PROYECTO FIN DE GRADO

TÍTULO: Sistema de Posicionamiento en Interiores basado en Arduino utilizando Bluetooth Low Energy (BLE)

AUTOR/A: Daniel Porras Fernández

TITULACIÓN: Grado en Ingeniería Electrónica de Comunicaciones

DIRECTOR/A: Ferrán Blasco Pauls

TUTOR/A: Gregorio Rubio Cifuentes

DEPARTAMENTO: DTE

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Marta Sánchez Agudo

TUTOR/A: Gregorio Rubio Cifuentes

SECRETARIO/A: Vicente Hernández Díaz

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

El posicionamiento en interiores de personas, animales y objetos ha experimentado avances significativos, impulsado especialmente por el auge del Internet de las Cosa (IoT), esto ha supuesto una demanda creciente en servicios centrados en la localización. Aunque el GPS ha sido la tecnología principal para este propósito, su uso en espacios interiores se ve limitado debido a la pérdida de señal del satélite recibida por el dispositivo a localizar. Para contrarrestar estas limitaciones en espacios de interior, se están desarrollando tecnologías complementarias al GPS, como la navegación inercial, el posicionamiento basado en Wi-Fi o Bluetooth, y la integración de sensores adicionales en los dispositivos para mejorar la precisión de la ubicación en estos entornos.

Para la realización de este proyecto se ha centrado el desarrollo en dos metodologías de localización: trilateración y “fingerprinting”. Estos métodos utilizan la tecnología Bluetooth de baja energía (BLE) y la intensidad de la señal recibida (RSSI) para estimar la posición. Para su implementación se utilizan balizas BLE que se distribuyen por el entorno, y un dispositivo ESP32 programado en Arduino que recoge las señales. Estas señales son procesadas posteriormente de forma ‘offline’ en un ordenador.

Se ha seleccionado un dispositivo con microprocesador ESP32 por su capacidad de procesamiento, recursos y software existente. El ESP32 es empleado para recoger señales BLE y guardar la información en ficheros CSV. Debido a la tarea sencilla que va a tener este microprocesador, no se ha visto la necesidad de emplear otras opciones más complejas como pueden ser los STM32 de STMicroelectronics.

En cuanto al desarrollo del sistema de posicionamiento, como primera opción de sistema se utiliza el método de la trilateración. Este método utiliza el cálculo de distancia en metros a partir del RSSI. No obstante, los resultados no han sido los esperados en cuanto a precisión del sistema, debido a las fluctuaciones de las señales RSSI provenientes de las balizas BLE.

Para suavizar estas fluctuaciones se implementó un filtro de Kalman, que sí reduce esas fluctuaciones, pero que se queda corto en el proceso de filtrado ante la aleatoriedad de estas fluctuaciones en los valores de RSSI.

Para solventa esta situación, se planteó la solución al problema del posicionamiento usando método de “fingerprinting”, en un intento de desarrollar un sistema con mayor grado de precisión y con mayor robustez a las fluctuaciones en las señales BLE. Este sistema requiere de una primera fase de creación de una base de datos que no requiere el método de trilateración, donde se mapea el área en su totalidad. En su segunda fase es capaz de posicionar al objetivo en el entorno con un rango de precisión aceptable, limitado por el mapeo realizado en la fase 1. De este modo, si se realiza un mapeo del entorno cada 2 metros de separación, se va a tener un error de 2 metros, y asi sucesivamente.

Este método compara mediante un algoritmo clasificador las señales recogidas durante la ruta realizada por el ESP32 con los valores de la base de datos creada durante el mapeo del

entorno, y devuelve la posición de la ruta más cercana a los valores registrados durante el recorrido.

Al avanzar en el desarrollo, se valoró el implementar una interfaz de usuario sencilla para acompañar al método de “fingerprinting” desarrollado. Por ello se ha proporcionado un cliente de escritorio para Windows con el objetivo de que un usuario final de este sistema de posicionamiento no se vea obligado a instalarse todo un entorno de programación para poder utilizarlo. Este cliente está desarrollado en Python, se ha encapsulado todo el código en un ejecutable y se ha implementado una interfaz de usuario sencilla, dentro de las dimensiones del proyecto, para su uso.

En resumen, el objetivo de este Proyecto Fin de Grado plantea el diseño, caracterización y desarrollo de un sistema de posicionamiento en interiores basado en la tecnología Bluetooth BLE, con el fin de aprovechar sus ventajas en términos de bajo consumo de energía, bajo costo y compatibilidad con dispositivos modernos.

Abstract

Indoor positioning of people, animals and objects has experienced significant advancements, driven especially by the rise of the Internet of Things (IoT), this has led to a growing demand for location-centric services. Although GPS has been the main technology for this purpose, its use in indoor spaces is limited due to the loss of satellite signal received by the device to be located. To counteract these limitations in indoor spaces, complementary technologies to GPS are being developed, such as inertial navigation, Wi-Fi or Bluetooth-based positioning, and the integration of additional sensors into devices to enhance location accuracy in these environments.

To carry out this project, the development has focused on two localization methodologies: trilateration and fingerprinting. These methods use Bluetooth Low Energy (BLE) technology and Received Signal Strength Indicator (RSSI) to estimate position. For its implementation, BLE beacons are used and are distributed throughout the environment, and an ESP32 device programmed in Arduino collects the signals. These signals are subsequently processed offline on a computer.

A device with an ESP32 microprocessor has been selected due to its processing capacity, resources and existing software. The use of this ESP32 is to collect BLE signals and save the information in CSV files. Due to the simple task that this microprocessor will have, there has not been a need to use other more complex options such as the STM32 from STMicroelectronics

Regarding the development of the positioning system, the trilateration method is used as the first system option. This method uses the calculation of distance in meters from the RSSI. However, the results have not been as expected in terms of system accuracy, due to the fluctuations of the RSSI signals coming from the BLE beacons.

To smooth out these fluctuations, a Kalman filter was implemented, which does reduce these fluctuations, but falls short in the filtering process due to the randomness of these fluctuations in the RSSI values.

To solve this situation, the solution to the positioning problem was proposed using the fingerprinting method, in an attempt to develop a system with a higher degree of precision and greater robustness to fluctuations in BLE signal. This system requires a first phase of creating a database that does not requires the trilateration method, where the area is mapped in its entirety. In its second phase the system is capable of positioning the target in the environment with an acceptable range of precision, limited by the mapping carried out in phase one. In this way, if a mapping of the environment is carried out every 2 meters of separation, it will have an error of 2 meters, and so on.

This method compares, using a classifier algorithm, the signals collected during the route carried out by the ESP32 with the values in the database created during the mapping of the

environment, and returns the position of the route closest to the values recorded by the ESP32.

As development progressed, it was valued to implement a simple user interface to accompany the “fingerprinting” method developed. For this reason, a desktop client for Windows has been provided so that an end user of this positioning system is not forced to install an entire programming environment to be able to use it. This client is developed in Python, all the code has been encapsulated in an executable and a simple user interface has been implemented, within the dimensions of the project, for its use.

In summary, the objective of this Final Degree Project is the design, characterization and development of an indoor positioning system based on Bluetooth BLE technology, to take advantage of its advantages in terms of low energy consumption, low cost and Compatibility with modern devices.

Índice de figuras

Índice de tablas

Índice de diagramas

Lista de acrónimos

ATT - Attribute Protocol

BLE - Bluetooth Low Energy

GATT - Generic Attribute Profile

GFSK - Gaussian Frequency Shift Keying

I2C - Inter-Integrated Circuit

IoT - Internet of Things

L2CAP - Logical Link Control and Adaptation Protocol

MEMS - Microelectromechanical Systems

NTP - Network Time Protocol

ODS - Objetivos de Desarrollo Sostenible

RSSI - Received Signal Strenght Indicator

RTC - Real-Time Clock

SIG - Special Interest Group

SoC - System on Chip

TFT - Thin Film Transistor

Índice de contenidos

1. Introducción

Uno de los avances tecnológicos más relevantes que ha transformado nuestra sociedad ha sido la disponibilidad de los sistemas de posicionamiento utilizando GPS. Este avance ha cambiado radicalmente nuestra forma de desplazarnos y localizar objetos, animales o personas. Hoy en día, lo sistemas de posicionamiento basados en GPS son una tecnología madura y ampliamente utilizada.

Sin embargo, el GPS no funciona bien en recintos interiores, donde la señal de los satélites no alcanza. Con al auge del *Internet of Things* (IoT) en los últimos años, han surgido nuevas tecnologías que permiten implementar sistemas de posicionamiento en interiores, utilizando distintos dispositivos y técnicas.

Uno de los principales desafíos de los sistemas de posicionamiento en interior es la precisión. Es relativamente sencillo determinar si un objeto está en una habitación, pero indicar su posición exacta es más complicado. Este desafío se incrementa cuando el objeto está en movimiento. Este proyecto aborda precisamente este último caso.

El objetivo de este Proyecto Fin de Grado es desarrollar un sistema de posicionamiento en interiores para objetos móviles en ambientes diáfanos, que sea de bajo costo y alta precisión. Para ello, se han analizado varios aspectos de los sistemas de posicionamiento, como tecnologías, dispositivos y algoritmos de cálculo de posición y filtrado de señal.

Dos de los dominios donde se pretende aplicar este sistema son:

- Deporte: Para analizar el recorrido de un deportista en la pista de juego (baloncesto, balonmano, tenis, hockey, etc.).
- Ganadería: Para observar el comportamiento de animales en corrales en diferentes circunstancias de salud y medioambientales.

La solución propuesta utiliza la tecnología Bluetooth Low Energy (BLE) como protocolo de comunicación, un dispositivo de bajas capacidades ESP32 basado en Arduino como dispositivo de adquisición de datos, balizas (o *beacons)* BLE distribuidos por el ambiente donde se desea instalar el sistema de posicionamiento, el empleo de la técnica de "fingerprinting" o huella dactilar junto a un algoritmo KNN (K-Nearest neighbor) para determinar las posibles posiciones, y un cliente de escritorio desarrollado en Python para adquirir los datos del ESP32 y procesar la información empleando la técnica "fingerprinting" y el filtro KNN.

El desarrollo del proyecto está más orientado al desarrollo software que al hardware, debido a las limitaciones temporales. Por ello se ha empleado una versión de desarrollo de un ESP32 que cuenta ya con una serie de módulos integrados, aunque se ha visto la necesidad de añadir algún módulo adicional como se explica a lo largo de esta memoria.

De esta manera, el tiempo se ha dedicado al desarrollo del sistema global y no sólo al desarrollo hardware de un dispositivo cuya función principal es recoger señales Bluetooth, y que es una pequeña parte de este sistema de posicionamiento.

Por otro lado, el hecho de utilizar un dispositivo existente y de librerías ya desarrolladas convierten este proyecto en un proyecto de integración al que se le añade una parte de desarrollo propio. Es por esto por lo que la integración del software ha sido mucho más elaborada que su contraparte hardware, donde se han utilizado entornos de desarrollo Arduino y Python.

Con respecto a la memoria, en este **capítulo 1** se presenta la **introducción**, junto a los objetivos técnicos y académicos; en el **capítulo 2** se presenta el **marco tecnológico**, que proporciona una visión global de las tecnologías utilizadas durante el desarrollo del proyecto; en el **capítulo 3** se detallan las **especificaciones y restricciones de diseño**; en el **capítulo 4** se presenta el **análisis el sistema**, donde se entra en detalle de las fases de desarrollo del proyecto, las elecciones de elementos hardware y el desarrollo software, así como la integración total del sistema; en el **capítulo 5** se explican los **resultados** obtenidos durante las fases de desarrollo explicadas en el apartado 4, además de las pruebas realizadas para esclarecer la viabilidad del sistema; en el **capítulo 6** se presentan las **conclusiones**, donde se aporta el impacto del proyecto junto a la visión de trabajos futuros; en el **capítulo 7** se detalla el **presupuesto**; y en el **capítulo 8** se aportan las **referencias.**

Además, se incluye un apartado de **Manual de Usuario** donde se explican los pasos a seguir para la configuración y puesta a punto del sistema de posicionamiento en interiores desarrollado; y un último apartado de **Anexos** donde se presentan las tablas numéricas con los resultados de las pruebas realizadas.

1.1 Objetivos técnicos y académicos

1.1.1 Objetivos técnicos

- Seleccionar adecuadamente los dispositivos para componer el sistema de posicionamiento interior, tomando en cuenta la disponibilidad en el mercado y los requisitos de funcionamiento.
- Desarrollar el software necesario para la captura de datos de los dispositivos emisores de señal bluetooth, asegurando la correcta adquisición y procesamiento de la información.
- Implementar el software para el cálculo y representación precisa de la posición del usuario en entornos interiores, empleando algoritmos diseñados para minimizar posibles errores en el posicionamiento.
- Determinar la disposición óptima del dispositivo receptor para maximizar la precisión del sistema de posicionamiento interior.
- Diseñar una interfaz de usuario intuitiva y amigable para la representación de la posición, facilitando la comprensión y la interacción del usuario con el sistema.
- Validar el funcionamiento del sistema en escenarios reales, demostrando su efectividad y fiabilidad en diferentes situaciones de uso.

1.1.2 Objetivos académicos

- Desarrollar habilidades en la selección y evaluación de tecnologías para aplicaciones específicas, considerando tanto los aspectos técnicos como los económicos.
- Mejorar las habilidades de programación y desarrollo de software, especialmente en entornos como Arduino y Python.
- Adquirir experiencia en el diseño e implementación de algoritmos para el cálculo y la representación de la posición en sistemas de posicionamiento interior.
- Fortalecer las habilidades de análisis y resolución de problemas, especialmente en la identificación y mitigación de errores en sistemas complejos.
- Desarrollar habilidades en la planificación y ejecución de pruebas en entornos reales, así como en la interpretación de resultados y la toma de decisiones basada en datos.

2. Marco tecnológico

En este apartado, se explorarán los conceptos fundamentales y las tecnologías clave que sustentan el desarrollo de un sistema de posicionamiento en interiores. A lo largo de esta sección, se proporcionará una comprensión detallada de:

- Concepto de posición y localización.
- Tecnologías inalámbricas.
- Metodologías de posicionamiento.
- Algoritmos para procesar señales.

A través de esta revisión de las tecnologías y metodologías disponibles, se establecerán las bases necesarias para el diseño y desarrollo de este sistema.

2.1 Posicionamiento y localización

A la hora de diseñar un sistema cuyo objetivo es determinar o calcular la posición de un objeto, persona o dispositivo en un entorno cerrado, es importante diferenciar entre posición y localización [1]. En el ámbito de los sistemas de posicionamiento se definen estos términos como:

- **Posición**: Las coordenadas específicas en un sistema de referencia, un ejemplo serían las coordenadas en dos dimensiones (x, y).
- **Localización**: La ubicación con relación a un objeto o evento, se trata de un concepto más amplio y puede involucrar la identificación de la región o entorno donde se encuentra el objeto, un ejemplo sería estimar que el objeto se encuentra frente a la entrada principal en un entorno como puede ser una recepción.

2.2 Tecnologías usadas en posicionamiento

Dentro del ámbito del posicionamiento en interiores, podemos diferenciar entre tecnologías basadas en ondas de banda ultra-ancha (UWB) y en ondas de banda estrecha (WiFi, BLE). Estas tecnologías ofrecen enfoques distintos para determinar la ubicación en interiores, cada una con sus propias características y aplicaciones específicas.

2.2.1 Ultra-Wide Band

Ultra Wide Band o banda ultra-ancha es una tecnología de comunicación inalámbrica que utiliza un espectro muy amplio de frecuencias para transmitir datos. Se denomina UWB a toda técnica que opera con un ancho de banda total superior a 1,5 GHz [2]. Esta tecnología tiene una función similar al Bluetooth, pues está pensada para el intercambio de gran cantidad de datos entre dispositivos.

A medida que la tecnología se consolida, se posiciona como alternativa para la gestión de espacios interiores y la interacción con dispositivos inteligentes. Entre sus ventajas se encuentra:

- Transmisión de datos a alta velocidad.
- Baja interferencia con otras tecnologías.
- Alta precisión de localización (<1 metro).

No obstante, presenta las siguientes limitaciones:

- Alcance limitado.
- Necesidad de infraestructura específica (Transceptor UWB).
- Mayor costo inicial comparado con otras tecnologías inalámbricas.

Estas características hacen del UWB una opción a considerar en el desarrollo de sistemas de posicionamiento en interior.

2.2.2 WiFi

Wifi es una tecnología de red inalámbrica mediante la cual los dispositivos electrónicos se conectan para interactuar con internet. Su uso para posicionamiento de objetos en interiores implica emplear estas señales Wifi para determinar la posición, utilizando los puntos de acceso existentes (Routers) como referencias de ubicación.

Estos sistemas miden los valores de RSSI desde varios puntos para estimar la distancia al punto de acceso. De esta manera, mediante técnicas como triangulación o “fingerprinting”, explicadas en detalle más adelante, podemos determinar la posición del dispositivo.

El punto fuerte de esta tecnología se encuentra en su gran distribución y uso, pues el utilizar la infraestructura Wifi ya presente en infinidad de edificios reduce en gran medida los costos de instalación del sistema. A su vez presenta una gran área de cobertura.

No obstante, presenta inconvenientes como pueden ser una precisión limitada (> 5 metros) y una gran sensibilidad a la interferencia y variabilidad de la señal.

2.2.3 Bluetooth

Bluetooth es un protocolo de comunicación inalámbrica que posibilita la transmisión de voz y datos entre diferentes dispositivos mediante ondas de radio. Los inicios de esta tecnología remontan a la época de los 90, dando lugar al SIG (Special Interest Group), una asociación privada sin ánimo de lucro encargada de dirigir el desarrollo, implementación y comercialización de la tecnología Bluetooth.

Actualmente, podemos diferenciar entre Bluetooth clásico y Bluetooth Low Energy.

El Bluetooth Clásico se introdujo a finales de los 90 como una tecnología revolucionaria en el intercambio de información entre dispositivos electrónicos. Es el Bluetooth tradicional que se

ha utilizado durante años para la transferencia de archivos y la conexión inalámbrica entre dispositivos. Sin embargo, su consumo energético relativamente alto lo hace menos adecuado para sistemas que funcionan con batería o requieren de conexiones constantes.

El Bluetooth Low Energy, como sugiere el nombre, es una versión más moderna del Bluetooth clásico que fue diseñada para ofrecer las mismas características, pero mantener un consumo de energía reducido, lo que lo hace ideal para aplicaciones de IoT. Es una versión más ligera del Bluetooth clásico y se introdujo como parte de la especificación Bluetooth 4.0 en 2010.

Además, el protocolo Bluetooth está diseñado para que sus versiones ofrezcan retrocompatibilidad con versiones anteriores, de modo que el Bluetooth BLE posee todas las características del Bluetooth clásico. Este protocolo de baja energía se ha ido actualizando desde su aparición en 2010 y cuenta con las siguientes versiones:

Bluetooth v4.0 (2010)

Introducción oficial de BLE como parte de la especificación Bluetooth. Mejoras importantes en términos de eficiencia energética.

Bluetooth v4.1 (2013)

Mejoras en consistencia de redes LTE, optimización de conexiones.

Bluetooth v4.2 (2014)

Mayor seguridad y privacidad, capacidad de direccionamiento IPv6.

Bluetooth v5.0 (2016)

Aumento del rango (hasta 4 veces), velocidad (el doble) y capacidad de transmisión de datos (hasta 8 veces).

Bluetooth v5.1 (2019)

Mejoras en la localización. Ofreciendo la ubicación aproximada entre dispositivos conectados.

Bluetooth v5.2 (2020)

Mejora de rendimiento al aumentar el número de dispositivos conectados, mayor seguridad y menor consumo.

Bluetooth v5.3 (2022)

Mejoras de conexión, mayor seguridad y menor consumo.

Bluetooth v5.4 (2023)

Permite comunicación bidireccional entre los puntos de acceso y miles de nodos de baja energía.

La arquitectura de BLE se organiza en varias capas, cada una con funciones específicas que colaboran para proporcionar una comunicación inalámbrica robusta [3]. Su estructura está especificada en la Figura 1.

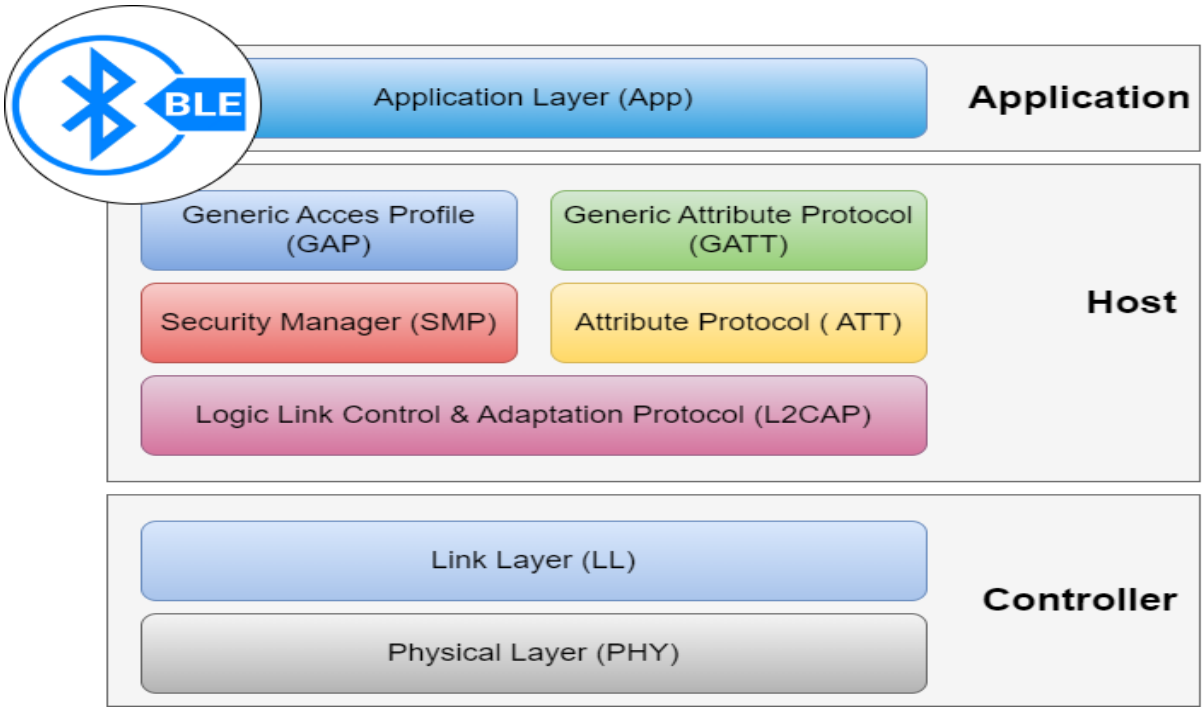

Figura 1. Arquitectura Bluetooth

- **Capa física**

La capa física se encarga de la transmisión y recepción de las señales de radio. Esta capa utiliza la modulación GFSK (Gaussian Frequency Shift Keying), que ayuda a reducir interferencias, asegurando una conexión estable. A su vez, dispone de hasta 40 canales en la banda ISM de 2.4 Ghz, de los cuales 33 se utilizan para advertising y 37 para la transmisión de datos. Esto puede apreciarse en la Figura 2.

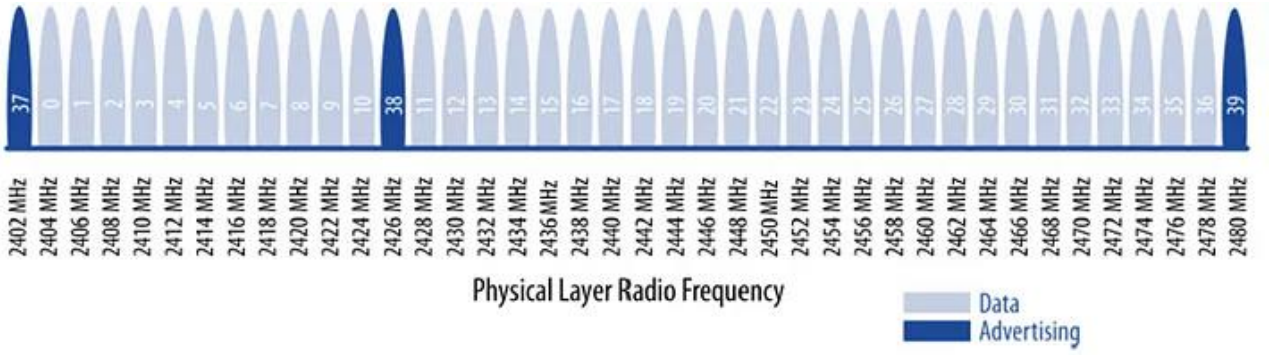

Figura 2. Capa física Bluetooth

FUENTE: https://pcng.medium.com/ble-protocol-stack-controller-2d2d5371deec

Además, esta capa emplea la técnica “frequency hopping” mediante saltos pseudo-aleatorios entre los 37 canales, que la vuelve robusta ante interferencias.

- **Capa de enlace**

La capa de enlace se encarga de gestionar las conexiones BLE ofreciendo definición de roles (Advertiser, Scanner, Master y Slave). En el modo "advertising", los dispositivos envían paquetes periódicamente, mientras que en el modo de escaneo quedan a la espera de recibir un paquete de "advertising".

De esta manera, una vez que un dispositivo central (master) y un periférico (slave) establecen conexión, la capa de enlace gestiona esa comunicación, asegurando una baja latencia y un control eficiente del consumo de energía mediante modos de suspensión y despertado.

- **Capa de Host**

La capa de host incluye varios protocolos esenciales para la comunicación entre un host y un controlador.

El Logical Link Control and Adaptation Protocol (L2CAP) gestiona el multiplexado de datos, fragmentación y reensamblaje de paquetes, permitiendo conexiones simultáneas con varios dispositivos.

El Attribute Protocol (ATT) define cómo se accede a los atributos en un dispositivo, permitiendo la lectura y escritura de datos.

El Generic Attribute Profile (GATT) organiza estos datos en servicios y características, proporcionando un marco estándar para la comunicación de datos entre dispositivos BLE.

- **Capa de aplicación**

La capa de aplicación es donde los perfiles y servicios específicos de BLE entran en juego. Esta capa incluye las interfaces y API que los desarrolladores utilizan para crear aplicaciones que interactúan con los dispositivos BLE, facilitando una integración más sencilla y eficiente de la tecnología en diversos productos y servicios.

Explicada la arquitectura, para entender cómo operan dos dispositivos BLE conectados, es necesarios centrarse en sus dos modos principales de funcionamiento: el modo de "advertising" y el modo de conexión.

En el modo de "advertising", un dispositivo BLE emite anuncios periódicos para informar a otros dispositivos de su presencia. Estos anuncios contienen información básica sobre el dispositivo, como su identificador único. Los intervalos de "advertising" se pueden configurar para controlar la frecuencia de los anuncios, utilizando tres canales específicos para reducir las interferencias y mejorar la capacidad de detección.

Por otro lado, en el modo de conexión, dos dispositivos BLE establecen una conexión bidireccional, permitiendo el intercambio continuo de datos. Uno de los dispositivos actúa como "master", mientras que el otro periférico actúa como "slave". Una vez establecida la conexión, la gestión de esta incluye la configuración de parámetros como el intervalo de

conexión, la latencia y el tiempo de espera, optimizando el balance entre el rendimiento y el consumo de energía. Durante la conexión, los datos pueden intercambiarse continuamente entre los dispositivos, permitiendo interacciones complejas y la transferencia de datos en tiempo real.

Esta arquitectura modular y eficiente en términos de energía permite que BLE sea una tecnología versátil para una amplia gama de aplicaciones en el ámbito del IoT, permitiendo tanto la detección rápida de dispositivos como la comunicación continua y bidireccional entre ellos.

Entre los distintos protocolos para la transmisión de señales mediante la tecnología BLE, dos de los más utilizados son iBeacon, creado por Apple, y Eddystone, desarrollado por Google.

2.2.4 IBeacon

El protocolo iBeacon fue desarrollado por Apple en 2013 y transmite la siguiente información:

UUID: Este campo identifica de forma única al iBeacon. Es un identificador de 16 bytes que generalmente se utiliza para distinguir un conjunto específico de iBeacons, como los desplegados por una empresa o en una ubicación específica.

Major: Este campo se utiliza para diferenciar grupos de iBeacons dentro del mismo conjunto, es un identificador de 2 bytes. Por ejemplo, un establecimiento puede tener varios iBeacons desplegados, cada uno con el mismo UUID, pero con diferentes valores Major para distinguir entre diferentes áreas o departamentos.

Minor: Similar al campo Major, también de 2 bytes, se utiliza para identificar de manera única un iBeacon específico dentro de un grupo o área definida por el campo Major. Por ejemplo, dentro de un departamento específico de una tienda, cada estante podría tener un iBeacon con el mismo UUID y Major, pero con valores Minor diferentes para distinguir entre ellos.

Power Level (RSSI): Este campo sirve para indicar la potencia de la señal transmitida por el iBeacon a una distancia de un metro, es un identificador de 1 byte. Se utiliza para estimar la proximidad del dispositivo receptor al iBeacon y se mide en dBm.

En la Figura 3 se muestra la estructura de una trama iBeacon.

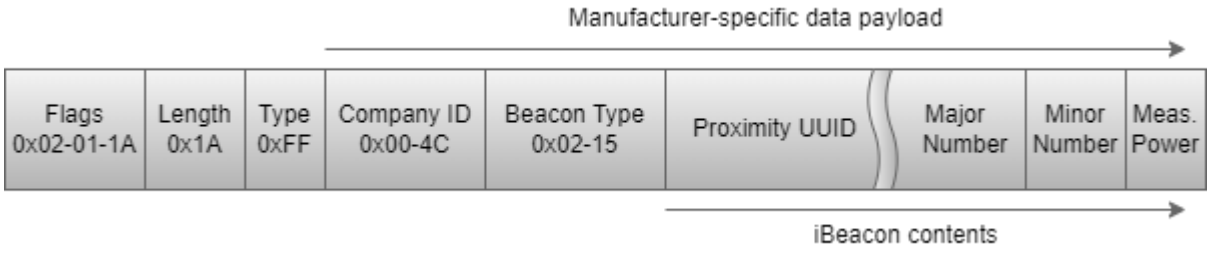

Figura 3. Trama iBeacon

2.2.5 Eddystone

El protocolo Eddystone [4] fue creado por Google y presentado en 2015. Se trata de un protocolo más completo al desarrollado por Apple al abarcar la funcionalidad del iBeacon y permitir, además, enviar otros tipos de paquetes. Una trama Eddystone puede ser de 4 tipos diferentes:

Eddystone-UID: Este formato transmite un identificador único de 16 bytes similar al UUID del iBeacon. Este identificador puede utilizarse para distinguir entre diferentes balizas.

Eddystone-URL: Este formato permite a la baliza transmitir una URL corta que puede ser detectada por dispositivos cercanos y utilizada para ofrecer contenido contextualizado. Por ejemplo, al detectar una baliza Eddystone-URL en una tienda, un teléfono inteligente podría mostrar automáticamente el sitio web de la tienda o una promoción específica relacionada con esa ubicación.

Eddystone-TLM: Este formato transmite datos telemétricos, como el voltaje de la batería de la baliza o la temperatura ambiente. Estos datos pueden utilizarse para monitorear y gestionar el estado de las balizas remotamente.

Eddystone-EID: Este formato utiliza identificadores efímeros rotatorios para transmitir una señal de baliza más segura. Este identificador cambia periódicamente a una tasa determinada con un servicio web.

La longitud total de la trama dependerá del tipo de paquete Eddystone y de los datos específicos que se estén transmitiendo. En la Figura 4 se muestran los 4 tipos de tramas.

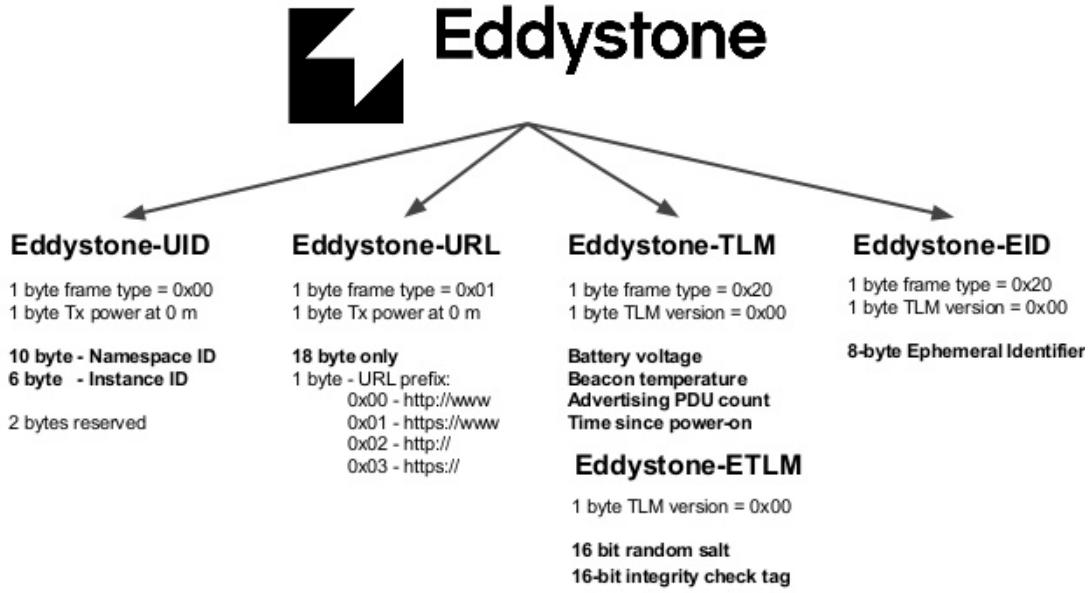

Figura 4. Tramas Eddystone

Fuente: https://www.gotoiot.com/pages/articles/beacons_intro/content.html

2.3 Comparativa de tecnologías

Entre las tecnologías propuestas para su uso en un sistema de posicionamiento, UWB ofrece una alta precisión de localización debido a su ancho de banda ultra-ancho, y, aunque su consumo energético es generalmente bajo, presenta un mayor coste y complejidad de implementación. WiFi, aunque común y de fácil implementación, ofrece una precisión de localización inferior entre las alternativas propuestas y un mayor consumo de energía. BLE, por otro lado, es una tecnología de bajo consumo y relativamente económica, con una precisión de localización adecuada para muchas aplicaciones en entornos de interior. Estas características se resumen en la Tabla 1.

	UWB	Wi-Fi	BLE v4.0	BLE v5.0
Frecuencia de trabajo	3.1-10.6 GHz	2.4-5 GHz	2.4 GHz	2.4 GHz
Alcance (m)	10 m	30-100 m	10 m	240 m
Precisión	< 1 m	> 5 m	Entre 1-5 m	1 m
Consumo energético	Moderado	Alto	Bajo	Bajo

Tabla 1. Comparativa de tecnologías inalámbricas

Teniendo en cuenta estas características, se ha decidido usar BLE por su balance entre costo, consumo de energía y precisión, haciéndola ideal como solución práctica y escalable en entornos donde la mayor precisión de UWB no es esencial.

2.4 Cálculo de posicionamiento en interiores

Existen varios métodos para calcular el posicionamiento en interiores, cada uno con sus ventajas y desventajas, diferenciados en el tipo de medición observada de la señal recibida.

2.4.1 Triangulación

La técnica de triangulación consiste en determinar la posición de un objeto o nodo objetivo en base al ángulo de llegada de señales inalámbricas. La posición del nodo objetivo puede calcularse matemáticamente, siempre que existan al menos dos dispositivos emisores. El proceso consta de dos fases:

1. **Medición de ángulos**: Los dispositivos emisores miden los ángulos en los que se reciben las señales provenientes del nodo objetivo.

2. **Cálculo de intersección**: Utilizando los ángulos medidos y las posiciones conocidas de los dispositivos emisores, se trazan líneas de dirección. La intersección de estas líneas determina la ubicación del objetivo.

Se trata de un método sencillo y eficiente, pero requiere de hardware adicional para su aplicación en entornos reales, ya que las antenas WiFi o Bluetooth no son suficientes para realizar estos cálculos. Además, se requiere una línea de visión directa entre las antenas emisoras y las receptoras, lo cual la hace más adecuada para aplicaciones en exteriores o en

espacios abiertos. En interiores, donde las obstrucciones son comunes, otras técnicas de localización, como la trilateración basada en la distancia o el uso de redes de sensores, pueden ser más efectivas.

2.4.2 Sensores inerciales

Este método de posicionamiento consiste en utilizar sensores que miden la aceleración, la orientación angular (acelerómetro y giroscopio) y el campo magnético (magnetómetro) para determinar la posición y el movimiento de un objeto o persona dentro de un entorno cerrado [5]. Estos dispositivos funcionan de la siguiente manera:

Acelerómetro

Detectan cambios en la aceleración. Se basan en el principio de la segunda ley de Newton (F=ma) para detectar fuerzas de aceleración, ya sea estática o dinámica, que hacen que una masa interna se desplace.

Estos dispositivos suelen medir en uno, dos o tres ejes, siendo estos últimos los más comunes, y realizan la medición tanto de la aceleración en m/s^2, cómo de la fuerza G, siendo esta de 9,8 m/s^2aproximadamente.

Giroscopio

Miden la velocidad angular, es decir, cómo cambia la orientación de un objeto con el tiempo. Utilizan el principio de conservación del momento angular para medir la tasa de rotación alrededor de uno o más ejes.

Los giroscopios modernos (MEMS) realizan la medición de la velocidad angular en grados por segundo $°/s$, o en radianes por segundo rad/s.

Magnetómetro

Detectan la intensidad y dirección del campo magnético. Se basan en la medición de los cambios en el campo magnético terrestre utilizando materiales sensibles a la magnetización, como los sensores de efecto Hall o los magnetorresistivos, para cuantificar en fuerza la señal magnética.

La Tierra genera un campo magnético que produce variaciones en la atmósfera. Estos dispositivos miden en los ejes X, Y, y Z para cuantificar esas variaciones en términos de flujo magnético, dando la medida en microteslas (μT).

El punto débil de estos sensores inerciales consiste en que el error generado por la medición de un sensor crece exponencialmente con el tiempo debido a la acumulación de errores en la integración de las señales, conocido como “deriva inercial”. Además, sensores como el magnetómetro pueden verse afectados por interferencias magnéticas locales.

Para mitigar estos problemas, el uso de estos sensores se combina con otros sistemas de localización, como la tecnología de posicionamiento por satélite (GNSS), WiFi o sistemas de visión por computadora, para mejorar la precisión del sistema.

2.4.3 Trilateración

Este método consiste en estimar las ubicaciones aproximadas mediante geometría de triángulos, concretamente utilizando las medidas de distancias entre los puntos de referencia y el dispositivo a posicionar. En el caso de trabajar sobre un plano de dos dimensiones, es necesario conocer las distancias entre la posición a determinar y al menos otros tres puntos de referencia preestablecidos. Cuando se utilizan más de tres puntos de referencia para estimar la ubicación, se denomina multilateración. El añadir más puntos de referencia ayuda a incrementar la precisión del sistema de posicionamiento.

Como se observa en la Figura 5, al trazar círculos imaginarios alrededor de cada punto de referencia, con un radio igual a la distancia desde cada uno de ellos hasta el punto que se desea localizar, se concluye que éste se encuentra en la intersección entre todos los círculos.

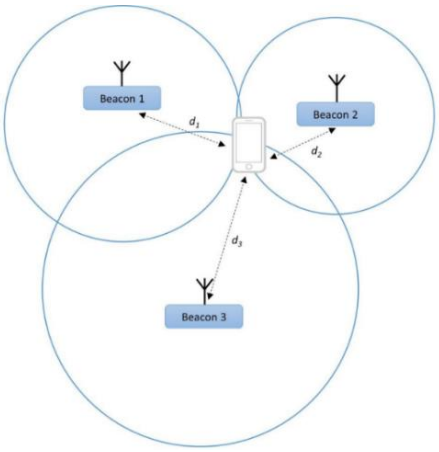

Figura 5. Método trilateración

Para poder aplicar ese método es necesario conocer la distancia de los nodos de referencia al objetivo, esto se consigue con la siguiente ecuación que relaciona el RSSI con la distancia en metros:

$$d = 10^{\frac{txPower - RSSI}{10 * FreeSpaceFactor}}$$

FreeSpaceFactor es una constante que depende del entorno en el que se utiliza el sistema de posicionamiento, TxPower es un valor preestablecido para el dispositivo emisor que indica el valor esperado del RSSI a un metro de distancia, y RSSI es el valor medido actualmente.

El uso de esta técnica presenta un desafío, la distancia entre el punto y los dispositivos emisores no es precisa, por lo que las intersecciones entre los círculos no son puntos exactos, sino áreas. No obstante, la sencillez de su aplicación hace que sea uno de los métodos de localización más utilizados a la hora de su implementación en interiores.

2.4.4 Fingerprinting

Esta técnica consiste en detectar valores medibles de las señales presentes en una determinada área con el fin de obtener un "fingerprint" disponible para uno o más dispositivos para su localización.

La posición se calcula utilizando un algoritmo de coincidencia de datos, es decir, encontrando, entre todas las detecciones previamente guardadas, aquella que mejor aproxime la detección realizada en tiempo real por el dispositivo.

Para aplicar esta técnica podemos diferenciar dos fases:

1. El entorno se analiza utilizando una cuadrícula de puntos con respecto a la cual se asociarán grupos de valores de las señales recibidas con una posición determinada. Una representación de un entorno cuadriculado se observa en la Figura 6.

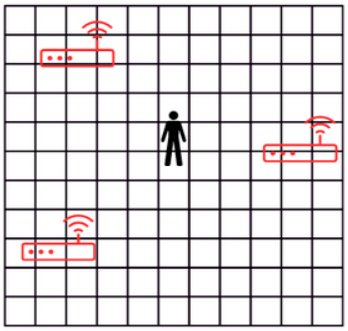

Figura 6. Cuadrícula de puntos

2. Los grupos de valores adquiridos en la fase 1 y aquellos adquiridos en tiempo real proporcionarán la posición más confiable del objeto mediante comparación.

En la Figura 7 se muestra una visión global de este método.

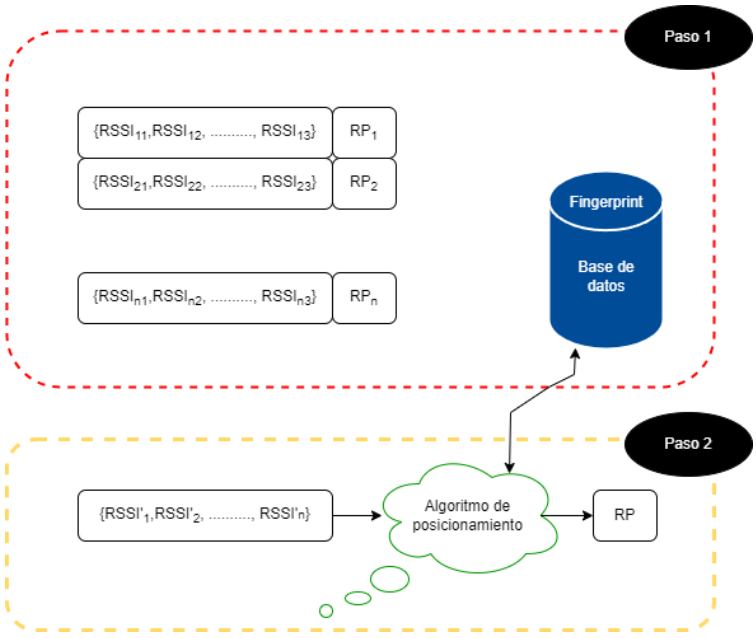

Figura 7. Esquema método fingerprinting

En el caso de utilizar el valor RSSI de las señales BLE recibidas, se crea un mapa que asocia para cada posición de un entorno cuadriculado, valores de RSSI recogidos de los distintos dispositivos emisores repartidos por el entorno. Al recoger la información recibida en cada uno de los puntos se crea una base que contiene los datos del mapeo realizado.

Estos datos se procesan mediante distintos algoritmos para determinar un punto del espacio asociado a cada muestra.

Aunque requiere un tiempo de implementación más largo y un costo computacional más elevado que otras técnicas, en comparación ha demostrado proporcionar resultados con muy bajo nivel de error.

2.5 Algoritmos de filtrado

En las técnicas de posicionamiento descritas anteriormente, es común la existencia de fallos al determinar la ubicación de forma precisa, debido a interferencias o desajustes en los datos. Por este motivo, suelen ir acompañados de algoritmos que sean capaces de mejorar estos comportamientos inusuales.

2.5.1 Filtro K-NN

El algoritmo de los “k-nearest neighbor” o k-vecinos más cercanos, “es un clasificador de aprendizaje supervisado no paramétrico, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual” [6].

Este algoritmo se basa en la idea de que cosas que están cerca unas de otras tienden a ser similares. Para su aplicación, han de seguirse los siguientes pasos:

1. Se parte de un conjunto de datos conocidos a los que se le ha asignado un parámetro, denominado “etiqueta”.
2. Una vez recopilados los datos junto a su etiqueta, se elige un valor de ‘k’, en función de cuantos vecinos más cercanos se quieran considerar.
3. Para cada punto de datos que se quiera clasificar, se asigna un valor de diferencia entre ese punto y los puntos de la base de datos, esta diferencia se denomina distancia. La distancia más utilizada es la euclidiana.
4. Una vez obtenidas todas las distancias del punto a clasificar y los puntos de la base de datos, se seleccionan los ‘k’ puntos más cercanos.

Al usar este algoritmo para clasificar, se compara la etiqueta de estos ‘k’ puntos que están más cercanos a la muestra. La etiqueta de destino con la mayor frecuencia entre estos ‘k’ puntos se asigna como la clase de destino para la nueva muestra.

Este clasificador se adapta bien a la técnica de “fingerprint”, ya que las muestras obtenidas se comparan por probabilidad con las muestras de entrenamiento, representadas por el mapa de radio de los puntos obtenidos. Las ventajas de este método son que no requiere

aprendizaje ni construcción de un modelo, puede adaptar sus límites de decisión arbitrariamente, produciendo una representación de modelo más flexible.

2.5.2 Filtro de Kalman

El filtro de Kalman es un algoritmo matemático utilizado para estimar el estado oculto (no medible) de un sistema a partir de una serie de mediciones ruidosas. Para su implementación diferenciamos dos fases [7], una fase que opera siguiendo un enfoque de predicción y otra fase de actualización con el objetivo de ajustar sus estimaciones y minimizar el error.

La fase de predicción consta de dos pasos.

- Predicción del estado: $X_{k|k-1} = \Phi_k X_{k-1|k-1}$
- Predicción de la covarianza: $P_{k|k-1} = \Phi_k P_{k-1|k-1} \Phi_k^T + Q_k$

La fase de actualización consta de tres pasos.

- Cálculo de la ganancia de Kalman: $K_k = P_{k|k-1} H_k^T (H_k P_{k|k-1} H_k^T + R_k)^{-1}$
- Actualización del estado: $X_{k|k} = X_{k|k-1} + K_k y_k$
- Actualización de la covarianza: $y_k = z_k - H_k X_{k|k-1}$

En la Figura 8 se proporciona una descripción detallada de la implementación de un filtro de Kalman:

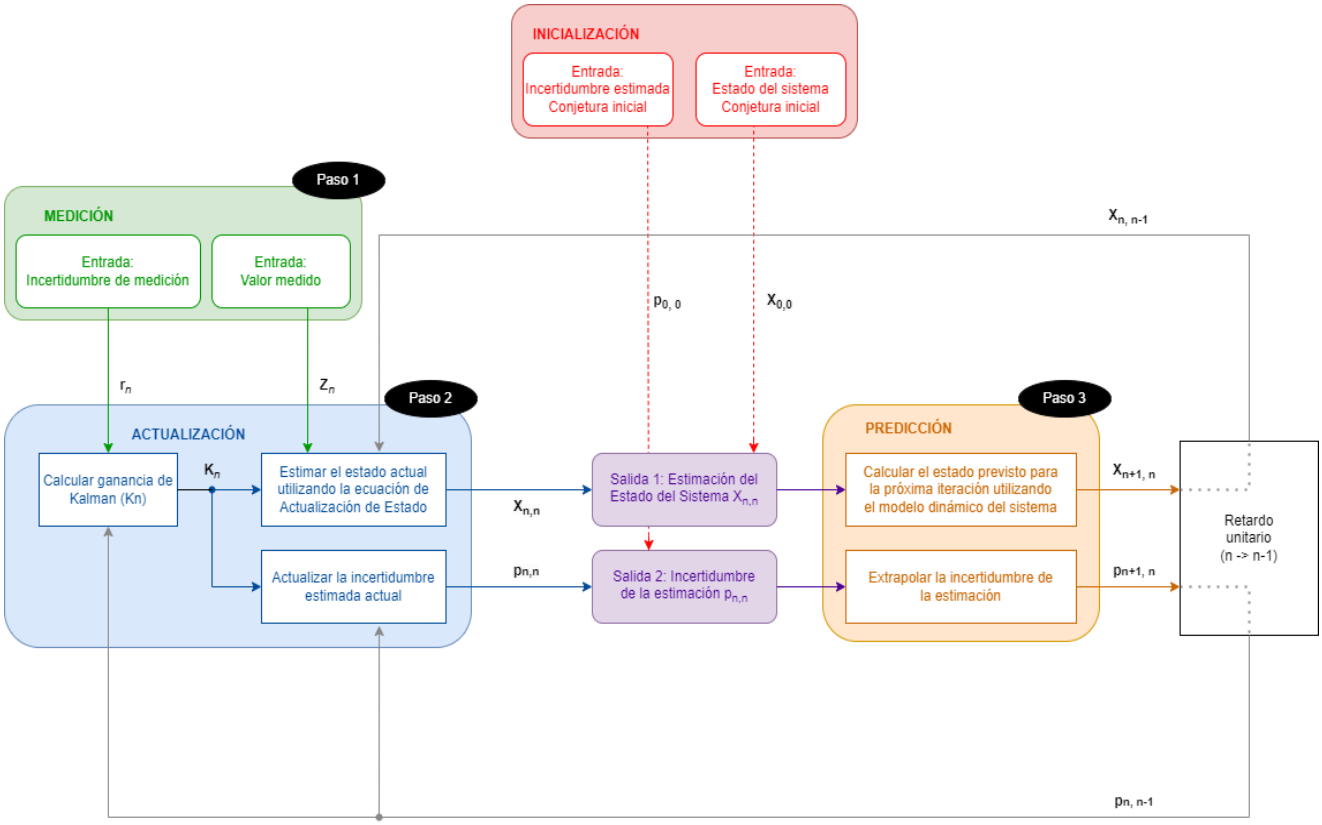

Figura 8. Esquema filtro de Kalman

3. Especificaciones y restricciones de diseño

- Los dispositivos utilizados deben estar disponibles comercialmente y no diseñados específicamente para este proyecto, garantizando así la viabilidad y la posibilidad de replicar el sistema en diferentes entornos.

- La comunicación entre los dispositivos se llevará a cabo mediante BLE, debido a su amplia disponibilidad, bajo consumo de energía y capacidad para comunicarse eficientemente en distancias cortas, lo que lo hace ideal para aplicaciones de posicionamiento en interiores.

- Es fundamental documentar adecuadamente el sistema para permitir futuras expansiones y modificaciones, describiendo la arquitectura del sistema, los componentes utilizados, los algoritmos implementados, los procedimientos de calibración y cualquier otra información relevante para comprender y mantener el sistema en el futuro.

- Se utilizará un dispositivo Arduino como receptor de la señal Bluetooth para el sistema de posicionamiento en interiores, aprovechando su condición de plataforma de hardware de código abierto ampliamente compatible con una amplia gama de sensores y módulos de comunicación.

- Todo el software utilizado en el desarrollo del sistema debe estar disponible bajo licencias de distribución libre, lo que garantiza que el sistema pueda ser compartido, modificado y mejorado.

4. Descripción de la solución propuesta

En este apartado se lleva a cabo una explicación de la solución propuesta al sistema de posicionamiento planteado en este proyecto de fin de grado. Para ello se explicará la elección de los componentes, la caracterización de los valores RSSI recibidos y el desarrollo de dos métodos diferentes para estimar la posición: trilateración y "fingerprinting".

4.1 Análisis del sistema

El sistema de posicionamiento en interiores desarrollado ha resultado en la implementación de un sistema basado en la técnica del "fingerprinting". Para su desarrollo se han utilizado espacios de 8x8m $(64m^2)$ junto a 4 balizas BLE posicionadas en las esquinas. El dispositivo encargado de recoger la información y que porta el activo a localizar por el sistema está basado en la arquitectura Arduino.

Este dispositivo utiliza un ESP32 como núcleo y se encarga de la recogida de muestras proveniente de las balizas BLE. El procesado de estas muestras se lleva a cabo de forma 'offline' en un ordenador personal mediante un programa desarrollado en Python.

Durante el proceso de investigación se estudiaron los métodos más utilizados para determinar la ubicación en espacios de interior, así como la realización de un análisis práctico de las balizas utilizadas y su valor de RSSI a distintas distancias.

Por último, se han empleado dos métodos diferentes de estimación de ubicación: trilateración y "fingerprinting". Este último posicionándose como solución exitosa en el marco de este proyecto.

El método basado en "fingerprinting" se ha complementado con el desarrollo de un cliente de escritorio desarrollado en Python que permite la comunicación entre un ordenador personal y el dispositivo Arduino encargado de la recogida de muestras.

A continuación, se entra en detalle de los distintos elementos y fases de desarrollo.

4.2 Hardware

En este apartado, se examina y justifica el hardware seleccionado para el desarrollo del proyecto.

4.2.1 Dispositivo receptor de señal

Entre las opciones disponibles en el mercado se ha seleccionado la arquitectura basada en ESP32, debido a su bajo consumo energético y su amplia difusión y uso por la comunidad Arduino. Su flexibilidad y compatibilidad con diversos sensores y tecnologías lo convierten en una opción versátil y económica.

Otras arquitecturas presentes en el mercado incluyen el STM32, que ofrece un alto rendimiento o la Raspberry PI, conocido por su robustez en el manejo de sistemas operativos complejos.

El dispositivo seleccionado es una versión de desarrollo basada en el microcontrolador ESP32, comercializada por la empresa M5Stack que desarrolla hardware y soluciones para proyectos de IoT. El dispositivo específico es el m5stack CORE 2, una versión de desarrollo basada en el chip ESP32 que permite su programación en Arduino, entorno de desarrollo elegido para la programación de este SoC (System on Chip).

Este chip ESP32 cuenta con Wifi y Bluetooth integrado de manera nativa y sus especificaciones técnicas se detallan en la Tabla 2.

CPU y Memoria	
Procesador	Xtensa dual-core 32-bit LX6 hasta 240MHz
Memoria ROM	448 KB
Memoria SRAM	520 KB
WI-FI	
Protocolo	802.11b/g/n
Bit Rate	802.11n superior a 150Mbps
Rango de frecuencias	2412-2484 MHz
Bluetooth	
Protocolo	v4.2 BR/EDR y Bluetooth BLE
Clase	Clase-1, clase-2 y clase-3

Tabla 2. Especificaciones ESP32

El m5stack CORE 2 cuenta con varios componentes, entre los cuales destacan su pantalla TFT (Thin Film Transistor), una batería de 500mAh o la inclusión de tres botones capacitivos. Estos elementos son de gran utilidad para facilitar el desarrollo de proyectos de IoT.

El hecho de que estos componentes vengan integrados permite el desarrollo del proyecto sin necesidad de realizar conexiones de cableado para los diferentes sensores que van a componer el sistema final.

Todos estos componentes vienen encapsulados en una pequeña caja de dimensiones 54x54x16mm y puede observarse en la Figura 9.

Figura 9: ESP32 versión M5stack CORE2

FUENTE: https://shop.m5stack.com/products/m5stack-core2-esp32-iot-development-kit

Además de utilizar este dispositivo, ha sido necesario el añadido de un magnetómetro, al no venir este incluido entre los componentes adicionales integrados.

La inclusión de este sensor es necesaria para aumentar la precisión del sistema de posicionamiento. La información obtenida permite procesar los datos de manera más exacta mediante el algoritmo de posicionamiento desarrollado, mejorando así la exactitud del sistema.

En concreto se ha seleccionado el GY-271 de Az-Delivery. Este magnetómetro se comunica mediante interfaz I2C (Inter-Integrated Circuit) y es capaz de funcionar en cualquier posición u orientación con una resolución de 1° a 2°. Es un módulo muy utilizado por lo que cuenta con muchas librerías y ejemplos por parte de la comunidad de Arduino para su aplicación en microcontroladores alimentados mediante 3.3V y 5V. En la Figura 10 se presenta el módulo GY-271 escogido.

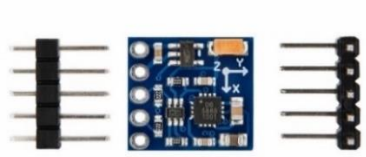

Figura 10. Magnetómetro GY-271

FUENTE: HTTPS://WWW.AZ-DELIVERY.DE/ES/PRODUCTS/GY-271-KOMPASSMODUL-KOMPASS-MAGNET-SENSOR-FUER-ARDUINO-UND-RASPBERRY-PI

4.2.2 Dispositivo emisor de señal

De entre las muchas balizas disponibles en el mercado, se ha optado por el uso de las balizas emisoras BLE del fabricante HolyIoT, debido a su relación calidad/precio. Estas balizas ofrecen compatibilidad con los estándares iBeacon y Eddystone a un bajo costo, lo que permite una integración fluida con el ESP32.

Este enfoque asegura una implementación coherente y eficiente del sistema de posicionamiento en interiores. Una visión a una de estas balizas se presenta en la Figura 11.

Figura 11: iBeacon fabricante Holyiot

FUENTE: HTTPS://ES.ALIEXPRESS.COM/I/32845366171.HTML?GATEWAYADAPT=GLO2ESP

Las balizas están fabricadas por Nordic semiconductors y su núcleo es el chip nRF51822 compatible con Bluetooth low energy v4.1 y el chip nRF52840 compatible con Bluetooth low energy v5.4. Están compuestas por el chip, que cuenta con microcontrolador y radio BLE integrados, una batería y un encapsulado.

Los parámetros hardware y firmware más característicos de estas balizas se detallan en las Tablas 3 y 4.

Parámetro	Valor
Battery	CR2477
Battery Capacity	1100 mAh (3 V)
Transmission Power	-30～4dBm (default：0dBm)
Frequency	100～2000 ms
Security	Password Security
Distance	0～60m (open air)
Battery Life	12～24 months

Tabla 3. Parámetros hardware baliza Holyiot

Parámetro	Valor
UUID	FDA50693-A4E2-4FB1-AFCF C6EB07647825
Major	0-65535 (default：10032)
Minor	0-65535 (default：)
Transmission Power	-30～4dBm (default: 0dBm)
Password	5bytes (default: AA14061112)
Broadcasting Interval	100-2000ms（default：500ms）
RSSI	-
iBeacon Name	1-14bytes，default：holyiot

Tabla 4. Parámetros firmware baliza Holyiot

Su configuración se realiza de manera muy sencilla a través de una app llamada “Holyiot-beacon”, disponible en la tienda de aplicaciones de los teléfonos Android e iOS. Esta aplicación facilita la configuración de las especificaciones firmware, como el tipo de trama BLE junto a sus características específicas.

A su vez, permite seleccionar el intervalo de muestreo y la potencia de transmisión. En la Figura 12 se observa la configuración realizada para la baliza 1. El resto de las balizas se han configurado de manera análoga.

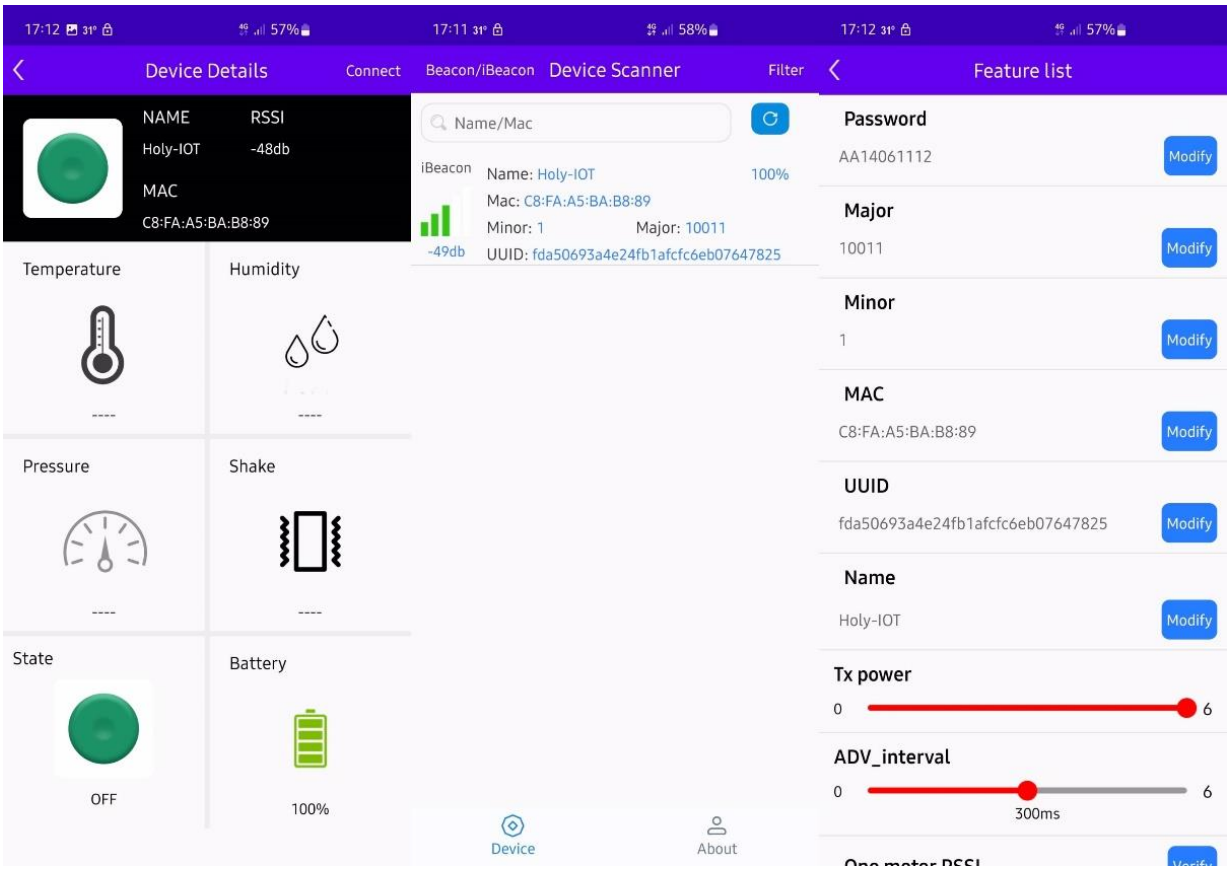

Figura 12. Interfaz app Holyiot

Para la aplicación de estas balizas en el sistema de posicionamiento, se ha optado por establecer la potencia de transmisión al máximo (4 dBm), pues interesa que estas señales se propaguen cubriendo el máximo espacio posible.

Esta configuración permite, además, la dilatación en la variabilidad de la señal en el espacio, facilitando su identificación con la distancia a la que se encuentra.

Esta fluctuación se esclarece en la Figura 13, donde aparece representado para el Beacon 1 una serie de medidas de RSSI capturadas de 1 a 10 metros de distancia en orden creciente.

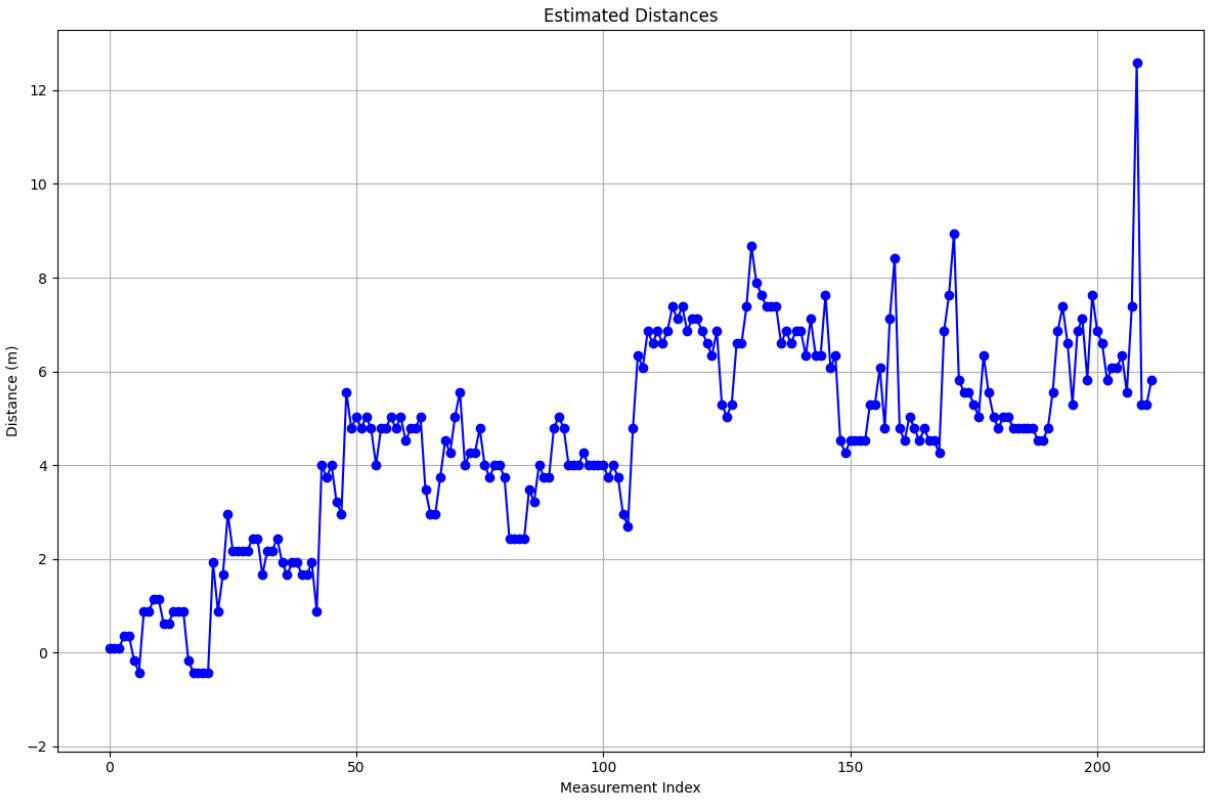

Figura 13. Variabilidad señal RSSI baliza 1

La reducción en la potencia de transmisión se traduce en un estrechamiento horizontal de estas mediciones representadas para la baliza 1, haciendo más complejo el cálculo de la distancia y reduciendo drásticamente su aplicación práctica en entornos reales.

Estos datos han sido tomados en un espacio abierto con el fin de evitar interferencias, ya sean procedentes de señales externas, como otros dispositivos Bluetooth o WiFi, o por procesos como la refracción o reflexión de estas señales por obstáculos en el espacio de prueba.

El resto de los campos del protocolo iBeacon se han configurado con el fin de identificar las balizas en el procesado posterior. Los campos modificados aparecen en azul en la Figura 14

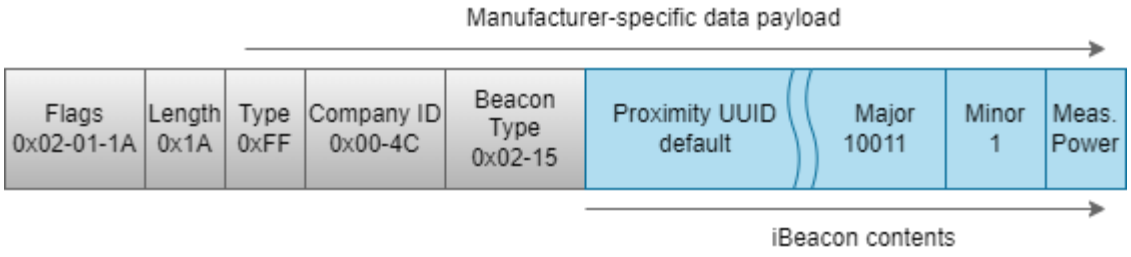

Figura 14. Configuración trama iBeacon baliza 1

El campo “Major” es común a todas las balizas, el campo “Minor” depende del número de baliza y es el campo que sirve para su identificación dentro del sistema de posicionamiento. El campo “Measured Power” depende de la potencia de transmisión seleccionada y de la propia baliza.

4.3 Software

Como se ha detallado en el apartado de Hardware, para la realización de este sistema de posicionamiento se ha decidido utilizar el dispositivo M5stack CORE2 basado en un ESP32. Por ello, la primera tarea a realizar a realizar ha sido la preparación del software de recogida de datos de las tramas iBeacon enviadas por las balizas BLE.

4.3.1 Arduino y ESP32

Junto a este código de recogida de datos, y aprovechando las características del ESP32, se han implementado una serie de programas que mejoran la experiencia del usuario final. Estas características incluyen:

1. La implementación de un servidor web para la descarga de los ficheros CSV generados por el Arduino.
2. El uso de un servidor NTP (Network Time Protocol) para la recogida de la hora de un servidor de internet.
3. La implementación de un modo de ahorro de energía durante la ejecución del modo automático. Encargado de la recogida de datos de manera autónoma.

Estas mejoras están basadas en el WiFi integrado por el ESP32. Para la gestión del usuario y contraseña, se crea un fichero en la memoria SD llamado “config.txt” donde el usuario puede modificar las credenciales de autentificación.

El **servidor web** se ha implementado como un servidor web asíncrono. Esto permite al ESP32 manejar solicitudes simultáneas sin bloquear el flujo principal del programa. Este servidor gestiona la interacción con la tarjeta SD, permitiendo al usuario listar y descargar los ficheros almacenados.

El sistema de **servidor NTP** implementado recoge la hora de un servidor de internet y la almacena en el RTC (Real-Time Clock) del ESP32. Esta sincronización entre servidor NTP y RTC asegura que el sistema opere con una referencia temporal precisa.

El **modo ahorro de energía** desarrollado duerme al microprocesador durante el intervalo de tiempo entre la recogida de muestras BLE. Este modo de ahorro de energía se ha seleccionado debido a su reducción significativa en el consumo, mientras permite conservar las conexiones establecidas de WiFi y BLE. Esto permite al servidor web responder a las solicitudes recibidas. Además, este modo permite al ESP32 dormirse durante los periodos de inactividad, despertándose rápidamente para recoger nuevas muestras.

La integración de todos los módulos converge en el software mostrado en el Diagrama 1.

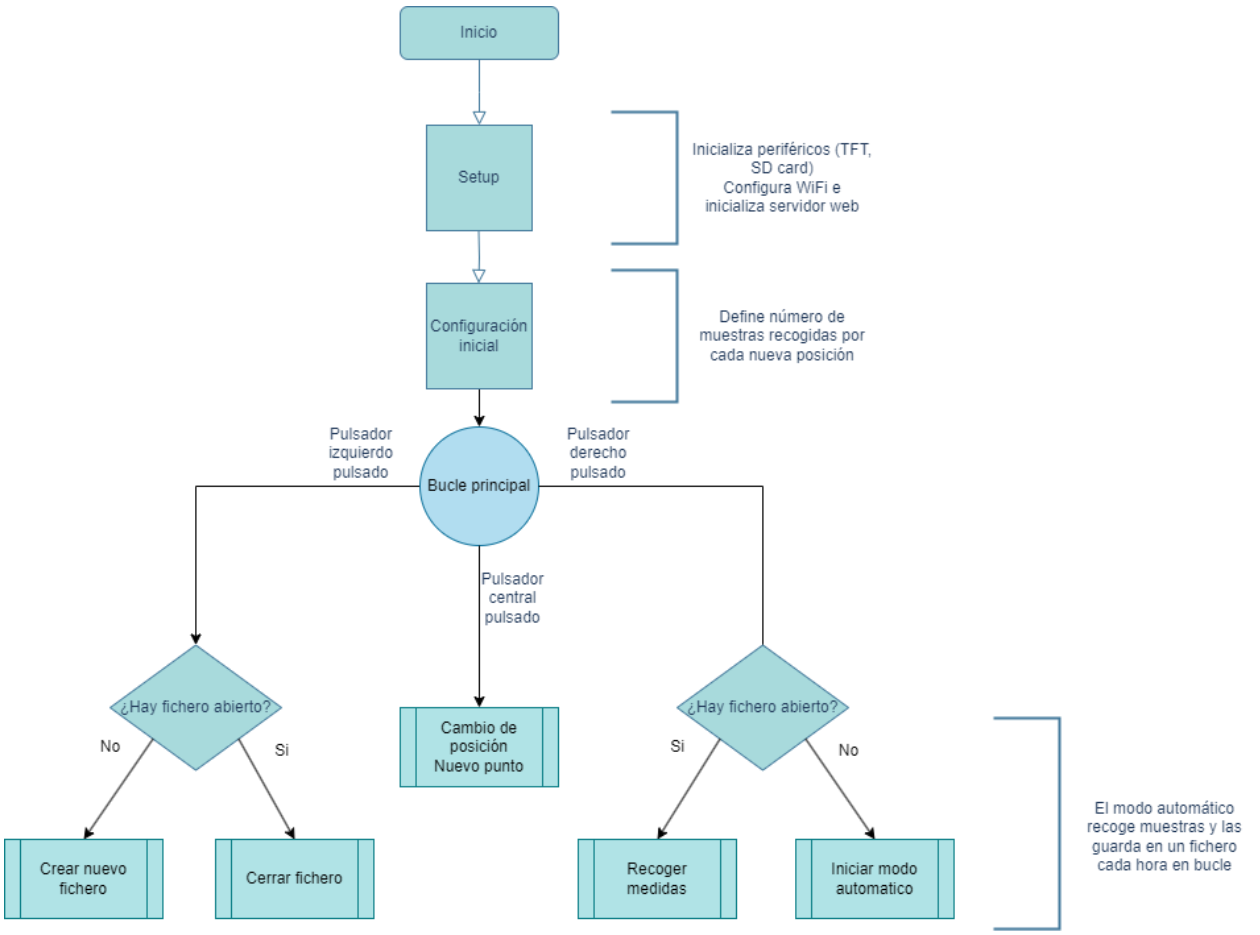

Diagrama 1. Programa Arduino ESP32

El desarrollo de este código ha sido construido apoyándose en las librerías Arduino de código libre detalladas en la Tabla 5.

Configuración	Librerías	Uso
StackM5 CORE 2	M5Core2.h	Configuración de periféricos integrados en StackM5 CORE 2
	Wire.h	Configuración del protocolo de comunicación I2C
Baliza BLE	BLEDevice.h	Funcionalidad general de dispositivos BLE
	BLEUtils.h	Funciones auxiliares como conversión de datos
	BLEScan.h	Escaneo de dispositivos BLE cercanos
	BLEAdvertisedDevice.h	Maneja dispositivos BLE encontrados durante escaneo
	BLEBeacon.h	Creación y manejo de beacons BLE
	BLEEddystoneURL.h	Gestiona el protocolo Eddystone-URL
	BLEEddystoneTLM.h	Gestiona el protocolo Eddystone-TLM
Memoria externa	SD.h	Manejo de tarjetas SD
	SPIFFS.h	Manejo sistema de archivos SPIFFS
Servidor WiFi	WiFi.h	Gestiona la conectividad WiFi
	AsyncTCP.h	Comunicación TCP asíncrona
	ESPAsyncWebServer.h	Creación de servidor web asíncrono
Implementación DNS	WiFiAP.h	Configuración de dispositivos como AP (Access Point)
	ESPmDNS.h	Resolución de nombres de host en una red local (DNS)

Configuración de hora con servidor NTP	NTPClient.h	Maneja sincronización de tiempo utilizando protocolo NTP
	WiFiUdp.h	Proporciona funciones para la comunicación UDP
Modo Light Sleep	esp_sleep.h	Configuración y manejo de modos de ahorro de energía
Magnetómetro	QMC5883LCompass.h	Facilita funciones para configuración y manejo del magnetómetro

Tabla 5. Librerías Arduino

Estas librerías se encuentran bajo la licencia MIT, una licencia de software de código abierto muy permisiva que concede "el derecho de usar, copiar, modificar, fusionar, publicar, distribuir, sublicenciar, y/o vender copias del Software, y a permitir a las personas a las que se les proporcione el Software a hacer lo mismo".

4.3.2 Python

Como se ha indicado en el apartado anterior, Arduino ha sido la plataforma seleccionada para el desarrollo del dispositivo encargado de la recogida de datos. En este apartado se va a abordar el procesado de estos datos de forma 'offline'. Para ello, se ha decidido utilizar Python en su versión 3.12.2, las distintas versiones de Python disponibles pueden descargarse desde su web [8].

Se ha escogido este lenguaje debido tanto a la flexibilidad que ofrece a la hora de trabajar con datos como a su compatibilidad multiplataforma. Para el uso de Pyhton, una herramienta indispensable es Pip, el sistema de gestión de paquetes que se utiliza para instalar y administrar paquetes y dependencias. Mediante este gestor se han instalado todas las librerías utilizadas para el desarrollo Python de este proyecto fin de grado.

Esta herramienta viene preinstalada a partir de Python 3.4. Sin embargo, su instalación es muy sencilla a través del script "get-pip.py", disponible en [9].

Una vez instalado Python y haber descargado este script, mediante la ventana de comandos de Windows basta con ejecutar el siguiente comando desde la ubicación del archivo:

```
C:> py get-pip.py
```

El gestor es muy sencillo de utilizar, se ejecuta también desde la ventana de comandos de Windows y entre sus comandos encontramos los siguientes:

```
pip install nombre_del_paquete          -> Instalar un paquete
pip uninstall nombre_del_paquete        -> Desinstalar un paquete
pip --upgrade nombre_del_paquete        -> Actualizar un paquete
pip show nombre_del_paquete             -> Mostrar información sobre un paquete
pip list                                -> Listar paquetes instalados
```

4.4 RSSI

Una vez terminado el código Arduino y habiendo configurado las balizas, el siguiente paso consiste en el procesado de las señales recibidas y su valor RSSI.

Este procesado se ha realizado a partir de las señales recogidas en una serie de pruebas. En esta prueba se han trazado líneas rectas en el que se han colocado marcadores a cada metro hasta cubrir una distancia de 10 metros. Se han recogido las señales procedentes de las balizas y guardado en un fichero CSV.

Posteriormente, este archivo CSV se ha procesado con el objetivo de obtener una visión clara de los valores RSSI y del comportamiento de la señal en relación con la distancia.

4.4.1 Procesado de valores RSSI

El programa utilizado para procesar estas mediciones ha sido desarrollado en Python. Este desarrollo ha resultado en la construcción de un programa que puede visualizarse de manera sencilla mediante el Diagrama 2, proporcionando una visión clara de la lógica y el flujo del programa.

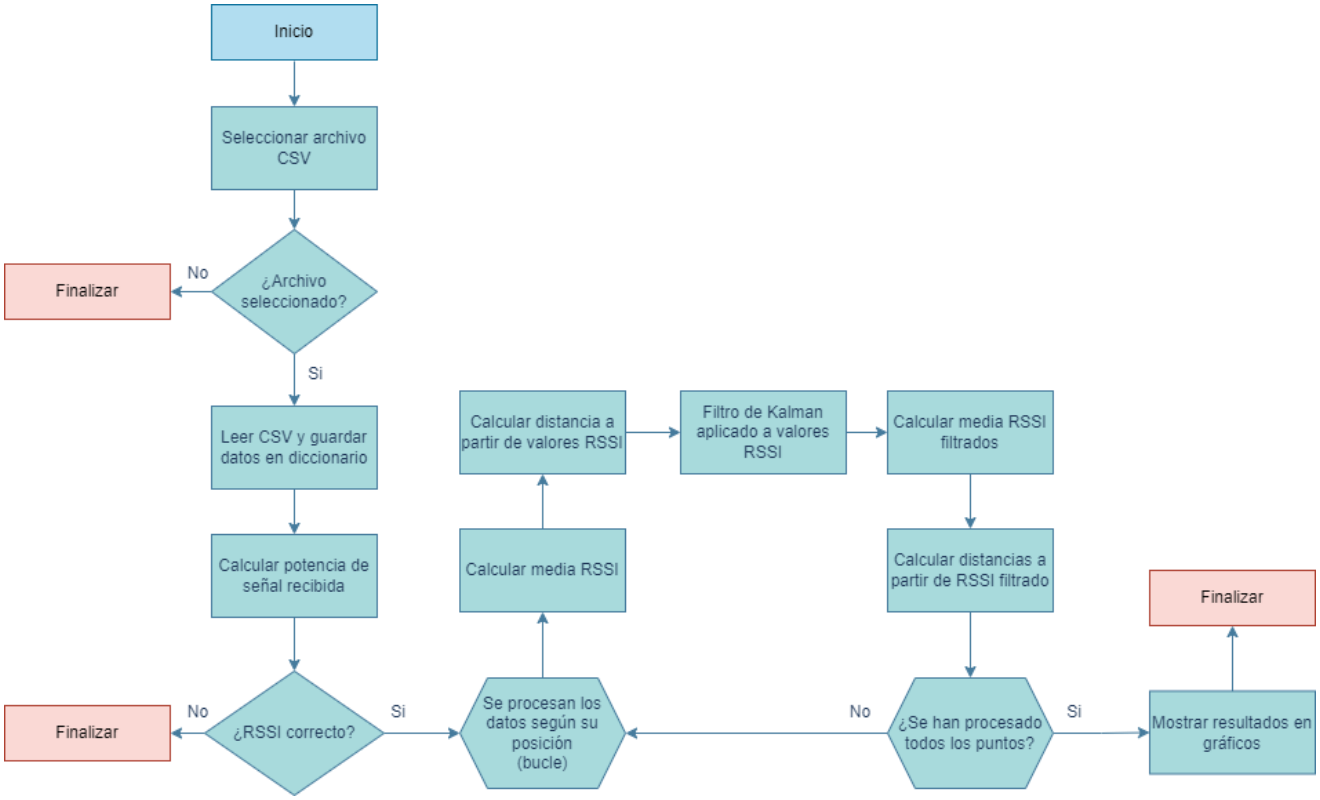

Diagrama 2. Programa procesado valores RSSI

El objetivo de este análisis y, por ende, del desarrollo de este programa, es esclarecer el comportamiento de la señal enviada por las balizas BLE, con el propósito de poder parametrizar los valores recibidos. Para ello el programa puede dividirse en tres bloques detallados a continuación.

4.4.2 Cálculo del RSSI

El cálculo del RSSI surge de la necesidad de conocer su valor teórico a una distancia conocida, de tal manera que se pueda analizar su comportamiento real partiendo de unos valores conocidos y asociados a su valor de distancia.

Para el cálculo del RSSI teórico a partir de la distancia conocida (1 metro), se ha utilizado el modelo "Log Distance Path-Loss Model", que asume que la potencia de la señal recibida decrece con la distancia siguiendo un comportamiento logarítmico. La fórmula general de este modelo que relaciona la distancia con el RSSI es:

$$d = 10^{\frac{txPower - RSSI}{10 * FreeSpaceFactor}}$$

Despejando de esta fórmula podemos obtener el valor del RSSI en función de la distancia:

$$RSSI = txPower - 10 * \log_{10} \frac{d}{FreeSpaceFactor}$$

El significado de estos parámetros se ha definido en el marco tecnológico, entre ellos destacan el RSSI, que es la potencia recibida a un metro de distancia, donde el fabricante ajusta el valor a -55 dBm. No obstante, ese valor no se ha visto reflejado en las mediciones realizadas en un entorno real.

Para definir el RSSI real perteneciente a cada baliza se ha calculado el promedio de una serie de medidas realizadas a un metro de distancia, excluyendo los valores límite (máximo y mínimo). El resultado obtenido se compara con el valor obtenido del fabricante y con el recogido por la trama iBeacon proveniente de las balizas. En la Tabla 6 se muestran los valores de RSSI obtenidos para cada baliza.

RSSI balizas BLE				
Baliza	1	2	3	4
RSSI teórico (dBm)	-55	-55	-55	-55
RSSI Trama iBeacon (dBm)	-60	-55	-64	-58
RSSI (dBm)	-53.8	-61	-67.81	-63.34

Tabla 6. RSSI a 1 metro calculado

Una vez definidos los valores de RSSI de las distintas balizas se deja de tomar en consideración el valor enviado por la trama iBeacon para su uso en el cálculo de la distancia, pues se considera que esa diferencia es suficiente para alterar su valor de manera significativa.

Por último, el "FreeSpaceFactor", constante que depende del espacio donde se realiza la medición, tiene un valor que ronda el 2 para entornos cerrados. Este valor se ha ajustado a lo largo del proceso de pruebas hasta aproximar su valor a 1,8.

A partir de este momento, conociendo el RSSI y el valor de la constante "FreeSpaceFactor" para cada baliza, se recogen las señales en una distancia de hasta 10 metros, esta longitud se

ha tomado como distancia máxima. A partir de esa distancia las señales no siempre se recogen por parte del ESP32.

4.4.3 Filtro de Kalman

El filtro de Kalman se ha implementado en Python a partir de una librería realizada en JavaScript, para más información puede consultarse [10]. Esta librería está enfocada en mejorar la precisión de mediciones al reducir el ruido contenido en sistemas de una dimensión.

La construcción de un filtro de Kalman se realiza empleando las funciones definidas en la clase "*KalmanFilter*". Estas funciones son las siguientes:

_init__(self, R, Q): Inicializa el filtro de Kalman junto al ruido del proceso (R) y el ruido de medición (Q).
filter(self, measurement): Aplica el filtro de Kalman a una medición nueva. Actualiza el estado estimado (x) y la covarianza del error (cov).
last_measurement(self): Devuelve la última actualización del estado (x).
set_measurement_noise(self, noise): Establece un nuevo valor del ruido de medición (Q).
set_process_noise(self, noise): Establece un nuevo valor para el ruido de proceso (R).

La relación entre las funciones definidas proporciona una manera sencilla de implementar un filtro de Kalman en el que el orden de pasos a seguir queda claramente establecido.

No obstante, debido a la naturaleza del sistema de localización que se está construyendo, es necesario el desarrollo de dos funciones adicionales, con el objetivo de conseguir un valor para el ruido de medición y el ruido de proceso de una forma más dinámica. Estas funciones son:

estimate_process_nosie(rsii_values): Estima el ruido de proceso (R) a partir de los valores RSSI específicos de cada medición.
estimate_measurement_noise(rssi values): Estima el ruido de medición (Q) a partir de los valores RSSI específicos de cada medición.

4.4.4 Muestra de resultados

La muestra de resultados se realiza mediante una serie de gráficas. Estas gráficas resultantes se han desarrollado con la ayuda de la librería "matplotlib" [11]. Se trata de una librería de código abierto de Python especializada en la visualización de datos mediante gráficos en 2D. Esta manera de representar los datos aporta claridad a la gran cantidad de datos que se procesan.

En concreto, se crean dos tipos de figuras. En la primera se visualizan cuatro parcelas. Se muestra por un lado los valores RSSI y por otro su distancia calculada en base a esos valores de RSSI. Entrando en detalle:

En el gráfico en la **posición inferior izquierda** se representa en rojo la media de los valores RSSI recogidos en cada posición (de 1 a 10 metros).

En el gráfico en la **posición superior izquierda** se representa en azul la distancia calculada respecto de los valores RSSI representados en rojo.

En las dos gráficas del lado derecho (verde y amarillo) se representa el mismo tipo de dato, pero filtrado mediante el filtro de Kalman. De esta manera en amarillo tenemos los valores RSSI filtrados y encima en color verde el cálculo de la distancia respecto esos valores de RSSI filtrados.

A continuación, en la Figura 15 se representa una gráfica obtenida en una de las pruebas realizadas.

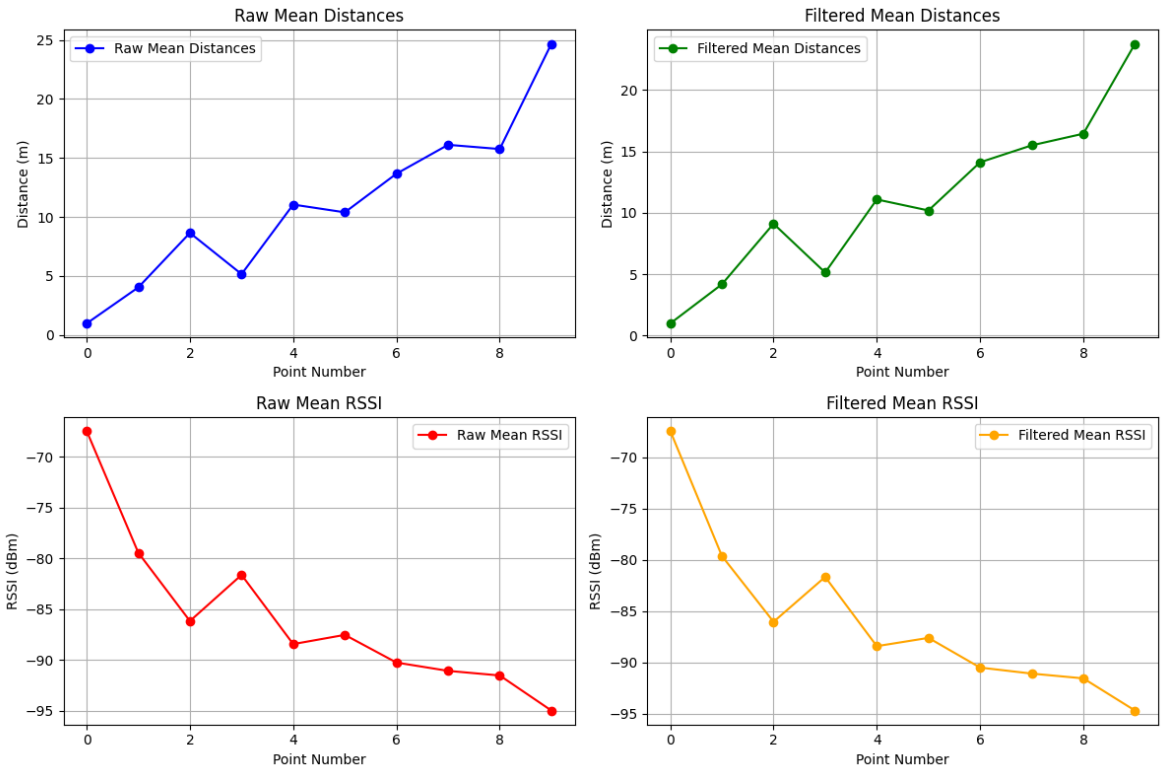

Figura 15. Representación media señal RSSI y distancia

Esta figura evidencia el comportamiento logaritmo del valor RSSI y la relación inversamente proporcional que existe entre distancia (metros) y potencia de la señal (dBm).

El segundo tipo de gráfico entra en detalle en cada posición procesada. En esta nueva figura se representan dos gráficos. En el **gráfico superior** se observa el valor RSSI (azul), el valor RSSI filtrado mediante le filtro de Kalman (verde), y el valor teórico que debe tener el RSSI para esa posición conocida (rojo).

En el **gráfico inferior** tenemos representada la distancia calculada a partir de los valores RSSI sin procesar (azul), la distancia a partir de los valores RSSI filtrados (verde) y la distancia real de la posición que se está analizando en la figura. Esto se ve representado en la Figura 16.

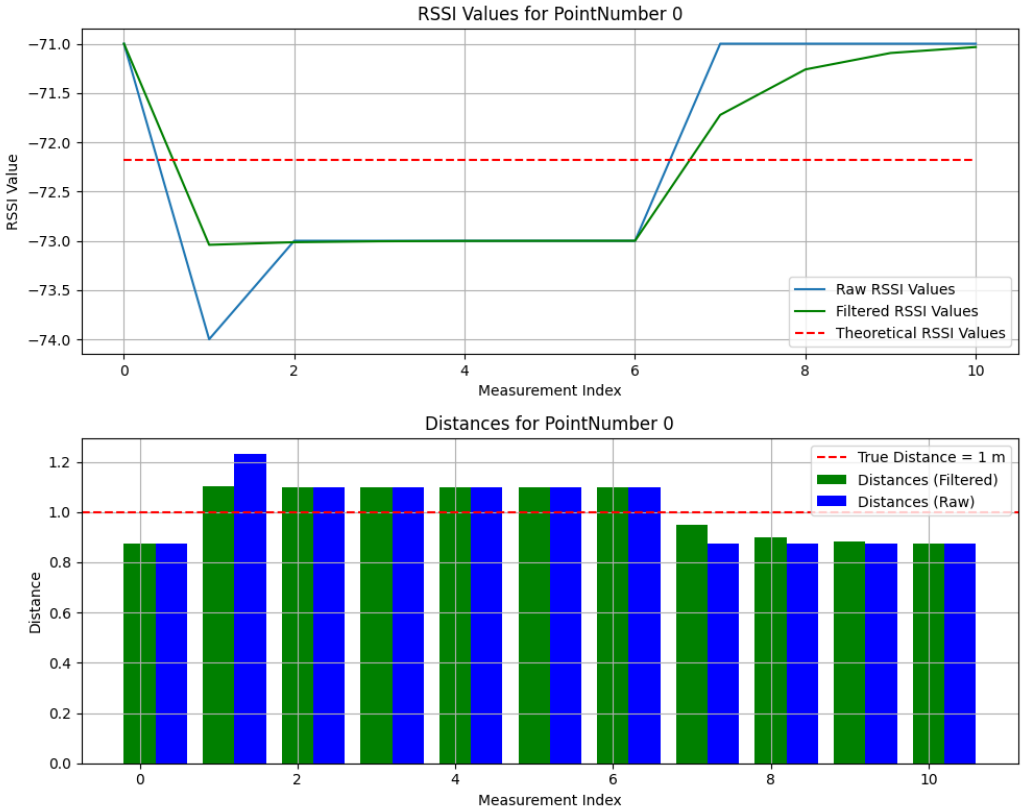

Figura 16. Representación señal RSSI y distancia para posición conocida (1m)

Este último gráfico se muestra en ventanas individuales para cada posición procesada del CSV. El análisis de las pruebas y resultados obtenidos a lo largo de este estudio se profundizará en el apartado 5 de esta memoria. Junto a estas gráficas se devuelve también los valores numéricos de cada valor representado.

Una vez realizado el análisis de la señal recibida, el siguiente paso es el desarrollo del algoritmo o sistema capaz de utilizar el valor del RSSI para determinar una posición en el espacio. Este paso se ha abordado desde dos enfoques diferentes, el primero se realiza mediante trilateración y el segundo mediante la técnica de "fingerprinting". A continuación, se detallan los dos métodos implementados.

4.5 Método trilateración (Solución 1)

El planteamiento de este método surge debido a lo sencillo que resulta, a priori, el cálculo de la distancia a partir del valor del RSSI. Esta implementación está presente en infinidad de sistemas de posicionamiento en interiores que hacen uso de tecnologías WiFi o Bluetooth, y combinándolo con la trilateración consiguen imitar el comportamiento del GPS.

El desarrollo de esta propuesta ha seguido las siguientes fases:

4.5.1 Implementación de trilateración

Una introducción a la trilateración ha sido explicada en el marco tecnológico de esta memoria. Como se explica en ese apartado, el empleo de la trilateración implica conocer las posiciones

de los puntos de referencia (las balizas BLE) y la distancia del dispositivo a posicionar hasta esos puntos de referencia. Por ello se utiliza la expresión que relaciona el RSSI con la distancia en metros. Esa expresión es la siguiente:

$$d = 10^{\frac{txPower - RSSI}{10 * FreeSpaceFactor}}$$

La precisión aportada por esta expresión quedará determinada por el valor txPower y RSSI recibido por parte de la baliza y recogida por el ESP32. La potencia recibida de una señal debe rondar entre -50 y -100 dBm para considerarse aceptable, valores más próximos a 0 se considera una señal fuerte lo que se traduce en cercanía. A medida que estos valores disminuyen se entiende que la distancia aumenta.

A continuación, se entra en el proceso de cálculo junto a sus fórmulas matemáticas. Como apoyo a la explicación se incorpora la Figura 17.

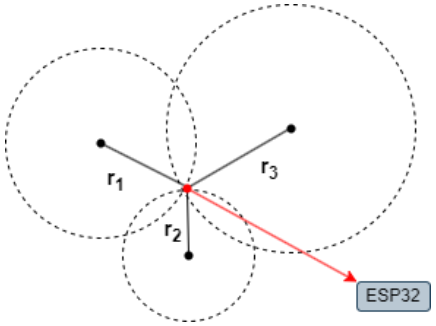

Figura 17. Método trilateración

Conocidas las posiciones (x, y) de las balizas, representadas como puntos negros, y las distancias al ESP32, punto rojo, se pueden aplicar las ecuaciones para la trilateración. Estas siguen el siguiente proceso:

- **Planteamiento de las ecuaciones de los círculos.**

$$1. {r_1}^2 = (x - x_1)^2 + (y - y_1)^2$$

$$2. {r_2}^2 = (x - x_2)^2 + (y - y_2)^2$$

$$3. {r_3}^2 = (x - x_3)^2 + (y - y_3)^2$$

- **Eliminar términos cuadráticos.**

Para simplificar estas ecuaciones, se restan la segunda ecuación de la primera.

$$(x - x_1)^2 + (y - y_1)^2 - (x - x_2)^2 - (y - y_2)^2 = {r_1}^2 - {r_2}^2$$

Expandiendo y simplificando:

$$x^2 - 2xx_1 + {x_1}^2 + y^2 - 2yy_1 + {y_1}^2 - x^2 + 2xx_2 - {x_2}^2 - y^2 + 2yy_2 - {y_2}^2 = {r_1}^2 - {r_2}^2$$

$$-2x(x_1 - x_2) - 2y(y_1 - y_2) + {x_1}^2 - {x_2}^2 + {y_1}^2 - {y_2}^2 = {r_1}^2 - {r_2}^2$$

$$-2x(x_1 - x_2) - 2y(y_1 - y_2) + ({x_1}^2 - {x_2}^2) + ({y_1}^2 - {y_2}^2) = {r_1}^2 - {r_2}^2$$

Esto puede reescribirse como:

$$A_x + B_y = E$$

Donde:

$$A = -2(x_2 - x_1), \qquad B = -2(y_2 - y_1), \qquad E = {r_1}^2 - {r_2}^2 + {x_2}^2 - {x_1}^2 + {y_2}^2 - {y_1}^2$$

De la misma manera, se resta la tercera ecuación de la primera.

$$(x - x_1)^2 + (y - y_1)^2 - (x - x_3)^2 - (y - y_3)^2 = {r_1}^2 - {r_3}^2$$

Simplificando y reescribiendo se obtiene:

$$C_x + D_y = F$$

Donde:

$$C = -2(x_3 - x_1), \qquad D = -2(y_3 - y_1), \qquad F = {r_1}^2 - {r_3}^2 + {x_3}^2 - {x_1}^2 + {y_3}^2 - {y_1}^2$$

- **Ecuaciones lineales.**

Como resultado se tienen dos ecuaciones lineales:

$$A_x + B_y = E$$

$$C_x + D_y = F$$

- **Resolver el sistema.**

Para resolver el sistema de ecuaciones, se puede utilizar la regla de Cramer, que permite resolver sistemas de ecuaciones calculando su determinante.

$$determinante = (B \cdot C - A \cdot D)$$

- **Obtener las coordenadas del punto desconocido.**

$$x = \frac{E \cdot D - B \cdot F}{determinante}$$

$$y = \frac{A \cdot F - E \cdot C}{determinante}$$

4.5.2 Trilateración en Python

El desarrollo del código sigue los pasos del desarrollo matemático a partir de la información obtenida del archivo CSV procesado. Este programa está representado en el Diagrama 3, reflejando la secuencia y la lógica del programa.

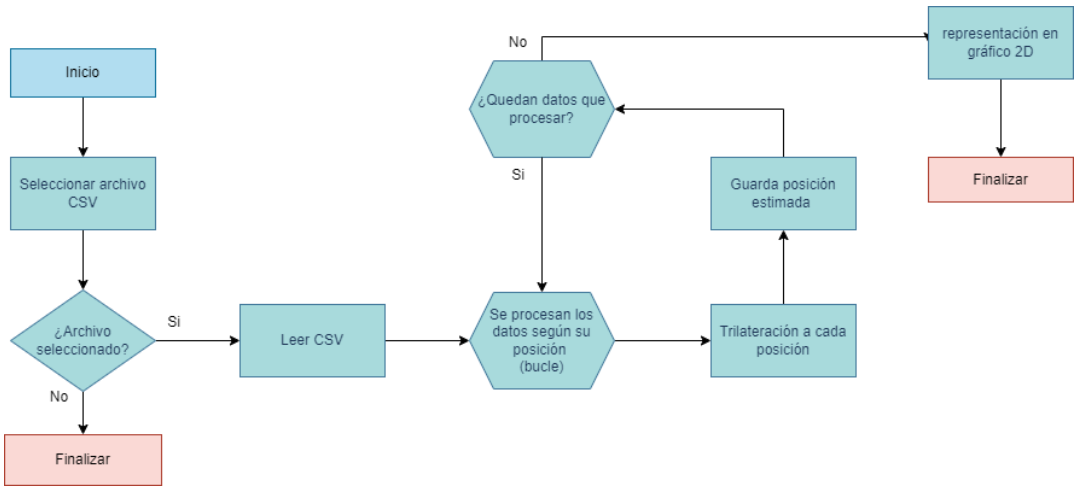

Diagrama 3. Programa trilateración

El código está formado por diferentes módulos, en el algoritmo de trilateración es donde se aplica el desarrollo de ecuaciones explicado en al apartado anterior. Esta implementación se ha desarrollado de la siguiente manera:

```
def trilaterate_2d (points, distances):
    # Filter the three closest points and their corresponding distances
    closest_points, closest_distances = filter_closest_points (points, distances)

    # Get coordinates of the three closest points
    x1, y1 = closest_points [0]
    x2, y2 = closest_points [1]
    x3, y3 = closest_points [2]

    # Get distances to the three closest points
    r1, r2, r3 = closest_distances

    # Calculate differences
    delta_x_21 = x2 - x1
    delta_y_21 = y2 - y1
    delta_x_31 = x3 - x1
    delta_y_31 = y3 - y1

    # Calculate intermediary values
    A = -2 * delta_x_21
    B = -2 * delta_y_21
    C = -2 * delta_x_31
    D = -2 * delta_y_31
    E = r1**2 - r2**2 - x1**2 + x2**2 - y1**2 + y2**2
    F = r1**2 - r3**2 - x1**2 + x3**2 - y1**2 + y3**2

    # Solve linear equation system
    denominator = (B * C - A * D)
    x = (E * D - B * F) / denominator
    y = (A * F - E * C) / denominator

    return [(x, y)]
```

Como la implementación del sistema utiliza cuatro balizas, se descarta la más lejana y se realiza la trilateración con las tres restantes.

La implementación de este sistema presenta buenos resultados cuando existe línea de visión directa entre el ESP32 y las balizas BLE. En caso de no existir, los valores de RSSI se disparan hacia abajo, lo que conlleva que las distancias calculadas aumenten exponencialmente, llevando a estimaciones erróneas con muy bajo nivel de precisión.

4.5.3 Implementación de multilateración

En el uso de la trilateración para calcular la posición se ha encontrado un problema, el hecho de que el valor de la distancia venga dado por la potencia recibida de la señal BLE la vuelve muy inestable. El RSSI se utiliza comúnmente para evaluar la calidad de una señal y se mide en dBm (decibelio/milivoltio). El medirse en decibelios lo relaciona a una naturaleza logarítmica. Esto implica que cada incremento en la escala se traduce en una multiplicación del valor anterior. En la Figura 18 se observa en amarillo un comportamiento lineal y en verde un comportamiento logarítmico sobre valores de distancia calculados a partir del RSSI en un transcurso de 10 metros.

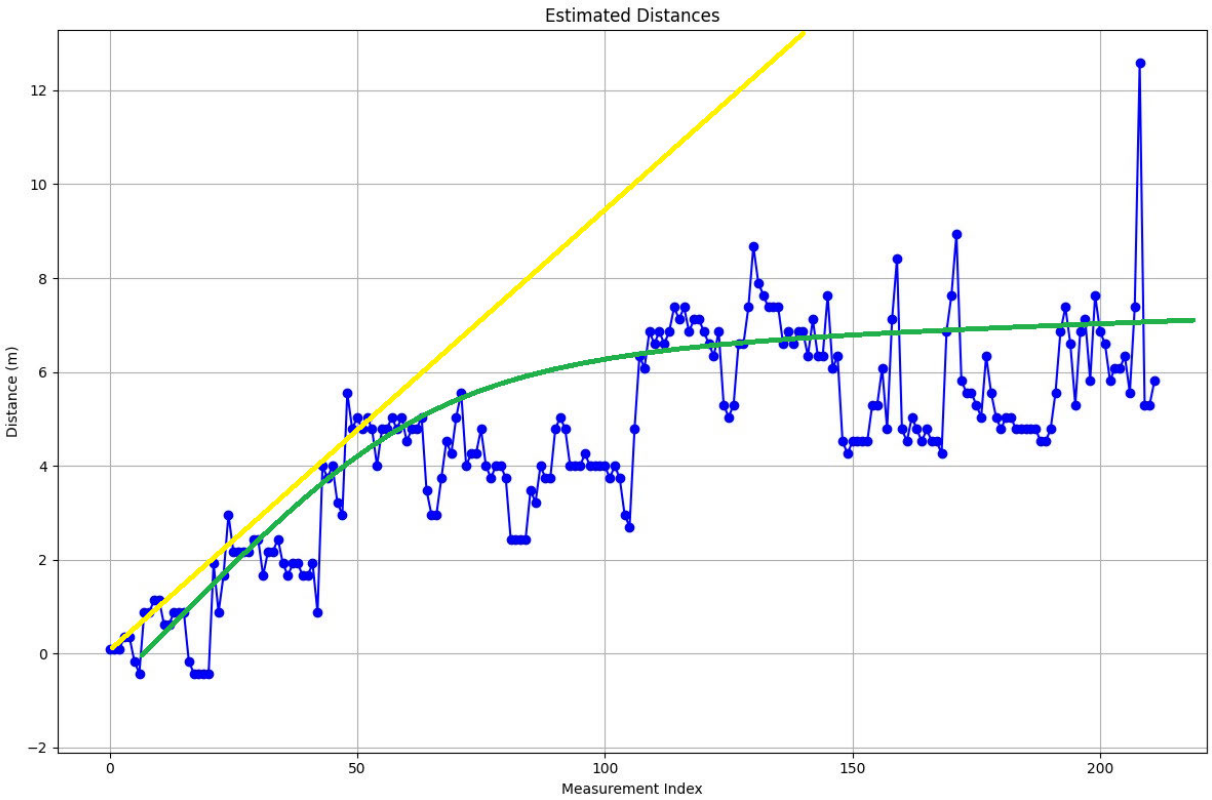

Figura 18. Escala logarítmica

Esto conlleva que, en un escenario real, si el dispositivo a posicionar se encuentra obstaculizado por el propio portador del dispositivo o por un obstáculo del terreno, éste se vuelve muy complicado de posicionar utilizando trilateración a partir de RSSI, pues su valor de distancia entre el portador y la baliza crece enormemente, provocando que el cálculo de su posición sea erróneo.

Debido a esta situación, y teniendo desarrollado el algoritmo de trilateración, se añade el uso de más balizas (multilateración) y se aplica un filtro de Kalman a los valores de RSSI recogidos con el objetivo de suavizar el ruido presente en estas medidas.

El proceso de desarrollo de esta nueva visión del problema a solucionar ha pasado por varias versiones hasta llegar al resultado explicado a continuación, y que puede dividirse en dos bloques principales.

4.5.3.1 Fundamentos matemáticos de la multilateración

Para aplicar la multilateración a partir de la potencia recibida (RSSI), las fórmulas aplicadas en la trilateración dejan de funcionar. Esto es debido a que, si bien trilateración y multilateración son técnicas relacionadas, las ecuaciones para multilaterar tienden a tener un mayor grado de complejidad y, por tanto, requieren de métodos numéricos adicionales o algoritmos de optimización para su resolución.

Matemáticamente, partiendo de <n> puntos conocidos (x_i, y_i), y las distancias <d_i> desde esos <n> puntos al punto desconocido (x, y), la ecuación para la distancia entre el punto desconocido y los puntos conocidos es:

$$d_i = \sqrt{(x - x_i)^2 + (y - y_i)^2} \text{ para i = 1, 2, …, n}$$

Sin embargo, debido a errores de medición y ruido en los datos, es poco probable llegar a una solución válida. Por este motivo, se usan algoritmos de optimización para hallar una solución que minimice el error.

A continuación, se explica la función de costo y la optimización por mínimos cuadrados.

- **Función de costo**

Se trata de una función que evalúa si una posición candidata (x, y) concuerda con las distancias medidas desde esa posición a los puntos conocidos Específicamente, calcula la diferencia, o error, entre las distancias estimadas desde la posición candidata y las distancias calculadas a partir del RSSI. Las ecuaciones son las siguientes:

El cálculo de la distancia estimada desde la posición candidata a los puntos conocidos.

$$d_{i\_estimada} = \sqrt{(x - x_i)^2 + (y - y_i)^2}$$

El error para cada punto conocido.

$$e_i = d_{i\_estimada} - d_i$$

La función de costo total es la suma de los cuadrados de estos errores

$$\sum_{i=1}^{n} (\sqrt{(x - x_i)^2 + (y - y_i)^2} - d_i)^2$$

- **Mínimos cuadrados**

La optimización por mínimos cuadrados se clasifica dentro de los algoritmos de ajuste. Su uso tiene como objetivo encontrar la posición (x, y) que minimice la función de costo total para hallar una posición válida.

Partiendo de la función de costo, se tiene un conjunto de puntos conocidos (x_i, y_i), para i=1, 2, …, n, y unas distancias conocidas d_i, donde los puntos conocidos son las variables independientes y la distancia la variable dependiente.

Suponiendo que se parte de cuatro puntos conocidos (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), la ecuación de la distancia para cada punto es:

$$d_1 = \sqrt{(x - x_1)^2 + (y - y_1)^2}$$

$$d_2 = \sqrt{(x - x_2)^2 + (y - y_2)^2}$$

$$d_3 = \sqrt{(x - x_3)^2 + (y - y_3)^2}$$

$$d_4 = \sqrt{(x - x_4)^2 + (y - y_4)^2}$$

Para poder aplicar el método de mínimos cuadrados se linealizan las ecuaciones.

$${d_1}^2 = (x - x_1)^2 + (y - y_1)^2$$

$${d_2}^2 = (x - x_2)^2 + (y - y_2)^2$$

$${d_3}^2 = (x - x_3)^2 + (y - y_3)^2$$

$${d_4}^2 = (x - x_4)^2 + (y - y_4)^2$$

Se resta la primera ecuación de las demás para eliminar los términos cuadráticos.

$${d_2}^2 - {d_1}^2 = (x - x_2)^2 + (y - y_2)^2 - (x - x_1)^2 - (y - y_1)^2$$

$${d_3}^2 - {d_1}^2 = (x - x_3)^2 + (y - y_3)^2 - (x - x_1)^2 - (y - y_1)^2$$

$${d_4}^2 - {d_1}^2 = (x - x_4)^2 + (y - y_4)^2 - (x - x_1)^2 - (y - y_1)^2$$

Simplificando.

$${d_2}^2 - {d_1}^2 = ({x_2}^2 + {y_2}^2 - {x_1}^2 - {y_1}^2) + 2(x_1 - x_2)x + 2(y_1 - y_2)y$$

$${d_3}^2 - {d_1}^2 = ({x_3}^2 + {y_3}^2 - {x_1}^2 - {y_1}^2) + 2(x_1 - x_3)x + 2(y_1 - y_3)y$$

$${d_4}^2 - {d_1}^2 = ({x_4}^2 + {y_4}^2 - {x_1}^2 - {y_1}^2) + 2(x_1 - x_4)x + 2(y_1 - y_4)y$$

A partir de este momento es necesario el uso de matemática adicional en comparación con el método de trilateración. Esto es debido a que el sistema puede no tener una solución exacta, asi que se utiliza mínimos cuadrados para encontrar la mejor aproximación.

Para aplicar mínimos cuadrados se escriben las ecuaciones en forma matricial $A_x = b$, donde x es el vector de incógnitas (x, y).

$$\begin{matrix} 2(x_1 - x_2) & 2(y_1 - y_2) \\ 2(x_1 - x_3) & 2(y_1 - y_3) \\ 2(x_1 - x_4) & 2(y_1 - y_4) \end{matrix} \cdot \begin{matrix} x \\ y \end{matrix} = \begin{matrix} {d_2}^2 - {d_1}^2 - ({x_2}^2 + {y_2}^2 - {x_1}^2 - {y_1}^2) \\ {d_3}^2 - {d_1}^2 - ({x_3}^2 + {y_3}^2 - {x_1}^2 - {y_1}^2) \\ {d_4}^2 - {d_1}^2 - ({x_4}^2 + {y_4}^2 - {x_1}^2 - {y_1}^2) \end{matrix}$$

La solución por mínimos cuadrados viene dada por la expresión:

$$x = (A^T A)^{-1} A^T b \;\; siendo \; x = (x, y)$$

De esta forma, se obtiene una solución que minimiza el error entre las distancias estimadas y las calculadas.

4.5.3.2 Filtro de Kalman

El filtro de Kalman aplicado mantiene una implementación análoga a la desarrollada en el estudio de la señal recibida y del RSSI, con la diferencia de que se ha implementado utilizando una librería de Python.

Esta librería llamada “filterpy” sigue el mismo proceso que el utilizado anteriormente, pero añade el respaldo de ser una librería ampliamente utilizada y verificada por la comunidad. Puede encontrarse más información en su página web [12].

4.5.4 Multilateración en Python

La necesidad de utilizar mínimos cuadrados para multilaterar se ha abordado mediante el uso de la libraría “scipy” de Python. Esta librería proporciona algoritmos para optimización, ecuaciones algebraicas, estadísticas y muchas otras clases de problemas. Puede encontrarse más información en su página web [13].

A su vez, se ha utilizado la librería “filterpy” para la implementación del filtro de Kalman en 1D, debido a la sencillez ofrecida para su integración.

Este programa está representado mediante el Diagrama 4.

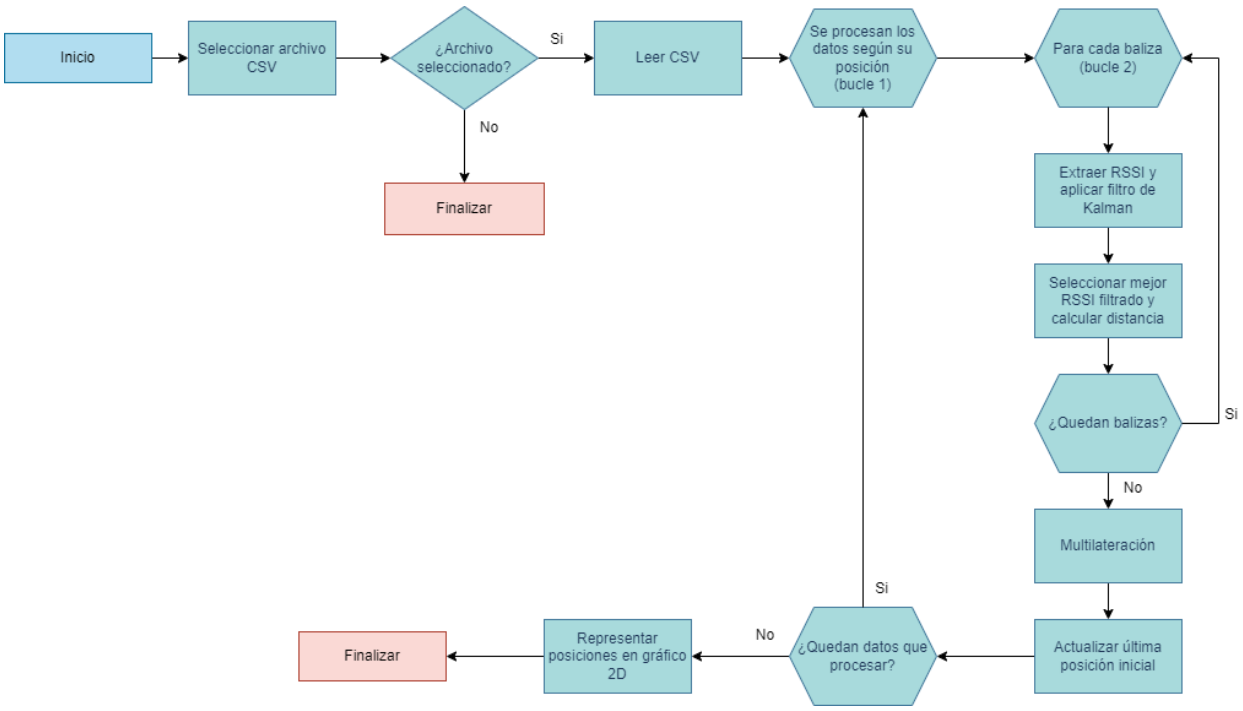

Diagrama 4. Programa multilateración

Entrando en detalle en el algoritmo de multilateración, se han definido dos funciones. Con ellas se calcula la función de costo y después, se realiza el ajuste por mínimos cuadrados.

El cálculo del coste se realiza mediante la función '*cost_function()*':

```
def cost_function(position, known_points, distances):
    # Compute the estimated distances from the given position
    estimated_distances = np.linalg.norm(known_points - position, axis=1)

    # Compute the difference between estimated and actual distances
    error = estimated_distances – distances

    return error
```

Dentro de esta función se utilizan los siguientes parámetros:

'position' es la posición candidata (x, y).

'known_points' representa las coordenadas (x_i, y_i).

'distances' contiene las distancias desde los 'known_points' a 'position'.

'estimated_distances' calcula las distancias estimadas donde:

- 'position' se resta de 'known_points' dando como resultado $(x - x_i, y - y_i)$.
- 'np.lianlg.norm()' calcula la distancia euclidiana para cada diferencia.

'error' calcula los errores para cada punto conocido.

Revisada la función de costo, para aplicar el cálculo por mínimos cuadrados se desarrolla la función '*multilaterate_least_squares()*'. La parte más interesante de esta función radica en las siguientes líneas:

```
def multilaterate_least_squares(known_points, distances, initial_guess=None):
   /* Código adicional*/
……
   /* Código adicional*/
    # Set the initial guess
    if initial_guess is None:
        initial_guess = np.mean(points, axis=0)
    else:
        initial_guess = np.array(initial_guess)  # Ensure it's a numpy array
    print("Initial guess:", initial_guess)
    # Use least squares optimization to find the position
    result = least_squares(cost_function, initial_guess, args=(points,
distances))
    # Extract the estimated position
    estimated_position = result.x
    return estimated_position
```

'*known_points'* es un diccionario con los puntos conocidos y sus coordenadas (x, y).

'*distances'* es un alista que contiene las distancias desde el punto a estimar hasta cada punto conocido.

'*initial_guess'* es la suposición inicial para la posición candidata (opcional).

'result' hace uso de la función *'least_squares'* para minimizar la función *'cost_function'* mediante *'args=(points, distances)'* que contiene los puntos conocidos y las distancias.

'estimated_position' extrae la posición candidata del resultado de la optimización.

Estos pasos siguen la solución matemática explicada con anterioridad, donde la aplicación de los mínimos cuadrados se deja en manos de la función '*least_squares()'* proporcionada por la librería de Python "scipy".

El uso de la función '*least_squares()*' proporciona al usuario la siguiente información:

'Initial guess'**:** Es el punto de partida para la optimización.

'Result:message'**:** Indica si la optimización terminó en éxito o fracaso**.**

'success'**:** Puede tener valores TRUE o FALSE.

'status'**:** Indica la condición de finalización de la optimización.

'fun'**:** Valores de la función de costo para cada una de las distancias medidas.

'x'**:** Coordenadas del punto estimado.

'cost'**:** Resultado de la función de costo.

'jac'**:** Matriz que representa las derivadas de la función de costo con respecto a las coordenadas (x, y)**.**

'grad'**:** Vector que indica la dirección y la magnitud del cambio en la función de costo. Un valor cercano a 0 indica una buena optimización.

'optimality'**:** Vector que indica qué tan cerca está la solución de ser óptima.

'nfev': Número de evaluaciones de la función.

'njev'**:** Número de evaluaciones del jacobiano.

4.6 Método fingerprinting (Solución 2)

Este método es el resultado de buscar una solución que presente un mayor grado de precisión en la posición a estimar, minimizando el error de posicionamiento obtenido con la trilateración. Para ello se ha desarrollado un sistema de posicionamiento mediante la técnica de "fingerprinting".

"Fingerprinting" utiliza la intensidad de las señales recibidas desde diferentes ubicaciones para crear un mapa de referencia. Al comparar luego estas señales recibidas con el mapa, se consigue estimar la posición de un usuario con precisión.

En la Figura 19 se muestra la disposición desplegada durante las pruebas realizadas para construir el modelo.

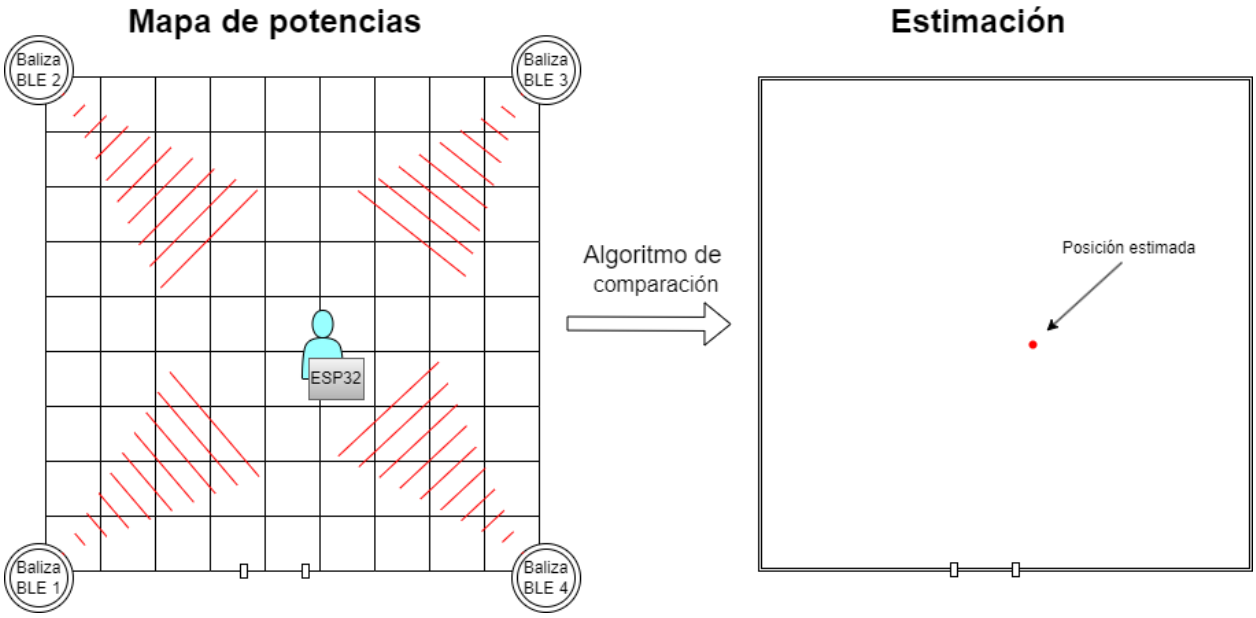

Figura 19. Método Fingerprinting

La idea consiste en mapear el área donde se quiera instalar un sistema de posicionamiento en interior, de manera que obtengamos un mapa de potencias RSSI de los puntos que se quieran identificar. Las pruebas realizadas del modelo se han mapeado con divisiones de metro cuadrado [m^2], por tanto, la precisión conseguida tendrá como limitante esa distancia de ≈1m.

Este método sigue las fases explicadas a continuación.

4.6.1 Creación del mapa de potencias RSSI

La creación del mapa de potencias es parte indispensable del éxito final del sistema de posicionamiento. Para una correcta recogida de datos, se recorren los puntos establecidos para realizar las mediciones y se recogen los valores de RSSI en cuatro direcciones diferentes.

En el caso concreto de este sistema, se han recogido valores de RSSI, primero entendiendo las distintas direcciones como perpendiculares a las paredes del área donde se implementa el sistema, y segundo, una vez finalizado el desarrollo de este método, mediante el magnetómetro añadido al modelo para terminar de reforzar el algoritmo de clasificación, de esta forma, se toman cuatro direcciones que se corresponden con [Norte, Este, Sur, Oeste].

De esta manera, se logran mediciones para cuando la señal es obstaculizada por el propio portador, cubriendo una gran cantidad de casos. Hay que tener en cuenta que cuantas más medidas se requieran, más tedioso es el proceso de creación del mapa de potencias. Por ello se busca un equilibrio entre cantidad de posibles escenarios y medidas realizadas.

Una vez obtenido el mapa se le asigna a cada punto unas coordenadas. Se ha optado por utilizar coordenadas cartesianas (x, y), entendiéndose la esquina inferior izquierda como el punto (0, 0). Al tratarse de un sistema pensado para entornos de interior, con conocer su posición en relación con el área donde se despliegue es suficiente para ubicar al usuario.

4.6.2 Algoritmo de comparación (k-nearest neighbor)

El algoritmo de comparación utilizado está basado en el algoritmo “k-nearest neighbor”. Este algoritmo compara mediante aproximación, es decir, calcula la distancia como la diferencia de la muestra a procesar con las muestras de la base de datos y aproxima a la muestra de la base de datos cuya diferencia sea menor.

Para poder aplicar este algoritmo es de vital importancia el parámetro <k>, que hace referencia al número de puntos que se tienen en consideración a la hora de clasificar nuevos puntos. Para este proyecto se ha decidido optar por k=1 pues queremos dar con la solución más próxima, y no un conjunto de ellas. Este hecho se ejemplifica en la Figura 20.

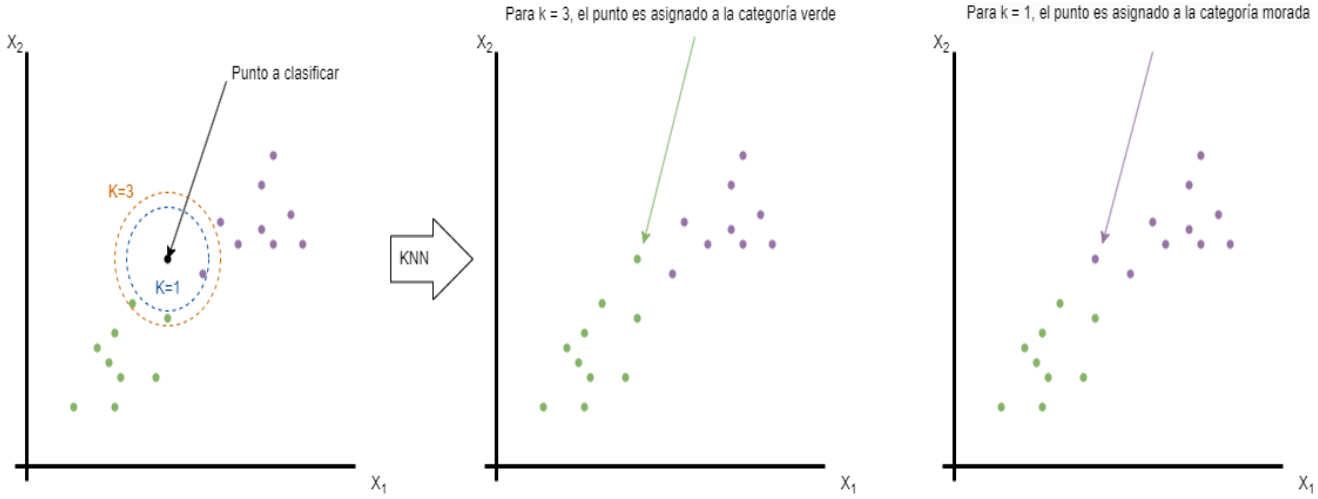

Figura 20. Algoritmo K-NN

Al seleccionar k=3, se toman los 3 puntos más próximos al punto a clasificar. Esto son 2 puntos con la etiqueta 'verde' y un punto con la etiqueta 'morada'. Por tanto, el punto seleccionado se clasifica como perteneciente al grupo con etiqueta 'verde'.

De forma similar, para el caso de k=1, el punto se clasifica como perteneciente al grupo de etiqueta 'morada'.

Aparte de seleccionar un valor para 'k', es necesaria la creación de un mapa de potencias. Las entradas del mapa creado presentan la siguiente configuración:

- **Posición:** Formado por las coordenadas (x, y) y la dirección (hasta 4 direcciones diferentes).
- **Minor ID:** Es el Minor enviado por la trama iBeacon que identifica el número de baliza.
- **Mean RSSI:** La media de los RSSI recogidos para esa posición del 'minor ID'.

Donde los distintos datos de la base de datos son el conjunto de 'minor ID' y valor de RSSI, y las posiciones (x, y, dirección) son las etiquetas.

Por último, el empleo del sensor de brújula (magnetómetro) posee puntos o zonas muertas, esto quiere decir que hay ángulos o determinadas posiciones del sensor donde éste no es capaz de determinar en qué dirección se encuentra. Estas zonas se han tratado como que no existe dirección. En este caso el algoritmo KNN compara la posición de la ruta con todas las direcciones que tiene guardadas en la base de datos para ese punto.

A partir de este momento, se aplica el algoritmo y se siguen una serie de pasos partiendo de una ruta que contiene puntos a clasificar:

1. Iteración sobre cada punto de la ruta

Se recorren los puntos de la ruta para obtener por cada punto los 'minor ID' de las balizas junto a sus valores de RSSI.

2. Iteración sobre cada posición (etiqueta) de la base de datos

Para cada conjunto de datos de la ruta, se obtienen los diferentes datos de la base de datos para procesar

3. Verificación de la dirección

Se comprueba la dirección de la ruta con los datos de la base de datos. De esta manera se ahorra trabajo computacional al procesar solo los datos con la misma dirección. En caso de no existir dirección de ruta, se compara con todas las posiciones guardadas en la base de datos.

4. Cálculo de la distancia entre valores RSSI

Se calcula para cada baliza (minor ID) la diferencia entre valores de RSSI y se suma el total. Esto quiere decir que para cuatro balizas se calcula la diferencia individual entre los valores de RSSI y se agrupan en un total. Este valor total es el que se tiene en consideración para clasificar.

5. Consideración de la posición previa

Se toma en consideración la posición previa para el cálculo de la distancia. Esto se tiene en cuenta debido a que posiciones más cercanas al punto estimado en el instante de tiempo t-1 tienen mayor probabilidad de ser correctas que distancias que estén ubicadas más lejos en el área total cubierta por el sistema de posicionamiento.

Se calcula la diferencia mediante la distancia euclidiana. Esta distancia es la distancia ordinaria entre dos puntos de un espacio euclídeo. Para un espacio bidimensional la distancia euclidiana se calcula como:

$$d_{P1,P2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

De este modo, esta diferencia se añade al total del valor RSSI utilizado para clasificar. En caso de no existir una posición previa, este cálculo no se realiza.

6. Retorno de la posición estimada

Se selecciona la posición (etiqueta) de menor valor entre las posiciones procesadas de la base de datos. Esta posición es considerada la solución del algoritmo de clasificación.

4.6.3 Método fingerprinting con Python

El programa desarrollado es fiel al proceso de creación de mapa e implementación del algoritmo de comparación recién explicados. Este programa sigue el flujo de ejecución mostrado en el Diagrama 5.

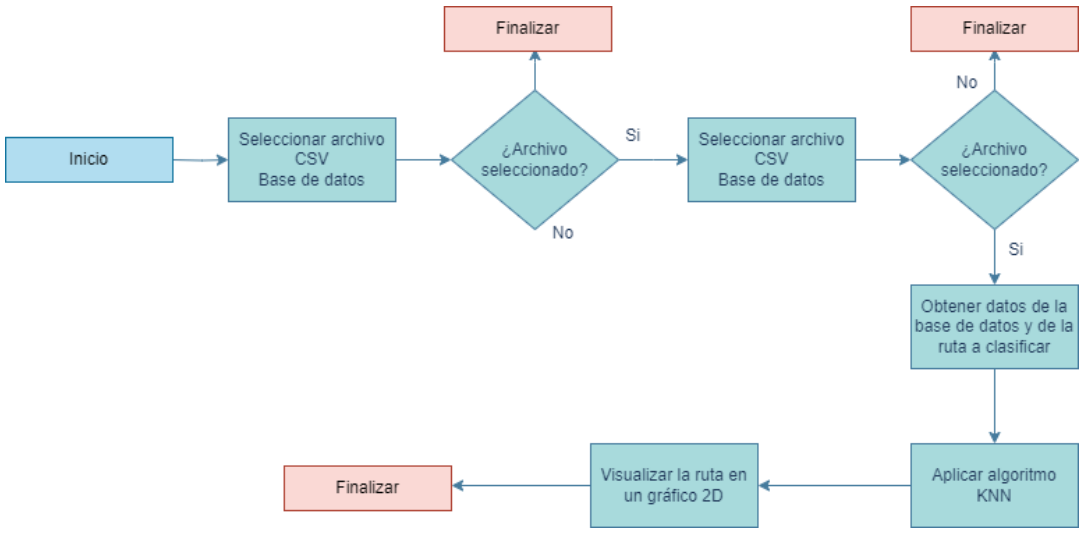

Diagrama 5. Programa fingerprinting

La aplicación del algoritmo K-NN sigue el orden de pasos explicado en el apartado anterior. Partiendo de una base de datos y una ruta a procesar, estos pasos son:

i. Inicialización

Se crea un diccionario para almacenar las posiciones estimadas.

ii. Iteración sobre cada punto de la ruta

Para cada posición de la ruta y sus datos asociados (Minor, RSSI, Dirección).

iii. Iteración sobre cada posición de la base de datos

Para cada posición de la base de datos y sus datos asociados (Minor, RSSI, Dirección, [x, y]).

iv. Verificar dirección

Se comparan las direcciones de la ruta y la base de datos. Si coinciden o si no existe dirección, continúa.

v. Calcular distancia entre valores RSSI

Para cada baliza (Minor) y sus valores de RSSI asociados se calcula la media.

Si la baliza (Minor) se encuentra también en la base de datos se calcula la distancia estimada.

vi. Considerar distancia previa

Si existe una posición estimada previa distinta de 0, se calcula la distancia entre la posición estimada y la posición previa y se ajusta a la distancia estimada.

vii. Actualizar la mejor posición

Si la distancia estimada actual es menor que la última distancia calculada, se actualiza su valor y se repite el proceso hasta completar la base de datos

viii. Retorno de posición estimada

Se retorna el diccionario que contiene las posiciones estimadas. Este diccionario contiene la posición asociada a la distancia de menor valor para cada posición de la ruta.

Este algoritmo KNN recién explicado proporciona unos resultados con escaso error donde la trayectoria seguida por el usuario es claramente diferenciable.

La ejecución de este programa da como resultado un gráfico en 2D donde se visualizan los puntos en su posición estimada. Un ejemplo de ejecución para un recorrido diagonal da como resultado la Figura 21.

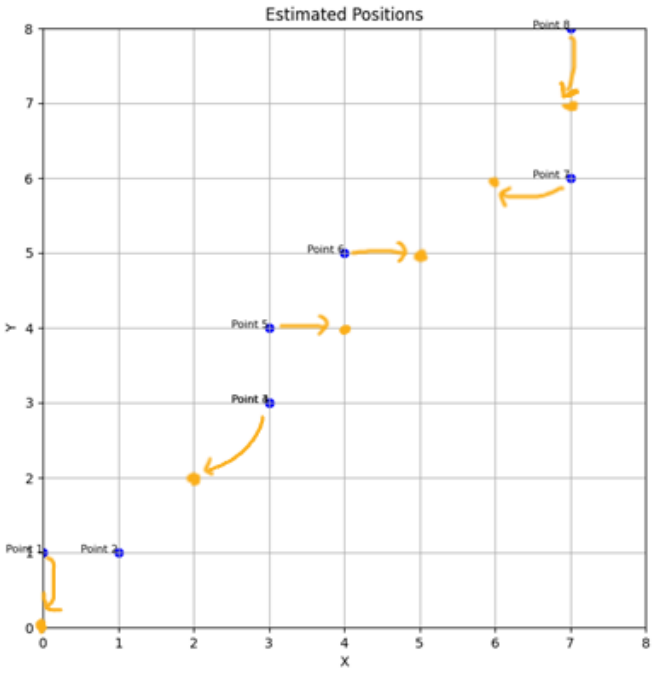

Figura 21. Ejemplo mapa de ruta

En esta figura se observa que el nivel de error para las posiciones es exactamente de un metro, nivel de error establecido como mínimo para el sistema implementado. Este hecho provoca la elección de este segundo método como método seleccionado. Pues la ruta seguida por el dispositivo es claramente diferenciable, empieza en la esquina inferior izquierda, pasa por el centro y termina en la esquina inferior derecha.

4.7 Interfaz de usuario

Con la aplicación del modelo de "fingerprinting" se ha alcanzado una solución óptima al problema de posicionamiento. El último paso es mostrar esta información de una manera sencilla y visualmente enriquecedora. Por ello, se ha creado una interfaz gráfica con la ayuda de Python, pues todo el procesado esta desarrollado en este lenguaje.

Para su desarrollo se ha utilizado la librería "tkinter" [14] junto a su extensión "customtkinter" [15]. 'tkinter' es el paquete de interfaz por defecto de Python y contiene una gran cantidad de utilidades como pueden ser las ventanas gráficas para interactuar con el usuario.

Por otro lado, “customtkinter” es una extensión de “tkinter” que proporciona elementos extra y actualizados con una estética más moderna y limpia. En una analogía con el diseño de páginas web en HTML, se podría considerar “customtkinter” como la incorporación de un CSS al código con el objetivo de realizar modificaciones estéticas.

El uso del cliente de escritorio desarrollado es muy sencillo. Consta de tres botones que realizan acciones diferentes. Con el primero, se procesa un archivo CSV, es decir, se ejecuta el algoritmo de “fingerprinting”. Es necesario especificar las dimensiones del área del sistema de posicionamiento para una correcta visualización de los datos en el mapa creado, para ello el usuario debe ingresar las dimensiones X e Y máximas del entorno mapeado. Con el segundo, se envía una petición al ESP32 para que liste los ficheros contenidos en su tarjeta SD. Por último, el tercer botón envía una petición de descarga del fichero especificado en el cuadro de texto. Para que estos dos últimos botones funcionen correctamente, el usuario debe especificar la dirección IP del ESP32 o su nombre de dominio, por defecto viene configurado como “beaconhost”.

El cliente de escritorio se muestra en la Figura 22.

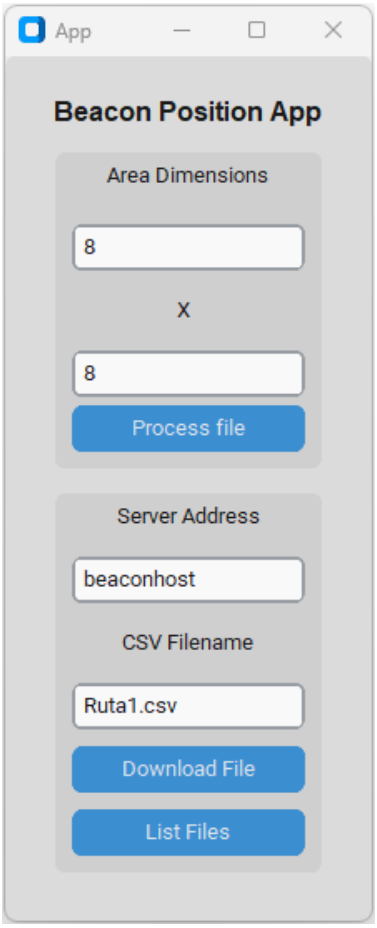

Figura 22. Cliente de escritorio

Esta versión con interfaz gráfica incluida se ha encapsulado en un ejecutable (.exe) con la finalidad de no necesitar de Python instalado en el ordenador donde se ejecute. La creación de este ejecutable ha sido posible gracias a la herramienta Pyinstaller [16].

Pyinstaller agrupa un programa en Python junto a todas sus dependencias en un único paquete. Esto facilita al usuario la ejecución del programa sin la necesidad de instalar un intérprete de Python, así como el resto de las librerías utilizadas.

Descargar esta herramienta requiere de tener instalada la herramienta Pip y ejecutar el siguiente comando

```
pip install pyinstaller
```

Esta herramienta se ejecuta mediante la ventana de comandos de Windows desde el directorio donde están localizados los archivos del programa que se quiere convertir.

Una vez terminado el código y asegurarse que no contiene errores, se ejecuta la herramienta pyinstaller. Esta herramienta dispone de una gran cantidad de comandos para realizar diferentes acciones sobre el ejecutable. Para el caso de este programa se han utilizado las siguientes:

```
--onefile: Empaqueta todo en un solo archivo .exe

--windowed: Para aplicaciones GUI, suprime la consola

--icon=<icon_path>: Especifica un icono para el fichero .exe

--name nombre_de_ejecutable: Especifica el nombre asignado al fichero .exe

--specpath: Genera archivo con extensión .spec en el directorio actual
```

Al ejecutar el comando incluyendo *specpath*, se puede realizar el proceso de creación del ejecutable en dos pasos.

El primer comando genera un ejecutable junto a un archivo *mi_programa.spec* con una serie de parámetros configurables. Se trata de un archivo de especificaciones para personalizar el proceso de empaquetado realizado por pyinstaller, puede utilizarse en caso de necesitar especificar un paquete que la herramienta haya pasado por alto o añadir ficheros adicionales.

El segundo comando crea el ejecutable a partir del archivo de especificaciones. Una vez finalizado, se crean dos directorios (dist y build). El ejecutable que nos interesa se encuentra dentro de la carpeta dist.

```
pyinstaller –onefile –specpath . mi_programa.py
pyinstaller mi_programa.spec
```

La aplicación desarrollada está pensada para comunicarse con el ESP32 y procesar los ficheros CSV que contienen las rutas con las posiciones a estimar. Esta comunicación se realiza a través de red local mediante IP o DNS con nombre “beaconhost”.

Como añadido adicional, el procesado de un fichero devuelve el mapa mediante un gráfico en 2D junto a un cuadro de texto donde viene detallada toda la información en formato JSON.

Esta información se devuelve en este formato de tal forma que el usuario final pueda modificar o tratar los datos de una manera más cómoda, pues es un formato ampliamente utilizado en web y basado en texto plano, facilitando su legibilidad y el procesamiento de la información mediante librerías y funciones específicas en infinidad de lenguajes de programación.

Se ha estructurado siguiendo el formato especificado en el siguiente cuadro de texto.

```
[
    {
        "Point Number": 1,
        "Position": {
            "X": 4.0,
            "Y": 4.0,
            "Direction": "East"
        },
        "Timestamp": "4/8/2024 18:20"
    }
]
```

Por último, la funcionalidad del programa en su totalidad se muestra en el Diagrama 6.

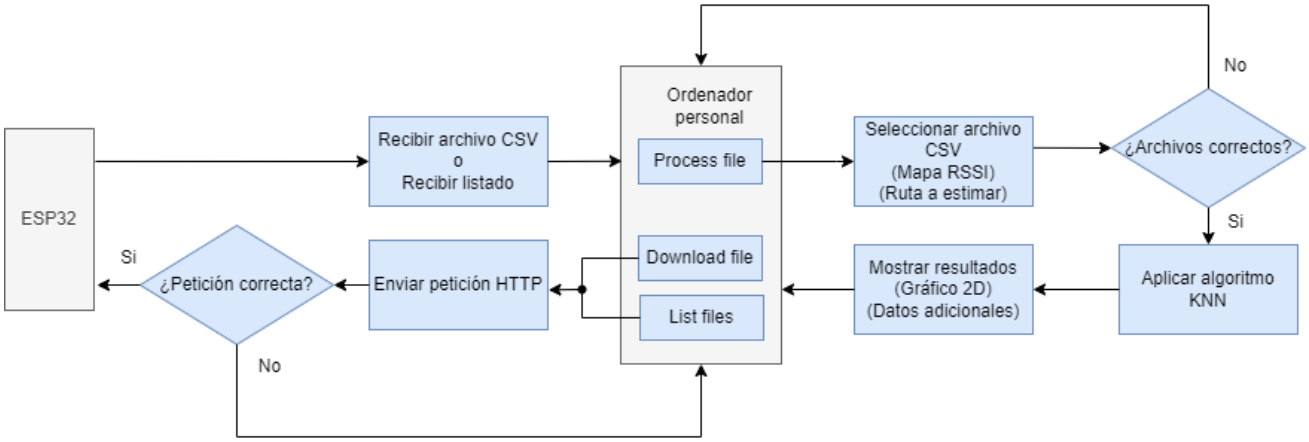

Diagrama 6. Cliente de escritorio

5. Resultados

En este capítulo se detallan las pruebas realizadas en cada fase de desarrollo explicada en el capítulo 4 de esta memoria. Estas pruebas se dividen en 3 fases.

5.1 Procesado RSSI

El objetivo de esta fase de experimentación es caracterizar la señal BLE recibida. Para lograrlo, se realiza un análisis de la señal recibida en diferentes distancias conocidas. Las balizas se colocan inicialmente a 0 metros y se establecen marcadores cada metro en línea recta hasta alcanzar una distancia de 10 metros. Esta distancia se ha determinado debido a las limitaciones observadas en los datos registrados por el ESP32, donde a partir de esta distancia la señal registrada es muy débil o no se recoge directamente.

Durante el transcurso de esta prueba se ha mantenido siempre visión directa entre baliza y ESP32. De tal forma que se asegura que la señal recibida presenta el nivel mínimo de ruido posible.

Además, las balizas se enumeran como sigue: baliza 1, 2, 3 y 4 corresponde a BLE v4.1, mientras que baliza 5 utiliza BLE v5.4.

Este estudio se divide en cuatro fases principales, cada una destinada a parametrizar la señal recibida para su posterior análisis en los métodos de posicionamiento desarrollados.

Para cada fase explicada a continuación, se incorporan tablas numéricas adicionales en el apartado de Anexo A.1 Procesado RSSI.

1. Posicionamiento balizas eje vertical/horizontal (plano XY o plano ZY)

Prueba realizada para determinar la posición óptima de las balizas. Entre las posibles colocaciones se ha considerado el posicionamiento de la baliza en vertical (plano ZY) y en horizontal (plano XY). La diferencia fundamental radica en la posición de la antena dentro del encapsulado de la baliza, donde dependiendo de su colocación la señal puede verse más o menos obstruida por el propio cuerpo de la baliza. Estas posiciones se muestran la Figura 23.

Figura 23. Posicion de baliza

La prueba se ha realizado con las balizas BLE v4.1. y ha arrojado los datos mostrados en las Figuras 24 y 25.

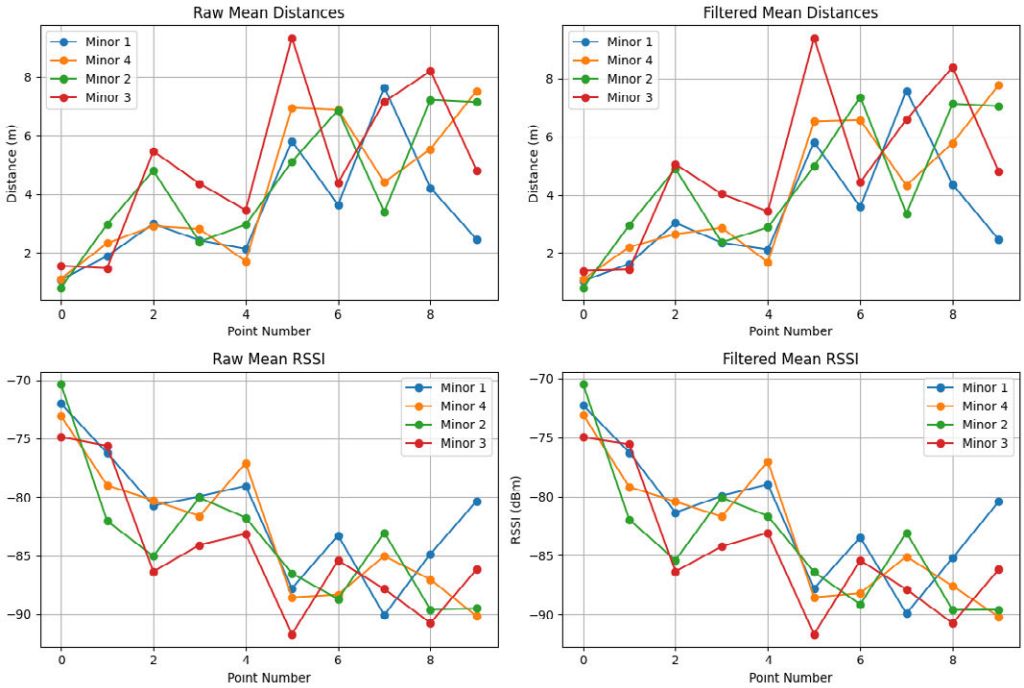

Figura 24. Posición horizontal 4 balizas

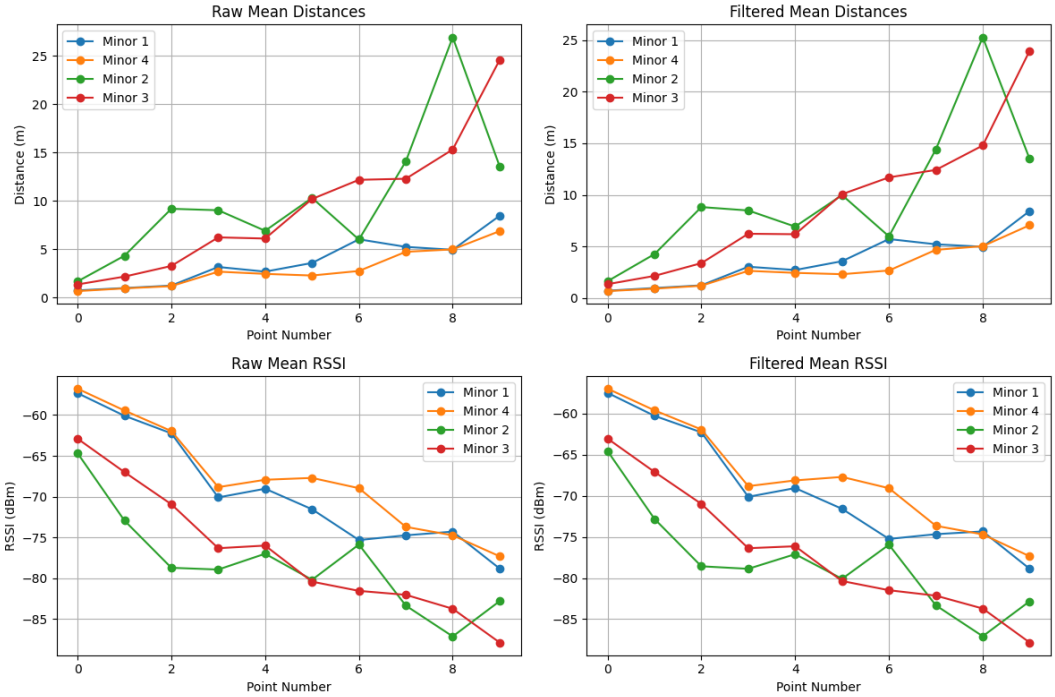

Figura 25. Posición vertical 4 balizas

Se presentan cuatro gráficas por figura, mostrando los resultados de dos configuraciones de posicionamiento (horizontal y vertical) de las balizas. En las gráficas de la izquierda se muestra:

1. **Cálculo de distancia a partir de RSSI (Gráfica superior)**: Muestra la relación entre valor medio de RSSI con su distancia estimada en las diferentes distancias analizadas. Se calcula mediante la fórmula de "Log Distance Path-Loss Model". La fórmula es:

$$d = 10^{\frac{txPower - RSSI}{10 * FreeSpaceFactor}}$$

2. **Media de valores RSSI a cada distancia analizada (Gráfica inferior):** Muestra la media del valor RSSI en dBm para las diferentes distancias analizadas (De 1 a 10 metros).

En las gráficas de la derecha se representa la misma información que en las gráficas de la izquierda, pero tras haber sido filtrados los valores de RSSI mediante el filtro de Kalman. Este filtro se emplea para mejorar la precisión de las mediciones al reducir el ruido y las fluctuaciones en los valores de RSSI.

1. **Cálculo de distancia a partir de RSSI filtrado (Gráfica superior)**: Muestra la relación entre valor de RSSI filtrado con su distancia estimada en las diferentes distancias analizadas. Se calcula de la misma manera que la gráfica de la izquierda.
2. **Valores RSSI filtrados a cada distancia analizada (Gráfica inferior):** Muestra el valor del RSSI en dBm tras aplicar el filtro de Kalman al conjunto de valores crudos para las diferentes distancias analizadas.

Junto a estas Figuras 24 y 25, se incluyen figuras individuales que detallan las medidas tomadas en cada metro analizado para cada baliza (véase Figura 16. Representación señal RSSI y distancia para posición conocida (1m) del apartado 4.4.4). Estas medidas muestran un comportamiento similar entre las cuatro balizas, con ligeras variaciones en sus valores, como se observa en la comparativa de las Figuras 24 y 25.

Para mejorar la comprensión de esta enorme cantidad de datos y figuras, en la Tabla 7 se representan los datos numéricos correspondientes a la baliza 1 para las dos posiciones tratadas.

Distancia	Horizontal				Vertical			
	Max	Min	Media	Desviación estándar	Max	Min	Media	Desviación estándar
1	-65.7612	-81.4049	-72.2722	4.3087	-55.2318	-62.0000	-57.4726	1.6543
2	-72.2564	-84.4808	-76.2748	3.9334	-58.1469	-63.0000	-60.2670	1.4250
3	-76.2220	-88.0778	-81.3607	4.3139	-61.2406	-64.1636	-62.2747	0.7072
4	-76.7676	-85.0854	-79.9527	2.0548	-69.0106	-73.6041	-70.1149	1.0154
5	-76.0000	-81.1130	-78.9815	1.1459	-66.6150	-70.9088	-69.0547	1.0044
6	-85.0000	-91.9420	-87.7918	1.8314	-69.6078	-75.0000	-71.5800	1.4601
7	-79.1069	-87.0925	-83.4709	2.0309	-72.0579	-79.4198	-75.2442	2.6135
8	-85.0000	-95.5030	-89.9032	2.7275	-72.0000	-77.7939	-74.6630	1.5790
9	-82.1389	-94.0000	-85.2009	3.3069	-72.0809	-78.3068	-74.3276	1.4259
10	-79.1425	-83.0000	-80.4300	0.8034	-76.3486	-82.0000	-78.8708	1.4317

Tabla 7. Comparativa RSSI baliza 1

El resto de las tablas pertenecientes a las otras balizas analizadas se encuentran en el apartado Anexo A.1.1 Posicionamiento Horizontal / Vertical.

Parte de esta prueba consiste en identificar el valor del RSSI a 1 metro de distancia. Al observar los resultados obtenidos para la baliza1, los rangos de RSSI en la posición vertical son más cercanos a lo definido por el fabricante (-55 dBm).

Tras analizar las dos configuraciones de posicionamiento (vertical y horizontal) mediante las figuras presentadas, y apoyándose en los datos numéricos de las tablas. Se concluye que la posición vertical es la óptima. Esta decisión se basa en respuesta a los siguientes puntos.

Mejor precisión de RSSI: En las gráficas individuales y en los valores numéricos de las tablas, los valores RSSI para cada distancia que menos varían en rango corresponden con la posición vertical.

Comparación de datos crudos y filtrados: La aplicación del filtro de Kalman muestra el mismo nivel de suavizado en las dos configuraciones. Sin embargo, es más efectiva en la posición vertical debido a su mayor estabilidad en el rango de valores RSSI por distancia analizada.

Si bien es cierto que el error aumenta con la distancia. Este hecho ocurre en las dos posiciones analizadas, con la diferencia de que en la posición vertical los valores registrados realizan una transición más suave en todo el rango de valores RSSI. Por ello la posición seleccionada es la vertical.

2. Comportamiento balizas en diferentes ambientes

El objetivo de esta prueba es observar el comportamiento de las señales RSSI en dos ambientes diferentes: uno con presencia de ruido y otro con ruido e interferencias mitigados.

Las características de estas pruebas se recogen en la Tabla 8.

Prueba	Ambiente	Ubicación	Factores de Ruido	Interferencias Potenciales
1	Con ruido	Parque público	Presencia de personas y coches con dispositivos electrónicos	Dispositivos móviles, sistemas de navegación de vehículos, otros dispositivos electrónicos portátiles
2	Sin ruido	Espacio controlado (campo)	Ausencia de personas y vehículos, mínimo ruido electrónico	Sin dispositivos electrónicos ajenos, ni presencia de personas o vehículos

Tabla 8. Comparativa ambientes prueba 2

Para conseguir estas dos situaciones, la primera prueba se ha realizado en un parque público, con personas y coches pasando alrededor. Se entiende que estas personas y vehículos portaban dispositivos electrónicos. La segunda prueba se ha realizado en mitad de una arboleda.

Los resultados se muestran en dos figuras diferentes. Estas figuras recogen el mismo tipo de información que la figura análoga de la prueba anterior. Figuras 26 y 27.

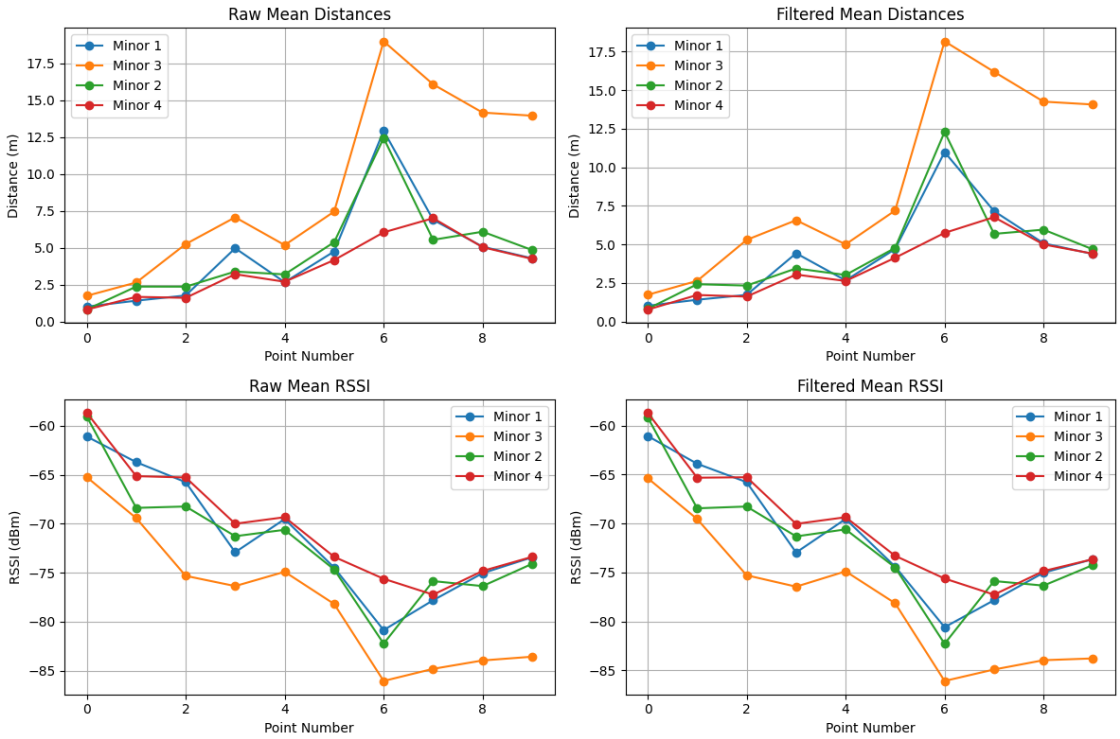

Figura 26. Balizas en presencia de ruido

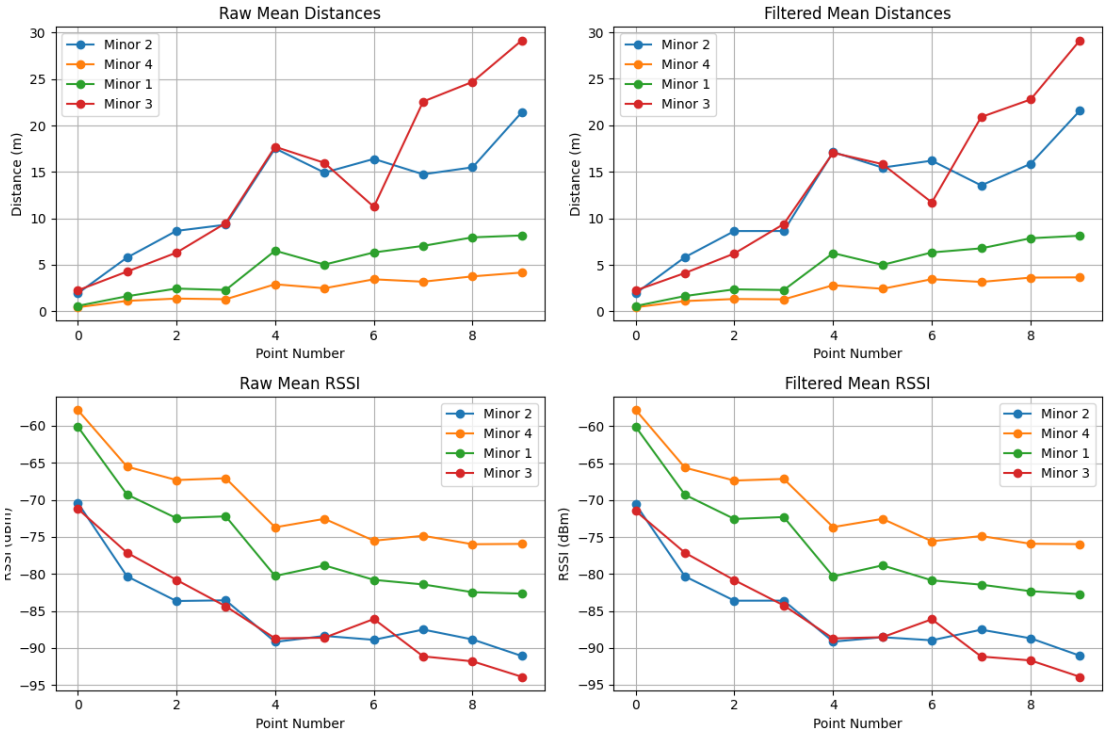

Figura 27. Balizas en ausencia de ruido

La comparación de los resultados obtenidos en estos dos ambientes refleja claramente como el ruido y las interferencias afecta la precisión y estabilidad de las mediciones de RSSI. Aunque estas fluctuaciones se ven mitigadas por el filtro de Kalman, no desaparecen por completo. Esta situación crea la necesidad de incluir y analizar una nueva baliza basada en BLE v5.4 para

comparar su comportamiento y evaluar si se logra mayor estabilidad y precisión al traducir el valor de RSSI a distancia.

3. Baliza 5 (BLE 5.4) y su comportamiento individual y junto a otras balizas

El objetivo de esta prueba es comparar el comportamiento de las balizas BLE v4.1 junto a las balizas BLE v5.4, pues estas ofrecen mejoras significativas en términos de alcance y potencia de señal, lo que podría reducir las fluctuaciones de RSSI observadas hasta el momento.

Al analizar esta baliza frente a las otras basadas en BLE v4.1 se obtiene la Figura 28.

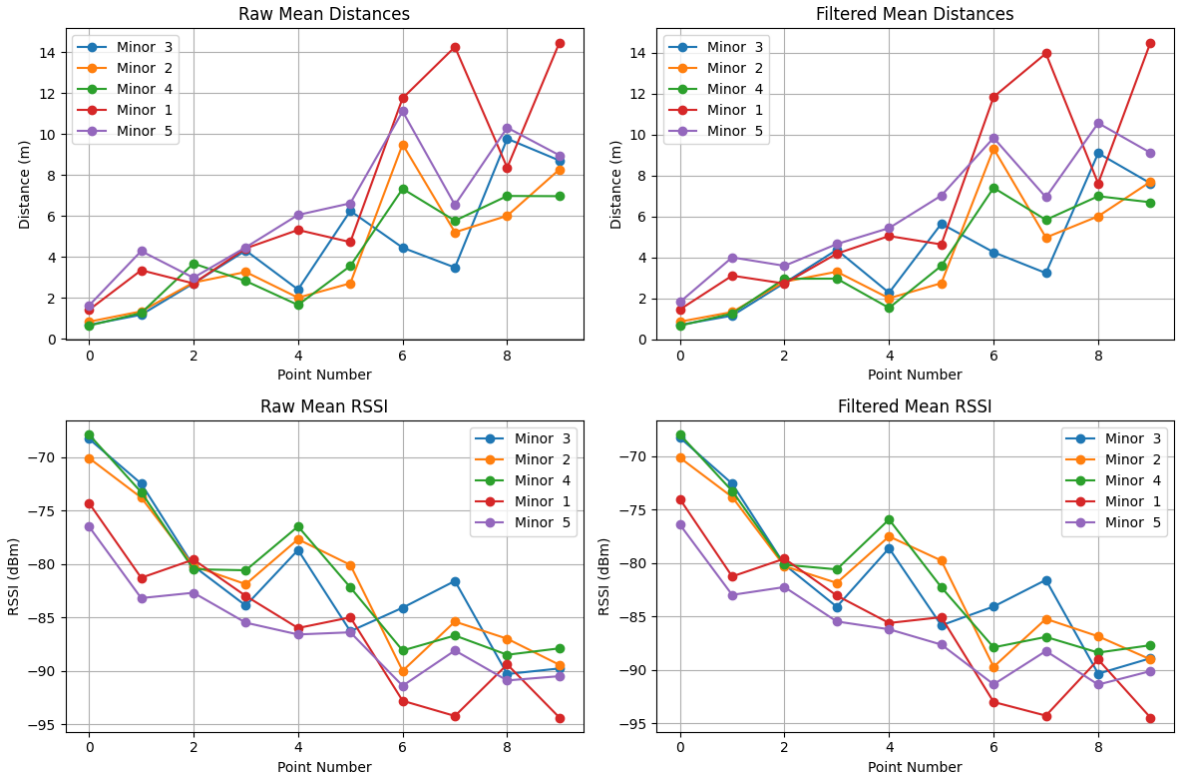

Figura 28. Comparativa entre balizas BLE 4.1 vs balizas BLE 5.4

Ante la comparativa de la baliza BLE v5.4 respecto a las balizas BLE v4.1, no se observan mejoras significativas. De hecho, la baliza 5 (BLE v5.4) muestra un comportamiento similar al de una baliza basada en BLE v4.1. Con total probabilidad esta similitud se ve condicionada por la versión BLE del dispositivo ESP32 utilizado, compatible únicamente con la versión BLE v4.1. Como resultado, las ventajas teóricas de la BLE v5.4 no se manifiestan en la práctica debido a esta restricción técnica.

5.2 Método trilateración (Solución 1)

En esta fase del proyecto, se utilizan las balizas basadas en BLE v4.1 para estimar la posición mediante valores RSSI recibidos por el dispositivo ESP32. La meta es obtener una ruta legible que presente el mínimo nivel de error.

Para ello, se comparan los dos métodos desarrollados: trilateración y multilateración. Ambos métodos utilizan el valor de RSSI para calcular la distancia desde el ESP32 hasta las diferentes

balizas, con la diferencia de que en el caso de trilateración se descartan las balizas sobrantes y sus distancias, utilizando los tres valores más próximos únicamente, mientras que la multilateración utiliza todos los datos recogidos por el ESP32.

Las pruebas se han realizado sobre una superficie cuadrada de 8x8 metros. La primera prueba se posiciona únicamente en el centro del cuadrado y se realizan cuatro mediciones. La segunda prueba sigue una línea recta desde la baliza 1 hasta la baliza 4.

Esta prueba permite evaluar la efectividad de la trilateración y la multilateración en la estimación de rutas en un entorno controlado y estos son los resultados de las dos rutas analizadas, Figuras 29 y 30.

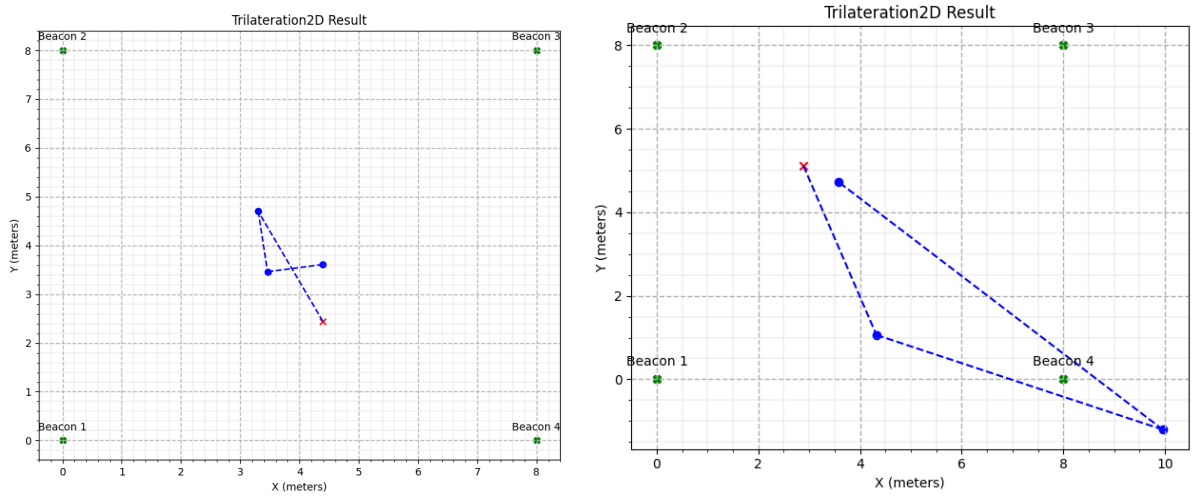

Figura 29. Ruta 1 y 2 método trilateración

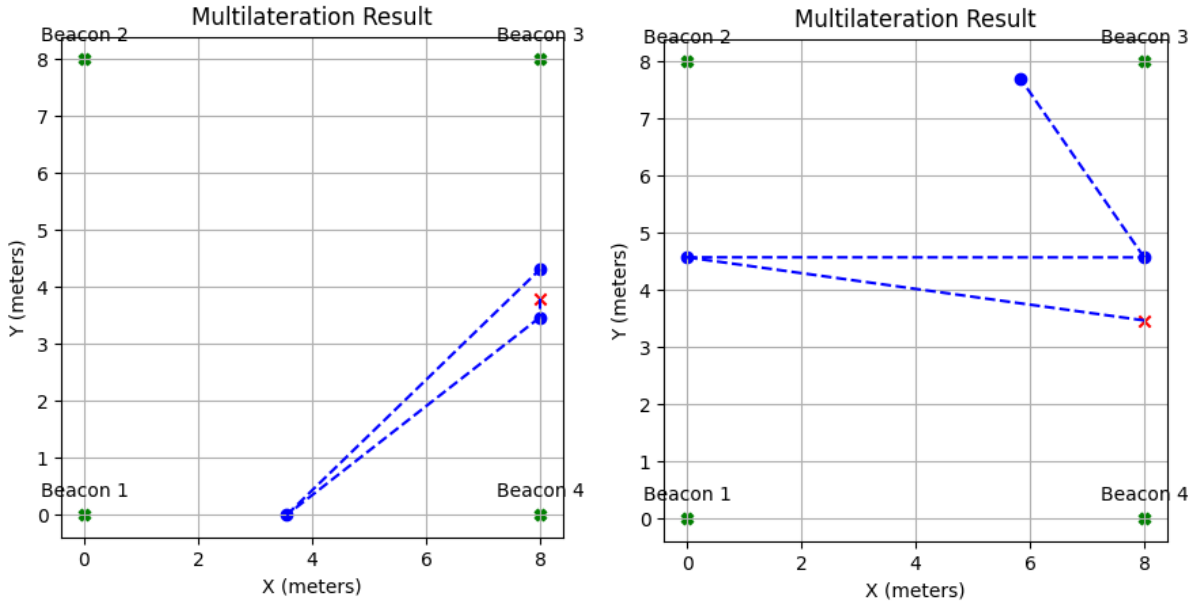

Figura 30. Ruta 1 y 2 método multilateración

Los resultados demuestran que las fluctuaciones en la señal y su valor RSSI afectan significativamente la precisión del cálculo de distancia estimada y de la posición. Sin embargo,

el algoritmo de trilateración parece tener un mejor comportamiento al mitigar parte de este problema, descartando las medidas de las balizas más lejanas y utilizar sólo lo valores más próximos, mejorando la precisión en comparación a la multilateración. Al utilizar este último método las señales recibidas de todas las balizas disponibles, la estimación de la posición es más susceptible a las fluctuaciones de la señal.

Estas fluctuaciones en la señal han sido tratadas mediante un filtro de Kalman que ayuda a reducir las fluctuaciones al suavizar los valores. No obstante, no ha sido suficiente, bien por la naturaleza de la señal o por un nivel de filtrado muy ligero.

Por tanto, este método de trilateración y multilateración no han mostrado los resultados necesarios para tener en consideración su uso como parte del sistema de posicionamiento en interiores desarrollado.

5.3 Método fingerprinting (Solución 2)

El algoritmo basado en la técnica del "fingerprinting" se desarrolla como segunda solución ante la escasa precisión y gran cantidad de errores de estimación presentes en los modelos de trilateración y multilateración. Esta técnica se ha implementado siguiendo los pasos descritos en los apartados 2 y 4 de esta memoria, que se refieren a la técnica de "fingerprinting" y al algoritmo KNN.

Las pruebas se han realizado en una superficie máxima de 8x8 metros. Se han trazado varias rutas en dos versiones: una sin magnetómetro y otra con magnetómetro.

En el caso de tener magnetómetro el algoritmo de los KNN toma en consideración las muestras que corresponden a la misma dirección de la base de datos. Esto mejora y agiliza el proceso de estimación de la posición del ESP32.

En caso de no tener magnetómetro o no tener un valor de dirección asociado (Norte, Sur, Este, Oeste) el algoritmo de los KNN compara las posiciones de la ruta con todas las direcciones de la base de datos y da una estimación.

A continuación, se muestran cuatro figuras, dos por cada versión. En la primera ruta se han bordeado los laterales del cuadrado, comenzando en la esquina inferior izquierda. En la segunda prueba se ha seguido una línea recta desde la esquina inferior izquierda hasta la esquina superior izquierda. Estas gráficas se muestran en las Figuras 31 y 32.

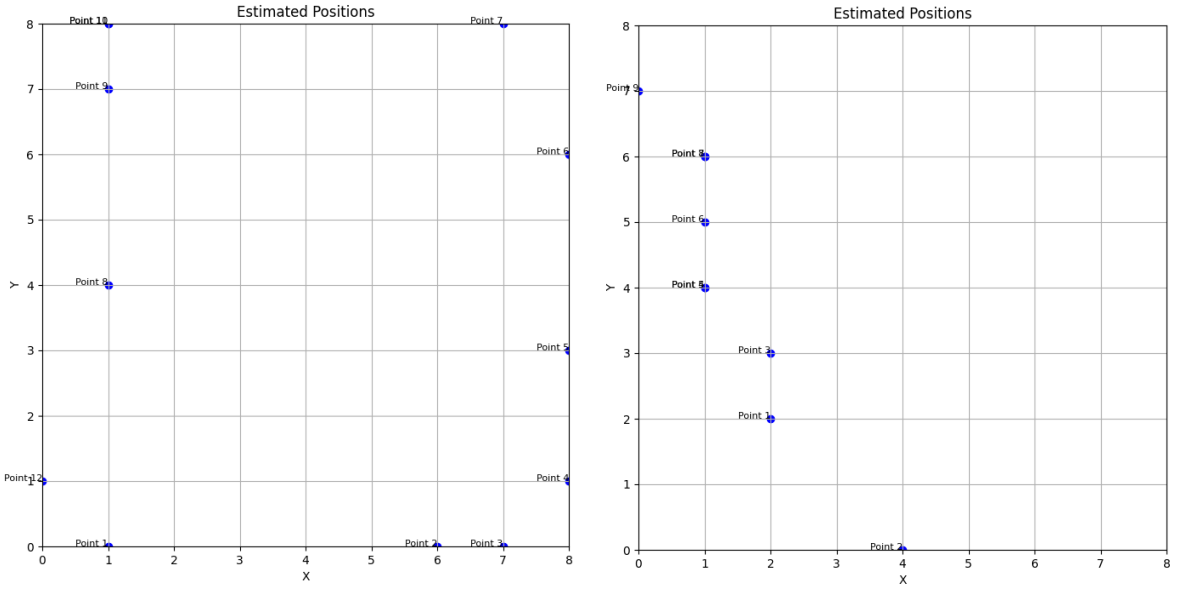

Figura 31. Rutas 1 y 2 método fingerprinting sin magnetómetro

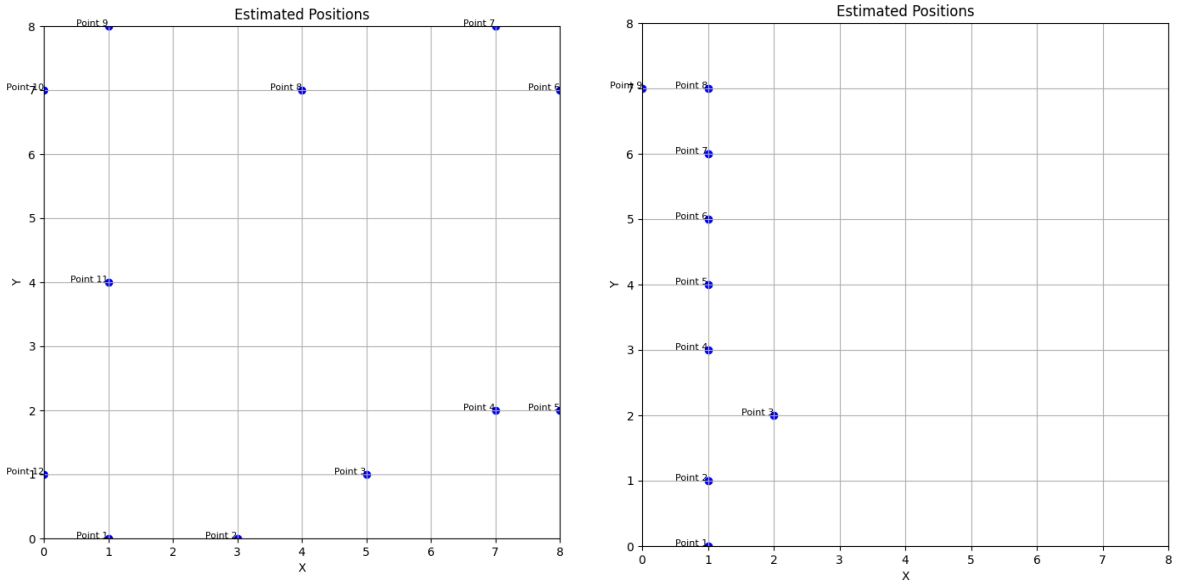

Figura 32. Rutas 1 y 2 método fingerprinting con magnetómetro

Las tablas con las coordenadas numéricas de estas cuatro rutas se encuentran en el apartado de Anexo A.2 Resultados método fingerprinting.

Los resultados obtenidos reflejan el éxito de este segundo método (fingerprinting) en la estimación de las posiciones basadas en los valores de RSSI recibidos para distintas posiciones de un espacio mapeado con anterioridad. Este éxito se establece tanto para la solución sin y con magnetómetro pues en ambos casos el número de errores es reducido. No obstante, esta implementación mejora significativamente con el añadido de la información recogida por el magnetómetro al reducir el número de elementos a comparar por el algoritmo, facilitando un ajuste más preciso.

En general, este método ha demostrado ser más preciso y fiable en la estimación de la posición comparado con los métodos de trilateración y multilateración. Por ello, se ha seleccionado como la solución principal para el problema de posicionamiento en interiores planteado en este proyecto fin de grado.

5.4 Simulación de entorno real

Como prueba adicional, se ha realizado una simulación de un entorno real utilizando seis balizas situadas en los laterales de un rectángulo de dimensiones 15x5 metros cuadrados, simulando las dimensiones de un corral de ganado, donde uno de los lados más largos es una pared. Cuatro de estas balizas se han colocado en las esquinas y las otras dos se han situado en el centro de los lados más largos, cubriendo de esta manera un área mayor con señales BLE. Como recordatorio, la viabilidad práctica de las balizas se sitúa en un rango máximo de 10 metros.

A continuación, en la Figura 33 se muestra una representación del espacio donde se ha realizado la prueba.

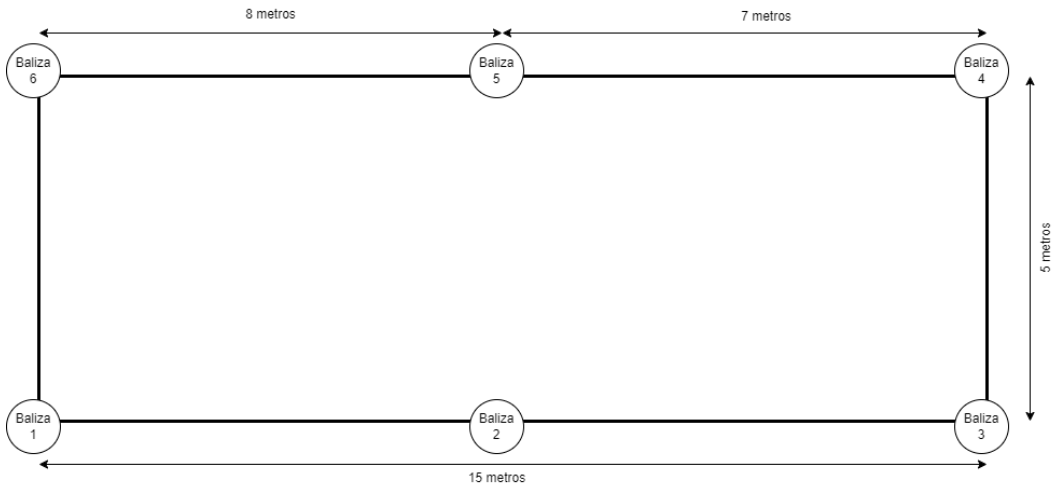

Figura 33. Dimensiones recinto prueba collar

Para la realización de esta prueba, se ha mapeado la zona para crear un mapa de dimensiones 15x5 metros cuadrados que registre la señal proveniente de las seis balizas utilizadas. Además, se ha incluido el uso de un collar para ganado, y se han realizado dos tipos de rutas para comprobar su rendimiento: una prueba sin collar y otra prueba con collar.

El ESP32 en las pruebas con collar para ganado va colocado dentro de una caja en un lateral del collar, quedando de esta forma pegado al cuello del animal en posición vertical.

El objetivo de esta prueba es determinar si el uso del dispositivo en una posición diferente a la utilizada para crear el mapa, junto a su uso cubierto por una caja, entorpece el registro de las señales BLE por parte del ESP32. Además, se pretende analizar el funcionamiento del sistema cuando se superan las condiciones de distancia para la señal RSSI (máximo 10 metros) y cómo se comporta al solventar esta característica añadiendo más balizas en el entorno.

Los dos tipos de rutas realizadas y sus resultados pueden observarse en las siguientes figuras:

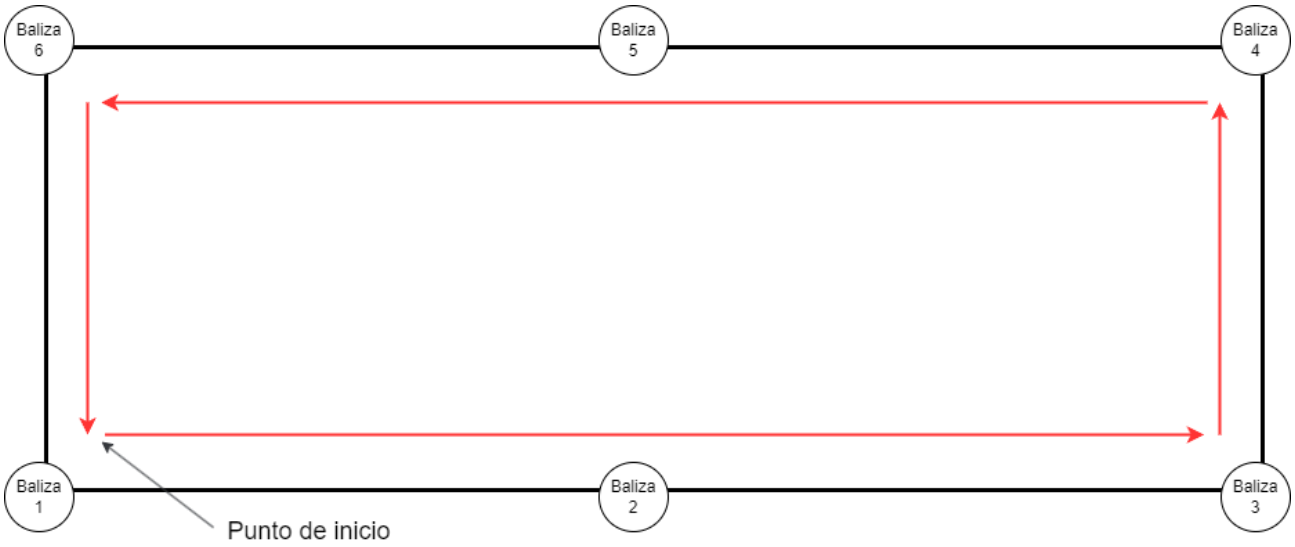

Figura 34. Esquema ruta 1

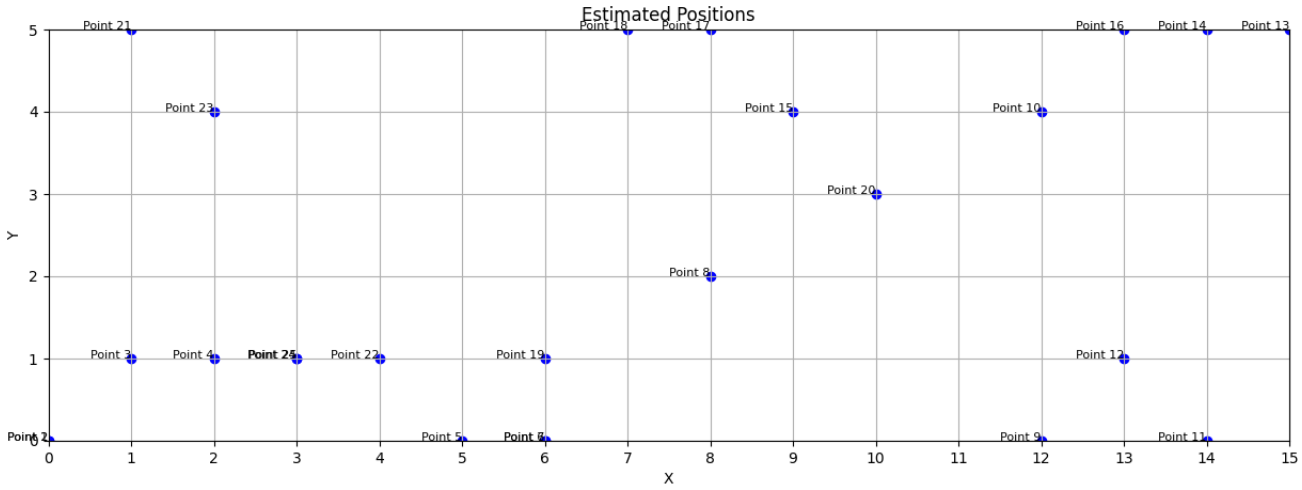

Figura 35. Ruta 1 sin collar

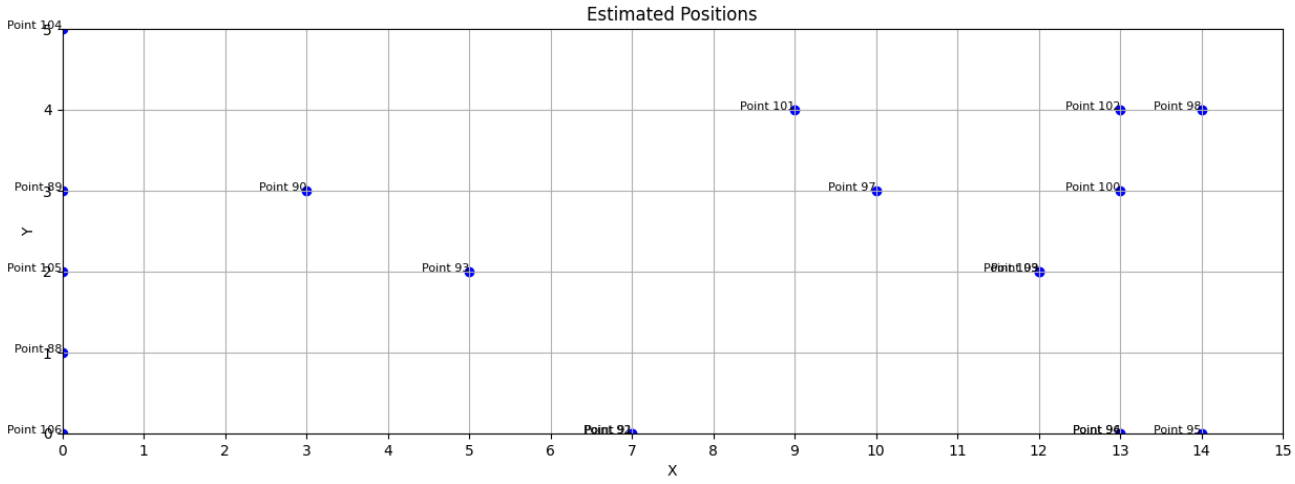

Figura 36. Ruta 1 con collar

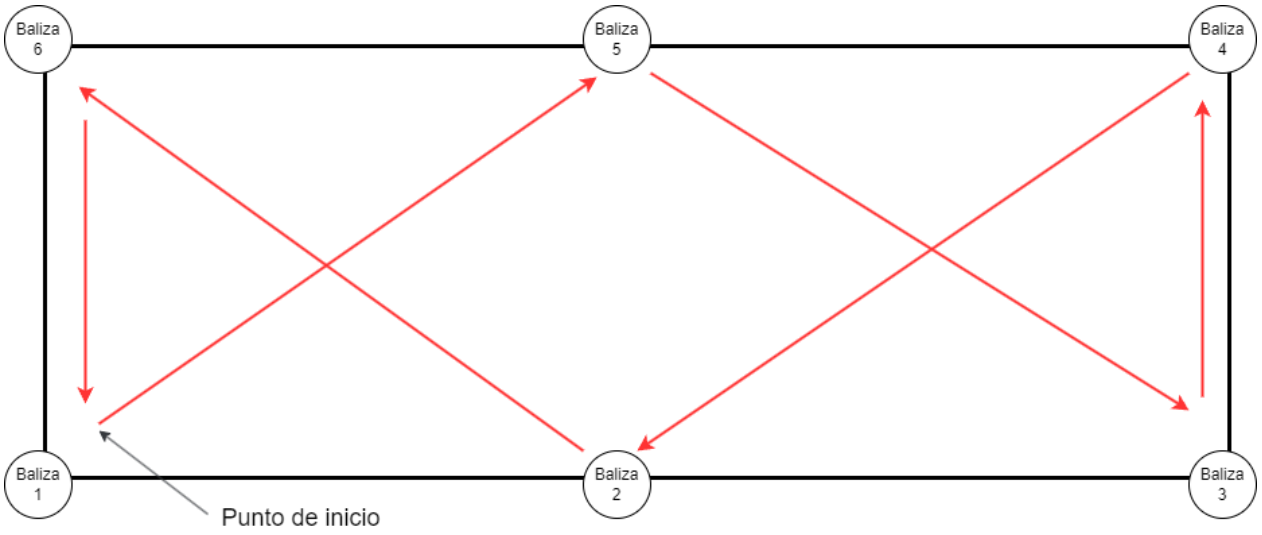

Figura 37. Esquema ruta 2

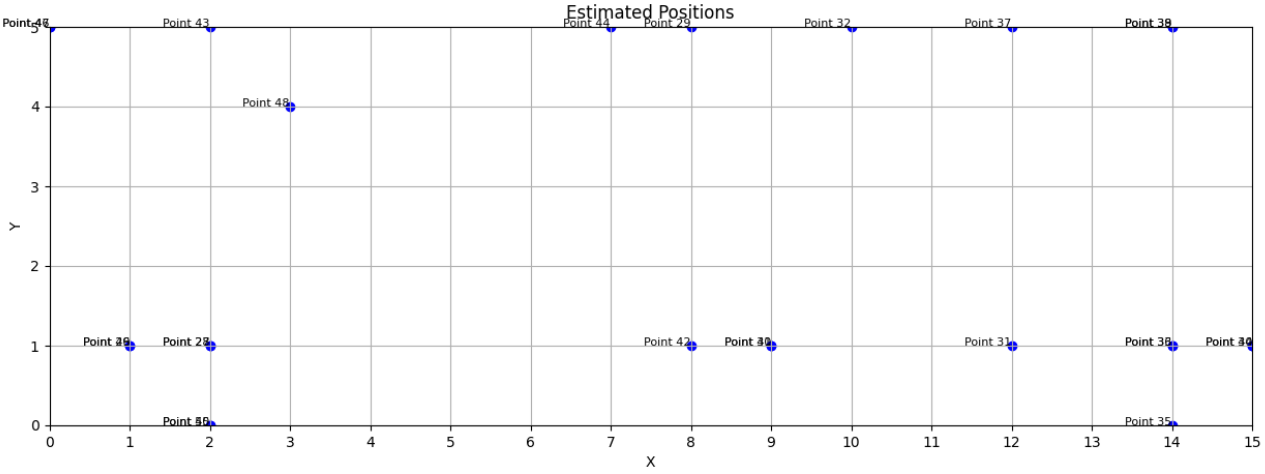

Figura 38. Ruta 2 sin collar

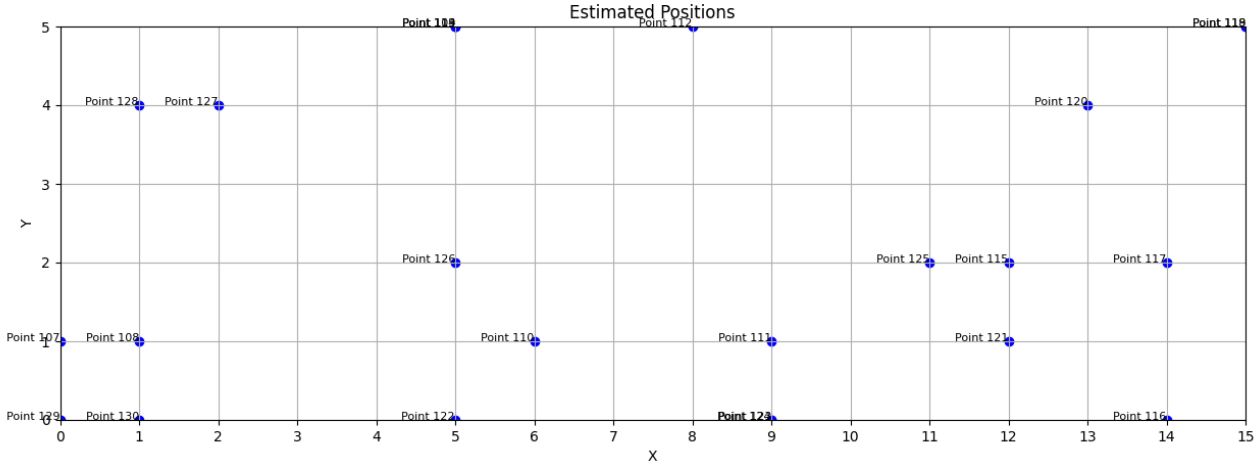

Figura 39. Ruta 2 con collar

Las pruebas realizadas en un entorno de 15x5 metros han revelado que al utilizar el ESP32 en la mano del usuario o en el collar, dentro de una caja de plástico y en posición vertical, los recorridos presentan un nivel de error considerable. Estos errores están presentes en ambas pruebas y son suficiente para dificultar la comprensión visual de la ruta seguida por el usuario. Sin embargo, los resultados de coordenadas numéricos permiten descartar fácilmente las estimaciones de posición erróneas al seguir la ruta punto por punto, ya que estas desentonan entre las posiciones correctas adyacentes en el tiempo.

Para agilizar este análisis se incluyen las tablas numéricas en el apartado de Anexo A.3 Resultados de simulación en entorno real.

El empleo de las balizas BLE en un entorno con distancias de hasta 15 metros dificulta la adquisición de los valores RSSI, pues a partir de 10 metros estos valores se reciben con un alto nivel de saturación. En el análisis del mapa realizado se han observado valores recogidos que superan los -90dBm para las balizas más alejadas, lo que dificulta enormemente el proceso de clasificación implementado mediante el algoritmo KNN al no contar con un conjunto de valores diferenciables entre sí.

En comparación con las pruebas realizadas durante el desarrollo del método “fingerprinting” en áreas de 8x8 metros y utilizando únicamente 4 balizas, esta prueba de simulación en un entorno real muestra un nivel de error más elevado con una presencia de estimaciones erróneas en torno al 20% para la prueba sin collar y que se eleva hasta el 30% en los casos realizados con el collar de ganado.

En resumen, el sistema puede utilizarse con las dimensiones actuales (15x5 metros) siempre y cuando se aumente el número de balizas, pero es importante considerar que habrá estimaciones de posición erróneas entre el conjunto de posiciones estimadas. Por lo tanto, depende del usuario final considerar o no esta limitación para el uso del sistema de posicionamiento en entornos que superen las dimensiones establecidas como óptimas de 8x8 metros.

6. Conclusiones

Con este proyecto fin de carrera se ha pretendido diseñar un sistema de posicionamiento en interiores capaz de estimar la ubicación de un usuario o activo con la mayor precisión posible, utilizando la tecnología Bluetooth BLE y un dispositivo Arduino ESP32. En concreto se han utilizado las balizas BLE basadas en el chip NRF51822 y el dispositivo Arduino stackM5 CORE2.

Durante el desarrollo de este proyecto se han abordado dos soluciones. El primer sistema de posicionamiento está basado en el método de trilateración, método que está implementado en diversos sistemas de posicionamiento de tecnologías de radiofrecuencia. Sin embargo, con este sistema no se ha logrado el grado de desempeño deseado para ser viable en entornos y situaciones reales. El principal problema identificado es el ruido presente en la señal recibida, provocado por interferencias, ya sea del propio usuario, de características del terreno o de solapamiento con otras señales del entorno.

Este problema ha requerido del análisis de la señal proveniente de las balizas, en concreto de su valor RSSI, el cual ha arrojado resultados desfavorables para la implementación de un sistema de posicionamiento utilizando este método.

Las fluctuaciones entre valores de RSSI han representado un desafío significativo para la precisión de estos dos métodos: trilateración y multilateración. Las pruebas realizadas en entornos con ruido y afluencia de personas y vehículos han demostrado que las interferencias afectan en gran medida la estabilidad de las mediciones. Estas limitaciones resaltan la necesidad de mejorar los algoritmos de filtrado para reducir el impacto de las interferencias y las fluctuaciones de la señal.

Para solventar esta situación, se ha enfocado el problema desde un método diferente, la técnica de “fingerprinting” o huella dactilar. Este método ha resultado en un sistema exitoso, capaz de ubicar a usuarios o activos dentro de un espacio mapeado con anterioridad. No obstante, presenta una clara desventaja respecto al primer método, la necesidad de una fase inicial de mapeo de la zona en la que se desea implementar el sistema. Este requerimiento imprescindible requiere de tiempo al tener que realizar las medidas de toda la zona, cuanta más precisión se quiera conseguir, más tiempo se emplea en esta fase. Sin embargo, esta fase solo se requiere durante la instalación del sistema, una vez montado y mapeado es un sistema que se comporta de manera sólida y robusta al compararlo con su predecesor.

Con la realización de este proyecto se ha logrado desarrollar una metodología robusta para la estimación de la posición utilizando balizas BLE y ha proporcionado un análisis comparativo de los métodos de trilateración, multilateración y “fingerprinting”. Este desarrollo ha permitido identificar las ventajas y limitaciones de cada método. Además, la implementación del método de “fingerprinting”, mejorado con el uso del magnetómetro, ha demostrado ser una solución efectiva del problema planteado, estableciendo una base sólida para futuros proyectos y mejoras en esta área.

6.1 Impacto del proyecto

En este apartado se pondrá de manifiesto las implicaciones sociales, de salud y seguridad, ambientales, económicas, tecnológicas o industriales relacionadas con el sistema de posicionamiento realizado [17], así como la posible aportación a los ODS (Objetivos de Desarrollo Sostenible) [18].

1. Innovación y desarrollo

   El proyecto fomenta la investigación y desarrollo del empleo de tecnologías como el Bluetooth BLE en el campo del posicionamiento y la monitorización.

2. Impacto ambiental

   La tecnología de posicionamiento en interiores contribuye a una gestión eficiente del ganado, reduciendo sobreexplotación y mejorando la calidad de vida del animal.

   Reducción de la huella ambiental al optimizar el uso de recursos para la explotación ganadera.

3. Responsabilidad social

   Mejora en las condiciones de trabajo de los ganaderos mediante el uso de tecnología que facilite su trabajo.

   Mejora el rendimiento deportivo aplicado en la monitorización de deportistas.

4. Beneficio económico

   El sistema de posicionamiento está diseñado con componentes asequibles. Lo que facilita su empleo por parte de pequeños y medianos autónomos.

En cuanto a la aportación del proyecto a los ODS:

1. ODS 2: Hambre Cero

   El sistema de posicionamiento aplicado en ganado mejora el bienestar animal, monitorizando y mejorando su salud. Se espera una mejora en la productividad ganadera y en la calidad de los productos animales.

2. ODS 3: Salud y Bienestar

   En el ámbito deportivo, el sistema contribuye a la monitorización de atletas mejorando su rendimiento. Estas mejoras previenen lesiones y mejora la planificación de los entrenamientos.

3. ODS 9: Industria, Innovación e Infraestructura

   El desarrollo e implementación de tecnologías de posicionamiento fomentan la innovación y el avance tecnológico en sectores como la agricultura y el deporte.

Este enfoque del proyecto promueve el bienestar social y la protección del medio ambiente.

6.2 Trabajos futuros

Para expandir y mejorar el sistema de posicionamiento planteado en este proyecto fin de grado, se recomiendan las siguientes líneas de trabajo:

1. Mejoras en la configuración de Hardware

Analizar y desarrollar hardware específico basado en un ESP32, un lector de tarjeta SD y un magnetómetro. Una configuración que presenta unos costes muy reducidos en comparación con la versión de desarrollo empleada en la realización del sistema de posicionamiento.

2. Mejoras en conectividad

Integración de conectividad WiFi con servidor web y realizar en el servidor el procesamiento de los datos recolectados por el sistema. El servidor actuaría como almacenamiento y procesamiento.

3. Implementar sistema en tiempo real

Investigar métodos de implementar un software especializado que permita determinar la ubicación de una persona, animal u objeto en tiempo real.

4. Tecnología empleada

Explorar el uso de dispositivos BLE más avanzados, como aquellos que soportan versiones más recientes de BLE, para reducir las limitaciones de hardware actuales.

5. Optimización del proceso de mapeo

Investigar métodos y herramientas que puedan acelerar y automatizar el proceso de mapeo inicial, reduciendo el tiempo y gasto de recursos empleado.

6. Algoritmos de filtrado

Continuar desarrollando y refinando algoritmos de filtrado como el filtro de Kalman, para mitigar las fluctuaciones en los valores de RSSI.

7. Pruebas controladas a largo plazo

Probar el sistema de posicionamiento y analizar su comportamiento a largo plazo (aprox. 1 año) con el objetivo de observar su rendimiento de manera más exhaustiva.

Este proyecto ha establecido una base sólida para la investigación y desarrollo de sistemas de posicionamiento en interiores utilizando tecnología BLE. Las conclusiones y recomendaciones aquí presentadas ofrecen una guía clara para futuros trabajos y mejoras en esta área, con el objetivo de alcanzar niveles superiores de precisión y robustez.

7. Presupuesto

El presupuesto se ha dividido en recursos materiales y recursos humanos. Se ha desglosado como indica la Tabla 9.

Ref	Concepto	Unidades	Precio/ud	Total
1	Ordenador personal	1	1368.89 €	1368.89 €
2	m5stack CORE2	1	43.07 €	43.07 €
3	Baliza BLE NRF51822	7	8.76 €	61.32 €
4	Pila litio 3V CR2477	7	1.26 €	8.82 €
5	Baliza BLE NRF52840	1	14.82 €	14.82 €
6	Pila litio 3V CR2032	1	1.48 €	1.48 €
SUBTOTAL				**1498.40 €**
Ref	Concepto	Horas	Precio/hora	Total
1	Ingeniero Junior	320	24.4 €/h	7808 €
TOTAL				**9306.40 €**

Tabla 9. Presupuesto

Todos los elementos software utilizados son gratuitos y de libre distribución.

8. Referencias

[1] «unigis,» [En línea]. Available: https://www.unigis.es/posicionamiento-indoor/.

[2] «zapt.tech,» [En línea]. Available: https://zapt.tech/es/blog/innovaci%C3%B3n/banda-ultra-ancha-qu%C3%A9-es-uwb/.

[3] «smart-lighting,» [En línea]. Available: https://smart-lighting.es/bluetooth-low-energy-introduccion-la-tecnologia/.

[4] A. Bassi, «gotoiot,» [En línea]. Available: https://www.gotoiot.com/pages/articles/beacons_intro/content.html.

[5] «5hertz,» [En línea]. Available: https://www.5hertz.com/index.php?route=tutoriales/category.

[6] «ibm,» [En línea]. Available: https://www.ibm.com/es-es/topics/knn.

[7] A. Becker, «kalmanfilter.net,» [En línea]. Available: https://www.kalmanfilter.net/ES/default_es.aspx.

[8] «pyhon.org,» [En línea]. Available: https://www.python.org/downloads/.

[9] «pypi.org,» [En línea]. Available: https://pypi.org/project/pip/.

[10] W. Bulten, «wouterbulten,» [En línea]. Available: https://www.wouterbulten.nl/posts/lightweight-javascript-library-for-noise-filtering/.

[11] «matplotlib.org,» [En línea]. Available: https://matplotlib.org/.

[12] FilterPy, «FilterPy.readthedocs.io,» [En línea]. Available: https://filterpy.readthedocs.io/en/latest/.

[13] SciPy, «scipy.org,» [En línea]. Available: https://scipy.org/.

[14] [En línea]. Available: https://docs.python.org/es/3/library/tkinter.html.

[15] «pypi.org,» [En línea]. Available: https://pypi.org/project/customtkinter/0.3/.

[16] «pyinstaller.org,» [En línea]. Available: https://pyinstaller.org/en/stable/.

[17] «Guía para trabajar la Responsabilidad Social y Ambiental,» [En línea]. Available: https://oa.upm.es/35542/1/Guia_Responsabilidad_Social_y_Ambiental-V2-1.pdf.

[18] «Objetivos de Desarrollo Sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/.

Manual de usuario

Este manual explica el proceso de instalación del sistema de posicionamiento que utiliza balizas BLE y un microcontrolador ESP32 para recoger datos de ubicación. Este sistema se puede separar en dos partes. Una parte corresponde al ESP32 y su programa Arduino, y otra parte corresponde al cliente de escritorio y su procesado de los datos recogidos por parte del ESP32.

A.1 Requisitos

- **Balizas BLE v4.2** configuradas con la máxima potencia de transmisión y un intervalo de advertising de 300ms.
- Un **microcontrolador ESP32** con capacidad WiFi y Bluetooth.
- Un **magnetómetro** para determinar la dirección del ESP32. (Mejora la precisión del sistema de posicionamiento).
- **Tarjeta SD** para almacenar los datos.

Para el proyecto desarrollado se han utilizado los siguientes dispositivos:

- Balizas: Baliza BLE v4.2 basadas en NRF51288 del fabricante Holyiot.
- ESP32: Versión de desarrollo m5stack CORE2 basada en ESP32 del fabricante m5stack.
- Magnetómetro: GY-271 de Az-Delivery.
- Tarjeta SD: Genérica.

A.2 Configuración del ESP32

Como requisito de uso de este apartado, el usuario debe tener instalado el IDE de Arduino. Una vez instalado, ejecutar los siguientes pasos para instalar el programa Arduino proporcionado:

A.2.1. Configurar IDE de Arduino

- Ve a Archivo > Preferencias y en URLs adicionales de gestor de placas añade: “https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json” si utilizas un m5stack CORE2.

En caso de utilizar otro dispositivo distinto basado en ESP32 deberá añadir la URL correspondiente.

- Ve a Herramientas > Placa > Gestor de placas y busca “esp32”. Instala el paquete correspondiente.

A.2.2. Cargar el programa en el ESP32

- Conecta el ESP32 a tu computadora mediante un cable USB.
- Abre el programa a instalar en el IDE de Arduino.

- Ve a Herramientas > Placa y selecciona tu modelo de ESP32.
- Ve a Herramientas > Puerto y selecciona el puerto COM correspondiente.

A.2.3. Instalación de librerías

El programa Arduino para el ESP32 requiere de una serie de librerías. Su instalación se realiza mediante los siguientes pasos:

- Ve a Programas > Incluir librerías > Administrar Bibliotecas.
- En el cuadro de búsqueda, ingresa el nombre de cada librería y haz clic en Instalar.

Configuración	Librerías	Uso
StackM5 CORE 2	M5Core2.h	Configuración de periféricos integrados en StackM5 CORE 2
	Wire.h	Configuración del protocolo de comunicación I2C
Baliza BLE	BLEDevice.h	Funcionalidad general de dispositivos BLE
	BLEUtils.h	Funciones auxiliares como conversión de datos
	BLEScan.h	Escaneo de dispositivos BLE cercanos
	BLEAdvertisedDevice.h	Maneja dispositivos BLE encontrados durante escaneo
	BLEBeacon.h	Creación y manejo de beacons BLE
	BLEEddystoneURL.h	Gestiona el protocolo Eddystone-URL
	BLEEddystoneTLM.h	Gestiona el protocolo Eddystone-TLM
Memoria externa	SD.h	Manejo de tarjetas SD
	SPIFFS.h	Manejo sistema de archivos SPIFFS
Servidor WiFi	WiFi.h	Gestiona la conectividad WiFi
	AsyncTCP.h	Comunicación TCP asíncrona
	ESPAsyncWebServer.h	Creación de servidor web asíncrono
Implementación DNS	WiFiAP.h	Configuración de dispositivos como AP (Access Point)
	ESPmDNS.h	Resolución de nombres de host en una red local (DNS)
Configuración de hora con servidor NTP	NTPClient.h	Maneja sincronización de tiempo utilizando protocolo NTP
	WiFiUdp.h	Proporciona funciones para la comunicación UDP
Modo Light Sleep	esp_sleep.h	Configuración y manejo de modos de ahorro de energía
Magnetómetro	QMC5883LCompass.h	Facilita funciones para configuración y manejo del magnetómetro

Tabla 10. Librerías Arduino

En caso de utilizar otros componentes, el programa requerirá de cambios en código y en las librerías utilizadas.

A.3 Calibración

El magnetómetro ha de calibrarse, pues su funcionamiento depende de la ubicación terrestre donde se implemente el sistema. Para ello se dispone de un programa Arduino pensado para este propósito.

Esta calibración requiere unos conocimientos previos en el campo de la programación. En caso de no tener esos conocimientos el proceso puede parecer complicado. Se aconseja seguir los siguientes pasos:

- Cargar el programa de calibración “calibrationSketch.ino” en el Arduino.
- Visitar https://www.magnetic-declination.com/ para determinar la declinación magnética en tu ubicación.
- Dividir los minutos entre 60 y cambiar el valor de la variable “magneticDeclination”.
- Iniciar el programa y mover la brújula en todos los ángulos para determinar los valores de offset máximos y mínimos del campo magnético en los ejes X e Y.
- Copiar la línea resultado de ejecutar el programa. Esta línea tiene la siguiente configuración: “compass.setCalibration(-15823, -8646, -9588, -5728, -4631, 973);”
- Cargar el programa “CalibrationCardinalSet.ino” y copiar la línea anterior.
- Modificar los valores de Heading, X e Y en el programa hasta que el magnetómetro sea capaz de identificar los cuatro puntos cardinales y guardar.
- Por último, cambia esta configuración por la del programa “ESP32_location_program.ino”. Para ello abre el fichero functions.ino y en la función “String estimateDirection()” modifica los valores para igualar a los del programa “CalibrationCardinalSet.ino” modificado en el paso anterior.

A.4 Uso del Programa Principal del ESP32

A.4.1. Configuración de red WiFi

Para que el ESP32 pueda conectarse a una red WiFi, es necesario crear un archivo ‘config.txt’ en la tarjeta SD. Este archivo debe contener las credenciales de la red WiFi en el siguiente formato:

```
SSID: tu_ssid
Password: tu_password
```

Este archivo debe terminar con una línea en blanco y debe guardarse en la raíz de la tarjeta SD.

A.4.2. Funcionamiento del Programa

El programa ESP32 utiliza tres pulsadores (izquierdo, central y derecho) para gestionar la creación de archivos CSV y la recolección de datos. A continuación, se describe su funcionamiento:

- **Pulsador izquierdo**

Sin archivo CSV abierto: Crea un nuevo archivo y lo abre.

Con archivo CSV abierto: Cierra el archivo abierto.

- **Pulsador central**

Con archivo CSV abierto: Aumenta en 1 la posición a mapear, registrando así el cambio de posición en el área.

- **Pulsador derecho**

Con archivo CSV abierto**:** Recoge medidas y las guarda en el archivo CSV.

Sin archivo CSV abierto**:** Inicia modo automático, que crea archivos y recoge medidas de forma autónoma.

A.5 Generación del mapa

A.5.1. Posicionamiento de las Balizas BLE

Para garantizar una cobertura adecuada del área donde se desea instalar el sistema de posicionamiento, se recomienda colocar las balizas BLE de la siguiente manera:

- Distancia: Cada baliza BLE debe estar a una distancia máxima de 8 metros de las demás, formando cuadrados de 64 metros cuadrados.
- Configuración: Configura las balizas para que emitan con la máxima potencia y un intervalo de advertising de 300 ms.

A.5.2. Mapeo del área

- Con el ESP32 configurado, recorre el área que desea mapear.
- En cada punto a mapear, recoge señales en cuatro direcciones diferentes.
- Las señales recogidas se almacenarán en un archivo CSV en la tarjeta SD del ESP32.

Al finalizar el mapeo, tendrás un conjunto de datos en un archivo CSV que representan las señales recogidas en cada posición del área. Una vez creado, descarga el archivo y asigna a cada posición del mapa un valor de X y un valor de Y. Estas casillas corresponden a als coordenadas de cada posición mapeada en tu sistema de posicionamiento.

A.6 Uso del Cliente de Escritorio

El cliente de escritorio viene preinstalado y basta con descargar y ejecutar el archivo llamado “Beacon Position App.exe” dentro de la carpeta ‘dist’.

La interfaz del cliente de escritorio y su funcionamiento se presenta en la Figura 40.

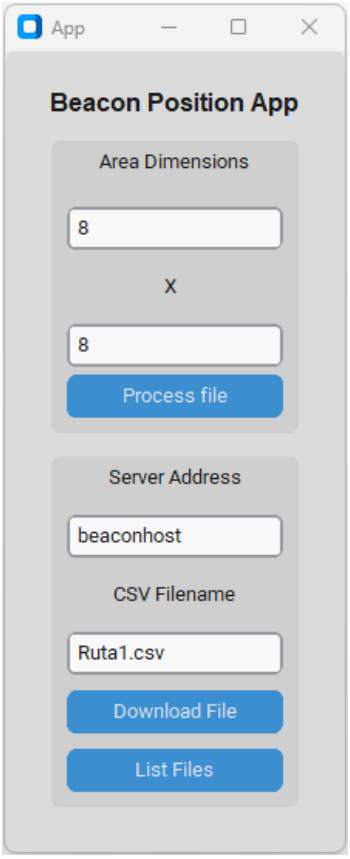

Figura 40. Cliente escritorio

Area Dimensions

Permite ingresar las dimensiones del mapa a crear mediante el botón Process File. En el cuadro de texto superior se ingresa el ancho y en el cuadro inferior se ingresa el alto.

Process File

Permite cargar y procesar archivos CSV recogidos por el ESP32. Este botón requiere primero de la selección de un mapa creado con anterioridad y posteriormente de un archivo que contenga la ruta a estimar.

Enter Server Address

Permite ingresar el nombre del servidor o dirección IP del ESP32 para la comunicación.

Enter Filename

Permite ingresar el nombre del archivo que se desea descargar.

Download File

Descarga un archive específico del ESP32 según el nombre proporcionado.

List Files

Muestra una lista de archivos disponibles en la memoria SD del ESP32.

Anexo

A.1 Procesado RSSI

A.1.1. Posicionamiento Horizontal / Vertical

Distancia	Horizontal				Vertical			
	Max	Min	Media	Desviación estándar	Max	Min	Media	Desviación estándar
1	-65.7612	-81.4049	-72.2722	4.3087	-55.2318	-62.0000	-57.4726	1.6543
2	-72.2564	-84.4808	-76.2748	3.9334	-58.1469	-63.0000	-60.2670	1.4250
3	-76.2220	-88.0778	-81.3607	4.3139	-61.2406	-64.1636	-62.2747	0.7072
4	-76.7676	-85.0854	-79.9527	2.0548	-69.0106	-73.6041	-70.1149	1.0154
5	-76.0000	-81.1130	-78.9815	1.1459	-66.6150	-70.9088	-69.0547	1.0044
6	-85.0000	-91.9420	-87.7918	1.8314	-69.6078	-75.0000	-71.5800	1.4601
7	-79.1069	-87.0925	-83.4709	2.0309	-72.0579	-79.4198	-75.2442	2.6135
8	-85.0000	-95.5030	-89.9032	2.7275	-72.0000	-77.7939	-74.6630	1.5790
9	-82.1389	-94.0000	-85.2009	3.3069	-72.0809	-78.3068	-74.3276	1.4259
10	-79.1425	-83.0000	-80.4300	0.8034	-76.3486	-82.0000	-78.8708	1.4317

Tabla 11. Comparación RSSI horizontal/vertical baliza 1

Distancia	Horizontal				Vertical			
	Max	Min	Media	Desviación estándar	Max	Min	Media	Desviación estándar
1	-66.5701	-76.0000	-70.4826	2.3608	-62.0127	-67.9563	-64.5547	1.9443
2	-76.0000	-90.8463	-81.9602	2.9070	-66.0000	-76.7003	-72.8080	1.9549
3	-78.3250	-93.8173	-85.3951	4.7365	-72.8471	-90.6387	-78.5686	4.5208
4	-78.6050	-83.0000	-80.0865	1.0653	-76.2907	-83.4899	-78.8764	2.0064
5	-79.0000	-84.6374	-81.6593	1.5825	-74.0669	-79.3242	-77.0960	1.5134
6	-83.0743	-93.6527	-86.4006	2.8286	-75.0000	-87.1744	-80.0939	2.9365
7	-81.8437	-98.5519	-89.1303	4.6087	-73.5426	-81.0000	-75.9457	1.6192
8	-81.3212	-85.1253	-83.0642	1.0534	-79.3246	-90.2076	-83.3205	2.8438
9	-86.9788	-94.2736	-89.6028	1.8747	-79.4384	-96.8931	-87.0804	5.4764
10	-87.3344	-94.0000	-89.5905	1.4714	-79.4553	-89.9340	-82.8265	2.6370

Tabla 12. Comparación RSSI horizontal/vertical baliza 2

Distancia	Horizontal				Vertical			
	Max	Min	Media	Desviación estándar	Max	Min	Media	Desviación estándar
1	-69.8044	-83.3962	-74.9863	3.8448	-61.2011	-68.0000	-63.0397	1.5271
2	-72.7255	-80.0733	-75.5881	2.1097	-64.1267	-69.0000	-67.0915	1.4267
3	-81.9107	-93.4458	-86.3573	3.1852	-69.6342	-72.9730	-70.9805	1.0378
4	-79.5496	-92.4375	-84.2346	3.6718	-74.5055	-81.0000	-76.3710	1.2149
5	-79.0000	-85.0517	-83.0768	1.4880	-73.9913	-79.3838	-76.1322	1.7473
6	-85.2341	-94.5969	-91.6488	2.3923	-76.9663	-83.3377	-80.3609	1.5532
7	-82.7041	-88.7345	-85.4297	1.5640	-78.4434	-85.1302	-81.4809	2.4635
8	-80.6053	-95.5679	-87.8575	4.8279	-78.8424	-86.0000	-82.1408	1.9656
9	-87.0000	-95.4514	-90.7259	2.2057	-81.1184	-90.5416	-83.7093	2.3697
10	-84.8628	-88.0248	-86.1792	0.9159	-84.2335	-93.2778	-87.8406	2.2570

Tabla 13. Comparación RSSI horizontal/vertical baliza 3

Distancia	Horizontal				Vertical			
	Max	Min	Media	Desviación estándar	Max	Min	Media	Desviación estándar
1	-69.1927	-76.9245	-73.0630	2.4350	-54.5307	-63.3885	-56.9618	2.1612
2	-74.7139	-87.8277	-79.1999	3.3860	-58.0702	-63.0000	-59.5989	1.3616
3	-75.5322	-87.1243	-80.4296	3.8202	-59.0000	-63.7283	-61.9285	0.9244
4	-78.8547	-88.4390	-81.7100	2.4516	-67.0288	-70.8388	-68.8193	0.9907
5	-75.2722	-78.9664	-77.0411	1.0850	-66.5183	-72.0000	-68.1219	1.6884
6	-84.0000	-95.7423	-88.5399	3.2557	-66.4127	-68.9113	-67.6990	0.5609
7	-80.0000	-96.4213	-88.1827	4.3330	-68.0259	-74.0000	-69.0703	1.1063
8	-81.4251	-88.2418	-85.0890	1.9802	-70.7597	-76.7547	-73.6444	1.9156
9	-83.9907	-97.0000	-87.5825	3.6346	-71.1089	-95.8996	-74.6898	4.6412
10	-85.9156	-94.7470	-90.1903	2.2466	-75.3568	-79.5647	-77.3496	1.1049

Tabla 14. Comparación RSSI horizontal/vertical baliza 4

A.1.2. Comportamiento balizas en diferentes ambientes

Distancia	Con Ruido		Sin Ruido	
	Distancia calculada	Error	Distancia calculada	Error
1	1.02548742	0.02548742	0.98812613	0.01187387
2	2.11459235	0.11459235	3.1038788	1.1038788
3	6.70562012	3.70562012	8.60898875	5.60898875
4	4.86144964	0.86144964	9.94487832	5.94487832
5	7.84473099	2.84473099	5.03889418	0.03889418
6	16.1081569	10.1081569	7.97801164	1.97801164
7	19.5196506	12.5196506	12.4367589	5.43675893
8	7.96534755	0.03465245	9.66732315	1.66732315
9	10.5329044	1.53290444	6.66717281	2.33282719
10	15.3934915	5.39349149	11.4160827	1.41608271

Tabla 15. Baliza 1

Distancia	Con Ruido		Sin Ruido	
	Distancia calculada	Error	Distancia calculada	Error
1	1.01274753	0.01274753	1.00303331	0.00303331
2	1.32592193	0.67407807	2.43871077	0.43871077
3	3.67498824	0.67498824	4.28228856	1.28228856
4	2.76744887	1.23255113	5.04282654	1.04282654
5	3.21393715	1.78606285	5.9577786	0.9577786
6	3.50091729	2.49908271	6.94272948	0.94272948
7	10.4975819	3.49758193	7.49645718	0.49645718
8	4.8800162	3.1199838	9.87618792	1.87618792
9	3.09234843	5.90765157	9.53858031	0.53858031
10	4.07274644	5.92725356	12.9754401	2.97544005

Tabla 16. Baliza 2

Distancia	Con Ruido		Sin Ruido	
	Distancia calculada	Error	Distancia calculada	Error
1	1.0392642	0.0392642	1.04068694	0.04068694
2	1.27079468	0.72920532	2.56457031	0.56457031
3	1.01972656	1.98027344	4.68431631	1.68431631
4	1.20610592	2.79389408	4.39813918	0.39813918
5	1.16551292	3.83448708	9.94248422	4.94248422
6	2.39923062	3.60076938	9.54027445	3.54027445
7	3.84682703	3.15317297	20.2597312	13.2597312
8	3.19072398	4.80927602	11.8015796	3.80157965
9	2.27723391	6.72276609	9.48918663	0.48918663
10	3.46015167	6.53984833	17.0106758	7.01067583

Tabla 17. Baliza 3

Distancia	Con Ruido		Sin Ruido	
	Distancia calculada	Error	Distancia calculada	Error
1	1.07493523	0.07493523	1.00855231	0.00855231
2	0.93853277	1.06146723	3.24523923	1.24523923
3	1.59372059	1.40627941	4.90630771	1.90630771
4	1.80283839	2.19716161	5.3230897	1.3230897
5	1.9480508	3.0519492	7.18965957	2.18965957
6	4.64782365	1.35217635	7.1738833	1.1738833
7	13.9541178	6.95411778	6.58116535	0.41883465
8	2.796189	5.203811	9.50795231	1.50795231
9	2.6005178	6.3994822	9.96859977	0.96859977
10	2.25783026	7.74216974	10.1431774	0.14317743

Tabla 18. Baliza 4

A.1.3. Baliza BLE v5.4

Distancia	RSSI				Distancia	
	Max	Min	Media	Desviación estándar	Distancia calculada	Error
1	-64.19839439	-77.53007836	-67.47877176	3.902999065	0.920111745	0.079888255
2	-69.50571347	-82.99672539	-76.09005033	4.938483219	2.95191174	0.95191174
3	-74.4967697	-85.30367759	-79.88323254	3.537404041	4.344821226	1.344821226
4	-71.09446763	-89.08759774	-79.84068962	6.459911746	4.799253036	0.799253036
5	-77.58025318	-84.44262686	-80.59645592	2.549189378	4.594597741	0.405402259
6	-83.08398826	-90.30348185	-85.97610483	2.418908305	8.395972338	2.395972338
7	-82.49108655	-88.71841737	-86.09301874	2.044951535	8.750544665	1.750544665
8	-79.71593834	-93.8022746	-85.16315899	3.686291272	7.39677262	0.60322738
9	-87.19481819	-94	-91.28328504	2.335850462	16.11428007	7.114280072
10	-91	-97.85180125	-94.49300383	2.478435025	22.97174076	12.97174076

Tabla 19. Comportamiento Baliza 5

A.2 Resultados método fingerprinting

En este apartado del Anexo se presentan las tablas con las coordenadas específicas de las rutas obtenidas como resultado de probar el método "fingerprinting" para estimar la posición. A continuación, se presentan las Tablas 20 y 21.

	Con magnetómetro		Sin magnetómetro	
Punto	Posición	Dirección	Posicion	Direccion
1	(1.0, 0.0)	South	(1.0, 0.0)	None
2	(3.0, 0.0)	South	(6.0, 0.0)	None
3	(5.0, 1.0)	South	(7.0, 0.0)	None
4	(7.0, 2.0)	East	(8.0, 1.0)	None
5	(8.0, 2.0)	East	(8.0, 3.0)	None
6	(8.0, 7.0)	East	(8.0, 6.0)	None
7	(7.0, 8.0)	North	(7.0, 8.0)	None
8	(4.0, 7.0)	North	(1.0, 4.0)	None
9	(1.0, 8.0)	North	(1.0, 7.0)	None
10	(0.0, 7.0)	West	(1.0, 8.0)	None
11	(1.0, 4.0)	West	(1.0, 8.0)	None
12	(0.0, 1.0)	West	(0.0, 1.0)	None

Tabla 20. Comparativa resultados Ruta 1 fingerprinting

	Con magnetómetro		Sin magnetómetro	
Punto	Posición	Dirección	Posicion	Dirección
1	(1.0, 0.0)	East	(2.0, 2.0)	None
2	(1.0, 1.0)	East	(4.0, 0.0)	None
3	(2.0, 2.0)	East	(2.0, 3.0)	None
4	(1.0, 3.0)	East	(1.0, 4.0)	None
5	(1.0, 4.0)	East	(1.0, 4.0)	None
6	(1.0, 5.0)	East	(1.0, 5.0)	None
7	(1.0, 6.0)	East	(1.0, 6.0)	None
8	(1.0, 7.0)	East	(1.0, 6.0)	None
9	(0.0, 7.0)	East	(0.0, 7.0)	None

Tabla 21. Comparativa resultados Ruta 2 fingerprinting

En rojo están indicadas las posiciones estimadas consideradas erróneas que entorpecen el visualizado de la ruta seguida por el dispositivo.

A.3 Resultados de simulación en entorno real

Tabla 22 donde se representan los valores numéricos de coordenadas obtenidos para la Ruta 1 de la simulación en entorno real.

Sin collar			Con collar		
Punto	Posición	Dirección	Punto	Posición	Dirección
1	(0.0, 0.0)	South	88	(0.0, 1.0)	North
2	(0.0, 0.0)	South	89	(0.0, 3.0)	South
3	(1.0, 1.0)	West	90	(3.0, 3.0)	East
4	(2.0, 1.0)	South	91	(7.0, 0.0)	South
5	(5.0, 0.0)	South	92	(7.0, 0.0)	East
6	(6.0, 0.0)	South	93	(5.0, 2.0)	North
7	(6.0, 0.0)	South	94	(13.0, 0.0)	East
8	(8.0, 2.0)	South	95	(14.0, 0.0)	West
9	(12.0, 0.0)	South	96	(13.0, 0.0)	East
10	(12.0, 4.0)	South	97	(10.0, 3.0)	Sur
11	(14.0, 0.0)	South	98	(14.0, 4.0)	East
12	(13.0, 1.0)	East	99	(12.0, 2.0)	South
13	(15.0, 5.0)	East	100	(13.0, 3.0)	West
14	(14.0, 5.0)	North	101	(9.0, 4.0)	West
15	(9.0, 4.0)	West	102	(13.0, 4.0)	South
16	(13.0, 5.0)	North	103	(12.0, 2.0)	East
17	(8.0, 5.0)	North	104	(0.0, 5.0)	East
18	(7.0, 5.0)	North	105	(0.0, 2.0)	North
19	(6.0, 1.0)	North	106	(0.0, 0.0)	West
20	(10.0, 3.0)	West			
21	(1.0, 5.0)	North			
22	(4.0, 1.0)	East			
23	(2.0, 4.0)	West			
24	(3.0, 1.0)	North			
25	(3.0, 1.0)	West			

Tabla 22. Ruta 1 simulación entorno real

Para la Ruta 1 sin collar se obtiene un error de medición del 20%.

Para la Ruta 1 con collar se obtiene un error de medición del 31.58% aproximadamente.

Tabla 23 donde se representan los valores numéricos obtenidos durante la Ruta 2 de la simulación en entorno real

Sin collar			Con collar		
Punto	Posición	Dirección	Punto	Posición	Dirección
26	(1.0, 1.0)	South	107	(0.0, 1.0)	South
27	(2.0, 1.0)	East	108	(1.0, 1.0)	South
28	(2.0, 1.0)	South	109	(5.0, 5.0)	West
29	(8.0, 5.0)	East	110	(6.0, 1.0)	East
30	(9.0, 1.0)	South	111	(9.0, 1.0)	East
31	(12.0, 1.0)	West	112	(8.0, 5.0)	West
32	(10.0, 5.0)	East	113	(5.0, 5.0)	East
33	(14.0, 1.0)	South	114	(5.0, 5.0)	West
34	(15.0, 1.0)	West	115	(12.0, 2.0)	West
35	(14.0, 0.0)	South	116	(14.0, 0.0)	West
36	(14.0, 1.0)	West	117	(14.0, 2.0)	South
37	(12.0, 5.0)	South	118	(15.0, 5.0)	West
38	(14.0, 5.0)	North	119	(15.0, 5.0)	East
39	(14.0, 5.0)	North	120	(13.0, 4.0)	East
40	(15.0, 1.0)	North	121	(12.0, 1.0)	North
41	(9.0, 1.0)	North	122	(5.0, 0.0)	East
42	(8.0, 1.0)	North	123	(9.0, 0.0)	North
43	(2.0, 5.0)	West	124	(9.0, 0.0)	North
44	(7.0, 5.0)	East	125	(11.0, 2.0)	North
45	(2.0, 0.0)	East	126	(5.0, 2.0)	South
46	(0.0, 5.0)	West	127	(2.0, 4.0)	East
47	(0.0, 5.0)	East	128	(1.0, 4.0)	North
48	(3.0, 4.0)	West	129	(0.0, 0.0)	South
49	(1.0, 1.0)	West	130	(1.0, 0.0)	South
50	(2.0, 0.0)	East			

Tabla 23. Ruta 2 simulación entorno real

Para la Ruta 2 sin collar se obtiene un error de medición del 16%.

Para la Ruta 2 con collar se obtiene un error de medición del 17.24%.