uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Electrónica Industrial y Automática
Curso 2018-2019

Trabajo Fin de Grado

"Diseño de un sistema de localización en interiores basado en BLE"

Diego Ventura Henríquez

Tutor

Luis Alfonso Entrena Arrontes

Leganés, 17 de junio de 2019

RESUMEN

En este documento se recogen los resultados de un estudio cuyo objetivo principal es analizar la viabilidad de un sistema de localización en interiores basado en comunicaciones inalámbricas (concretamente Bluetooth). El análisis se centra en conocer en profundidad los pros y contras de emplear este tipo de tecnología para tal fin.

En primer lugar, se estudiarán los entresijos del protocolo Bluetooth Smart/Bluetooth Low Energy y los sistemas de localización en interiores, exponiéndose a su vez las razones principales por las que se emplean hoy en día y los diferentes proyectos basados en ellos llevados a cabo en los últimos años.

Posteriormente, se diseñará un sistema de localización simple basado en este protocolo. Este consta de dos partes principales: un emisor (encargado de enviar una señal con información aleatoria) y un receptor (encargado de reconocer al emisor y medir la potencia con la que recibe su señal). El hardware de ambos componentes del sistema consiste en dos placas de desarrollo idénticas que incluyen, entre otros, un módulo bluetooth y un microcontrolador ARM de 32 bits.

Tras explicar los detalles del diseño del sistema, (placas de desarrollo empleadas, software empleado para la programación de estas y características de las señales Bluetooth con las que se trabaja) se describirá el proceso de desarrollo del mismo, haciendo especial hincapié en el funcionamiento de los códigos en C++ del receptor y el emisor y los resultados obtenidos.

Por último, se realizará un análisis de los resultados obtenidos en el apartado anterior. Debido a la inestabilidad de estos (por la fluctuación constante de la potencia de la señal), se llevará a cabo un filtrado estadístico que atenúe el problema. En función de la efectividad de este último y los factores citados al principio del documento se concluirá si el sistema es viable o no.

Palabras clave: Bluetooth, Microcontroladores, C++, ARM, Mbed.

AGRADECIMIENTOS

Este trabajo no es sólo la culminación de cuatro años de estudio, sino de una etapa maravillosa en la que he tenido la suerte de poder crecer enormemente en numerosos ámbitos. Sin embargo, y por suerte, no soy el único responsable de esto. Por ello, me gustaría aprovechar este espacio para expresar mi gratitud a todas aquellas personas que, de una forma u otra, me han acompañado a lo largo del camino.

Mis primeras palabras de agradecimiento van dirigidas a todas aquellas que no solemos recordar. Todos aquellos "olvidados" que, durante generaciones, han luchado para que hoy estas líneas puedan ser escritas. Porque mis compañeros y yo nunca podríamos haber llegado hasta aquí si sus esfuerzos no hubiesen convertido la educación en uno de los motores principales de nuestra sociedad.

En segundo lugar, a mis padres, mi hermano, mi tío y mi familia en general, que siempre han alimentado mis ganas de crecer con cariño, alegría, ánimos, paciencia e inspiración. Porque no hay mejor escuela que el hogar, y siempre deberíamos tenerlo presente.

También quiero recordar a todos mis amigos/as. Algunos, como Bendi, Baju, Rober, Camilo, Rubén, Alba y otros muchos que no alcanzo a nombrar, han vivido conmigo esta travesía y me han permitido crecer con ellos. Otros, como Joni, Guille, Pablo y Lore, han estado siempre ahí pese a la distancia, mostrando su incomparable capacidad de hacer que un océano parezca un simple charco. A todos les une algo: la enorme huella que han dejado en mí con todos los momentos vividos juntos, y que espero que nunca deje de crecer.

No me olvido tampoco de todas las personas que generosamente me han transmitido sus conocimientos en el ámbito de la ingeniería, tanto dentro de la universidad (todo el profesorado con el que he coincidido en estos años) como fuera de ella (Emilio, Paco, Luismi, Oscar y el resto del equipo de Terasalud y el Grupo Arquímea en general).

Por último, y no por ello menos importante, debo agradecer a mi tutor, Luis Entrena, todo lo que me ha aportado, tanto en el ámbito personal como el académico. Sus constantes ánimos, su bondad, su grandeza como profesor y el respeto que despierta en mis compañeros y en mí hacen que sea difícil no considerarle una fuente de inspiración. Espero que muchas generaciones más tengan la suerte de aprender de él.

Por todo esto y lo que no puede expresarse con palabras, gracias a todos y a todas de corazón.

ÍNDICE

ÍNDICE DE IMÁGENES

ÍNDICE DE TABLAS

LISTA DE ABREVIATURAS

AOA	Angle of Arrival
API	Application Programming Interface
ATT	Attribute Protocol
BLE	Bluetooth Low Energy
CLI	Command Line Interface
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GPS	Global Positioning System
ECTS	European Credit Transfer and Accumulation
EMC	Electromagnetic Compatibility
IDE	Integrated Development Environment
IOT	Internet of Things
IPS	Indoor Positioning System
ISM	Industrial Scientific and Medical
L2CAP	Logical Link Control and Adaptation Protocol
LE	Low Energy
LVD	Low Voltage Directive
MIT	Massachusetts Institute of Technology
OS	Operative System
RED	Radio Equipment Directive
RF	Radiofrecuencia
RoHS2	Restriction of Hazardous Substances (recast)
RSS	Received Signal Strength
RSSI	Received Signal Strength Indicator
RTLS	Real Time Location System
RTOS	Real Time Operating System

SDK	Software Development Kit
SLI	Sistema de Localización en Interiores
TFG	Trabajo de Fin de Grado
TDOA	Time Difference of Arrival
TOA	Time of Arrival
UC3M	Universidad Carlos III de Madrid
UWB	Ultra-Wide Band

1. INTRODUCCIÓN

1.1. Motivación

Este proyecto está orientado a estudiar la viabilidad de un sistema de localización basado en tecnología inalámbrica, más concretamente Bluetooth Low Energy (BLE). Existen numerosas razones para escoger este tema, aunque la principal y más obvia es el sinfín de aplicaciones que pueden derivar de los sistemas de localización basados en comunicaciones inalámbricas. Entre ellas podríamos destacar las siguientes:

- Localización de productos en grandes superficies: En naves industriales o edificios de grandes dimensiones puede resultar difícil encontrar ciertos productos/herramientas en momentos determinados. La existencia de un sistema de localización en interiores acaba con este problema y además ahorra tiempo de búsqueda a los usuarios.
- Localización de personas: Localizar personas dentro de un edificio puede resultar muy útil en diversos campos como, por ejemplo, el sector de la salud. Tomando como referencia el caso concreto de las residencias de ancianos, puede verse que dejar de supervisar un momento a personas de edad avanzada puede desencadenar consecuencias negativas para su salud (sedentarismo, caídas, acceso a áreas restringidas, etc.). Esto puede evitarse monitorizando su posición mediante un Sistema de Localización en Interiores (SLI).
- Guiado en interiores: Al igual que puede localizar a una persona, un SLI puede servir para guiarlas hacia un punto concreto de la infraestructura si la situación lo requiere.

Además, emplear el protocolo Bluetooth Low Energy como piedra angular de estos sistemas se ha convertido en una opción ampliamente escogida en los últimos años (debido principalmente al bajo consumo de energía que implica), por lo que resulta de gran utilidad familiarizarse con este tipo de tecnologías.

Del mismo modo, no hay que olvidar que sistemas como este constituyen un ejemplo de red de sensores (como el IOT), por lo que el auge de este tipo de sistemas en nuestra era lo convierten en una inversión de futuro en lo que a formación se refiere. Algunas de las áreas de aplicación más claras se muestran en la Figura 1.

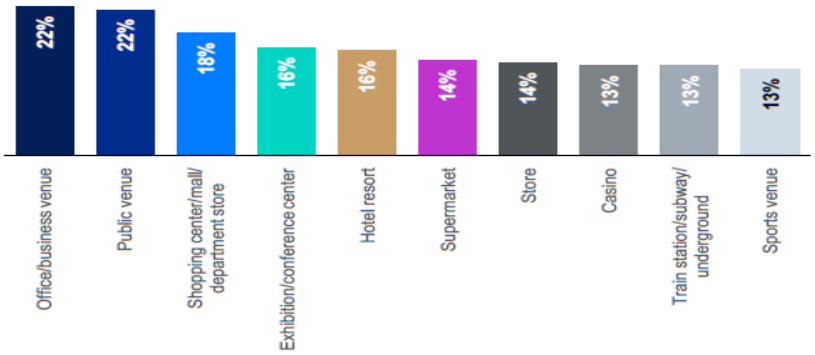


Figura 1: Top 10 futuras áreas de aplicación de los SLI [19]

1.2. Objetivos

El objetivo principal del trabajo es analizar la viabilidad de usar BLE como protocolo de comunicación de un SLI. Sin embargo, para hacerlo es necesario diseñar e implementar un sistema con estas características y analizar su funcionamiento. Por lo tanto, consta de los siguientes objetivos:

- Estudiar el funcionamiento del protocolo BLE.
- Establecer las bases del SLI a implementar (componentes y funciones de los mismos).
- Seleccionar la plataforma y las placas de desarrollo necesarias para implementar el sistema.
- Desarrollar el emisor y el receptor del sistema.
- Medir la señal recibida y calibrar el SLI.
- Llevar a cabo un análisis de viabilidad con los datos obtenidos.

1.3. Estructura del documento

Para facilitar que el lector pueda estudiar las partes de la obra que más le interesen sin perder tiempo, este trabajo sigue una estructura de diferenciación por capítulos, conteniendo cada uno de ellos una etapa distinta del estudio.

Estos son los siguientes:

- **Capítulo 1: Introducción.** Breve introducción al objeto de estudio y las razones que han llevado a escogerlo.
- **Capítulo 2: Estado de la técnica.** Análisis y recopilación de los artículos/proyectos relacionados con el tema de estudio que han surgido en los últimos años o siguen activos en la actualidad.
- **Capítulo 3: Plataforma de desarrollo.** Explicación detallada de la plataforma de desarrollo empleada para elaborar el estudio, es decir, el hardware escogido (placas para el emisor y el receptor), el software empleado para programar el mismo (sistema operativo y lenguaje de programación) y el protocolo de comunicación inalámbrica citado en apartados anteriores (Bluetooth Low Energy).
- **Capítulo 4: Desarrollo del sistema.** Este capítulo representa la parte más voluminosa del estudio, comprendiendo el desarrollo y la implementación de todas las partes del sistema.
- **Capítulo 5: Resultados experimentales.** Resultados iniciales obtenidos tras el desarrollo del sistema y descripción de la batería de pruebas realizadas. Como se podrá ver más adelante, las medidas varían enormemente dependiendo de varios factores, por lo que en este capítulo también se analizan las causas de dicho problema, se proponen soluciones para aumentar la exactitud en la localización y se vuelven a analizar los resultados obtenidos tras las modificaciones realizadas.

- **Capítulo 6: Conclusiones y trabajos futuros.** Recopilación de conclusiones tras completar el estudio, resumen de objetivos cumplidos y exposición de ideas para mejorar el sistema en un futuro.

Así mismo, al final del documento se incluyen los siguientes anexos:

- **Anexo A: Códigos transmisor y receptor.** Desarrollo de los códigos fuente empleados por el emisor y el receptor.
- **Anexo B: Programas MATLAB.** Desarrollo de los diversos programas implementados en MATLAB.
- **Anexo C: Marco regulador.** Recopilación de las normativas aplicables a este trabajo.
- **Anexo D: Entorno socioeconómico.** Análisis del impacto socioeconómico del proyecto y presupuesto necesario para llevar a cabo el proyecto.

2. SISTEMAS DE LOCALIZACIÓN EN INTERIORES

2.1. Introducción a los Sistemas de Localización en Interiores

Pese a que históricamente los sistemas de localización en exteriores han tenido más protagonismo que sus homólogos para interiores, en las últimas décadas estos últimos han ganado mucha importancia. Tanto es así que hoy en día pueden encontrarse una gran variedad de soluciones comerciales que integran RTLS (*Real Time Location Systems*) basados en diferentes metodologías y protocolos de comunicación.

Entre los múltiples métodos de localización en interiores conocidos destacan los siguientes:

- **Métodos basados en comunicaciones inalámbricas:** Son todos aquellos métodos de posicionamiento que emplean protocolos de comunicación inalámbrica como base. Entre ellos destacan los siguientes:
    - Fuerza de la señal recibida (Received Signal Strength (RSS)): Como su propio nombre indica, su principio de funcionamiento consiste en medir la potencia de la señal recibida y, en función de la misma, obtener la distancia a la que se encuentra el emisor (mediante una calibración previa o empleando la fórmula de Frii [1]). En la siguiente figura se puede observar una representación gráfica del método.

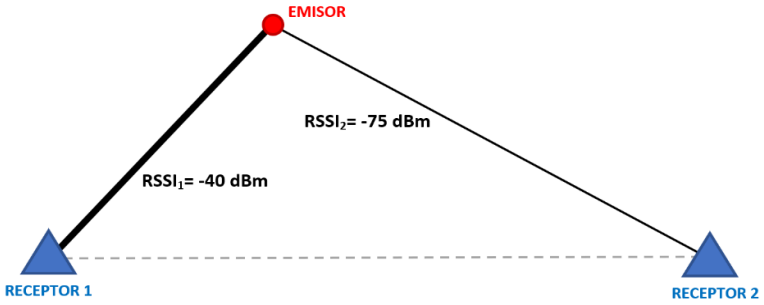


Figura 2: Localización por RSS

    - Ángulo de llegada (Angle of Arrival (AOA)): Se basa en medir el ángulo con el que llega la señal generada por el emisor al receptor (como se muestra en la figura 3).

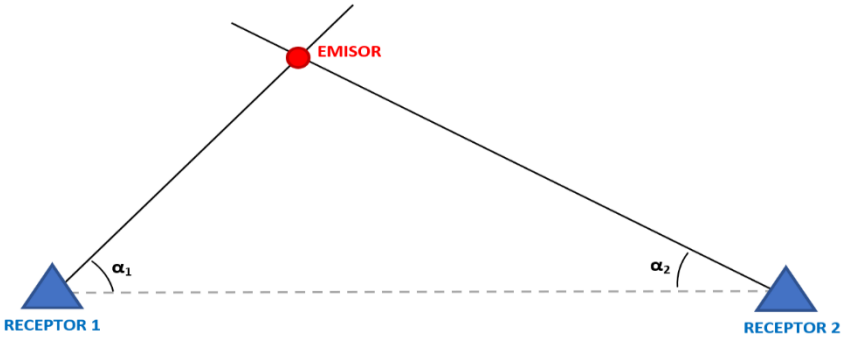


Figura 3: Localización por AOA

- Tiempo de llegada (Time of Arrival (TOA)): Se basa en medir el tiempo que tarda en llegar la señal generada por el emisor hasta el receptor (a grandes rasgos, esto equivale a medir el rango del emisor respecto al receptor, como se aprecia en la figura 4). Suele usarse más en sistemas de localización en exteriores, siendo el método empleado por los GPS (Global Positioning System).

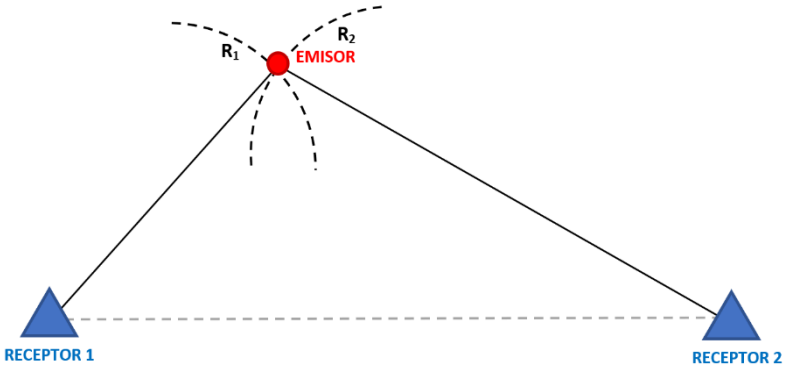


Figura 4: Localización por TOA

- Diferencia de tiempo de llegada (Time Difference of Arrival (TDOA)): Posee el mismo principio de funcionamiento que el TOA, sólo que en este caso se obtiene la distancia calculando la diferencia de los tiempos de llegada de las señales.
- *Fingerprinting*: Empleando el método RSS, se toman diversas medidas en un recinto y se almacenan en una base de datos. Posteriormente se accede a dicha base de datos para determinar la posición del emisor.

- **Métodos alternativos:** Pese a que los métodos basados en comunicaciones inalámbricas son los más empleados en la actualidad, existen otras opciones válidas para implementar un SLI. Entre ellas resultan de especial interés las siguientes:
  - Posicionamiento magnético: Generando un campo magnético y midiendo variaciones en el mismo ocasionadas por el elemento cuya posición se desea obtener se puede generar un sistema de localización efectivo.
  - Marcadores visuales: Situando marcadores visuales en posiciones estratégicas de un recinto (referencias del sistema) se puede detectar la posición de otro objeto introducido en el mismo.
  - Medidas inerciales: Otra forma posible de estimar la posición de un objeto es medir la aceleración con la que se mueve y cuánto tiempo dura dicho movimiento.
  - Ultrasonidos: La detección de objetos empleando ultrasonidos es una técnica clásica. Este método se basa en el mismo principio de funcionamiento.

Del mismo modo, los sistemas basados en comunicaciones inalámbricas pueden clasificarse según el protocolo empleado. Los más comunes son los siguientes:

- Radiofrecuencia.
- Wi-fi.
- Bluetooth.
- ZigBee.
- RFID.
- *Ultra-Wide Band* (UWB).

Todos y cada uno de ellos tienen una serie de ventajas e inconvenientes que los hacen más o menos convenientes dependiendo de la aplicación que se le quiera dar al SLI y las características que tenga (rango, consumo máximo, coste, etc.). En la siguiente gráfica se puede observar una comparativa en cuanto a exactitud y uso de los protocolos más empleados:

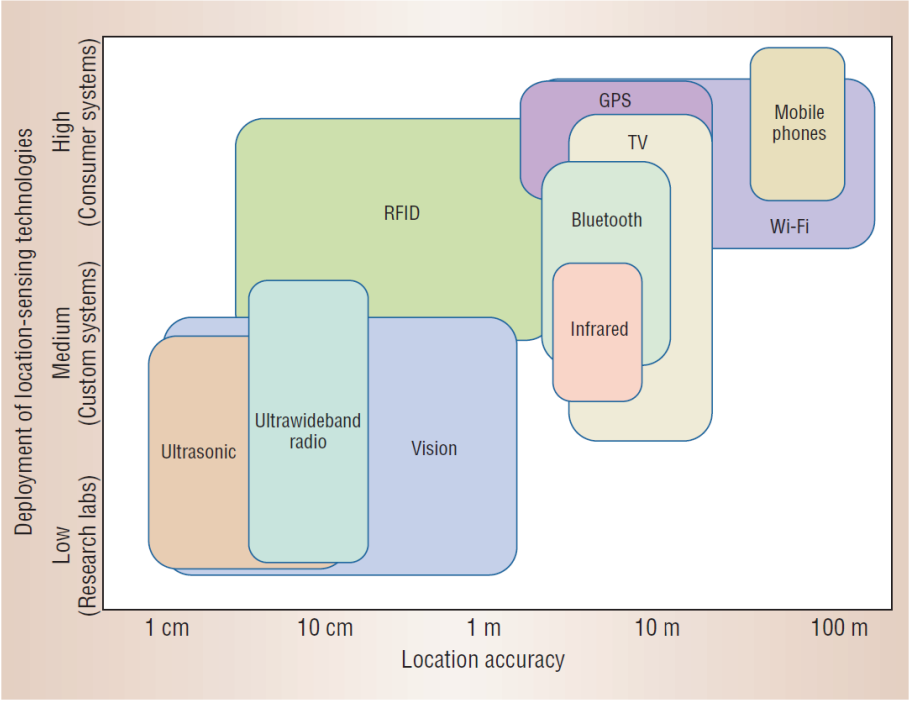


Figura 5: Comparativa protocolos inalámbricos [27]

Como puede verse, el Bluetooth (protocolo empleado en este trabajo) goza de una posición medianamente buena (entre 5 y 10 metros) y puede llegar a usarse en SLI comerciales. En el capítulo siguiente se analizarán en profundidad las características de este protocolo.

2.2. Proyectos previos

En las últimas décadas se han desarrollado un sinfín de proyectos comerciales y de investigación con estas características. Basándose en diferentes métodos de medida y protocolos de comunicación, los SLI son empleados en cada vez más ámbitos, aunque algunos proyectos han tenido más éxito que otros. Citarlos todos es casi imposible y no

tiene especial relevancia para explicar el desarrollo de este trabajo, pero sí existen algunos que merecen ser mencionados:

- Sonitor RTLS*:* Como se mencionó en el apartado anterior, existen diversas alternativas a las comunicaciones inalámbricas. Al usar ultrasonidos como tecnología base y sensores de movimiento como tecnología secundaria, Sonitor es un ejemplo comercial claro de esto. A grandes rasgos, como se puede observar en la figura 6, el sistema consta de tres tipos de componentes: *tags* (emisores de ultrasonido equipados con detectores de movimiento; únicamente envían señales al percibir una aceleración), detectores (micrófonos encargados de leer la señal emitida por los tags) y servidores de localización (elementos fijos que se conectan a los detectores por Ethernet y emplean la información recibida y calculan la posición de los tags). La versión IPS de este sistema también emplea el método TDOA, cubriendo un rango de 18m por detector con una exactitud de 2-3 cm.

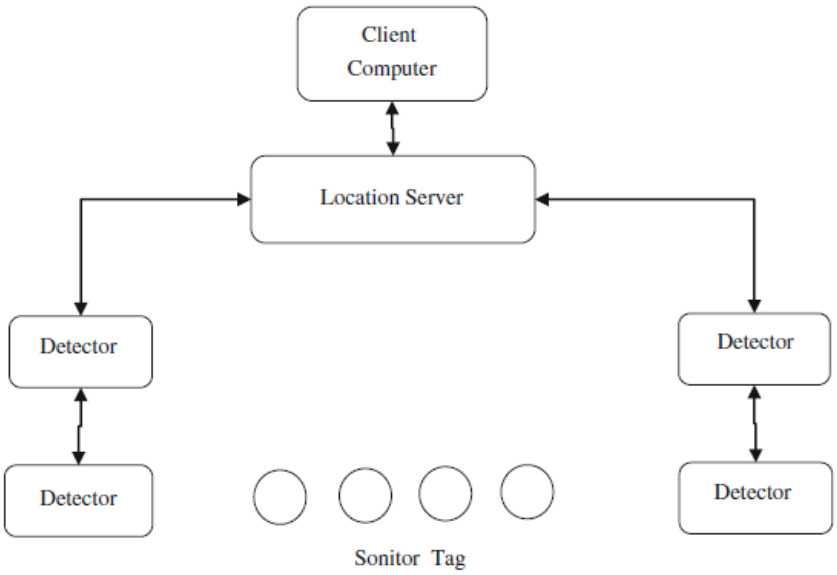


Figura 6: Esquema Sonitor RTLS [18]

- Aeroscout: Mientras Sonitor emplea los ultrasonidos como tecnología base, el SLI de Aeroscout, uno de sus principales competidores, se basa en el protocolo WiFi (aunque también emplea el RFID para activar los tags). Este consta de cuatro componentes principales: *tags* (emisores activos que producen señales WiFi continuamente), puntos de acceso WiFi., un motor de localización y, aunque no se trate de un componente físico, un software de gestión denominado MobileView. Para obtener la posición de los tags, el sistema emplea los métodos TDOA y RSS. En la figura 7 se puede observar un esquema de su arquitectura.

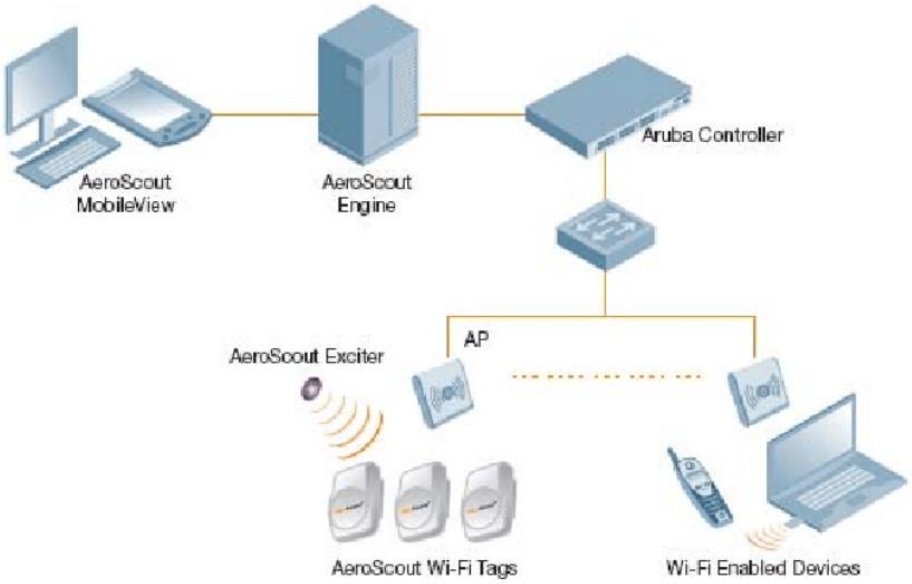


Figura 7: Esquema Aeroscout RTLS [18]

- Cricket: Este SLI "open source" desarrollado por el MIT se basa parcialmente en comunicaciones inalámbricas, pero se diferencia de otros sistemas de este tipo en que usa dos tecnologías base: Radiofrecuencia y ultrasonidos. La combinación de ambas permite obtener medidas con una exactitud de entre 1cm y 3cm, y el hardware empleado comprende dos tipos de dispositivos: *beacons* (emisores de radiofrecuencia y ultrasonidos que adoptan una posición fija estratégica, normalmente en paredes o techos (como se puede ver en la figura 9)) y *listeners* (receptores pasivos que leen la señal emitida con ultrasonidos únicamente si anteriormente han detectado la señal de radiofrecuencia esperada; posteriormente obtiene su posición con un algoritmo que procesa la información obtenida por ambas vías). Otra particularidad de *Cricket* es que es un sistema descentralizado (carece de un centro de operaciones / base de datos), por lo que es más fácil de implementar.

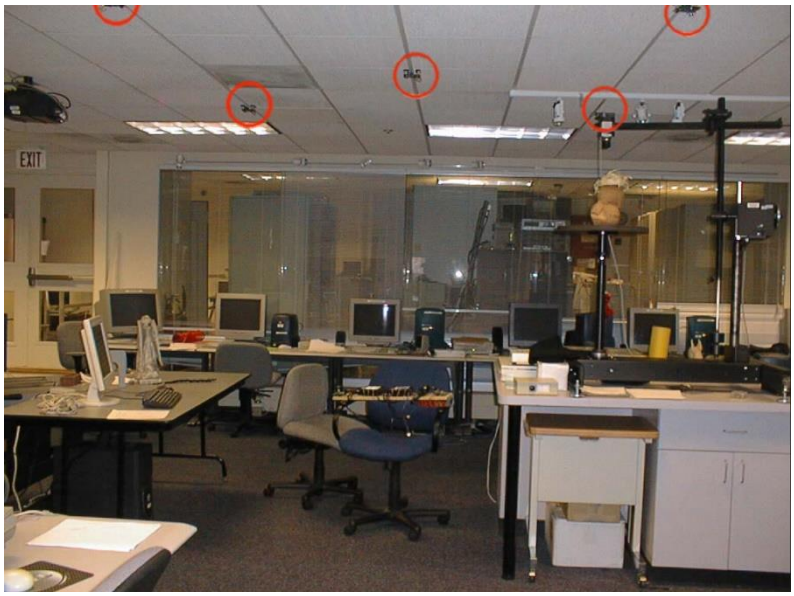


Figura 9: Implementación de listeners Cricket [20]

- *Awarepoint*: El punto fuerte de Awarepoint es su robustez, ventaja asociada al hecho de que emplea ZigBee como tecnología base. Su arquitectura es similar a la del SLI de Aeroscout, (como se aprecia en la figura 8) con la diferencia de que consta de cinco componentes: *Tags*, sensores, puentes, *appliances* y, aunque no se trate de hardware, el software de Awarepoint. Su exactitud a la hora de realizar medidas es de aproximadamente 1,5 m.

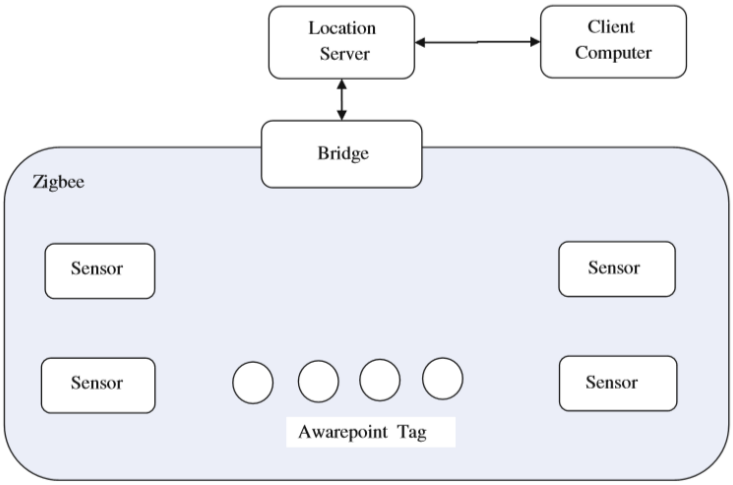


Figura 8: Esquema Awarepoint RTLS [18]

2.3. Estudios e investigaciones actuales

Del mismo modo, en los últimos años se han llevado a cabo diversas investigaciones acerca de los sistemas de localización en interiores. Al igual que ocurre en el ámbito comercial, las comunicaciones inalámbricas suelen ser el foco de atención, y aunque existen numerosos análisis enfocados hacia métodos como el TDOA o el AOA, el RSS goza de una especial popularidad en esta área.

Por supuesto, la cantidad de artículos de investigación de estas características es demasiado elevada como para citarlos todos, por lo que en este apartado se hará hincapié exclusivamente en aquellos que traten sobre SLI basados en BLE y RSS. Destacan los siguientes:

- *Accuracy analysis of the RSSI BLE SensorTag signal for indoor localization purposes* [2]
- *A BLE RSSI Ranking based Indoor Positioning System for Generic Smartphones* [3]
- *BLE Localization using RSSI Measurements and iRingLA* [4]
- *A Hybrid Method to Improve the BLE-Based Indoor Positioning in a Dense Bluetooth Environment* [5]
- *Blueps: sistema de localización en interiores utilizando Bluetooth* [6]

En concreto, el primero de ellos resulta de gran relevancia como apoyo a este trabajo, puesto que se centra en analizar la exactitud de un SLI de características muy similares al de este documento. En capítulos posteriores se explicará con más detalle el contenido de este.

3. PLATAFORMA DE DESARROLLO

3.1. Placa de desarrollo

La placa de desarrollo seleccionada para implementar tanto el emisor como el receptor del sistema es la "B-L475E-IOT01A2 Discovery Kit" de ST Microelectronics (Figura 10), un kit de desarrollo con gran capacidad de procesamiento y diversos sensores y protocolos de comunicación ideado para emplearse en proyectos de IOT y derivados.

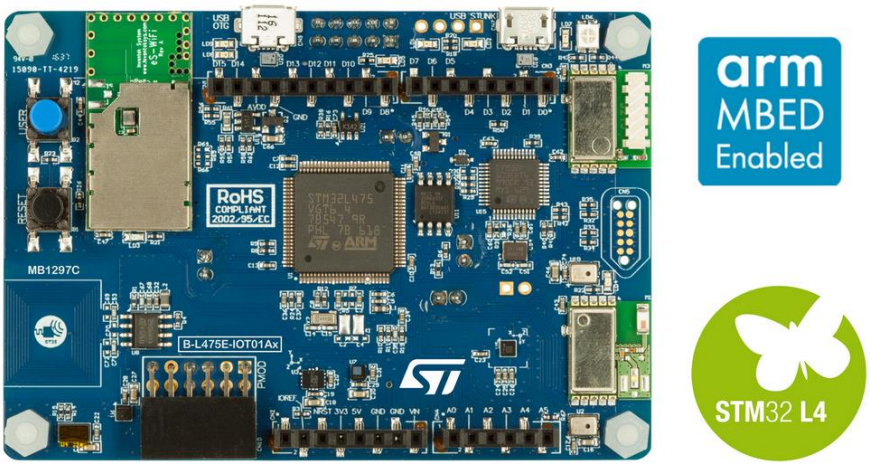


Figura 10: B-L475E-IOT01A [7]

Sus características principales, extraídas de la página oficial de ST Microelectronics [7], son las siguientes:

- MCUs de muy bajo consumo (serie STM32L4) basados en el núcleo Arm® Cortex®-M4 y con 100MB de memoria Flash y 128KB de memoria SRAM.
- Memoria Flash Quad-SPI de 64MB.
- Módulo Bluetooth v4.1 (SPBTLE-RF).
- Módulo RF (868 MHz) programable de bajo consumo (SPSGRF-868).
- Módulo Wi-Fi 802.11 b/g/n (Inventek Systems ISM43362-M3G-L44).
- Tag NFC dinámico basado en M24SR.
- 2 micrófonos omnidireccionales (MP34DT01).
- Sensor de humedadrelativa y temperatura (HTS221).
- Magnetómetro de 3 ejes (LIS3MDL).
- Giroscopio 3D y acelerómetro 3D (LSM6DSL).
- Barómetro digital (LPS22HB).
- Sensor de detección de gestos y tiempo de vuelo (VL53L0X).

Además, tiene soporte de diversos IDE como IAR o Keil, pero su gran atractivo en este ámbito es que puede ser manejada desde el sistema operativo ARM Mbed, lo que simplifica enormemente la programación del sistema.

3.2. ARM Mbed

La plataforma de desarrollo escogida para programar la placa descrita en el apartado anterior es el ARM Mbed OS, un sistema operativo en tiempo real destinado a facilitar la programación de microcontroladores ARM empleados en el desarrollo de proyectos de IOT. Las figuras 11 y 12 muestran su arquitectura y las capas que lo conforman respectivamente.

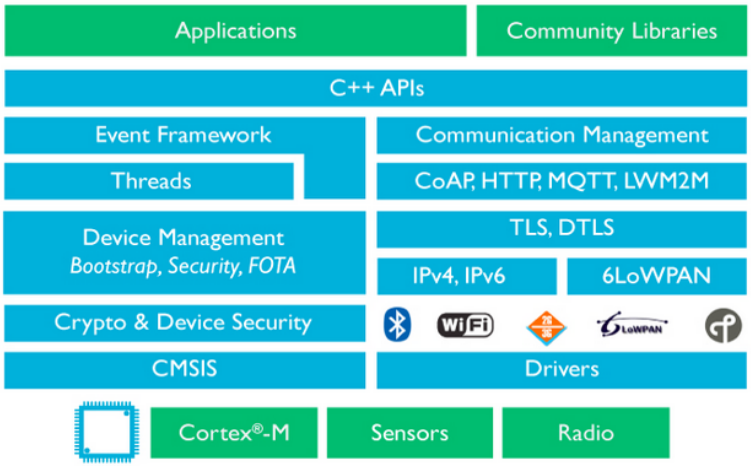


Figura 12: Arquitectura software Mbed OS [21]

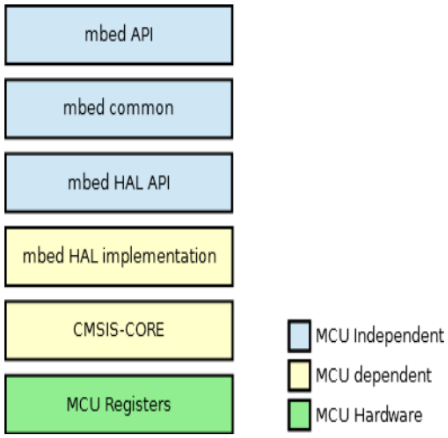


Figura 11: Capas Mbed OS [21]

El manejo de RTOS se ha convertido en una habilidad altamente demandada en el mercado laboral en los últimos tiempos, por lo que esto, sumado al amplio abanico de software de apoyo existente y la comunidad de desarrolladores que emplean Mbed OS, hacen del sistema operativo de ARM una opción ideal para desarrollar este proyecto.

Entre las diversas características que ofrece este RTOS podemos encontrar las siguientes:

- **Modular:** Las librerías necesarias para el proyecto se añaden automáticamente al dispositivo, lo que permite al desarrollador centrarse en la construcción del código.
- **Conectividad alta:** Mbed OS ofrece diversas opciones de comunicación (BLE, WiFi, RFID, NFC, Ethernet, 6LoWPAN, Thread, etc.).
- **Seguridad alta:** La seguridad multicapa y Mbed TLS hacen Mbed OS un RTOS seguro.
- **Seguridad *End-to-end*:** Además, el alto nivel de seguridad no se aplica sólo al hardware, sino también a las comunicaciones, el software e incluso el ciclo de vida del dispositivo.
- **Soporte de dispositivos y componentes:** Mbed OS da soporte a una gran variedad de dispositivos basados en ARM Cortex-M.
- **Ejecución de software en tiempo real:** Al estar basado en CMSIS-RTOS RTX, Mbed OS soporta ejecución determinista, multihilo y en tiempo real de software.
- **Drivers y librerías de soporte:** Mbed OS proporciona soporte de drivers para periféricos de una amplia gama de MCUs, así como librerías para cada *toolchain* soportada.

- ***Open source*:** Al ser *open source*, Mbed OS puede emplearse en proyectos de cualquier índole (incluyo aquellos que vayan a ser comercializados).
- **Comunidad amplia:** Mbed goza de una comunidad activa y global de desarrolladores y socios.
- **Fácil de usar:** La API de Mbed OS, la estructura modular de librerías y la enorme cantidad de código de ejemplo existente hacen que aprender a usar Mbed OS sea sencillo e intuitivo.

Otra particularidad de este RTOS es que otorga al desarrollador dos formas diferentes de trabajar:

- **Método online (Mbed Online Compiler):** El compilador online de Mbed tiene la gran ventaja de que permite comenzar proyectos de forma rápida y desarrollarlos empleando herramientas visualmente intuitivas (figura 13).

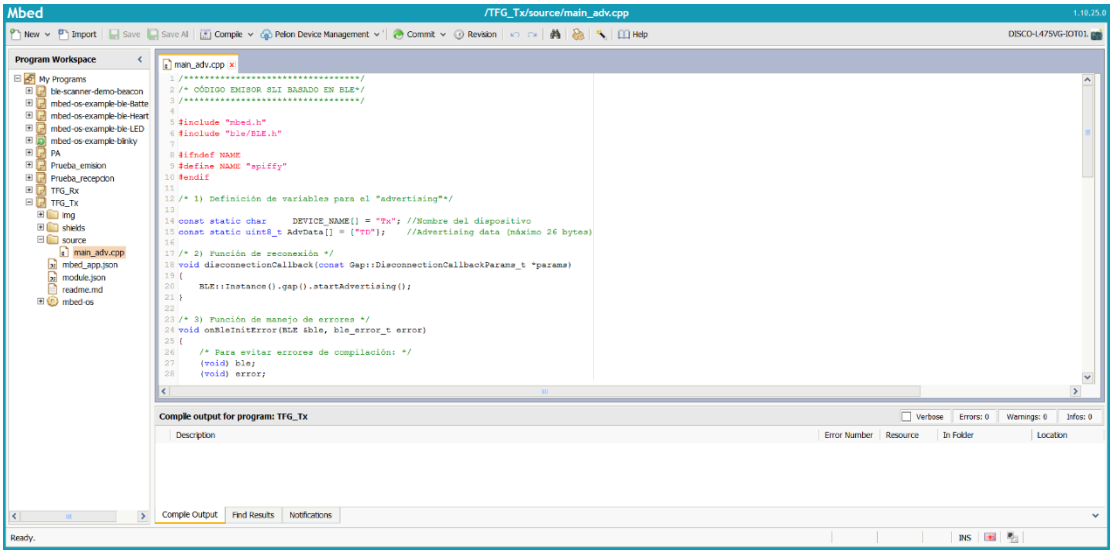


Figura 13: Entorno de trabajo Mbed Online Compiler

- **Método offline (Mbed CLI):** La alternativa al compilador online de Mbed consiste en una interfaz de línea de comandos (figura 14). Pese a que su instalación es algo más compleja y visualmente no es tan intuitiva, no sólo permite al desarrollador trabajar en cualquier sitio, sino que ofrece más opciones que su homólogo.

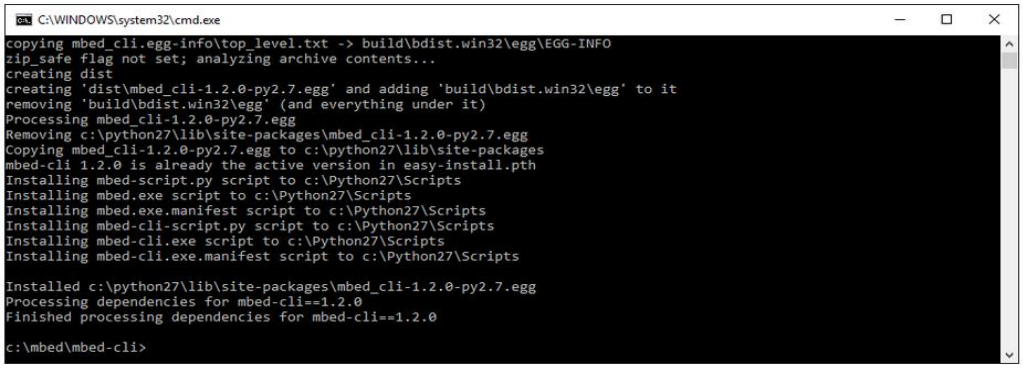


Figura 14: Entorno de trabajo Mbed CLI

Debido a problemas de instalación de los drivers de la placa en Mbed CLI, en este proyecto se optó por emplear exclusivamente el compilador online.

Como se aprecia en la figura 15, los proyectos y los archivos desarrollados en este espacio de trabajo se organizan por carpetas (al igual que en la mayoría de los sistemas operativos para PC empleados en la actualidad). De forma predeterminada, cada proyecto nuevo se genera dentro de la carpeta "My Programs" y, dentro de los mismos, se almacenan las diferentes carpetas de archivos y librerías generadas/importadas por el desarrollador.

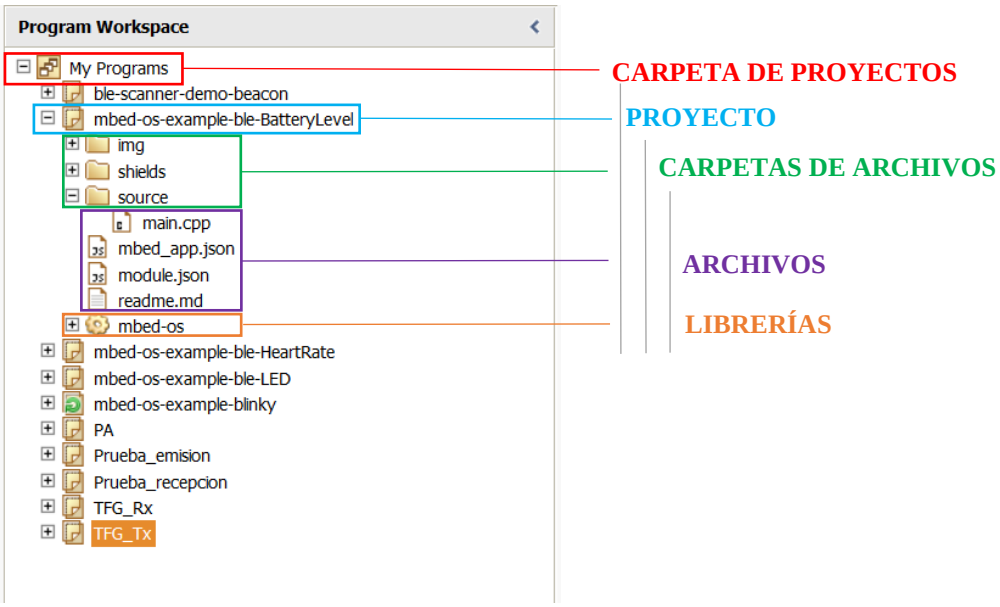


Figura 15: Organización "Program Workspace" Mbed Online Compiler

3.3. Bluetooth Low Energy (BLE)

El protocolo Bluetooth Low Energy (también conocido como Bluetooth Smart) es un subconjunto del protocolo Bluetooth (disponible a partir de la versión 4.0) diseñado para disminuir notablemente el consumo energético de los dispositivos.

En la figura 16 se muestra una comparativa de las características técnicas del Bluetooth convencional y el BLE.

Technical Specification	Classic Bluetooth technology	Bluetooth low energy technology
Radio Frequency	2.4 Ghz	2.4 Ghz
Distance/Range	30 meters	50 meters
Over the air data rate	1–3 Mbit/s	1 Mbit/s
Application throughput	0.7–2.1 Mbit/s	0.2 Mbit/s
Active slaves	7-16,777,184	Unlimited
Security	64/128-bit and application layer user defined	128-bit AES and application layer user defined
Robustness	Adaptive fast frequency hopping, FEC, fast ACK	
Adaptive fast frequency hopping		
Latency (from a non-connected state)	Typically 100 ms	6 ms
Total time to send data	100 ms	<6 ms
Government Regulation	Worldwide	Worldwide
Certification Body	Bluetooth SIG	Bluetooth SIG
Voice capable	Yes	Yes, with some limitation
Network topology	Scatternet	
Star-bus		
Power consumption	1 as the reference	0.01 to 0.5 (depending on use case)
Peak current consumption	<30 mA	<15 mA
Service discovery	Yes	Yes
Profile concept	Yes	Yes
Primary use cases	Mobile phones, gaming, headsets, stereo audio streaming, automotive, PCs etc.	Mobile phones, gaming, PCs, watches, sports and fitness, healthcare, security & proximity, automotive, home electronics, automation, Industrial, etc.

Figura 16: Comparativa Bluetooth – BLE [22]

En cuanto a su arquitectura, la pila de protocolos BLE (figura 17) consta de las siguientes capas:

- Capa física.
- Capa de enlace.
- Interfaz de controlador de host (*Host Controller Interface* (HCI)).
- L2CAP (*Logical Link Control and Adaptation Protocol).*

- ATT (*Attribute Protocol*).
- SM (*Security Manager).*
- GAP (*Generic Access Profile*).
- GATT (*Generic Attribute Profile*).
- Aplicación.

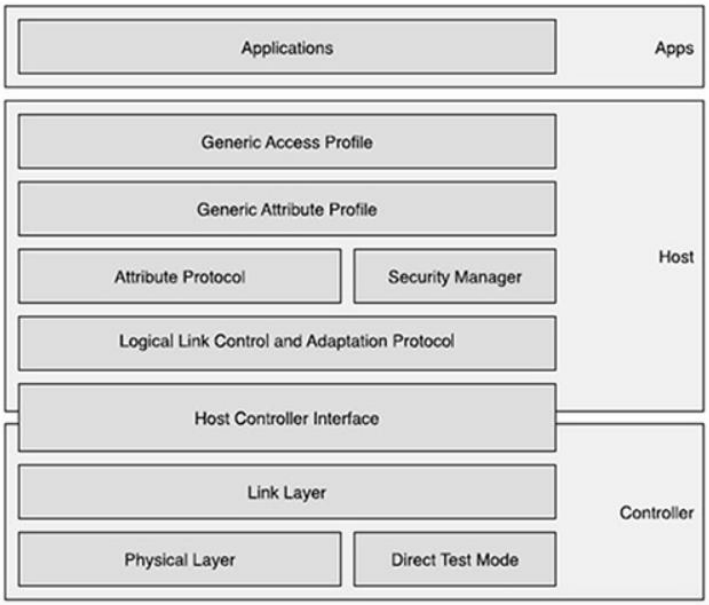


Figura 17: Pila de protocolos BLE [23]

Cabe prestar especial atención a las capas GAP, encargada de controlar las conexiones y el "advertising" (emisión de datos sin conexión), y GATT, responsable de establecer la forma en que dos dispositivos BLE intercambian información entre sí.

Capa GAP: Además de lo citado en el párrafo anterior, la capa GAP también define los dos roles que pueden adoptar los dispositivos con BLE:

- Periféricos: Dispositivos normalmente pequeños y de bajo consumo que se conectan a un dispositivo mayor más potente (central).
- Dispositivo central: Dispositivos de mayor potencia y consumo que los periféricos.

Del mismo modo, los periféricos emiten periódicamente una carga útil de datos denominada A*dvertising Data* que puede ser leída por los dispositivos centrales. De tal modo, si alguno de ellos necesita más información del periférico, puede enviarle una señal (*ScanRresponse Request*) solicitando más información (S*can Response Data*). Este proceso se ilustra perfectamente en la figura 18.

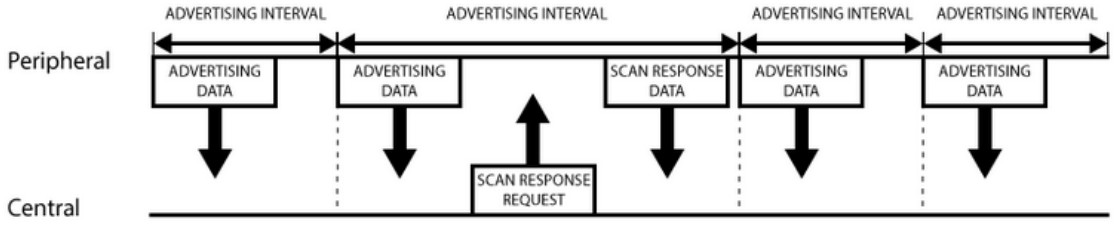


Figura 18: Transmisión de datos GAP [24]

Capa GATT: Una vez se establece la conexión entre un periférico y un dispositivo central, el protocolo GATT se encarga de gestionar la transmisión de datos en cualquiera de las dos direcciones. Este tipo de conexiones tiene una particularidad, y es que, mientras los dispositivos centrales pueden asociarse a varios periféricos, estos últimos no pueden vincularse a más de un dispositivo central. Además, si se necesita transmitir información de un periférico a otro, esta deberá pasar obligatoriamente por un dispositivo central al que ambos estén conectados. Por lo tanto, puede asumirse que el protocolo GATT establece una topología de conexión en red como la mostrada en la figura 19.

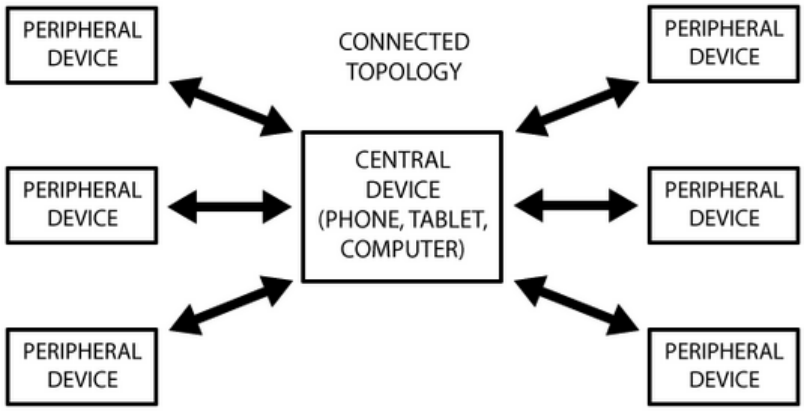


Figura 19: Topología de conexión en red GATT [24]

Por otro lado, al establecerse una conexión, el periférico pasa a ser el servidor GATT (esclavo) y el dispositivo central el cliente GATT (maestro). En la comunicación entre ambos el maestro siempre debe llevar la iniciativa, pudiéndose ver un ejemplo claro de esto en la figura 20.

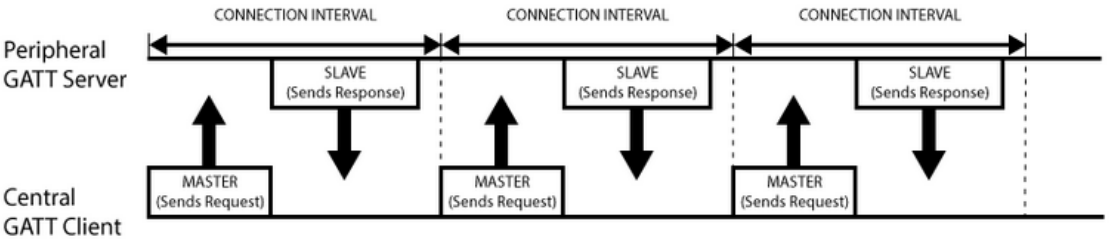


Figura 20: Comunicación bidireccional GATT [24]

4. DESARROLLO DEL SISTEMA

Como se mencionó anteriormente, para implementar el SLI se emplean dos dispositivos principales: un emisor (Tx) y un receptor (Rx), añadiéndose durante el desarrollo un ordenador para poder mostrar al usuario los datos. El emisor (periférico) es concebido como el elemento móvil del sistema, por lo que cumple la función de un "tag", mientras que el receptor (dispositivo central) se encarga de leer la información transmitida por este último y medir el RSSI (*Received Signal Strength Indicator*) de la señal (es decir, la fuerza con la que llega a él).

Figura 21: Estructura SLI desarrollado

Como se muestra en la figura 21, el SLI tiene una estructura muy simple: el emisor transmite la señal al receptor por BLE y este la recibe, extrae el RSSI y lo manda por el puerto serie al ordenador (que muestra su valor por pantalla mediante el emulador de terminal *Tera Term*).

Para llevar a cabo esto, se crearon dos proyectos de Mbed OS: uno para programar el emisor y otro para el receptor. Los códigos fuente de ambos (recogidos en el anexo A) están escritos en C++, y en ambos proyectos se emplean tres librerías principales:

- **Mbed.h:** *Fork* del SDK *(Software Development Kit)* de Mbed para C y C++.
- **Ble.h:** Librería básica del protocolo BLE.
- **Gap.h:** Librería básica del protocolo GAP. Incluye funciones que permiten controlar la comunicación entre dispositivos BLE.

Del mismo modo, todos los conceptos teóricos necesarios para entender las bases de la comunicación por BLE y profundizar en los tipos de datos manejados por este han sido extraídos de los siguientes documentos:

- *Bluetooth Core Specification (v5.1)* [8]
- *Supplement to the Bluetooth Core Specification (v8)* [9]

En los próximos apartados se profundizará en los procesos de implementación del emisor y el receptor por separado, explicando en detalle las diversas partes que los conforman y conceptos claves como el método de lectura empleado por el receptor o el formato adoptado por el A*dvertising Data*.

4.1. Implementación del emisor (Tx)

Antes de analizar el funcionamiento del código del emisor es necesario prestar especial atención a qué forma tienen los datos emitidos por el mismo (es decir, el *Advertising data*).

Figura 22: Formato de la carga útil del Advertising data

En primer lugar, la carga útil tiene 31 bytes de capacidad, pero, como refleja la figura 22, solo puede emplearse un máximo de 29 bytes para almacenar información. Esto se debe a que los dos primeros se emplean para indicar el tamaño de la carga útil (expresado en bytes) y el tipo de datos contenidos en la misma. Teniendo esto último en cuenta, se puede llegar a la conclusión de que existen 255 tipos de datos válidos (el valor 0x00 no representa ninguno), pero en este caso basta con saber que los más empleados son los de la tabla 1:

Tabla 1: Tipos de datos más comunes usados en el Advertising Data

Data Type Value	Data Type Name	Reference for Definition
0x01	«Flags»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.3 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.3 and 18.1 (v4.0)Core Specification Supplement, Part A, section 1.3
0x02	«Incomplete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x03	«Complete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x04	«Incomplete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, section 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x05	«Complete List of 32-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, section 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x06	«Incomplete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x07	«Complete List of 128-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x08	«Shortened Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0)Core Specification Supplement, Part A, section 1.2
0x09	«Complete Local Name»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0)Core Specification Supplement, Part A, section 1.2

Además, si se desea enviar varios "bloques" de datos a la vez, es posible concatenarlos dentro de la misma carga útil siempre y cuando se respete el formato expuesto anteriormente (figura 22).

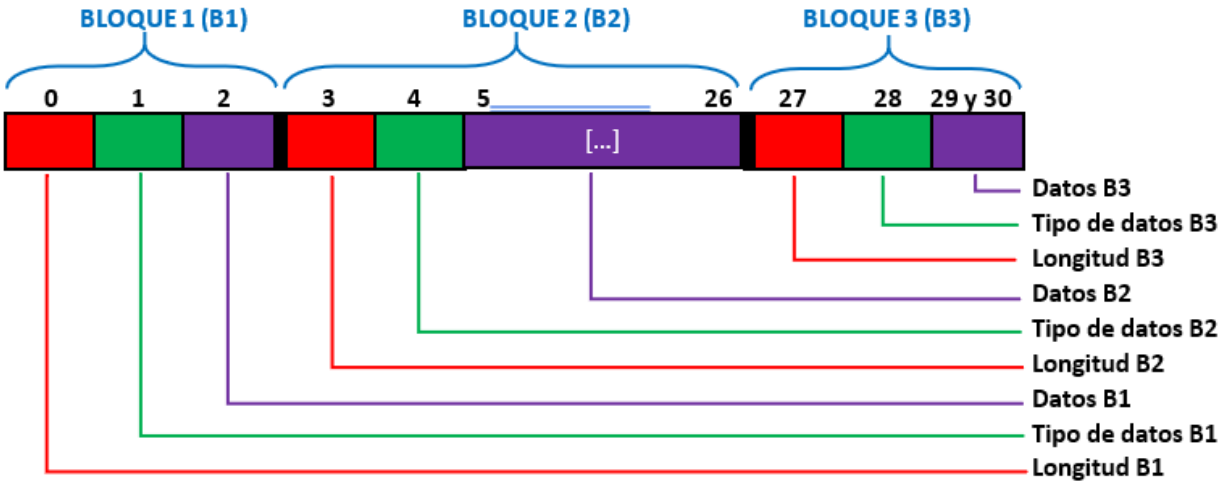


Figura 23: Carga útil generada por el emisor

Esta práctica, a la par que usual, resulta de gran utilidad para evitar tener que emplear el *Scan Response Data*, por lo que se optó por recurrir a ella en este mismo trabajo. Concretamente, la carga útil generada por el emisor consta de tres bloques de datos fácilmente diferenciables en la figura 23:

- **Bloque 1 (Byte 0 → Byte 2):** En esta primera parte de la carga útil se envían únicamente *flags* (Byte 1 bloque= 0x01). Concretamente, se activan el 0x01 y el 0x04 que, como puede observarse en los comentarios del código del anexo A, establecen que el emisor solo sea *Low Energy* (LE) y sea detectable constantemente.
- **Bloque 2 (Byte 3 → Byte 26 máximo):** La segunda parte de la carga útil se encarga de enviar la información de interés en forma de *Manufacturer Specific Data* (Byte 1 bloque= 0xFF). En realidad, como en este caso sólo se desea medir el RSSI, carece de importancia qué mensaje se transmita, pero aun así resulta conveniente saber esto para tener una identificación completa del programa.
- **Bloque 3 (Si el bloque 2 llega hasta el byte 26: Byte 27 → Byte 30):** En la tercera parte de la carga útil únicamente se almacena el nombre del dispositivo, denominado "Tx". Este valor es usado por el receptor para identificar la señal que debe leer (si no, podría confundirse con otros periféricos).

Con esto puede darse por finalizada la explicación del *Advertising Data*. Ahora bien, ¿cómo funciona el código fuente del emisor? A grandes rasgos, este consta de cinco partes bien diferenciadas:

- Declaración e inicialización de variables (1).
- Función de reconexión (2).
- Función de manejo de errores (3).

- Función postinicialización del *Baselayer* (4).
- Función principal (5).

Sin embargo, todas ellas convergen en un "proceso" lineal cuyo diagrama de flujo puede observarse en la figura 24:

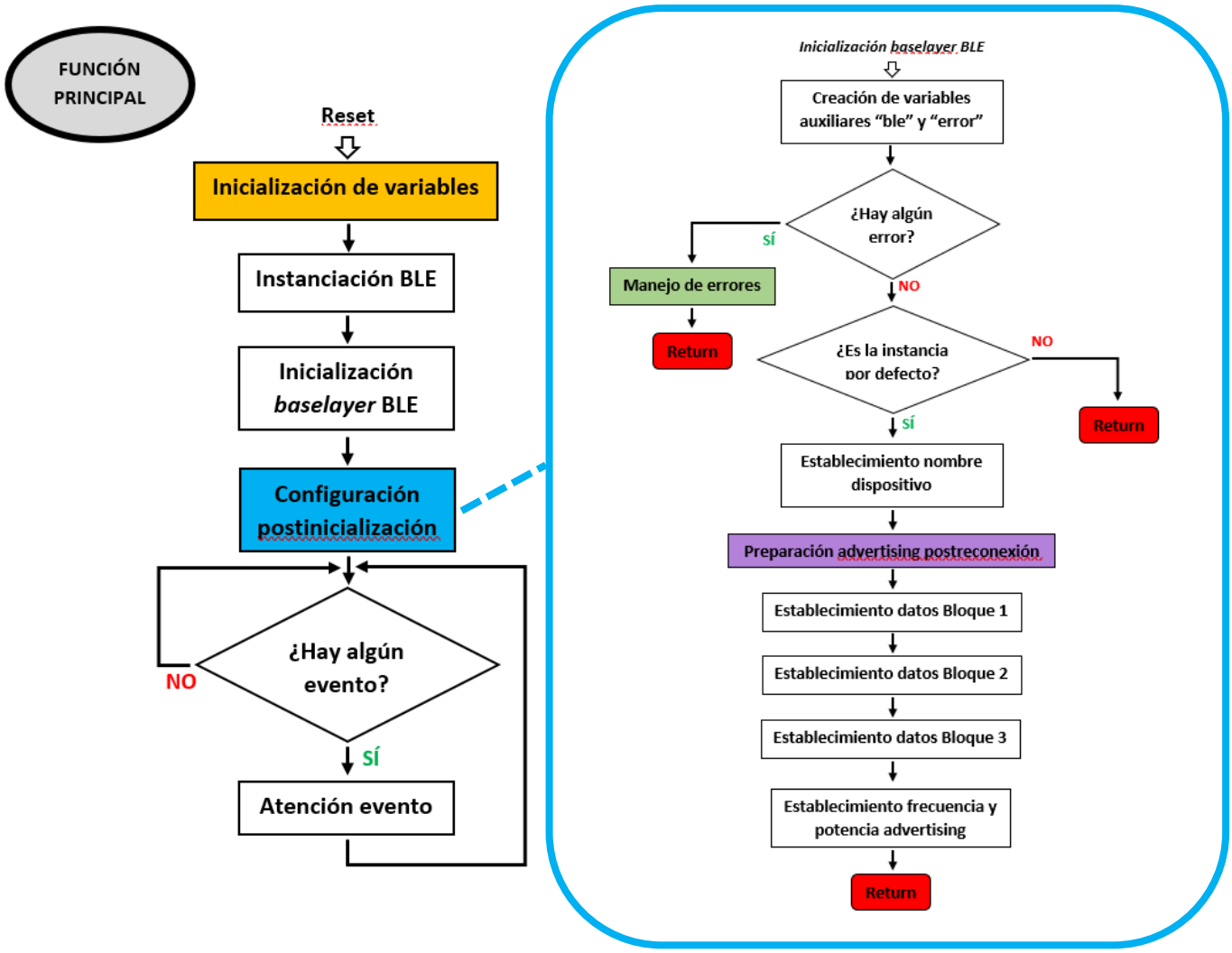


Figura 24: Diagrama de flujo del código fuente del emisor

Primero, se declaran y se inicializan las variables necesarias en el proceso. Estas son el nombre del dispositivo y los datos que se quieren transmitir mediante el *Advertising Data* (necesarios a la hora de "escribir" la carga útil tras la inicialización del *Baselayer*).

Posteriormente, se accede a la función principal (el paso anterior no pertenece a ella) y se crea una instancia/objeto de la clase BLE empleando una función propia de ble.h (*BLE::Instance(BLE::DEFAULT_INSTANCE)*). Una vez hecho esto, mediante la función *ble.init(bleInitComplete)* se inicializa el *Baselayer* BLE y posteriormente se hace un *Callback* a la función de postinicialización (bloque azul).

En cuanto se pasa a esta función, se generan dos variables auxiliares para los pasos posteriores: ble (con un constructor copia de BLE) y error (variable para detectar si se ha

producido algún error en los pasos anteriores). Luego, empleando esta última, se verifica que no existan errores. Si no los hay, se salta al siguiente paso, pero si se detecta alguno, se hace un *Callback* a la función de manejo de errores (en este caso, se ha diseñado de forma simple, prescindiendo de un método de manejo de errores y optando por evitar errores de compilación únicamente) y se sale de la función de postinicialización.

El siguiente paso es similar a este, con la diferencia de que lo que se comprueba no es la inexistencia de errores, sino que la instancia generada anteriormente coincida con la instancia por defecto. En caso negativo, se sale de la función de postinicialización y, en caso positivo, se establece el nombre del dispositivo.

Tras esto, mediante la sentencia *ble.gap().onDisconnection(disconnectionCallback)* proveniente de Gap.h, se hace otro *Callback*, pero en este caso a la función de reconexión y únicamente si se ha producido una desconexión (este comportamiento se programa dentro de la propia sentencia). En este caso, la función de reconexión únicamente pone en marcha de nuevo el *Advertising*.

A continuación, se añaden los bloques 1, 2 y 3 de datos (con todas las características especificadas anteriormente) a la carga útil concatenándolos. Para finalizar la función de postinicialización se establecen la frecuencia y la potencia (en dBm) con la que se hace el *Advertising*.

Por último, se vuelve a la función principal y se entra en un bucle infinito de espera a eventos (la función *ble.waitForEvent()* de ble.h permite hacer esto de forma muy sencilla, dado que al llamarla se fuerza la atención automática de cualquier evento próximo). De esta forma, el microcontrolador pasará a un estado de reposo si no tiene tareas que atender, reduciendo significativamente el consumo energético. Como el *Advertising* es un evento, este será continuo.

4.2. Implementación del receptor (Rx)

El código fuente del receptor también puede encontrarse en el anexo A, y consta de las siguientes partes:

- Declaración e inicialización de variables **(1)**.
- Función de escaneo **(2)**.
- Función de manejo de errores **(3)**.
- Función postinicialización del *Baselayer* **(4)**.
- Función principal **(5)**.

A primera vista puede apreciarse que su estructura es muy similar a la del código del emisor. Sin embargo, como puede apreciarse en el diagrama de flujo mostrado en la figura 25, ambos difieren en puntos clave:

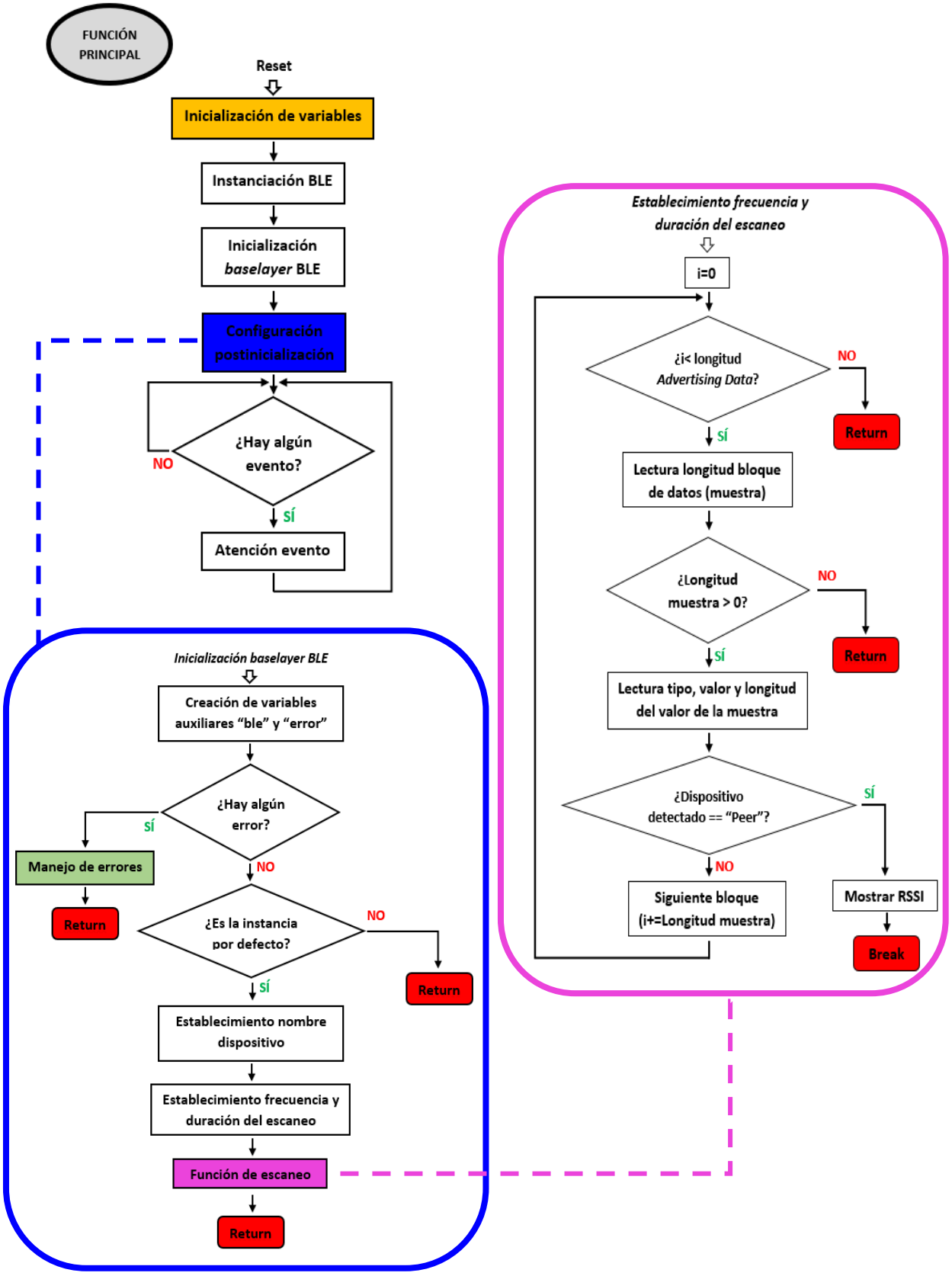


Figura 25: Diagrama de flujo del código fuente del receptor

En primer lugar, se declaran e inicializan las variables necesarias en el resto del proceso. Estas son el nombre del receptor y el del emisor que se desea detectar. De ahora en adelante, nos referiremos a este último como *Peer* (pareja).

Posteriormente, de la misma forma que ocurría con el emisor, es necesario generar un objeto BLE e inicializar el *Baselayer* para establecer el protocolo de comunicación. Una vez superado esto se hace un *Callback* a la función de postinicialización.

Las similitudes con el emisor también aparecen en ella, puesto que se implementan los mismos métodos de manejo de errores y verificación de instancia por defecto. Posteriormente se establecen el nombre del dispositivo y la frecuencia y duración del escaneo. En este caso se ha optado por que ambos tiempos sean iguales (500 ms) para así recoger más datos en menos tiempo a costa de aumentar el consumo. Sin embargo, las coincidencias acaban aquí, puesto que en el siguiente paso se comienza el escaneo mediante la función *ble.gap().startScan(advertisementCallback)*.

Esta función, proveniente de Gap.h, ordena al dispositivo comenzar el escaneo hasta detectar un periférico dentro de su rango que esté emitiendo datos. Cuando esto ocurre, se hace un *Callback* a otra función que comprueba si es el *Peer* o no.

Para ello, se usa un bucle "for" con una variable contadora "i" que recorre todas las posiciones de memoria de la carga útil del *Advertising Data*. Teniendo en cuenta que los datos esperados tienen el formato especificado en el apartado anterior (figura 23), la variable contadora accederá a la primera posición identificando su valor como la longitud del bloque de datos (denominado muestra en esta sección) en que se encuentra. Si es mayor que cero (existencia de información en la carga útil), se registrarán el tipo de datos de la muestra, su valor y la longitud de este último. A continuación, comparando estos elementos con sus equivalentes del *Peer* se determina si coinciden o no. En caso afirmativo, se lee el valor del RSSI y se muestra en la terminal del puerto serie del ordenador mediante un "printf", mientras que en caso negativo se pasa al siguiente bloque de datos y se repite el ciclo hasta que no queden más (esto se logra sumándole a la variable contadora el tamaño de la muestra anterior).

Por último, una vez hecho esto, se vuelve a la función principal y se entra en un bucle infinito de espera a eventos como el del emisor. Como el escaneo es un evento, esto equivale a un escáner periódico.

5. RESULTADOS EXPERIMENTALES

5.1. Medidas iniciales

Una vez implementados el emisor y el receptor, el siguiente paso para lograr el SLI era calibrar el sistema y generar un modelo matemático para obtener la distancia entre ambos dispositivos a partir del RSSI medido. Sin embargo, al tomar diversas muestras, se observó que, incluso dejando las dos placas a una distancia fija en una posición estática, este valor sufría variaciones muy bruscas que dificultaban el proceso (como se puede observar en la figura 26).

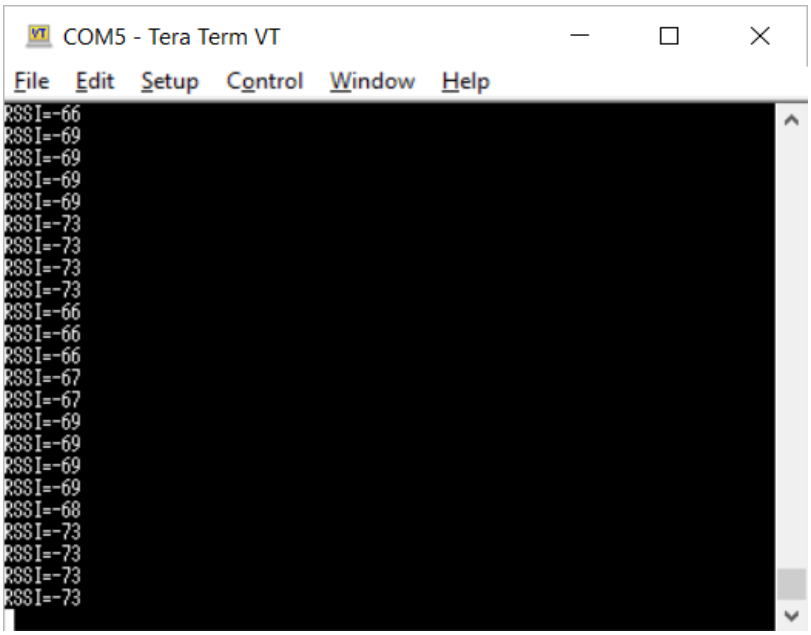


Figura 26: Lecturas RSSI mostradas en Tera Term

Para determinar si estas fluctuaciones siguen un patrón determinado o no, resulta más sencillo representar gráficamente la evolución del RSSI con el tiempo que emplear la consola del puerto serie. Por ello, se decidió exportar los valores del puerto serie a MATLAB y mostrarlos gráficamente desde allí

El programa, recogido en el primer apartado del anexo B de este documento, simplemente lee los valores del puerto serie y, con ayuda de una variable contadora, almacena las muestras en un vector. Posteriormente, al alcanzarse un número determinado de muestras (en este caso 200 para que se viese bien la evolución del RSSI), se representan todas en una gráfica.

En la figura 27 se puede ver el resultado de ejecutar este programa a diferentes distancias.

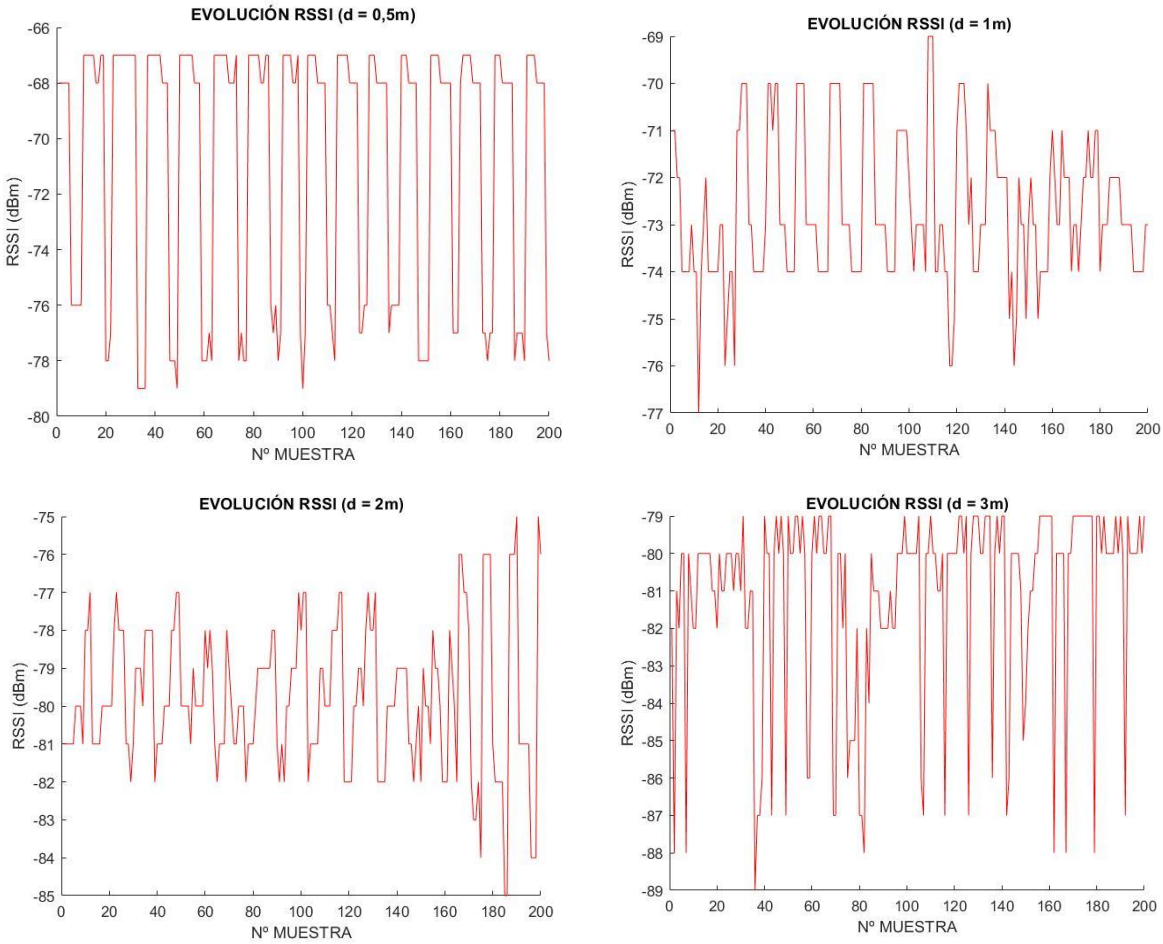


Figura 27: Representación gráfica RSSI a diferentes distancias

Analizando estas gráficas se extraen tres conclusiones principales:

- Las variaciones en la señal no siguen un patrón definido.
- La fluctuación de la señal aumenta con la distancia entre el emisor y el receptor.
- Si se toma esta señal sin filtrar para el proceso de calibración, la exactitud a la hora de medir la distancia disminuirá enormemente.

Tomando esto en cuenta se llega fácilmente a la conclusión de que, si se desea que el SLI tenga una exactitud aceptable, es necesario diseñar un filtro para esta señal. Por la forma de la misma, no parece que un filtro electrónico digital sea la opción más adecuada, pero sí parece eficaz implementar un filtrado estadístico. Llegados a este punto, la siguiente pregunta que debe responderse es qué grado de complejidad debe tener este filtro.

Para averiguarlo, en primer lugar, se optó por desarrollar un programa en MATLAB (Anexo B apartado 2) que obtuviese la media de cierta cantidad de muestras (50, 500 y luego 1000). Este funciona de forma muy similar al programa diseñado para representar gráficamente la evolución del RSSI, con la diferencia de que, al terminar de recoger las muestras, en vez de pasarlas a una gráfica se calcula la media de todas las medidas.

Colocando el emisor a una distancia fija del receptor (en este caso 1m), se pretendía analizar si la media era constante o, si en caso de variar, su fluctuación era lo suficientemente pequeña como para que implementar un filtro así de simple mejorase la exactitud del sistema. Los resultados obtenidos se reflejan en la tabla 2:

Tabla 2: Cálculo de la media del RSSI a 1m

	Medidas Laboratorio			
Distancia (m)	**Nº Medida**	**Número de muestras**	**Media RSSI (dBm)**	**Media de medias (dBm)**
1	1	50	-77,94	-78,452
	2	50	-78,78	
	3	50	-77,92	
	4	50	-78,16	
	5	50	-79,46	
	1	500	-77,53	-76,806
	2	500	-76,92	
	3	500	-76,47	
	4	500	-76,2	
	5	500	-76,91	
	1	1000	-75,03	-75,964
	2	1000	-75,31	
	3	1000	-75,84	
	4	1000	-76,03	
	5	1000	-77,61	

Atendiendo a ellos, se puede observar que la media no sólo no se mantiene constante, sino que fluctúa varias unidades. Además, aumentar el número de muestras no estabiliza demasiado su valor ni en un rango tan corto como este, por lo que no hace falta repetir el ensayo a otras distancias para concluir que emplear este método de filtrado es inviable.

Esto demuestra que hay que elevar la complejidad del filtro, pero no permite vislumbrar aún la forma que debe tener para solucionar el problema planteado. Para tener una perspectiva más amplia, en el siguiente paso del desarrollo se optó por analizar la distribución de las muestras con la intención de que esto permitiese intuir qué camino tomar en los ensayos siguientes. Nuevamente se diseñó un programa en MATLAB para tal fin (Anexo B apartado 3).

Su estructura es muy similar a la de los dos programas desarrollados anteriormente, con la diferencia de que, tras almacenarse las muestras en un array, dos variables contadoras van recorriéndolo de principio a fin y agrupando los valores en otro array. Posteriormente se genera el histograma con los datos de este último y se calculan parámetros estadísticos relevantes del conjunto (media, mediana, moda, desviación estándar y rango).

En la figura 28 se muestran varios histogramas generados manteniendo la distancia fija.

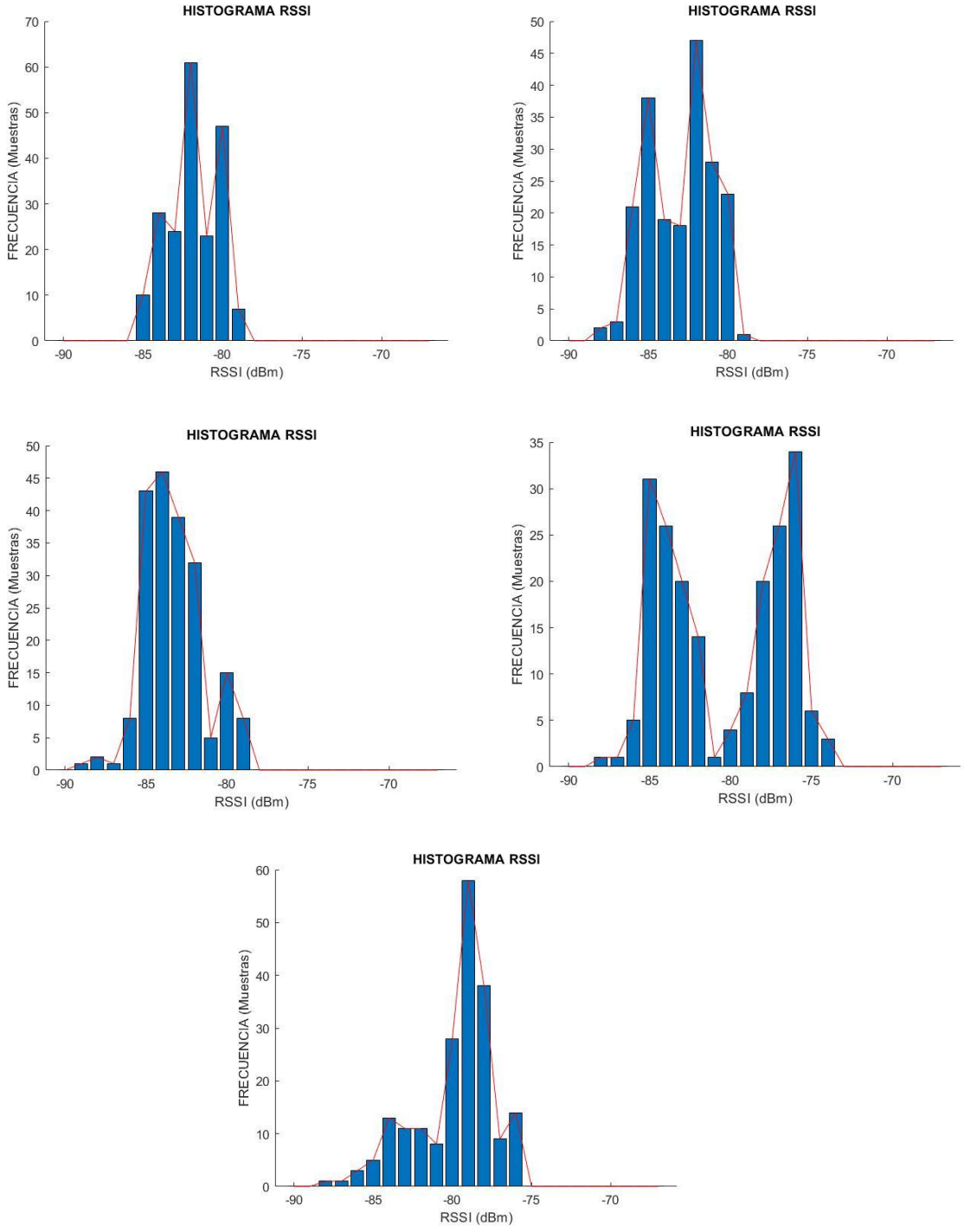


Figura 28: Histogramas RSSI

Analizando los histogramas conjuntamente no se detecta la repetición exacta de un patrón, pero sí se aprecia que en la mayoría de los casos la distribución de las muestras se asemeja a una distribución normal. Si tomamos el último histograma como ejemplo, incluso podemos advertir que la media, la mediana y la moda calculadas por el programa tienen el mismo valor (como se refleja en la figura 29), característica típica de este tipo de distribuciones.

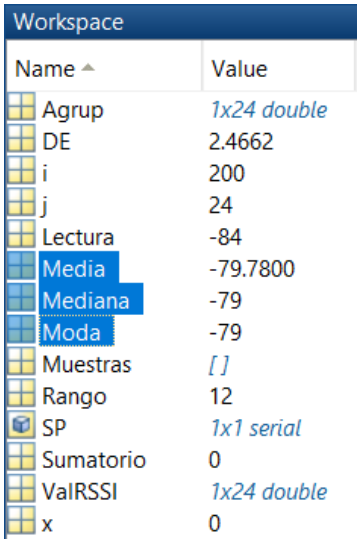


Figura 29: Media, mediana y moda de un conjunto de muestras aleatorio (Histograma 5)

Esto muestra que, aunque el rango de valores del RSSI sea muy amplio, la mayoría de las muestras suele agruparse en torno a la media. Esta información es crucial para desarrollar un método de filtrado efectivo, por lo que se explicará en más profundidad en el siguiente apartado.

5.2. Filtrado de la señal (teoría)

Asumiendo que la distribución de muestras tiende a comportarse como se describe en el apartado anterior, se llega a la conclusión de que desechar las muestras que se alejen demasiado de la media lograría reducir la fluctuación del RSSI.

Para ello, se plantea implementar un filtro que establezca límites controlados por los dos parámetros estadísticos que se suelen emplear para caracterizar una distribución normal: la media aritmética (μ) y la desviación estándar (σ). Ambos parámetros se pueden calcular a partir de las ecuaciones mostradas en la figura 30, siendo "x" la muestra actual y "N" el número total de muestras.

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i \qquad \sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

Figura 30: Ecuaciones de la media aritmética y la desviación estándar

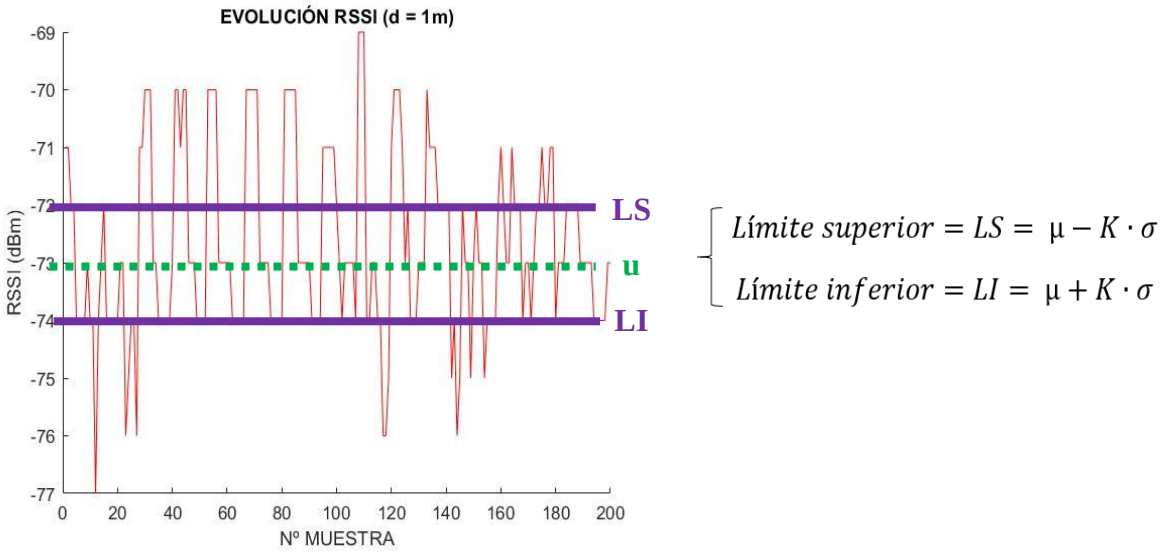


Figura 31: Límites estáticos RSSI

De tal forma, los valores que quedasen fuera de dichos límites no se tendrían en cuenta en cálculos posteriores. Estos se obtendrían como se muestra en la figura 31.

Sin embargo, hay que tener cuidado a la hora de establecerlos, dado que se deben tener en cuenta las siguientes consideraciones:

- Si se calculan μ y σ al final de un conjunto grande de muestras, el movimiento del emisor durante el muestreo conduciría a un error de posicionamiento.
- El valor de la constante "K" debe elegirse con mucho cuidado para no desechar valores demasiado cercanos a la media o dejar pasar valores alejados.

Así pues, se decidió añadir algunas modificaciones para solventar estos problemas:

- Para evitar un comportamiento errático del sistema, se emplea una media aritmética móvil (μ_{mov}) y se calcula la desviación estándar a partir de la misma (σ_{mov}). Dividiendo el total de las muestras en subconjuntos de valores de tamaño medio, ambos parámetros se calculan para un subconjunto y definen los límites que deben respetarse durante el muestreo del siguiente. Tras repetirse esto para cada uno de ellos, al llegarse al final se calcula la media total de las muestras y se vuelve a empezar de cero, como se ilustra en la figura 32. Eligiendo los tamaños adecuados (del conjunto y los subconjuntos) este método acaba con los errores de posicionamiento ocasionados por un movimiento constante del emisor.

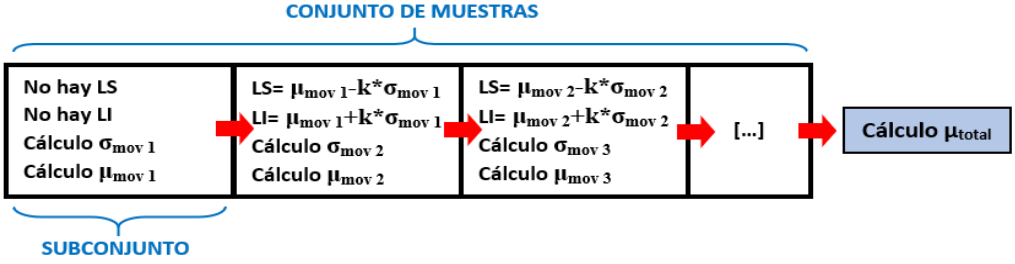


Figura 32: Diagrama cálculo de límites dinámicos RSSI

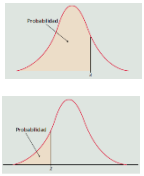
- Como en el método del punto anterior la desviación estándar varía dinámicamente, la "K" puede mantenerse estable. Para controlar qué porcentaje del conjunto se quiere cubrir aproximadamente se puede recurrir a las tablas de probabilidad acumulada de la distribución normal [10].

5.3. Filtrado de la señal (implementación)

Para poder representar gráficamente la diferencia entre la señal sin filtrar y la señal filtrada, se diseñó un programa en MATLAB (Anexo B apartado 4) que cumpliese los requisitos del apartado 5.2.

Tras realizar diversas pruebas variando el tamaño del conjunto (todas las muestras) y el subconjunto (muestras para la media móvil) se optó por asignarles 240 muestras y 15 muestras respectivamente. Otros valores también pueden dar resultados efectivos, pero estos ofrecen una tasa de error baja y un tiempo de medida del RSSI medio bastante bueno (en torno a los 30-40 segundos).

Por otro lado, se tomó K=1,28 debido a que así en una distribución normal perfecta se cubrirían aproximadamente el 80% de las muestras. La demostración puede verse en la figura 33.



$$P_+ = P(K = +1{,}28) = 0.8997$$

$$P_- = P(K = -1{,}28) = 0.1003$$

$$P_{Tot} = P_+ - P_- = 0.7994 \approx 0.8 \rightarrow \mathbf{80\%}$$

Figura 33: Demostración Probabilidad en función de "K"

El funcionamiento del programa puede verse explicado en forma de diagrama de flujo en la figura 34.

Para empezar, una vez comienza el proceso de ejecución se declaran e inicializan diversas variables que entran en juego en los pasos posteriores. Entre las más relevantes podemos encontrar variables contadoras para controlar las muestras válidas (x), las desechadas (fallos) y recorrer los subconjuntos de los que se sacan σ_{mov} y μ_{mov} (xmd), arrays para almacenar las lecturas de la media (ArrMed), la media móvil (ArrMedMov) y la desviación estándar móvil (ArrDEMov), los valores de RSSI filtrados (Muestras) y sin filtrar (Raw) y el objeto "puerto serie" SP, que permite leer los valores provenientes del receptor.

Posteriormente, se entra en un bucle de procesamiento de muestras que sólo se detiene cuando se alcanzan las 240 lecturas. Cada vez que se lee un valor, este se almacena en el array de muestras sin filtrar y se recalcula y registra la media. Si la muestra tomada es una de las 15 primeras del conjunto global, no se establecen límites y el valor leído pasa

al array de muestras filtradas, ya que en estas condiciones no hay lecturas suficientes para obtener valores de σ_{mov} y μ_{mov}. Sin embargo, al llenarse el primer subconjunto (15 muestras iniciales) comienzan a definirse los límites dinámicos planteados en el apartado anterior, variando los mismos al pasar de un subconjunto a otro. De aquí en adelante las muestras que se salen de estos límites son desechadas, mientras que las que no lo hacen se añaden a los arrays de lecturas de RSSI filtradas y muestras para la media móvil. La variable "xmd" se encarga de controlar que al acabar un subconjunto se recalculen y se registren los valores de σ_{mov} y μ_{mov}.

Una vez se alcanzan las 240 lecturas, se cierra el puerto serie y, para finalizar, se representan gráficamente todos los valores del RSSI puro, la media aritmética global y la media aritmética móvil.

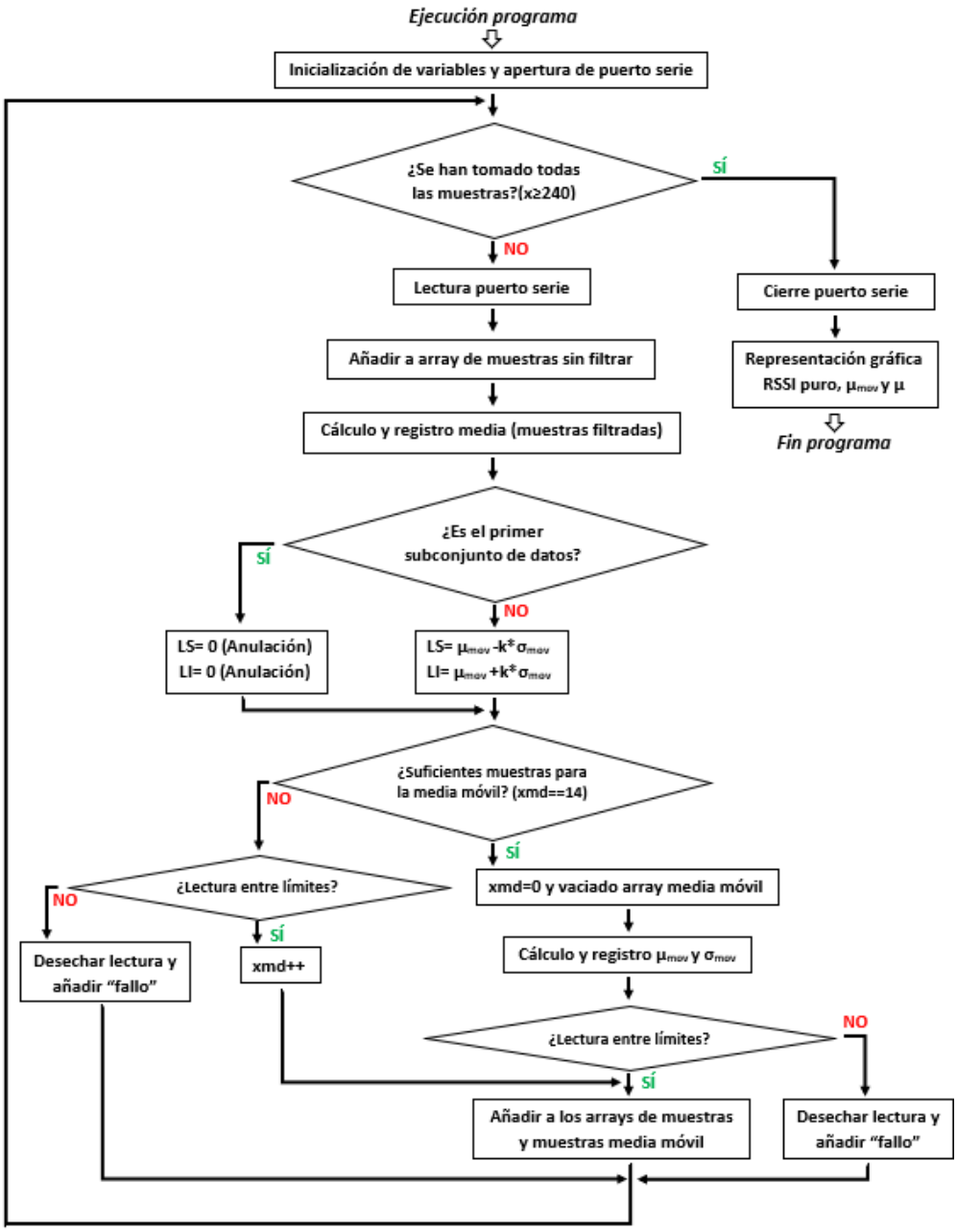


Figura 34: Diagrama de flujo filtro MATLAB

5.4. Medidas con filtrado

Como puede observarse en la figura 35, donde se muestra la gráfica obtenida al ejecutar el programa anterior a una separación emisor-receptor de 1m, los resultados obtenidos al implementar el filtro son notablemente mejores que los del apartado 5.1.

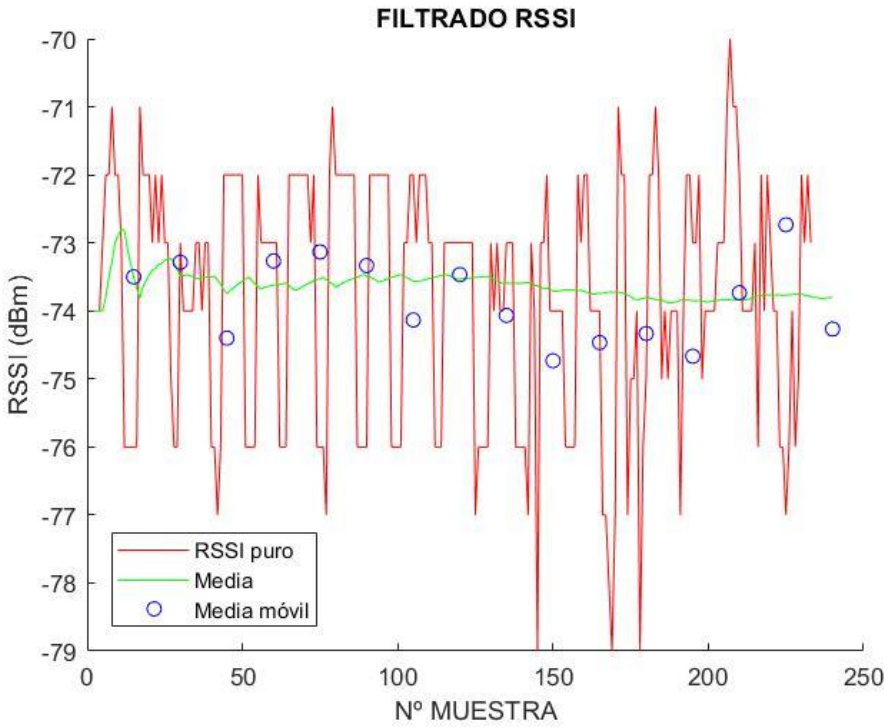


Figura 35: Comparativa señales RSSI con/sin filtro (d= 1m)

La media aritmética global comienza a estabilizarse gradualmente a partir de las primeras 15 muestras (primer establecimiento de límites), indicador de que el filtrado tiene el efecto buscado. Sin embargo, la efectividad del filtro no puede confirmarse hasta ver cómo varía el valor final de la media repitiendo el ensayo.

Por ello, se preparó una batería de pruebas que consistía en obtener la media final varias veces a diversas distancias sin obstáculos de por medio. En total se realizaron 20 ensayos a distancias de 0.5, 1, 2 y 3 metros (5 ensayos por cada distancia). Desgraciadamente, el rango de la antena del módulo BLE resultó ser mucho menor de lo esperado en un principio, observándose que aumentar la distancia más provocaba un comportamiento errático del sistema (pérdida de la mayoría de las muestras, enorme inestabilidad en las medidas y resultados contradictorios).

En consecuencia, los resultados de las pruebas (recogidos en la tabla 3) no contemplan distancias superiores.

Tabla 3: Medidas RSSI medio (con filtro)

Medidas Laboratorio				
Constante K (se multiplica por la D.E)	**Distancia (m)**	**Número de muestras**	**Media RSSI (dBm)**	**Media de medias (dBm)**
1,28	0,5	240	-67,51	-67,906
		240	-68,27	
		240	-67,58	
		240	-67,94	
		240	-68,23	
	1	240	-73,16	-73,126
		240	-73,13	
		240	-73,18	
		240	-73,37	
		240	-72,79	
	2	240	-78,81	-78,564
		240	-79,4	
		240	-77,99	
		240	-77,8	
		240	-78,82	
	3	240	-81,43	-82,088
		240	-82,6	
		240	-81,3	
		240	-82,45	
		240	-82,66	

Comparando los resultados obtenidos en el apartado 5.1 con estos se confirma una clara mejora en la estabilidad de la media RSSI.

Anteriormente este valor llegaba a oscilar más de dos unidades de una prueba a otra, pero tras el filtrado esta diferencia difícilmente supera la unidad. Además, puede verse una clara diferencia entre los valores devueltos a cada distancia, lo que indica una exactitud de medida del SLI bastante buena.

Por otro lado, atendiendo a los resultados del estudio destacado en el capítulo 2 [2], se observa que el comportamiento del SLI diseñado por los autores del artículo y el desarrollado en este trabajo presentan una similitud clara: la oscilación del RSSI. En una de las gráficas del documento (reproducida en la figura 36) se muestra cómo varía este valor con la distancia, pudiéndose ver que su fluctuación en una misma posición es de poco menos de 10 dBm.

Ciertamente, la señal medida en el apartado 5.1 presenta el mismo problema, pero el filtrado mejora atenúa tanto la media aritmética que hace a este SLI capaz de distinguir distancias menores.

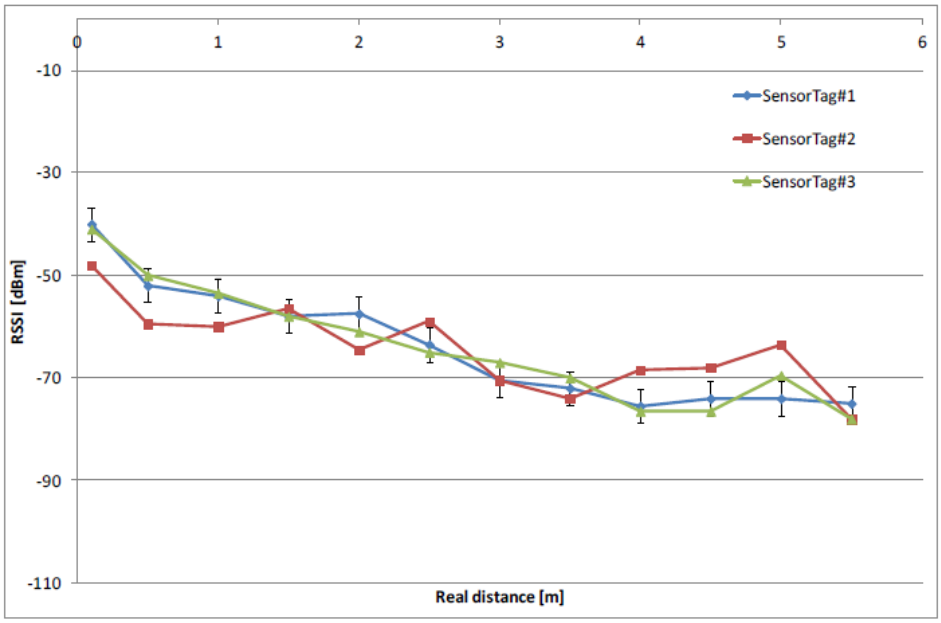


Figura 36: RSSI vs distance measurements for three TI CC2650 Sensor Tags direct view [2]

Tras completar todos los ensayos y analizar los resultados, se concluyó que, por todas las razones expuestas en los párrafos anteriores, el filtro resuelve satisfactoriamente el problema planteado.

A raíz de esto se tomó la decisión de implementarlo en las placas de desarrollo mediante Mbed OS y pasar a calcular la distancia.

5.5. Cálculo de la distancia y modificación del código Rx

Buena parte de los sistemas de localización en interiores basados en RSS emplean la fórmula de Frii [1] (figura 37) para obtener la distancia entre emisor y receptor a partir del RSSI.

$$d \approx d_0 \cdot 10^{\left(\frac{\Pi_{TX} - \Pi_{RX} + \Pi_0 - \sigma_G}{10\gamma}\right)}$$

Figura 37: Fórmula de Frii en forma de dBm [26]

d=Distancia a medir
d_0=Distancia de referencia
Π_{TX}- Π_{RX}=RSSI
σ_G=Ruido Gaussiano
Υ=Pérdidas

Sin embargo, debido a las fluctuaciones del RSSI y el poco rango de la antena, resulta complicado obtener valores realistas y estables del ruido gaussiano y las pérdidas ambientales en el proceso de calibración. Por ello, se optó por emplear un método mucho más sencillo, pero igualmente válido: la diferenciación por rangos.

Este consiste únicamente en estudiar en qué rango de valores se mueve el RSSI a diferentes distancias mediante un proceso de calibración previo. Tomando como base los resultados se establecen valores límite que, a la hora de medir, determinan si un objeto está en una posición u otra. Esta técnica sería ineficiente en sistemas con mucha más exactitud (el proceso de calibración sería demasiado largo y sería más eficiente emplear el otro método), pero la inexactitud del BLE la hace adecuada para esta aplicación.

El proceso de calibración consiste en tomar diversas medidas en un mismo entorno y a varias distancias, por lo que las pruebas realizadas para confeccionar la tabla 3 sirvieron como tal. Así pues, se determinó que en el laboratorio los valores límites son los mostrados en la figura 38:

Figura 38: Rangos de RSSI para determinar la distancia

Por otro lado, para implementar tanto esta función como el filtro sin necesidad de usar MATLAB basta con modificar el código del receptor, ya que el procesamiento de señales no implica al emisor en ningún momento.

El código Rx actualizado (Anexo A apartado 3) no es más que una combinación de la versión inicial y el filtro de MATLAB pasado a C++, por lo que el diagrama de flujo que describe su funcionamiento (figura 39) es casi idéntico al mostrado en el capítulo 4 (sólo se añaden la etapa de filtrado y el cálculo de la distancia).

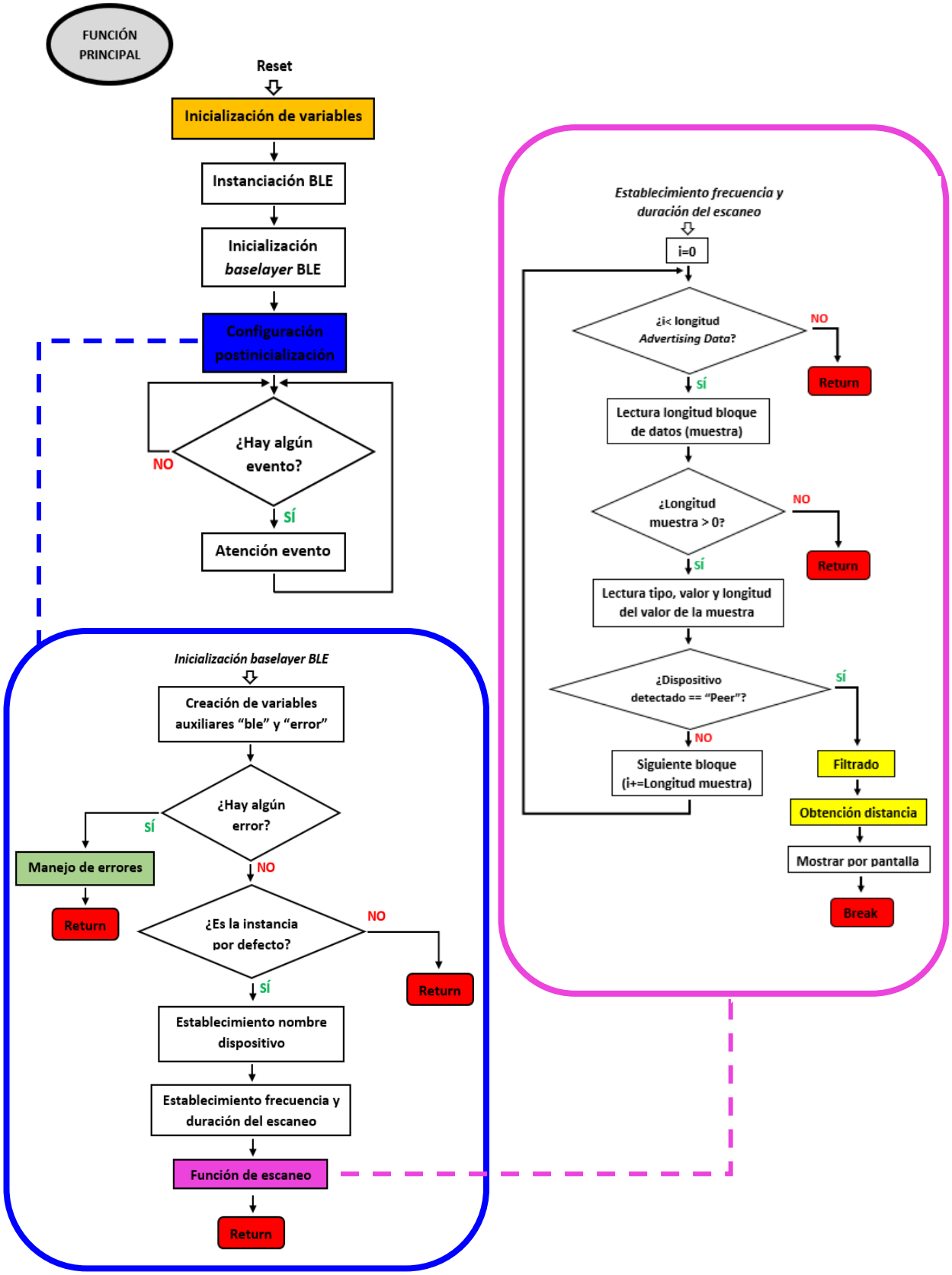


Figura 39: Diagrama de flujo del código fuente del receptor (con filtro)

5.6. Variación del RSSI por elementos externos

Todas las medidas mostradas anteriormente fueron tomadas en el mismo ambiente, a la misma altura y orientación y sin ningún elemento físico entre el emisor y el receptor.

Así, se forzaba una situación casi ideal en la que se minimizaban las alteraciones en la medida, pero a la hora de llevar el sistema a un entorno más realista hay que tener en cuenta que existen diversos factores que alteran el RSSI:

- **Obstáculos o paredes.** La existencia de cualquier objeto entre el emisor y el receptor disminuye el valor del RSSI en mayor o menor medida según su forma, su volumen y el material del que está compuesto. Un ejemplo claro de este fenómeno puede verse en la figura 40, que refleja los resultados obtenidos en el artículo del apartado 5.4 [2] al repetir el ensayo de la figura 36 con un obstáculo de por medio.

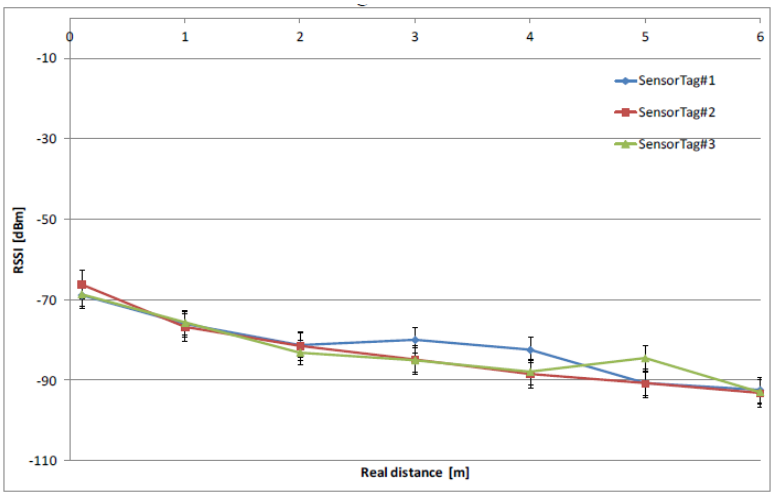


Figura 40: RSSI vs distance measurements for three TI CC2650 SensorTags with wall obstacle

- **Orientación y altura de los dispositivos.** La orientación en la que se coloquen el emisor y el receptor también afecta al RSSI. Colocar ambos dispositivos a diferentes alturas tiene un efecto muy similar.
- **Perturbaciones en el ambiente.** La existencia de otras señales de radiofrecuencia (RF) puede provocar interferencias con el BLE. Además, si estas señales son fruto de algún tipo de comunicación por Bluetooth, la interferencia es aún mayor al estar en la misma banda de frecuencias.

Por ello, a la hora de hacer la instalación del SLI resulta esencial colocar los receptores en posiciones estratégicas, aprovechando los obstáculos para diferenciar salas y calibrando perfectamente el sistema antes de comenzar a medir. Si no, los efectos de todos los factores citados anteriormente podrían inducir errores de medición del RSSI y, por lo tanto, también de la distancia.

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. Conclusiones

El estudio llevado a cabo para elaborar este documento ha seguido una metodología progresiva para abordar el problema clásico de la localización en interiores.

Volviendo a los objetivos planteados al principio, puede afirmarse que todos ellos se han cumplido satisfactoriamente:

- **Se ha hecho un estudio del funcionamiento del protocolo BLE**, observando y analizando tanto las bases teóricas del mismo como las características de las señales basadas en él. Respecto a esto último, como se mencionó anteriormente, resulta especialmente destacable la enorme fluctuación de la potencia con el paso del tiempo.
- **Se han establecido las bases de un SLI,** escogiéndose de forma razonada los componentes del mismo y sus respectivas funciones.
- **Se han escogido el hardware y la plataforma de desarrollo necesarios para implementarlo,** optando por soluciones comerciales innovadoras y con una tasa de uso alta en redes de sensores, IOT y derivados.
- **Se ha logrado desarrollar el emisor y el receptor del sistema correctamente,** funcionando ambos como se pretendía desde la concepción del sistema. Además, los programas diseñados en C++ para controlarlos manejan las comunicaciones por BLE.
- **Se han llevado a cabo numerosas medidas de la señal recibida en diferentes situaciones y se ha calibrado satisfactoriamente el SLI,** implementándose un filtro estadístico para mejorar notablemente los resultados y obteniéndose la distancia emisor-receptor a partir de los mismos. Como ventaja añadida, la exactitud alcanzada a la hora de medir la distancia acabó siendo mayor de lo esperado en un principio.
- **Se ha llevado a cabo un estudio de viabilidad a partir de los datos obtenidos,** demostrándose en los últimos apartados del capítulo 5 que un SLI basado en BLE puede ser viable si se diseña adecuadamente.

Además, los numerosos obstáculos encontrados durante el desarrollo del estudio han servido para ver de primera mano las dificultades que pueden presentarse al implementar sistemas basados en redes de sensores y comunicaciones inalámbricas.

Haber podido superarlos o al menos compensarlos demuestra que los sistemas de localización en interiores tienen un margen muy amplio de crecimiento por delante. Esto, sumado a su versatilidad en infinidad de campos, convierte a los SLI en un objeto de estudio muy interesante y una inversión de futuro para nuestra sociedad.

6.2. Mejoras futuras

El número de líneas de mejora que pueden desarrollarse como continuación a este proyecto es grande, dado que su objetivo era hacer un primer estudio de viabilidad. Dado que un trabajo Fin de Grado conlleva un esfuerzo limitado, no se ha podido ampliar el estudio abarcándolas, pero en versiones futuras resultaría extremadamente interesante abordar las siguientes:

1. **Modificación del hardware y ampliación del número de dispositivos.** Como se ha comentado en repetidas ocasiones a lo largo del documento, el hardware escogido presentaba un gran problema: el poco rango del BLE. Esto podría solucionarse cambiando la antena del módulo, pero para lograrlo habría que modificar la placa (invalidando así su marcado CE [11]), por lo que resultaría más conveniente optar por otra placa de desarrollo compatible con Mbed OS. Para reducir el consumo podría resultar interesante escoger una basada en los SoC de las familias nRF51 o nRF52 Nordic Semiconductor, caracterizados por emplearse en sistemas de IOT y derivados. Posteriormente, si se dispusiera de un presupuesto mayor, podrían adquirirse más dispositivos y hacer un SLI a gran escala.

2. **Estudio de otras tecnologías de comunicación inalámbricas.** Estudiar la posibilidad de usar otros protocolos como WiFi o ZigBee resultaría especialmente interesante desde el punto de vista de la investigación, puesto que permitiría comparar el SLI basado en BLE con ellos en términos de complejidad, consumo energético y exactitud de medida.

BIBLIOGRAFÍA

[1] «Antenna-Theory,» [En línea]. Available: http://www.antenna-theory.com/basics/friis.php. [Último acceso: 10 junio 2019].

[2] M. Kaczmarek, J. Ruminski y A. Bujnowski, «M. Ganzha, L. Maciaszek, M. Paprzycki (eds).,» *Annals of Computer Science and Information Systems,* vol. 8, pp. 1413-1416, 2016.

[3] Z. Ma, S. Poslad, Bigham John, X. Zhang y L. Men, «A BLE RSSI Ranking based Indoor Positioning,» de *2017 Wireless Telecommunications Symposium (WTS)*, Chicago, IL, USA, 2017.

[4] T. Adel, V. Thierry, N. Nejah y B. Damien, «BLE Localization using RSSI Measurements and iRingLA,» de *IEEE International Conference on Industrial Technology (ICIT),* Sevilla, España, 2015.

[5] H. Ke, H. Ke y D. Xuecheng, «A Hybrid Method to Improve the BLE-Based Indoor Positioning in a Dense Bluetooth Environment,» *Sensors (14248220),* vol. 19, pp. 424-425, 2019.

[6] M. Rodríguez, J. Pece y C. Escudero, «Blueps: Sistema de localización en interiores utilizando Bluetooth,» 2005.

[7] ST Microelectronics, «Sitio web oficial de ST Microelectronics,» [En línea]. Available: https://www.st.com/en/evaluation-tools/b-l475e-iot01a.html. [Último acceso: 03 junio 2019].

[8] Bluetooth SIG, «Bluetooth Core Specification v5.1,» 2019.

[9] Bluetooth SIG, «Supplement to the Bluetooth Core Specification (CSS) v8,» 2019.

[10] «Seactuario,» [En línea]. Available: http://seactuario.com/ContMatematicas/PROBABILIDAD25.htm. [Último acceso: 06 mayo 2019].

[11] Dekra Testing and Certification, «Marcado CE,» [En línea]. Available: https://www.marcado-ce.com/. [Último acceso: 13 junio 2019].

[12] Parlamento europeo, *Low Voltage Directive,* 2014/35/EU.

[13] Parlamento europeo, *Electromagnetic Compatibility Directive,* 2014/30/EU.

[14] Parlamento europeo, *Radio Equipment Directive,* 2014/30/EU.

[15] Parlamento europeo, *Restriction of Hazardous Substances,* 2011/65/EU.

[16] ISO, *Estándar C++17,* ISO/IEC 14882:2017-2017.

[17] IEEE, *Estándar 802.15.1,* 2002.

[18] S. Goswami, Indoor Location Technologies, Nueva York: Springer,New York,NY, 2013.

[19] IndoorAtlas, *The Rise of Indoor Positioning,* 2016.

[20] Massachusetts Institute of Technology, «Página web oficial de "Cricket",» [En línea]. Available: http://cricket.csail.mit.edu/. [Último acceso: 03 junio 2019].

[21] Greenwaves Technologies, «Página web oficial de Greenwaves Technologies,» [En línea]. Available: https://greenwaves-technologies.com/manuals/BUILD/MBED-OS/html/index.html. [Último acceso: 03 junio 2019].

[22] RTC Magazine, «Web RTC Magazine,» [En línea]. Available: http://archive.rtcmagazine.com/articles/view/103741. [Último acceso: 05 junio 2019].

[23] R. Heydon, Bluetooth Low Energy: The Developer's Handbook, Prentice Hall, 2012.

[24] K. Townsend, «Página web oficial de Adafruit,» [En línea]. Available: https://learn.adafruit.com/introduction-to-bluetooth-low-energy/gap. [Último acceso: 05 junio 2019].

[25] J. Hrisko, «Makers Portal,» [En línea]. Available: https://makersportal.com/blog/2017/1/9/using-raspberry-pi-hm-10-and-bluepy-to-develop-an-ibeacon-mesh-network-part-2. [Último acceso: 06 mayo 2019].

[26] IEEE, «Página web estándar 802.15,» [En línea]. Available: http://www.ieee802.org/15/pub/TG1.html. [Último acceso: 13 junio 2019].

[27] M. Hazas, J. Scott y J. Krumm, «Location-aware computing comes of age,» *Computer,* vol. 37, nº 2, pp. 95-97, 2004.

ANEXO A. CÓDIGOS TX Y RX

A.1. Código fuente emisor (Tx)

```
/**********************************/
/* CÓDIGO EMISOR SLI BASADO EN BLE*/
/**********************************/

#include "mbed.h"
#include "ble/BLE.h"

#ifndef NAME
#define NAME "spiffy"
#endif

/* 1) Definición de variables para el "advertising"*/

const static char     DEVICE_NAME[] = "Tx"; // Nombre del dispositivo
const static uint8_t    Datos[] = {"TD"};    // Advertising data (máximo 26 bytes)

/* 2) Función de reconexión */
void disconnectionCallback(const Gap::DisconnectionCallbackParams_t *params)
{
    BLE::Instance().gap().startAdvertising();
}

/* 3) Función de manejo de errores */
void onBleInitError(BLE &ble, ble_error_t error)
{
    /* Para evitar errores de compilación: */
    (void) ble;
    (void) error;

    //En este caso, se ha decidido no implementar manejo de errores
}

/* 4) Función post-inicialización BLE */
void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE&        ble   = params->ble;
    ble_error_t error = params->error;

    /* Comprobación/Manejo de errores */
    if (error != BLE_ERROR_NONE) {
        onBleInitError(ble, error);         // Llamado a la función de manejo de errores (en caso de que existan)
        return;
    }

    /* Comprobación de que el objeto detectado es la instancia por defecto  */
    if(ble.getInstanceID() != BLE::DEFAULT_INSTANCE) {
        return;
    }

    /* Establece el nombre de la característica para el nombre del dispositivo */
    ble.gap().setDeviceName(reinterpret_cast<const uint8_t *>(&DEVICE_NAME));

    /* Reconexión (comenzar "advertising" de nuevo) */
    ble.gap().onDisconnection(disconnectionCallback);

    /* Flags de "advertising" (Bloque 1 de datos) */
    //Activación flags 0x04 y 0x01 (mediante un OR) ->Sólo LE y detectable constantemente
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::BREDR_NOT_SUPPORTED | GapAdvertisingData::LE_GENERAL_DISCOVERABLE );
    ble.gap().setAdvertisingType(GapAdvertisingParams::ADV_CONNECTABLE_UNDIRECTED);

    /* Cabecera de los datos + Datos (Bloque 2 de datos) */
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::MANUFACTURER_SPECIFIC_DATA, Datos, sizeof(Datos));

    /* Añadir el nombre del dispositivo al Advertising Data Payload */
    ble.gap().accumulateAdvertisingPayload(GapAdvertisingData::COMPLETE_LOCAL_NAME, (uint8_t *)DEVICE_NAME, sizeof(DEVICE_NAME));

    /* Frecuencia de "advertising" */
    ble.gap().setAdvertisingInterval(100); /* 100ms */

    /* Comienzo "advertising" */
    ble.gap().setTxPower(0);        // Potencia de emisión
    ble.gap().startAdvertising();   // Inicio "advertising"
}
```

```
74
75  /* 5) Función principal */
76  int main(void)
77  {
78      /* Instanciación objeto BLE */
79      BLE& ble = BLE::Instance(BLE::DEFAULT_INSTANCE);
80
81      /* Inicialización "baselayer" BLE */
82      ble.init(bleInitComplete);
83
84      /* Bucle infinito (atención a eventos BLE) */
85      while (true) {
86          ble.waitForEvent();
87      }
88  }
89
```

A.2. Código fuente receptor (Rx)

```
/************************************/
/* CÓDIGO RECEPTOR SLI BASADO EN BLE */
/************************************/

#include <events/mbed_events.h>
#include <mbed.h>
#include "ble/BLE.h"
#include "ble/Gap.h"

/* 1) Definición de variables */

//Variables comunicación BLE
const static char DEVICE_NAME[] = "Rx";     // Nombre del receptor (Este dispositivo)
static const char PEER_NAME[] = "Tx";       // Nombre del emisor

//---------------------------

/* 2) Función de manejo de errores */
void onBleInitError(BLE &ble, ble_error_t error)
{
    /* Para evitar errores de compilación: */
    (void) ble;
    (void) error;

    // En este caso, se ha decidido no implementar manejo de errores
}

/* 3) "Callback" para comprobar si el elemento detectado es el que se desea localizar*/
void advertisementCallback(const Gap::AdvertisementCallbackParams_t *params)
{
    // Bucle de análisis de los datos recibidos
    for (uint8_t i = 0; i < params->advertisingDataLen; ++i) {

        const uint8_t record_length = params->advertisingData[i];   // Longitud de la muestra
        if (record_length == 0) {
            continue;
        }
        const uint8_t type = params->advertisingData[i + 1];        // Tipo de datos de la muestra
        const uint8_t* value = params->advertisingData + i + 2;     // Valor de la muestra
        const uint8_t value_length = record_length - 1;             // Longitud del valor de la muestra

        if(type == GapAdvertisingData::COMPLETE_LOCAL_NAME) {
            if ((value_length == sizeof(PEER_NAME)) && (memcmp(value, PEER_NAME, value_length) == 0))
            {
                printf("RSSI=%d\r\n",params->rssi;                  // Mostrar por pantalla el "RSSI" medido
                break;
            }
            i += record_length;
        }
    }
}

/* 4) "Callback" post-inicialización BLE*/
void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE&        ble   = params->ble;
    ble_error_t error = params->error;

    if (error != BLE_ERROR_NONE) {
        /* Llamada a la función de error */
        onBleInitError(ble, error);
        return;
    }

    /* Comprobación de que el objeto detectado es la instancia por defecto  */
    if(ble.getInstanceID() != BLE::DEFAULT_INSTANCE) {
        return;
    }

    /* Establece el nombre de la característica para el nombre del dispositivo */
    ble.gap().setDeviceName(reinterpret_cast<const uint8_t *>(&DEVICE_NAME));

    /* Escaneo */
    ble.gap().setScanParams(500, 500);          // Cada 500ms se escanea durante 500ms (escáner continuo)
                                                // Nota: Modificando estos valores se puede reducir notablemente el consumo
    ble.gap().startScan(advertisementCallback); // Se activa el escaneo llamando a la función "advertisementCallback"
                                                //(que comprueba si el elemento detectado es el que se desea)
}

```

```
/* 5) FUNCIÓN PRINCIPAL*/
int main(void)
{
    /* Instanciación de un objeto de la clase "BLE" */
    BLE& ble = BLE::Instance(BLE::DEFAULT_INSTANCE);

    /* Inicialización capas básicas BLE */
    ble.init(bleInitComplete);

    /* Bucle infinito de atención a eventos */
    while (true) {
        ble.waitForEvent();
    }
}
```

A.3. Código fuente receptor con filtro (Rx con filtro)

```
/**********************************/
/* CÓDIGO RECEPTOR SLI BASADO EN BLE*/
/**********************************/

#include <events/mbed_events.h>
#include <mbed.h>
#include <math.h>
#include "ble/BLE.h"
#include "ble/Gap.h"
//#include <fstream>

/* 1) DEFINICIÓN DE VARIABLES*/
//Variables comunicación BLEb
uint16_t intervaloescaneo=1000;             // Intervalo de escaneo (en ms)
const static char DEVICE_NAME[] = "Rx";     // Nombre del receptor (Este dispositivo)
static const char PEER_NAME[] = "SCN";      // Nombre del emisor
//Variables contadoras
int x=0;                                    // Variable contadora (va de 0 a 240)
int xmd=0;                                  // Variable contadora (para la media móvil)
//Variables análisis/filtrado estadístico
int MedMov=0;                               // Media móvil
double DEMov;                                // D.E móvil
int MuesMov[15];                            // Array de muestras móviles
int k,LS,LI;                                // Límites y constante para filtrado estadístico
float Distancia;
//Variables cálculo media
int Media;
int SumMues=0;
int TamMues=0;

//---------------------------

/* 2) Función de manejo de errores */
void onBleInitError(BLE &ble, ble_error_t error)
{
    /* Para evitar errores de compilación: */
    (void) ble;
    (void) error;

    // En este caso, se ha decidido no implementar manejo de errores
}

/* 3) "Callback" para comprobar si el elemento detectado es el que se desea localizar*/
void advertisementCallback(const Gap::AdvertisementCallbackParams_t *params)
{
    for (uint8_t i = 0; i < params->advertisingDataLen; ++i) {

        const uint8_t record_length = params->advertisingData[i];
        if (record_length == 0) {
            continue;
        }
        const uint8_t type = params->advertisingData[i + 1];
        const uint8_t* value = params->advertisingData + i + 2;
        const uint8_t value_length = record_length - 1;

        if(type == GapAdvertisingData::COMPLETE_LOCAL_NAME) {
            if ((value_length == sizeof(PEER_NAME)) && (memcmp(value, PEER_NAME, value_length) == 0)) {
                if (x<240) {
                    int8_t Lectura=params->rssi;

                    /* Limitación de muestras*/
                    if (MedMov!=0 && xmd==14) {
                        k=0.7881;                       //Constante de limitación
                        LS = MedMov-(k*DEMov);          //Asignación dinámica LS
                        LI = MedMov+(k*DEMov);          //Asignación dinámica LI
                    } else {
                        LS=0;
                        LI=0;
                    }

                    /* Cálculos y filtrado estadístico*/
                    if (xmd<14 && ((Lectura<LI && Lectura>LS) || LS==0)) {
                        MuesMov[xmd]=Lectura;    //Concatenación de muestras (móvil)
                        SumMues=SumMues+Lectura;
                        TamMues++;
                    } else if (xmd==14) {
                        xmd=0;                          //Reinicio contador
                        int sum=0;
                        int Dif=0;
                        double SumDif=0;

                        /* Cálculo media móvil */
                        for (int i=0; i<15; i++) {
                            sum=sum+MuesMov[i];
                        }
                        MedMov=sum/15;                  //Media móvil
```

```
                    /* Cálculo D.E móvil */
                    for (int j=0; j<15; j++) {
                        Dif=(MuesMov[j]-MedMov)^2;
                        SumDif=SumDif+Dif;
                    }
                    DEMov=sqrt(SumDif/15);          //D.E móvil

                    for (int k=0; k<15; k++) {
                        MuesMov[k]=0;
                    }
                }
            printf("x=%d  xmd=%d   RSSI=%d\r\n",x,xmd,Lectura);
            x++;
            xmd++;
            }
            else
            {
                Media=SumMues/TamMues;

                int a=-72;          //Lab=-70 |   Casa=-69
                int b=-76;          //Lab=-76 |   Casa=-77.5
                int c=-80;          //Lab=-80 |   Casa=-80
                int d=-83;          //Lab=-83 |   Casa=-83

                if (Media>a){
                    Distancia=0.5;
                }else if(Media<=a && Media>b){
                    Distancia=1;
                }else if(Media<=b && Media>c){
                    Distancia=2;
                }else if(Media<=c && Media>d){
                    Distancia=3;
                }else{
                    Distancia=4;
                }
                x=0;

                printf("MediaRSSI:%d Distancia: %f\r\n",Media,Distancia);
            }
        break;
        }
        i += record_length;
    }
  }
}

/* 4) "Callback" post-inicialización BLE*/
void bleInitComplete(BLE::InitializationCompleteCallbackContext *params)
{
    BLE&        ble   = params->ble;
    ble_error_t error = params->error;

    if (error != BLE_ERROR_NONE) {
        /* Llamada a la función de error */
        onBleInitError(ble, error);
        return;
    }

    /* Comprobación de que el objeto detectado es la instancia por defecto  */
    if(ble.getInstanceID() != BLE::DEFAULT_INSTANCE) {
        return;
    }

    /* Establece el nombre de la característica para el nombre del dispositivo */
    ble.gap().setDeviceName(reinterpret_cast<const uint8_t *>(&DEVICE_NAME));

    /* Escaneo */
    ble.gap().setScanParams(500, 500);          // Cada 500ms se escanea durante 500ms (escáner continuo)
                                                // Nota: Modificando estos valores se puede reducir notablemente el consumo
    ble.gap().startScan(advertisementCallback); // Se activa el escaneo llamando a la función "advertisementCallback"
                                                //(que comprueba si el elemento detectado es el que se desea)
}

/* 5) FUNCIÓN PRINCIPAL*/
int main(void)
{
    /* Instanciación de un objeto de la clase "BLE" */
    BLE& ble = BLE::Instance(BLE::DEFAULT_INSTANCE);

    /* Inicialización capas básicas BLE */
    ble.init(bleInitComplete);

    /* Bucle infinito de atención a eventos */
    while (true) {
        ble.waitForEvent();
    }
}
```

ANEXO B. CÓDIGOS MATLAB

B.1. Programa para representar gráficamente la evolución del RSSI con el tiempo

```
%% SCRIPT DE RECEPCIÓN DE DATOS DE UN PUERTO SERIE %%

% La función de este Script es leer los datos mostrados en el puerto serie
% y representar gráficamente su valor. Inicialmente es empleado para graficar
% medidas de RSSI devueltas por una placa de desarrollo "ST B-L475E-IOT01A"

clc                     %Borrado de la ventana de comandos

% 1) Inicialización de variables
x=0;                    %Variable contadora de muestras
Sumatorio=0;            %Sumatorio de muestras
k=200;                  %Número de muestras para sacar la media
xarray=1:k;
Muestras=[];

SP=serial('COM5');      %Instanciación de un objeto "Puerto serie"
fopen(SP);              %Apertura del puerto serie

% 2) Adquisición de muestras
while(x<k)              %Bucle de adquisición de datos
    Lectura  = fscanf(SP,'%d');         %Muestra actual
    Muestras = [Muestras Lectura];      %Concatenación de muestras
    x=x+1;                              %Paso a la siguiente muestra
end
fclose(SP);             %Cierre del puerto serie

% 3) Representación gráfica de los datos
hold on
plot(xarray,Muestras,'r')           %RSSI
title('EVOLUCIÓN RSSI (d = 3m)')  %Título
xlabel('N° MUESTRA')                %Rótulo eje X
ylabel('RSSI (dBm)')                %Rótulo eje Y
```

B.2. Programa para obtener la media de las muestras

```
%% SCRIPT DE RECEPCIÓN DE DATOS DE UN PUERTO SERIE %%

% La función de este Script es leer los datos mostrados en el puerto serie
% y obtener su media. Inicialmente es empleado para guardar medidas de RSSI
% devueltas por una placa de desarrollo "ST B-L475E-IOT01A"

clc                     %Borrado de la ventana de comandos

% 1) Inicialización de variables
x=0;                    %Variable contadora de muestras
Sumatorio=0;            %Sumatorio de muestras
k=20;                   %Número de muestras para sacar la media

Muestras=[];            %Conjunto de muestras
SP=serial('COM5');      %Creación de un objeto de la clase "Puerto serie"
fopen(SP);              %Apertura del puerto serie

% 2) Adquisición de muestras
while(x<k)              %Bucle de adquisición de datos
    Lectura  = fscanf(SP,'%d');         %Muestra actual
    Muestras = [Muestras Lectura];      %Concatenación de muestras
    x=x+1;                              %Paso a la siguiente muestra
end

% 3) Obtención de la media
for i=1:x
   Sumatorio=Sumatorio+Muestras(i);     %Cálculo del suamtorio de muestras
end
Media=Sumatorio/x;      %Cálculo de la media

fclose(SP);             %Cierre del puerto serie
```

B.3. Programa para obtener un histograma de las muestras

```
%% SCRIPT DE RECEPCIÓN DE DATOS DE UN PUERTO SERIE %%

% La función de este Script es leer los datos mostrados en el puerto serie
% y generar un histograma. Inicialmente es empleado para estudiar medidas de
% RSSI devueltas por una placa de desarrollo "ST B-L475E-IOT01A"

clc                     %Borrado de la ventana de comandos

% 1) Inicialización de variables
x=0;                    %Variable contadora de muestras
Sumatorio=0;            %Sumatorio de muestras

Muestras=[];            %Conjunto de muestras
Agrup=zeros(1,24);
ValRSSI=[-67 -68 -69 -70 -71 -72 -73 -74 -75 -76 -77 -78 -79 -80 -81 -82 -83 -84 -85 -86 -87 -88 -89 -90];            %Valores de RSSI posibles
SP=serial('COM5');      %Instanciación de un objeto "Puerto serie"
fopen(SP);              %Apertura del puerto serie

% 2) Adquisición de muestras
while(x<200)           %Bucle infinito de adquisición de datos
    Lectura  = fscanf(SP,'%d');         %Muestra actual
    Muestras = [Muestras Lectura];      %Concatenación de muestras
    x=x+1;                              %Paso a la siguiente muestra
end

% 3) Agrupamiento de muestras
for i=1:x
   for j=1:24
       if Muestras(i) == ValRSSI(j)
           Agrup(j)=Agrup(j)+1;
       end
   end
end

% 4) Obtención del histograma
hold on
bar(ValRSSI,Agrup);
plot(ValRSSI,Agrup,'r')
title('HISTOGRAMA RSSI')            %Título
xlabel('RSSI (dBm)')                %Rótulo eje X
ylabel('FRECUENCIA (Muestras)')     %Rótulo eje Y

% 5) Análisis estadístico
Media=mean(Muestras);      %Media
Mediana=median(Muestras);   %Mediana
Moda=mode(Muestras);        %Moda
DE=std(Muestras);           %Desviación Estándar
Rango=range(Muestras);      %Rango de las muestras
```

B.4. Filtro estadístico RSSI

```
%% FILTRO ESTADÍSTICO RSSI BLE %%

% La función de este Script es leer los datos mostrados en el puerto serie
% y obtener su media. Inicialmente es empleado para filtrar medidas de RSSI
% devueltas por una placa de desarrollo "ST B-L475E-IOT01A"

clc                     %Borrado de la ventana de comandos

%Inicialización de variables contadoras
x=0;                    %Contador de muestras 1 (1 muestra -> ++)
Sumatorio=0;            %Variable sumatorio de muestras
xmd=0;                  %Contador de muestras 2 (15 muestras -> ++)
fallos=0;               %Contador de muestras desechadas
y=0;

%Inicialización del resto de variables
Raw=[];                 %Conjunto de muestras sin filtrar
Muestras=[];            %Conjunto de muestras filtradas
MuesMov=[];             %Muestras para la media móvil
ArrMed=[];              %Array de valores de la media (recalculada cada muestra)
MedMov=0;               %Inicialización de la media móvil
ArrMedMov=[];           %Array de valores de la media móvil (recalculada cada 15 muestras)
ArrDEMov=[];            %Array de valores de la desv.estándar móvil (recalculada cada 15 muestras)

SP=serial('COM5');      %Instanciación de un objeto "Puerto serie"
fopen(SP);              %Apertura del puerto serie

% 1) Adquisición de muestras
while(x<240)            %Bucle de adquisición de datos
    Lectura  = fscanf(SP,'%d');         %Muestra actual
    Raw = [Raw Lectura];                %Muestras sin filtrar

    x=x+1;                              %Paso a la siguiente muestra
    Med=mean(Muestras);                 %Cálculo constante de la media
    ArrMed=[ArrMed Med];                %Array de medidas de la media

    if (MedMov~=0 && xmd==14)
        k=1.28;                         %Constante de limitación
        LS = MedMov-(k*DEMov);          %Asignación dinámica LS
        LI = MedMov+(k*DEMov);          %Asignación dinámica LI
    else
        LS=0;                           %Anulación de LS (subconjunto inic)
        LI=0;                           %Anulación de LS (subconjunto inic)
    end

    if (xmd<14 && ((Lectura<LI && Lectura>LS) || LS==0))
        xmd=xmd+1;
        MuesMov = [MuesMov Lectura];    %Concatenación de muestras (móvil)
    elseif xmd==14
        xmd=0;                          %Reinicio contador
        MedMov=mean(MuesMov);           %Media móvil
        ArrMedMov=[ArrMedMov MedMov];

        DEMov=std(MuesMov);             %Desviación estándar móvil
        ArrDEMov=[ArrDEMov DEMov];

        if ((Lectura<LI && Lectura>LS) || LS==LI==0)
            MuesMov=Lectura;
        else
            MuesMov=[];
        end
    end

    if ((Lectura<LI && Lectura>LS) || LS==0)
        Muestras(y+1) = Lectura;        %Concatenación de muestras
        y=y+1;                          %Siguiente posición del vector
    else
        fallos=fallos+1;
    end
end

% 2) Cierre del puerto serie
fclose(SP);             %Cierre del puerto serie

% 3) Representación gráfica de los datos
xarray=size(Muestras);

hold on
plot(1:xarray(2),Muestras,'r')                      %RSSI
plot(1:x,ArrMed,'g')                                %Media
xmov=15:15:240;
plot(xmov,ArrMedMov,'bo')                           %Media móvil
%plot(xmov,ArrDEMov,'go')                           %Desviación estándar móvil
title('FILTRADO RSSI')                              %Título
xlabel('Nº MUESTRA')                                %Rótulo eje X
ylabel('RSSI (dBm)')                                %Rótulo eje Y
legend('RSSI puro','Media','Media móvil','Location','southwest')   %Leyenda
```

ANEXO C. MARCO REGULADOR

El sistema desarrollado está sujeto a diversas normativas y estándares técnicos de varios ámbitos. En el caso de las directivas, aquellas que sean aplicables dependen del lugar en el que se emplee, por lo que en este apartado se asume que el SLI sólo será empleado en países de la Unión Europea.

En concreto, el hardware empleado debe respetar las siguientes directivas europeas:

- **Directiva de Baja Tensión (Low Voltage Directive; LVD) 2014/35/EU** [12]**.** Afecta a todos los dispositivos cuya alimentación esté entre 50 y 1000 V_{AC} o 75 y 1500 V_{DC}.

- **Directiva de Compatibilidad Electromagnética (Electromagnetic Compatibility; EMC) 2014/30/EU** [13]**.** Como su propio nombre indica, se emplea para garantizar que los equipos comercializados respetan la compatibilidad electromagnética.

- **Directiva de Equipos de Radio (Radio Equipment Directive; RED) 2014/53/EU** [14]**.** Afecta a todos los dispositivos que operen con ondas de radio (sean emisores o receptores).

- **Directiva de restricción de sustancias peligrosas v2 (Restriction of Hazardous Substances v2; RoHS2) 2011/65/EU** [15]**.** Asociada a dispositivos eléctricos/electrónicos, se aplica a los alimentados por debajo de los 1000 V_{AC}/1500V_{DC}. Restringe el uso de sustancias nocivas.

En relación al software empleado, el lenguaje de programación C++ se asocia con el estándar ISO C++. Este ha evolucionado con el paso del tiempo hasta llegar a su versión actual: ISO/IEC 14882:2017 [16] (comúnmente conocida como C++17). Cabe destacar que pronto será sustituida por la versión C++20, actualmente en desarrollo.

Por último, respecto a la tecnología de comunicación inalámbrica empleada, es necesario recalcar que, al ser un subconjunto del Bluetooth clásico, el BLE se basa en el estándar IEEE 802.15.1 [17]. Además, al funcionar a una frecuencia de 2,4 GHZ, se encuentra dentro de una de las bandas de radio ISM (*Industrial Scientific and Medical*).

ANEXO D. ENTORNO SOCIOECONÓMICO

D.1. Análisis del impacto socioeconómico

El impacto socioeconómico de este proyecto puede ser de magnitudes muy elevadas si se explota su potencial al máximo, pero, como ocurre en la mayoría de los casos, esto dependerá del grado de expansión que tenga.

Como se mencionó en la introducción del documento, existen muchos campos de aplicación para los SLI, pero entre ellos tiene especial peso el sector de la salud.

Dedicar esfuerzos al desarrollo y la implementación de sistemas de localización en entornos sanitarios aportaría beneficios enormes tanto al desarrollo de nuestra sociedad como a nuestra economía. No sólo se reducirían los costes de personal de forma masiva (no se necesitarían tantos asistentes para vigilar a los pacientes), sino que los recursos destinados a este fin anteriormente podrían aprovecharse para mejorar el servicio en otros aspectos (contratando a personal dedicado a otras labores, adquiriendo equipos mejores o aumentando el presupuesto de investigación entre otras cosas). Además, la calidad de vida de los pacientes y su tranquilidad dentro de hospitales o residencias aumentaría notablemente, puesto que el control de su ubicación permitiría a los sanitarios reaccionar mucho más rápido ante una emergencia.

Tal es el margen de crecimiento que tienen los SLI en este ámbito que muchas empresas de ingeniería ya se han puesto manos a la obra para entrar en el mercado cuanto antes, y el número de residencias de ancianos que demanda soluciones como esta aumenta cada vez más.

Por si esto fuera poco, conseguir que estos SLI se basen en BLE sin perder exactitud a la hora de medir distancias aumenta las razones de estas empresas para apostar por el sector, dado que emplear este tipo de tecnologías reduce notablemente los costes totales (mucho menor consumo y hardware muy barato).

Esto hace que el trabajo recogido en este documento gane especial importancia, demostrando que, tanto directa como indirectamente, afecta de forma positiva al entorno socioeconómico

D.2. Planificación y presupuesto

Las diferentes fases del proyecto pueden apreciarse en la tabla 4, pudiéndose agrupar todas las tareas llevadas a cabo en seis grupos:

1. Labor de documentación.
2. Familiarización con Mbed OS.
3. Diseño y desarrollo del SLI.
4. Diseño de la etapa de filtrado.
5. Modificaciones del SLI.
6. Redacción de la memoria.

En total, el tiempo dedicado por el alumno a desarrollarlas supera por poco las 300 horas, pero como el peso del Trabajo de Fin de Grado es de 12 ECTS (European Credit Transfer and Accumulation) y, en la UC3M (Universidad Carlos III de Madrid), este equivale a 25 horas de trabajo, se ha tomado ese valor como referencia para los cálculos posteriores. Así mismo, se ha considerado que el tutor ha dedicado unas 30 horas a asesorar al alumno y apoyarle con el desarrollo del proyecto

Además, para elaborar el presupuesto del proyecto (Tabla 5) se ha hecho uso de las siguientes estimaciones por falta de información fehaciente:

- Se estima que el salario del tutor, por ser catedrático de la UC3M, es de 3000€ mensuales aproximadamente. Suponiendo 20 días laborables al mes y una jornada laboral de 8 horas, se considera que su salario por hora es de 18,75€.

- Considerando al alumno un ingeniero junior, se supone que su salario mensual es de 2000€ aproximadamente. Con el mismo horario laboral que el tutor, se obtiene que su salario por hora es de 12,50€.

Cualquier otro factor a tener en cuenta se encuentra explicado en la descripción de los elementos del presupuesto.

Tabla 4: Distribución de tareas y dedicación temporal

	TAREA	FECHA INICIO	FECHA FIN	DURACIÓN (DÍAS)	DURACIÓN (HORAS)	FEBRERO				MARZO				ABRIL				MAYO				JUNIO	
						S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2	S3	S4	S1	S2
1	**Labor de documentación**	01/02/2019	28/02/2019	28	40																		
1.1	Documentación SLI (teoría)	01/02/2019	05/02/2019	5	7																		
1.2	Documentación proyectos previos	06/02/2019	10/02/2019	5	7																		
1.3	Documentación investigaciones actuales	11/02/2019	16/02/2019	6	9																		
1.4	Documentación BLE	17/02/2019	23/02/2019	7	12																		
1.5	Documentación B-L475E-IOT01A	24/02/2019	24/02/2019	1	2																		
1.6	Documentación Mbed OS	25/02/2019	28/02/2019	4	3																		
2	**Familiarización con Mbed OS**	**01/03/2019**	**17/03/2019**	**17**	**50**																		
2.1	Análisis programas de ejemplo (básicos)	01/03/2019	02/03/2019	2	10																		
2.2	Análisis programas de ejemplo (BLE)	03/03/2019	08/03/2019	6	15																		
2.3	Análisis librerías mbed.h y gap.h	09/03/2019	17/03/2019	9	25																		
3	**Diseño y desarrollo del SLI**	**18/03/2019**	**06/04/2019**	**20**	**55**																		
3.1	Diseño SLI	18/03/2019	19/03/2019	2	5																		
3.1	Programación Tx	20/03/2019	25/03/2019	6	15																		
3.2	Programación Rx	26/03/2019	04/04/2019	10	25																		
3.3	Medidas iniciales (varios entornos)	05/04/2019	06/04/2019	2	10																		
4	**Diseño etapa de filtrado**	**07/04/2019**	**04/05/2019**	**28**	**85**																		
4.1	Diseño programas MATLAB (gráficos)	07/04/2019	09/04/2019	3	10																		
4.2	Análisis medidas iniciales	10/04/2019	10/04/2019	1	5																		
4.3	Análisis comportamiento media RSSI	11/04/2019	11/04/2019	1	5																		
4.4	Diseño teórico filtro	12/04/2019	17/04/2019	6	20																		
4.5	Implementación filtro (MATLAB)	18/04/2019	01/05/2019	14	35																		
4.6	Medidas postfiltrado (varios entornos)	02/05/2019	04/05/2019	3	10																		
5	**Modificaciones del SLI**	**05/05/2019**	**18/05/2019**	**14**	**40**																		
5.1	Implementación filtro (Rx)	05/05/2019	08/05/2019	4	5																		
5.2	Implementación obtención distancia (Rx)	09/05/2019	10/05/2019	2	5																		
5.3	Calibración SLI (varios entornos)	11/05/2019	13/05/2019	3	10																		
5.4	Pruebas medición distancia (varios entornos)	14/05/2019	16/05/2019	3	10																		
5.5	Extracción de conclusiones	17/05/2019	18/05/2019	2	10																		
6	**Redacción de la memoria**	**19/05/2019**	**16/06/2019**	**29**	**32**																		
6.1	Redacción inicial	19/05/2019	13/06/2019	26	27																		
6.2	Corrección (tras feedback del tutor)	15/06/2019	15/06/2019	1	3																		
6.3	Revisión	16/06/2019	16/06/2019	1	2																		

Tabla 5: Presupuesto TFG

PRESUPUESTO PROYECTO TFG DIEGO VENTURA HENRÍQUEZ

Código	Elemento/Descripción	Unidad	Cantidad	Precio unitario	Presupuesto
A	**Materiales**				
A1	**Placa B-L475E-IOT01A** Placa de desarrollo empleada para implementar tanto el receptor como el emisor.	**ud.**	**2**	**48,14 €**	**96,28 €**
A2	**Ordenador** Ordenador portátil adquirido a un precio de 1100€ . Se considera amortizable en 4 años con unas 5 horas de uso diarias.	**h.**	**300**	**0,16 €**	**48,00 €**
A3	**Cable USB (A - Micro B)** Cable de USB A a microUSB B empleado para conectar las placas con el ordenador y el cargador.	**ud.**	**2**	**5,00 €**	**10,00 €**
A4	**Cargador (Vo=5V)** Cargador de 5V de tensión de salida empleado para alimentar el emisor.	**ud.**	**1**	**5,00 €**	**5,00 €**
	Subtotal Materiales				**159,28 €**
B	**Software**				
B1	**Licencia estudiante MATLAB** Licencia MATLAB Student (sin Simulink)	**ud.**	**1**	**35,00 €**	**35,00 €**
	Subtotal Software				**35,00 €**
C	**Costes de personal**				
C1	**Salario jefe de proyecto** Salario correspondiente al jefe de proyecto (tutor) por sus horas de trabajo.	**h.**	**30**	**18,75 €**	**562,50 €**
C2	**Salario ingeniero junior** Salario correspondiente al ingeniero junior (alumno) por sus horas de trabajo.	**h.**	**300**	**12,75 €**	**3.825,00 €**
	Subtotal Costes de personal				**4.387,50 €**
	TOTAL PROYECTO				**4.581,78 €**