [6] Davide Giovanelli, Elisabetta Farella, Daniele Fontanelli, David Macii. "Bluetooth-based Indoor Positioning through ToF and RSSI Data Fusion". *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 24-27 September 2018, Nantes, France.
- [7] García-Paterna, P.J.; Martínez-Sala, A.S.; Sánchez-Aarnoutse, J.C. "Empirical Study of a Room-Level Localization System Based on Bluetooth Low Energy Beacons". *Sensors* 2021, 21, 3665. <https://doi.org/10.3390/s21113665>.
- [8] Ramirez, R.; Huang, C.-Y.; Liao, C.-A.; Lin, P.-T.; Lin, H.-W.; Liang, S.-H. "A Practice of BLE RSSI Measurement for Indoor Positioning". *Sensors* 2021, 21, 5181. <https://doi.org/10.3390/s21155181>.
- [9] Lu Bai, Fabio Ciravegna, Raymond R. Bond, Maurice D. Mulvenna. "A Low-Cost Indoor Positioning System Using Bluetooth Low Energy". (2020) *IEEE Access*, 8. pp. 136858-136871. <https://doi.org/10.1109/ACCESS.2020.3012342>.
- [10] Liu, Liu, Bofeng Li, Ling Yang, and Tianxia Liu. 2020. "Real-Time Indoor Positioning Approach Using iBeacons and Smartphone Sensors" *Applied Sciences* 10, no. 6: 2003. <https://doi.org/10.3390/app10062003>.
- [11] A. Khalajmehrabadi, N. Gatsis and D. Akopian, "Modern WLAN Fingerprinting Indoor Positioning Methods and Deployment Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1974-2002, thirdquarter 2017, doi: 10.1109/COMST.2017.2671454.

- [12] Kawecki, R.; Hausman, S.; Korbel, P. "Performance of Fingerprinting-Based Indoor Positioning with Measured and Simulated RSSI Reference Maps". *Remote Sens.* 2022, 14, 1992. <https://doi.org/10.3390/rs14091992>.
- [13] Bonavolontà, F.; Liccardo, A.; Schiano Lo Moriello, R.; Caputo, E.; de Alteriis, G.; Palladino, A.; Vitolo, G. "An Improved Method Based on Bluetooth Low-Energy Fingerprinting for the Implementation of PEPS System". *Sensors* 2022, 22, 9615. <https://doi.org/10.3390/s22249615>.
- [14] Proximity Beacon Specification. Release R1. Apple.
- [15] Bluetooth Core Specification - Version 4.2 [Vol 6, Part B] – Subsection 4.4.2.2.
- [16] Koen Vervloesem. "Develop your own Bluetooth Low Energy Applications for Raspberry Pi, ESP32 and nRF52 with Python, Arduino and Zephyr". Elektor Publication. ISBN 978-3-89576-501-8.
- [17] <https://c4model.com/>.
- [18] <https://github.com/google/eddystone/tree/master>.
- [19] Q. Li, R. Li, K. Ji and W. Dai, "Kalman Filter and Its Application," *2015 8th International Conference on Intelligent Networks and Intelligent Systems (ICINIS)*, Tianjin, China, 2015, pp. 74-77, doi: 10.1109/ICINIS.2015.35.
- [20] Kitagawa, G. (January 1993). "A Monte Carlo Filtering and Smoothing Method for Non-Gaussian Nonlinear State Space Models". *Proceedings of the 2nd U.S.-Japan Joint Seminar on Statistical Time Series Analysis*: 110–131.
- [21] <https://fastapi.tiangolo.com/>.
- [22] <https://streamlit.io/>.
- [23] <https://mynewt.apache.org/>.
- [24] B. Delaunay, "Sur la sphère vide. A la mémoire de Georges Voronoï", *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 1934, no. 6, 793–800.

## Anexo A. Estudio de viabilidad

Rescatamos en este anexo, con corrección de erratas, el estudio de viabilidad entregado como parte de la PEC2 (segunda Prueba de Evaluación Continua) el pasado 2 de abril de 2024:

Para este estudio preliminar de viabilidad se ha preparado un entorno de simulación que permite simular el despliegue de diferentes combinaciones de balizas y la presencia de un receptor en el escenario denominado "sala", el cual será el primer escenario de experimentación en el proyecto.

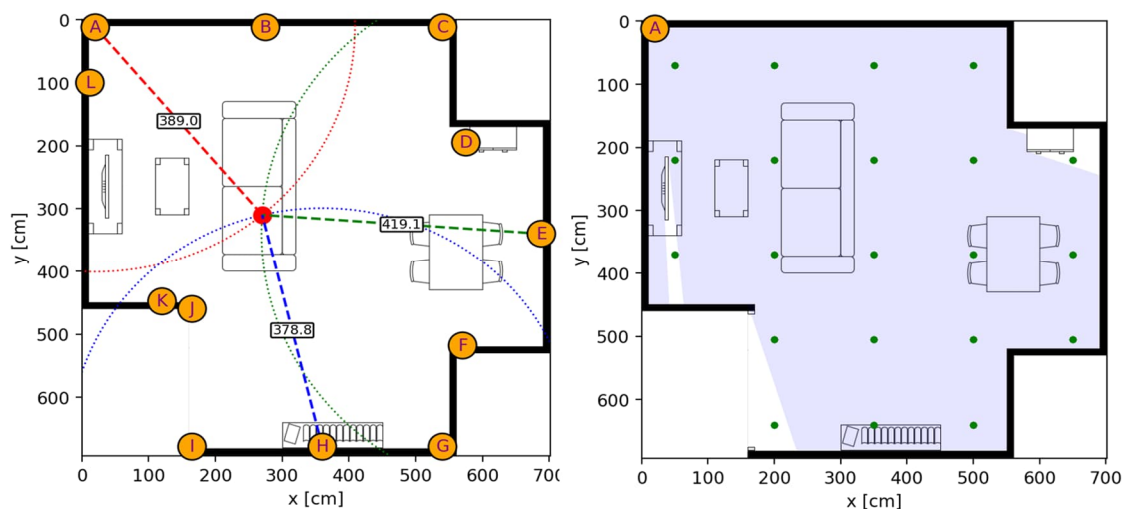
El entorno de simulación implementa un algoritmo de multilateración por mínimos cuadrados ponderados y un algoritmo basado en fingerprinting. Se han simulado medidas de RSSI basadas en el modelo de propagación *Log-Normal Shadowing Path Loss Model*, el cual se rige por la ecuación (1), donde la distancia de referencia se ha asumido a 1 m, y un valor RSSI de referencia de -50 dBm; y medidas de ToF según el modelo lineal de la ecuación (2), donde  $c$  es la velocidad de propagación de la luz en el vacío, el valor de distancia de referencia,  $d_0$ , es 1 m, y el valor de ToF de referencia es igual a 1 ms.

Nótese que los valores de referencia han de obtenerse empíricamente tras el despliegue de las balizas en los escenarios reales. Además, las dos ecuaciones presentan un ruido gaussiano aditivo.

$$RSSI(d) = RSSI_0 - 10 \cdot n \cdot \log\left(\frac{d}{d_0}\right) + X_\sigma \quad (1)$$

$$ToF(d) = ToF_0 + \frac{d-d_0}{c} + X_\sigma \quad (2)$$

Para el factor de pérdida por propagación,  $n$ , de la ecuación (1), se tiene en cuenta en la simulación si el receptor está en la línea de visión de la baliza. La línea de visión se calcula por *raycasting* desde cada posición candidata para las balizas.



La siguiente simulación se basa en un despliegue de 4 balizas en las posiciones: A, E, H y J (las posiciones no se han optimizado aún para esta PEC2) y un mallado de receptores uniforme localizado principalmente en el centro de la sala. La

simulación considera ruido gaussiano sobre la medida de RSSI con una desviación estándar de 5 y 10 dB. Para la medida de ToF se considera una desviación estándar de 100  $\mu$ s debido principalmente a la desincronización. Los resultados a grandes rasgos se presentan en la tabla siguiente:

Parámetro	Ruido ( $\sigma$ )	Algoritmo	RMSE [m]	MAE [m]
RSSI	5 dB	Fingerprinting	1.32	0.86
RSSI	5 dB	Multilateración	2.08	1.09
RSSI	10 dB	Fingerprinting	1.6	1.17
RSSI	10 dB	Multilateración	2.24	1.39
ToF	100 $\mu$ s	Fingerprinting	2.14	1.75
ToF	100 $\mu$ s	Multilateración	> 5 m	> 5 m

Si las simulaciones son representativas, la solución planteada cumpliría con cada uno de los objetivos definidos si se utiliza un algoritmo basado en fingerprinting. Además, se ha de probar el uso de algoritmos híbridos que combinen diferentes parámetros.

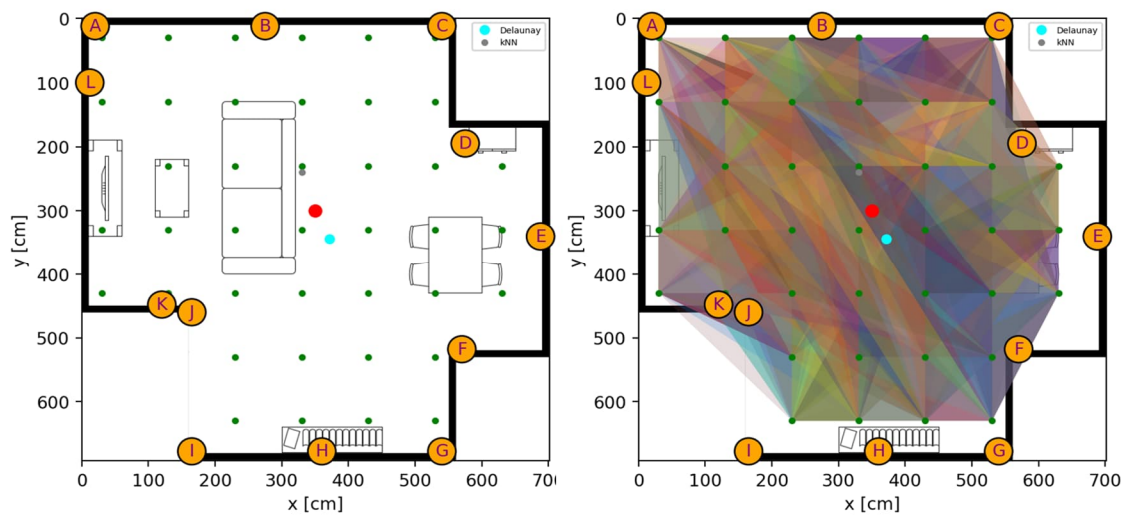
Sin embargo, es importante resaltar que, según el estándar Bluetooth 4.2, al intervalo fijo de transmisión de tramas de *advertisement* se le añade un retardo aleatorio adicional de entre 0 y 10 ms [15] para evitar colisiones en el canal (BLE no deja de ser, a fin de cuentas, un protocolo de comunicación, no de radiolocalización). Si este retardo no es medible a través de las APIs de Bluetooth de las balizas, o si no se introduce algún mecanismo corrector, este retardo introducirá un ruido adicional en la medida de tiempo en el terminal receptor de un orden de magnitud superior al valor de ToF. Esto puede suponer un error de posicionamiento que convierta en inviable una solución híbrida basada en medidas de tiempo. El estudio de viabilidad no es concluyente para el parámetro ToF (cualquier medida de tiempo, en general), por lo que se requerirán estudios empíricos durante el desarrollo del proyecto para calibrar, parametrizar y evaluar el modelo de estimación de ToF.

Consideramos los siguientes riesgos tras el estudio de viabilidad:

Riesgo	Consecuencia	Probabilidad	Impacto	Mitigación
Medida imprecisa de tiempo	Imposibilidad de utilizar medidas de tiempo. Esfuerzo destinado a mejora algorítmica sobre RSSI.	Alta	Alto	Estudio del estado del arte para probar mecanismos alternativos de estimación.
Calibración imprecisa	Aumento de los errores de posicionamiento en los algoritmos.	Baja	Alto	Incrementar esfuerzo en la fase de calibración.
Mala calidad del hardware	Baja cobertura BLE, aumento de errores de posicionamiento.	Media	Medio	No actuar.

La forma de solventar los riesgos mencionados no es otra que la de considerar un proyecto mínimo viable que sea capaz de cumplir con el objetivo principal o, en su defecto, demostrar que se han alcanzado los límites tecnológicos según el uso de BLE en el contexto del proyecto. Consideramos como proyecto mínimo viable el desarrollo de un sistema RTLS (Real-Time Location System) basado en RSSI mediante una tecnología estándar como iBeacon. En este contexto, se explicaría en la memoria del TFM las limitaciones técnicas encontradas para utilizar medidas de tiempo, y se propondrían líneas de trabajo para solventarlas, pero el enfoque principal caería en el desarrollo del RTLS.

Parte de los algoritmos de posicionamiento adoptados para la solución híbrida (fusión de RSSI con medidas de tiempo tales como TDoA y/o ToF) podrían reutilizarse para el proyecto mínimo viable. Por ejemplo, la siguiente imagen muestra una simulación basada en *RSSI fingerprinting* en la cual se prueban dos técnicas de estimación, a saber: una interpolación basada en un mallado Delaunay multidimensional, y una ponderación por kNN (k-Nearest Neighbors):



## Anexo B. Repositorios

Los repositorios con el código fuente de los diferentes elementos software desarrollados en este Trabajo Final de Máster se pueden encontrar en los enlaces siguientes:

- Datasets, librería de modelos y algoritmos, aplicación web y parte de la aplicación Streamlit: <https://github.com/alelince/tfm-ips>
- Aplicación Android: <https://github.com/alelince/tfm-app>
- Balizas BLE: <https://github.com/alelince/tfm-beacon>