UNIVERSIDAD DE VALLADOLID

E.T.S.I DE TELECOMUNICACIÓN

TRABAJO FINAL DE GRADO

GRADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Rastreador de dispositivos Bluetooth de baja energía

Autor: Alejandro Cabrero Domínguez

Tutor: Juan Blas Prieto

20 de septiembre de 2024

TÍTULO:	Rastreador de dispositivos Bluetooth de baja energía
AUTOR:	Alejandro Cabrero Domínguez
TUTOR:	Juan Blas Prieto
DEPARTAMENTO:	Teoría de la Señal y Comunicaciones e Ingeniería Telemática

TRIBUNAL

PRESIDENTE:	Evaristo José Abril Domingo
VOCAL:	Juan Carlos Aguado Manzano
SECRETARIO:	Juan Blas Prieto
SUPLENTE:	Jaime Gómez Gil
SUPLENTE:	Ignacio de Miguel Jiménez
FECHA:	27 de septiembre de 2024
CALIFICACIÓN:	

Resumen

El presente Trabajo de Fin de Grado se enmarca en el ámbito de las comunicaciones inalámbricas. En concreto está centrado en la tecnología Bluetooth Low Energy (BLE). Como parte fundamental del trabajo se hace un análisis de las especificaciones y de la torre de protocolos que emplea esta tecnología. Asimismo, se ilustra el procedimiento para configurar y hacer funcionar el kit de desarrollo BLE modelo nRF52840 fabricado por la empresa Nordic Semiconductor. Como ejemplo de aplicación del kit de desarrollo se utiliza el sistema operativo en tiempo real Zephyr para implementar una aplicación que permite rastrear la distancia entre dos dispositivos BLE empleando las propiedades físicas de las ondas electromagnéticas que intercambian los dispositivos BLE. Entre tales propiedades cabe señalar los tiempos de propagación y las variaciones de fase en distintas frecuencias. El código fuente empleado se basa en algoritmos propietarios del fabricante. Tras un estudio de campo en distintos escenarios reales de interés que proporcionan distintas condiciones de propagación se analizan y se comparan los distintos algoritmos. Como parte de este estudio de campo, se ha automatizado el procesado de las muestras obtenidas empleando código python, para analizar la precisión de los diferentes métodos de estimación de distancia analizados. Los resultados muestran un margen de error significativo, especialmente en algunos métodos, lo que sugiere la necesidad de optimización y una elección apropiada de método dependiendo del escenario de propagación de las señales.

PALABRAS CLAVE: Bluetooth Low Energy (BLE), Estimación de distancias, nrf_dm, Comunicaciones inalámbricas, Procesamiento de datos, Python.

Abstract

This Final Degree Project focuses on wireless communication technologies, specifically on using Bluetooth Low Energy (BLE) for distance estimation between devices. The main goal is to provide a theoretical and practical foundation for future projects in this field. A comprehensive technical explanation of BLE is included, alongside a tutorial for installing and configuring the Nordic Semiconductor development environment and using nRF52840 devices with the nrf_dm program.

A Python script was developed to process data obtained through nrf_dm, facilitating the graphical interpretation of measurements. This script, included in the project, is readily applicable for similar studies. Additionally, measurements were conducted in various environments, analyzing the accuracy of the different distance estimation methods in nrf_dm. Results revealed a significant error margin, especially in specific methods, highlighting the need for optimization.

The project proposes future research directions, such as improving the selection of the nrf_dm "Best" method or adjusting this selection based on environmental conditions. This work not only provides fundamental knowledge on BLE but also practical tools and key insights for improving distance estimation in future research.

KEYWORDS: Bluetooth Low Energy (BLE), Distance estimation, nrf_dm, Wireless communications, Data processing, Python.

Agradecimientos

En primer lugar quería dar las gracias a mi tutor de este proyecto, Juan, que ha tenido la paciencia, la serenidad y la seriedad necesarias para que este periodo se termine de forma exitosa.

Quiero hacer mención especial a mi familia, mis padres y mi hermana, quienes han confiado en mi potencial desde el primer momento, apoyándome en los momentos más duros la carrera, especialmente en estos últimos años. Este proyecto es por vosotros.

No puedo dejar de mencionar a mi círculo de amigos. Muchas horas en la biblioteca y numerosas experiencias compartidas han hecho el viaje más ameno, pero siempre al pie del cañón.

Y a ti, la persona que me ha acompañado incondicionalmente a lo largo de este proceso, en los buenos y en los malos momentos. Juntos hemos celebrado los éxitos y superado los desafíos, lo que ha hecho que este viaje sea aún más especial. Me siento muy afortunado de haberte encontrado; siempre formarás parte de mí. Ana.

Índice general

CAPÍTULO 1

INTRODUCCIÓN

1.1 Ámbito del tfg

El presente Trabajo de Fin de Grado se enmarca dentro del campo de las tecnologías de comunicación inalámbrica de bajo consumo energético, en particular Bluetooth Low Energy (BLE). BLE es una tecnología ampliamente utilizada en dispositivos IoT debido a su eficiencia energética y capacidad de operar en entornos de corto alcance. Este proyecto se centra en el uso de BLE para la estimación de distancias entre dispositivos, empleando el método Multi-Channel Phase Difference (MCPD) del programa nrf_dm, desarrollado por Nordic Semiconductor.

El ámbito de este trabajo abarca desde la configuración y utilización de hardware y software específico de BLE, hasta la recolección y análisis de datos en diversos escenarios. Se han realizado pruebas experimentales en entornos cerrados y abiertos, con distancias que varían de 1,5 a 20 metros, para evaluar el rendimiento del MCPD en diferentes condiciones.

Además de su enfoque técnico en la medición de distancias, el TFG busca sentar las bases para futuros proyectos, proporcionando tanto conocimientos teóricos como prácticos sobre BLE. La combinación de teoría, herramientas de desarrollo y análisis de datos posiciona este trabajo como un recurso valioso para estudiantes e investigadores interesados en la implementación de soluciones basadas en BLE.

1.2 Objetivos del tfg

El objetivo principal de este Trabajo de Fin de Grado es ofrecer una base sólida que pueda ser utilizada como referencia en futuros proyectos relacionados con la tecnología Bluetooth Low Energy (BLE) y la estimación de distancias. Para ello, el trabajo se plantea los siguientes objetivos específicos:

- **Proveer una base teórica sobre BLE:** El TFG busca ofrecer un compendio técnico detallado sobre los fundamentos de BLE. Esto permitirá a otros estudiantes o investigadores familiarizarse con esta tecnología, dotándolos de los conocimientos esenciales para iniciar trabajos en este campo.
- **Instrucciones detalladas para configurar el entorno Nordic:** Se desarrollará una guía práctica que describa el proceso completo de instalación y configuración del entorno de desarrollo proporcionado por Nordic Semiconductor. Esto incluye la descarga e instalación de herramientas necesarias para trabajar con dispositivos nRF52840 y programas como nrf_dm, permitiendo una iniciación eficiente en el uso de estos dispositivos.
- **Desarrollo de un código para procesamiento de datos:** Otro objetivo importante es la creación de un programa en Python que facilite el procesamiento de los datos obtenidos a través del método Multi-Channel Phase Difference (MCPD) de la libreía nrf_dm. El código permitirá convertir las muestras en gráficos interpretables de forma inmediata, lo cual agiliza el análisis de los resultados.
- **Análisis preliminar de las mediciones obtenidas:** Finalmente, el trabajo incluirá un análisis exploratorio de las muestras recopiladas en diferentes escenarios y distancias. Aunque no pretende ser exhaustivo, este análisis ofrecerá conclusiones iniciales y orientará futuras investigaciones sobre cómo optimizar la precisión de las mediciones de distancia en entornos reales.

1.3 Estructura del documento

El presente Trabajo de Fin de Grado está organizado en los siguientes capítulos, cada uno enfocado en una parte específica del desarrollo y análisis de la tecnología BLE aplicada a la estimación de distancias:

- **Capítulo 2: Marco teórico**

  Este capítulo proporciona una base técnica sólida sobre Bluetooth Low Energy (BLE). Se detalla el funcionamiento de BLE, sus características clave y su aplicabilidad en el ámbito de las comunicaciones inalámbricas de bajo consumo. El objetivo es ofrecer un marco teórico que sirva como fundamento para el trabajo experimental y de desarrollo posterior.

- **Capítulo 3: Configuración del entorno y selección de hardware y software**

  En este capítulo se describe el proceso de instalación y configuración del entorno de desarrollo de Nordic Semiconductor, necesario para trabajar con los dispositivos nRF52840. Este proceso no se limita únicamente a instalar el software, sino que implica la configuración detallada de varias herramientas, incluyendo la vinculación de dependencias, la compilación del código fuente y la realización de llamadas a librerías y funciones específicas del sistema operativo Zephyr. Se requiere configurar correctamente el compilador y gestionar los proyectos a través de herramientas como CMake y West, que requieren un entendimiento técnico para integrar el hardware BLE con el software de medición de distancias. Además, se expone la selección del hardware BLE utilizado, evaluando las ventajas y desventajas de cada dispositivo, y se justifica la elección de la librería de estimación de distancias nrf_dm.

- **Capítulo 4: Captación y procesamiento de muestras**

  Este capítulo aborda el proceso experimental de recolección de datos en diferentes entornos. Se describe el formato en que se presentan las muestras obtenidas, proporcionando contexto para la necesidad de un procesamiento adicional. Además, se explica el desarrollo de un programa en Python que permite procesar las muestras de manera eficiente, facilitando su análisis y visualización en gráficos. El código está incluido en el TFG, y puede ser utilizado de forma directa en un notebook de Python.

- **Capítulo 5: Métodos de estimación de distancia y análisis de resultados**

  En este capítulo se presentan los diferentes métodos que utiliza la librería nrf_dm para estimar la distancia entre dispositivos BLE, explicando en qué se basan y cuál sería su comportamiento esperado en diferentes escenarios. Posteriormente, utilizando el programa en Python desarrollado, se realiza el análisis de las muestras recolectadas, comenzando con el cálculo del offset y continuando con la separación de resultados por distancia. Se incluye un análisis de cómo se comporta cada método en los distintos entornos.

- **Capítulo 6: Conclusiones**

  Finalmente, en este capítulo se recogen las conclusiones más importantes del trabajo. Se exponen los hallazgos principales y se discute el potencial del TFG como base para futuros proyectos relacionados con la estimación de distancias mediante BLE.

CAPÍTULO 2

Marco Teórico

El siguiente marco teórico se basa en la información proporcionada por la Nordic Semiconductor Academy, específicamente en su curso sobre los fundamentos de Bluetooth Low Energy (BLE), que ofrece una visión técnica integral sobre esta tecnología [1].

2.1 ¿Qué es Bluetooth?

Bluetooth es una especificación industrial para redes inalámbricas de área personal (WPAN) creado por Bluetooth SIG (*Special Interest Group*). Permite transmitir voz y datos entre diferentes dispositivos mediante un enlace por radiofrecuencia en la banda de 2.4 GHz. Está diseñado para dispositivos de baja potencia y corto alcance, empleando transceptores baratos. En la tabla 2.1 se muestra una clasificación de los dispositivos Bluetooth atendiendo a su potencia de transmisión.

Tabla 2.1: Clasificación de dispositivos Bluetooth según su potencia de transmisión

Clase	Potencia máx. (mW)	Potencia máx. (dBm)	Alcance (m)
Clase 1	100	20	≈ 100
Clase 2	2,5	4	$\approx 5 - 10$
Clase 3	1	0	≈ 1
Clase 4	0,5	-3	$\approx 0,5$

El alcance siempre dependerá de varios factores relacionados con la configuración de software y hardware de los dispositivos utilizados, así como del entorno específico donde operan los dispositivos. Por lo tanto, es muy difícil tener una estimación precisa y generalizada del alcance. Enlace a una calculadora de alcance

2.2 Evolución de Bluetooth

En sus comienzos, allá por 1994, Bluetooth tenía una tasa de transmisión de 720 kb s^{-1}. Tras 3 décadas de evolución se ha llegado a velocidades de transmisión de hasta 50 Mb s^{-1}. El alcance de los dispositivos también ha ido incrementándose con el tiempo, pasando de un metro a más de 1 km. En la tabla 2.2 se incluye un resumen del rendimiento de las versiones principales.

2.2.1 Bluetooth Clásico

Es la versión habitual de Bluetooth para uso personal. Incluye dispositivos como auriculares inalámbricos. Es ideal para aplicaciones como la transmisión de música

Tabla 2.2: Clasificación de dispositivos Bluetooth según su velocidad de transmisión

Versión	Ancho de banda Mb s^{-1}
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	32 Mbit/s
Versión 5	50 Mbit/s

debido a su alta velocidad de transferencia de datos y a que es fácil de usar. En dispositivos como teléfonos inteligentes y altavoces inalámbricos requiere recargas periódicas de batería, pero esto no suele ser un problema en este tipo de aplicaciones.

2.2.2 Bluetooth de Baja Energía (BLE)

Introducido a partir de 2010 en la versión 4.0 de la Especificación Principal de Bluetooth por el Bluetooth SIG (*Bluetooth Special Interest Group*). Tiene su propia pila de protocolos. Está diseñado para dispositivos de bajo consumo de energía, como dispositivos ponibles o aplicaciones masivas de IoT (*Internet Of Things*). Se logra ahorrar batería reduciendo la velocidad de transferencia de datos y optimizando el uso de energía. De modo que BLE es más adecuado para dispositivos que necesitan operar con una potencia mínima y enviar únicamente pequeñas ráfagas de datos. Además de la diferencia en el consumo de energía y la velocidad de transferencia de datos, Bluetooth LE también difiere en otras áreas, como las topologías de red admitidas y los tipos de nodos.

2.3 Pila de protocolo BLE

2.3.1 Host

- **GATT (*Generic Attribute Profile*):** Define los subprocedimientos necesarios para utilizar la capa ATT.

- **GAP (*Generic Access Profile*):** Se encarga de controlar las conexiones y los anuncios en BLE. GAP es lo que permite que tu dispositivo sea público hacia el exterior y determina como dos dispositivos pueden interactuar entre ellos.
- **ATT (*Attribute Protocol*):** Permite que un dispositivo exponga ciertos datos a otro dispositivo.
- **SMP (*Security Manager Protocol*):** Define y proporciona métodos para que la comunicación sea segura.
- **L2CAP (*Logical Link Control & Adaptation Protocol*):** Proporciona servicios de encapsulamiento de datos a las capas superiores.

2.3.2 Controller:

- **LL (*Link Layer*):** Administra el estado de la radio (*Standby, Advertising, Scanning, Initiating, Connection*).
- **PHY (*Physical Layer*):** Determina la modulación de las ondas de radio y cómo se transmiten y reciben.

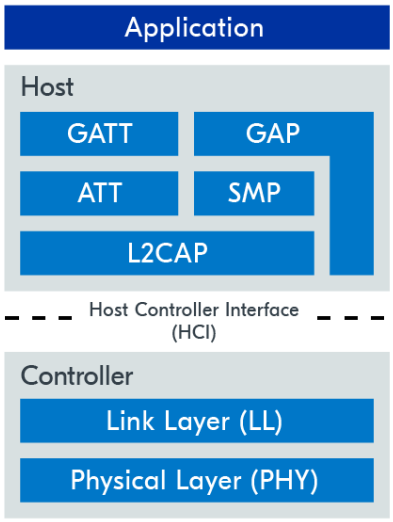

Figura 2.1: Arquitectura de BLE

2.4 Roles y topologías de dispositivos

2.4.1 Estilos de conexión:

El protocolo Bluetooth LE admite dos estilos de comunicación diferentes:

- **Connection-oriented communication:** Cuando hay una conexión dedicada entre dispositivos, formando una comunicación bidireccional.
- **Broadcast communication:** Cuando los dispositivos se comunican sin establecer una conexión primero, transmitiendo paquetes de datos a todos los dispositivos dentro del alcance.

2.4.2 Roles de dispositivo:

La capa GAP define roles específicos para los dispositivos en una red Bluetooth LE. Estos roles determinan cómo anuncia su presencia el dispositivo o cómo escanea y se conecta a otros nodos. Para que dos dispositivos Bluetooth LE se conecten entre sí, uno de ellos necesita anunciar su presencia y disposición para conectarse, mientras que el otro escaneará dichos dispositivos. Los roles que define la capa GAP son:

- **Central:** Un rol de dispositivo que escanea y establece conexiones con dispositivos periféricos. El dispositivo central puede enviar solicitudes de conexión a más de un dispositivo periférico simultáneamente y asume el rol de anfitrión en esta conexión. Dado que el central actúa como anfitrión, es responsable de tareas típicas de anfitrión como la gestión de conexiones y gran parte del procesamiento de datos.
- **Peripheral:** Un rol de dispositivo que anuncia y acepta conexiones que solicitan un rol central. Los roles periféricos también pueden aceptar solicitudes de conexión de otros roles centrales reiniciando el proceso de publicidad después de que se haya establecido una conexión. Los roles periféricos generalmente consumen menos energía que uno central.
- **Broadcaster:** Un tipo especial de rol periférico que transmite regularmente paquetes de *advertising* con datos sin aceptar solicitudes de conexión. Estos datos son accesibles para cualquier dispositivo que esté a la escucha.

- **Observer:** Un tipo especial de dispositivo central que escucha los paquetes de *advertising* sin iniciar una conexión.

2.4.3 Tipos de topologías:

La topología está definida en la capa GAP

- **Broadcast topology:** La transferencia de datos ocurre sin que los dispositivos establezcan una conexión. La ventaja de una topología de difusión es que no hay límite en cuántos dispositivos se pueden transmitir datos. Esto también es mucho más eficiente en cuanto a energía que la comunicación orientada a la conexión. Sin embargo, debido a los datos limitados disponibles en los paquetes de publicidad, el rendimiento es limitado. Además, no hay confirmación de los dispositivos receptores.

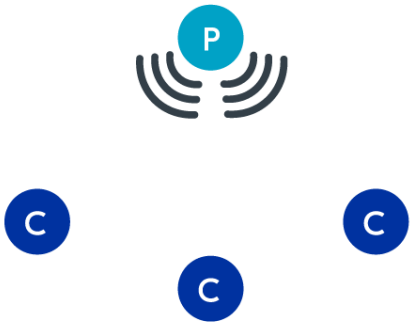

Figura 2.2: Broadcast-Topo

- **Connected topology:** Una topología de red conectada establece una conexión antes de que ocurra la transferencia de datos. A diferencia de la topología de difusión, la comunicación es ahora bidireccional. La ventaja de una topología conectada es el aumento del rendimiento que viene con el establecimiento de un enlace directo antes de la comunicación. Con la introducción de Periodic Advertising with Responses(PAwR) en Bluetooth 5.4, la comunicación bidireccional en modo sin conexión es posible.

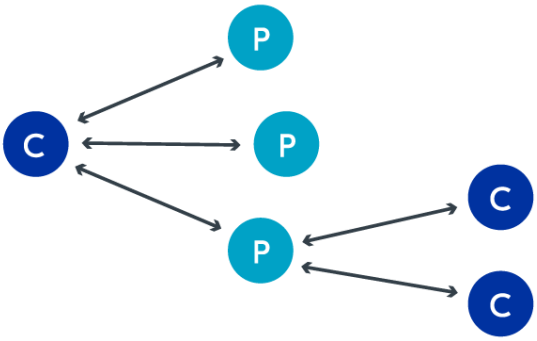

Figura 2.3: Connect-Topo

- **Multi-role topology:** Un solo dispositivo también puede operar en múltiples roles diferentes simultáneamente. Por ejemplo, el mismo dispositivo puede actuar como un periférico en un entorno y como un central en otro.

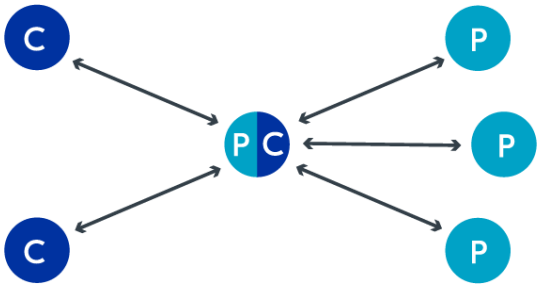

Figura 2.4: Multi-Topo

2.5 Representación e intercambio de datos

Las capas ATT y GATT se ocupan de la fase después de que se ha establecido una conexión, a diferencia de la capa GAP que se encarga del proceso de publicidad que ocurre antes de que se establezca una conexión.

2.5.1 The Attribute Protocol

El Protocolo de Atributos (ATT) es la base sobre la cual se transmite, recibe y maneja datos en la fase de conexión de los dispositivos Bluetooth LE. Se basa en una arquitectura cliente-servidor donde el servidor posee los datos y puede enviarlos directamente al cliente o el cliente puede solicitar los datos al servidor.

Los roles de cliente y servidor definidos en esta capa se asignan de forma independiente a los roles de periférico y central definidos en la capa GAP. Por lo tanto, un central puede ser tanto un cliente como un servidor, al igual que un periférico. Todo esto depende del caso de uso de la aplicación y la naturaleza de los datos que se transfieren. Estos roles son utilizados por la capa GATT, por lo que a menudo se les denomina servidor GATT y cliente GATT.

- **GATT server:** Dispositivo que almacena datos y proporciona métodos para que el cliente GATT acceda a los datos.
- **GATT client:** Dispositivo que accede a los datos en el servidor GATT.
- **Attribute:** Un formato de representación de datos estandarizado definido por el protocolo ATT. La capa ATT define una estructura de datos llamada atributo, que es utilizada por el servidor GATT para almacenar datos. El servidor puede contener varios atributos diferentes al mismo tiempo.

2.5.2 The Generic Attribute Profile

El Perfil de Atributos Genéricos (GATT) se sitúa directamente sobre la capa ATT, y se basa en ella clasificando jerárquicamente atributos en perfiles, servicios y características. La capa GATT utiliza estos conceptos para regular la transferencia de datos entre dispositivos Bluetooth LE. Lista completa de perfiles GATT definidos por el Bluetooth SIG se puede encontrar aquí.

2.6 Capa física: modos radio

En la parte inferior de la pila de Bluetooth LE se encuentra la capa física (PHY). PHY se refiere a las especificaciones de radio de la capa física que rigen el funcionamiento de la radio Bluetooth LE. Esta capa define diferentes esquemas de modulación y codificación adoptados por los transmisores de radio Bluetooth LE que afectan a cosas como el rendimiento de la radio. A su vez, esto cambia el consumo de batería del dispositivo o el alcance de la conexión.

- **1M PHY:** 1M PHY, o 1 Megabit PHY, es el PHY clásico compatible con todos los dispositivos Bluetooth LE. 1M PHY utiliza 1 megabit por segundo. Cuando se inicia una conexión entre dos dispositivos Bluetooth LE, este es el modo que se utilizará inicialmente. Luego, los pares pueden solicitar otro modo si ambos dispositivos lo admiten.
- **2M PHY:** El 2 Megabit PHY es un nuevo modo introducido en Bluetooth v5.0. Duplica efectivamente la velocidad de datos a 2 megabits por segundo, o 2 Mbps. Dado que los datos se transmiten a una velocidad de datos más alta (más rápida), la radio necesita permanecer encendida durante menos tiempo, lo que reduce el consumo de batería. La desventaja es la disminución de la sensibilidad del receptor, lo que se traduce en un rango de comunicación menor.
- **Coded PHY:** Mientras que el 2M PHY existe para usuarios dispuestos a sacrificar alcance por una mayor velocidad de datos, el PHY codificado fue introducido para servir a aplicaciones donde los usuarios pueden lograr un rango de comunicación más largo al sacrificar la velocidad de datos. El PHY codificado utiliza esquemas de codificación para corregir errores de paquetes de manera más efectiva, lo que también significa que un solo bit es representado por más de 1 símbolo. El PHY codificado utiliza 2 modos, S=2 y S=8:
  - **S=2:** 2 símbolos representan 1 bit, por lo tanto, la velocidad de datos es de 500 kbps.
  - **S=8:** 8 símbolos para representar un bit y la velocidad de datos se convierte en 125 kbps.

2.7 Estados de la capa de enlace:

2.7.1 Standby

En este estado, el dispositivo BLE no realiza ninguna actividad de comunicación. Es un modo de espera en el que no envía ni recibe paquetes. Los dispositivos entran en standby para conservar energía cuando no están implicados en operaciones de publicidad o conexión.

2.7.2 Advertising

Advertising es el acto de publicitar datos. Cuando un dispositivo Bluetooth LE se encuentra en un estado de publicidad, envía paquetes de publicidad para anunciar su presencia y potencialmente conectarse a otro dispositivo. Estos paquetes de publicidad se envían periódicamente a intervalos de publicidad. Cuanto menor sea el intervalo de publicidad, más frecuentemente se envían los paquetes de publicidad, y en consecuencia, se consume más energía. Por lo tanto, el compromiso aquí es entre el consumo de energía y qué tan rápido los paquetes de publicidad del anunciante serán recibidos por un escáner, comúnmente conocido como capacidad de descubrimiento. Para evitar colisiones de paquetes, se agrega una demora aleatoria de 0-10 ms antes de cada paquete de publicidad. Esto asegura que los dispositivos con el mismo intervalo de publicidad no terminen con colisiones de paquetes de publicidad todo el tiempo.

- **Advertising intervals:** El intervalo en el que se envía un paquete de publicidad. En el rango de 20 ms a 10.24 s, con un incremento de paso de 0.625 ms.
- **Advertising channels:** Los dispositivos Bluetooth LE se comunican a través de 40 canales de frecuencia diferentes. Estos canales se dividen en tres canales de publicidad primarios y 37 canales de publicidad secundarios. Los canales de publicidad primarios son los canales principalmente utilizados con fines de publicidad. Los canales de publicidad secundarios a veces también pueden ser utilizados con fines de publicidad, pero principalmente se utilizan para la transferencia de datos después de establecer una conexión. Para garantizar un cierto grado de redundancia, los paquetes de publicidad se envían en los tres canales primarios de publicidad, los canales 37, 38 y 39. Simultáneamente, un dispositivo de escaneo escaneará estos tres canales para buscar dispositivos

publicitarios. Dado que los paquetes de publicidad son esenciales para establecer la conectividad, los canales primarios de publicidad se eligen cuidadosamente. Los canales 37, 38 y 39, a pesar de ser números consecutivos, en realidad no son canales contiguos, como se puede ver en la imagen. La separación entre los tres canales sirve para evitar la interferencia de bandas adyacentes. Además, estos tres canales específicos sufren menos ruido de otras tecnologías que utilizan la banda ISM, como Wi-Fi. Con esta función, los paquetes de publicidad transmitidos en los canales de publicidad primarios apuntan a información complementaria que se está publicitando en los canales de publicidad secundarios.

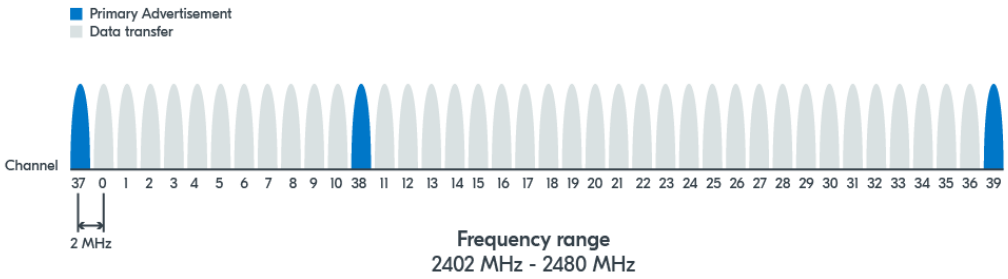

Figura 2.5: Channels

- **Tipos definidos de Advertising:** Hay 4 tipos principales de Advertising para cubrir en las legacy advertisements, así como un quinto que se utiliza en las respuestas de escaneo. Hay muchas formas diferentes en las que un periférico puede publicitar. Se puede hacer una clasificación antendiendo a las siguientes características:
  - *Connectable* vs. *non-connectable*: Determina si el central puede conectarse al periférico o no.
  - *Scannable* vs. *non-scannable*: Determina si el periférico acepta solicitudes de escaneo de un escáner.
  - *Directed* vs. *undirected*: Determina si los paquetes de publicidad están dirigidos a un escáner específico o no.

  Los tipos de Advertising son:

  1. **ADV_IND**: *Scannable and Connectable*. Connectable undirected advertising. Este es el tipo de publicidad más común. Si un periférico utiliza este tipo de

publicidad, significa que es tanto escaneable como conectable. Esto significa que el periférico está anunciando su presencia y permite que el central envíe una solicitud de escaneo y responderá con una respuesta de escaneo (por lo tanto, escaneable), lo que se seguirá de establecer una conexión (por lo tanto, conectable).

2. **ADV_DIRECT_IND** *(Directed connectable)*: Connectable directed advertising. Este tipo de publicidad se utiliza para publicidad dirigida donde el anunciante no acepta solicitudes de escaneo. Es dirigido, conectable pero no escaneable. Esto puede ser utilizado en casos donde el anunciante ya conoce al escáner y solo desea volver a conectarse rápidamente. Un buen ejemplo para este escenario es un ratón Bluetooth que ha perdido la conexión con la PC y solo quiere volver a conectarse nuevamente. En este caso, no es necesario aceptar solicitudes de escaneo y es más rápido enviar un paquete de publicidad dirigida para acortar el proceso de conexión.

3. **ADV_SCAN_IND** *(Non-connectable and scannable)*: Scannable undirected advertising. Un anunciante que utiliza este tipo de publicidad solo aceptará solicitudes de escaneo, pero no permitirá establecer una conexión con él (por lo tanto, no conectable).

4. **ADV_NONCONN_IND** *(*Non-connectable and non-scannable): Es del tipo Non-connectable undirected advertising. Este tipo de publicidad no acepta solicitudes de escaneo ni permite establecer conexiones. Un caso de uso típico para este tipo de publicidad es un beacon, donde el dispositivo no necesita cambiar la radio al modo de receptor ya que no permite recibir ningún dato, lo que a su vez reduce el consumo de batería.

Tabla 2.3: Tipos definidos de Advertising

	Connectable	Scannable	Directed
ADV_IND	x	x	
ADV_DIRECT_IND	x		x
ADV_SCAN_IND		x	
ADV_NONCONN_IND			

2.7.3 Scanning

El proceso de escuchar paquetes publicitarios.

- **Scan interval:** Similar a un intervalo de publicidad, el intervalo de escaneo se refiere a la pausa temporal en la búsqueda de paquetes de publicidad.
- **Scan window:** La ventana de escaneo se refiere al tiempo que el escáner pasa buscando paquetes, lo que en la práctica representa el ciclo de trabajo en el que el dispositivo está escaneando vs no escaneando durante cada intervalo de escaneo. Ambos van desde 2,5 ms hasta 10,24 segundos con un incremento de paso de 0,625 ms.
- **Scan request:** Cuando un periférico está anunciando, un central también puede optar por enviar una solicitud de escaneo al periférico, solicitando información adicional que no está incluida en los paquetes de publicidad. El advertiser tiene que entrar en un período RX para esperar la scan request. Este período RX también se puede usar para recibir la Connect request. Como se puede ver en la figura 2.6.
- **Scan response:** Si la solicitud de escaneo es aceptada, el periférico responderá con lo que se llama una respuesta de escaneo, también transmitida a través de los tres canales primarios de publicidad. Esta es una forma para que los periféricos envíen datos adicionales sin tener que establecer primero una conexión con el central. El periférico también puede optar por enviar una respuesta de escaneo vacía si no tiene más información adicional que proporcionar. Otra forma de aumentar la cantidad de datos que un periférico puede publicitar a la vez es con una función llamada extended advertising.

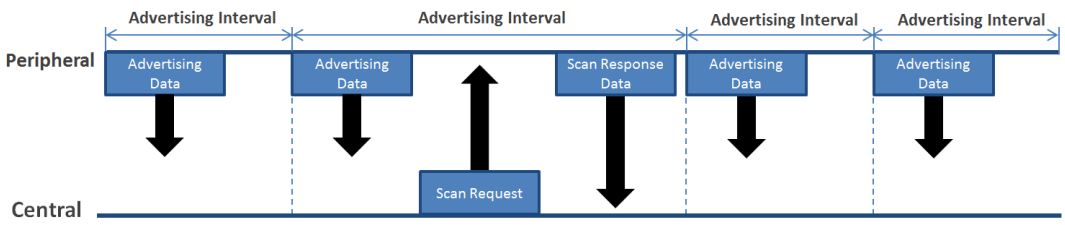

Figura 2.6: Scan-Request

2.7.4 Initiating:

En este estado, el dispositivo actúa como un iniciador de conexión. Escanea los paquetes de publicidad enviados por otros dispositivos y, cuando encuentra uno con el que quiere conectarse, envía una solicitud de conexión. Este proceso es el paso previo a establecer una conexión directa entre dos dispositivos.

2.7.5 Connection

Una vez que se ha enviado y aceptado la solicitud de conexión, ambos dispositivos entran en el estado de conexión. En este estado, intercambian datos de manera sincronizada mediante intervalos de conexión establecidos. La comunicación continua hasta que uno de los dispositivos decide terminar la conexión o se pierde el enlace por alguna razón (por ejemplo, si los dispositivos salen del rango de alcance).

2.8 Bluetooth address:

Cada dispositivo Bluetooth LE está identificado por una dirección única de 48 bits. Las direcciones Bluetooth se clasifican como direcciones públicas o direcciones aleatorias. Las direcciones aleatorias pueden clasificarse aún más en estáticas o privadas, dependiendo de si cambian o no. Además, las direcciones privadas pueden ser resolubles o no resolubles.

Todos los dispositivos Bluetooth LE deben tener una dirección pública o una dirección aleatoria estática. Opcionalmente, también pueden tener direcciones privadas resolubles o no resolubles, generalmente añadidas por razones de privacidad. La dirección pública asignada a un dispositivo se extrae del mismo pool de direcciones IEEE que las direcciones MAC (por ejemplo, para Ethernet, Wi-Fi), y por lo tanto también se conoce comúnmente como dirección MAC de Bluetooth.

2.8.1 Public address

Una dirección pública es una dirección fija que se programa en el dispositivo en el momento de fabricación. Debe ser registrada en la autoridad de registro de IEEE, y

es única en todo el mundo para ese dispositivo y no se puede borrar. Hay una tarifa asociada con la obtención de este tipo de dirección.

2.8.2 Random address

Una dirección aleatoria es mucho más comúnmente utilizada, ya que no requiere registro con el IEEE y puede ser configurada manualmente por el usuario. Se programa dentro del dispositivo o se crea durante la ejecución. Puede ser estática o privada.

- **Random static address:** Una dirección aleatoria estática puede asignarse y luego permanecer fija durante toda la vida útil del dispositivo. Puede ser modificada en el arranque, pero no durante la ejecución. Esta es una alternativa de bajo costo a una dirección pública porque no requiere registro. Como se mencionó anteriormente, todos los dispositivos Bluetooth LE deben utilizar una dirección pública o una dirección aleatoria estática, siendo esta última mucho más común.
- **Random private address:** Una dirección privada puede ser utilizada además de la dirección pública o aleatoria estática cuando un dispositivo desea proteger su privacidad. Esta es una dirección que cambia periódicamente y se utiliza para ocultar la identidad del dispositivo y disuadir el rastreo de dispositivos. Una dirección privada aleatoria puede ser resoluble o no resoluble.

  1. **Resolvable random private address** Una dirección privada aleatoria resoluble, fiel a su nombre, es resoluble tal como lo indican las intenciones, ya que los oyentes previstos tienen una clave precompartida mediante la cual pueden descifrar la nueva dirección cada vez que cambia. La clave precompartida es la Clave de Resolución de Identidad (IRK) y se utiliza tanto para generar como para resolver la dirección aleatoria. La dirección aleatoria básicamente solo es utilizada por el par para poder resolver la dirección real del dispositivo Bluetooth LE, que sigue siendo ya sea la dirección pública o la dirección aleatoria estática. La IRK permite al par traducir la dirección privada aleatoria en la dirección Bluetooth LE real del dispositivo.
  2. **Non-resolvable random private address** Una dirección privada no resoluble no es resoluble por otros dispositivos y solo está destinada a evitar el seguimiento. Este tipo de dirección no se usa comúnmente.

2.9 Advertisement packet:

El paquete BLE se muestra a continuación en la figura 2.7, con la parte principal yendo a lo que se llama la Unidad de Datos del Protocolo (PDU). La PDU consiste en una PDU de publicidad (a veces llamada PDU de canal de publicidad) o una PDU de datos (a veces llamada PDU de canal de datos), dependiendo de si el paquete BLE se utiliza para publicidad o transmisión de datos.

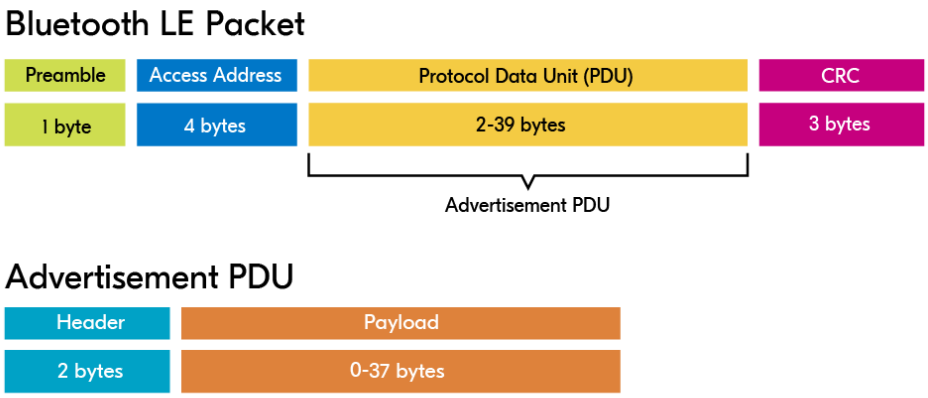

Figura 2.7: Paquete BLE y Advertisement

Como podemos ver en la figura 2.8, la PDU de publicidad consta de un encabezado y una carga útil. La parte del encabezado de la PDU de publicidad consta de:

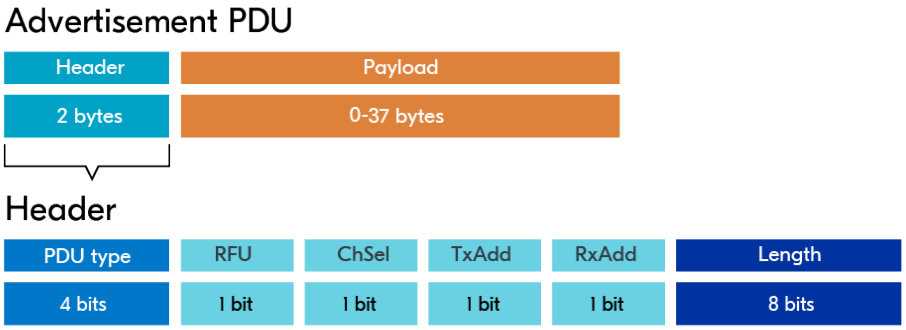

Figura 2.8: Advertisement PDU Header

- **PDU type** Determina el tipo de publicidad que discutimos en los tipos de publicidad, como ADV_IND.
- **RFU:** Reservado para uso futuro.
- **ChSel:** Se establece en 1 si se admite el Algoritmo de Selección de Canal LE #2.
- **TxAdd (Tx Address):** 0 o 1, dependiendo de si la dirección del transmisor es pública o aleatoria.
- **RxAdd (Rx Address):** 0 o 1, dependiendo de si la dirección del receptor es pública o aleatoria.
- **Length:** La longitud de la carga útil. La carga útil de la PDU de publicidad se divide en dos secciones, donde los primeros 6 bytes representan la dirección del anunciante (AdvA) y el resto se destina a los datos de publicidad reales (AdvData).

  La parte de payload consta de:

- **AdvA:** Dirección Bluetooth del dispositivo anunciante.

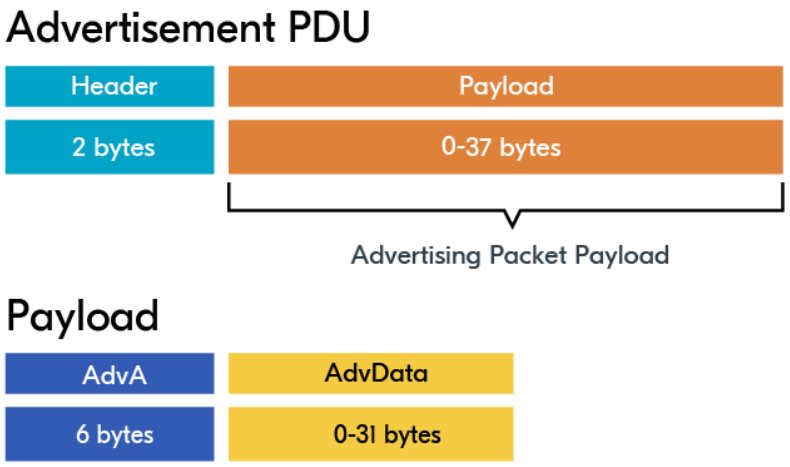

Figura 2.9: Advertisement PDU Payload

- **AdvData:** El paquete de datos de publicidad está compuesto por múltiples estructuras llamadas estructuras de datos de publicidad (AD structures). Cada estructura de AD tiene un campo de longitud, un campo para especificar el tipo (AD Type) y un campo para los datos reales (AD Data). Es importante tener en cuenta que el tipo de AD más común tiene una longitud de 1 byte.

  A continuación, se muestran algunos tipos comúnmente utilizados:

  - **Complete local name (BT_DATA_NAME_COMPLETE):** Este es simplemente el nombre del dispositivo que ve el usuario humano al escanear dispositivos cercanos (a través de un teléfono inteligente, por ejemplo).
  - **Shortened local name (BT_DATA_NAME_SHORTENED):** Una versión más corta del nombre local completo.
  - **Uniform Resource Identifier (BT_DATA_URI):** Se utiliza para publicitar una URI como direcciones de sitios web (URL).
  - **Service UUID:** El Identificador Universalmente Único del Servicio es un número universalmente único para un servicio específico. Puede ayudar a los escáneres a identificar dispositivos interesantes para conectarse. Hay diferentes opciones disponibles aquí.

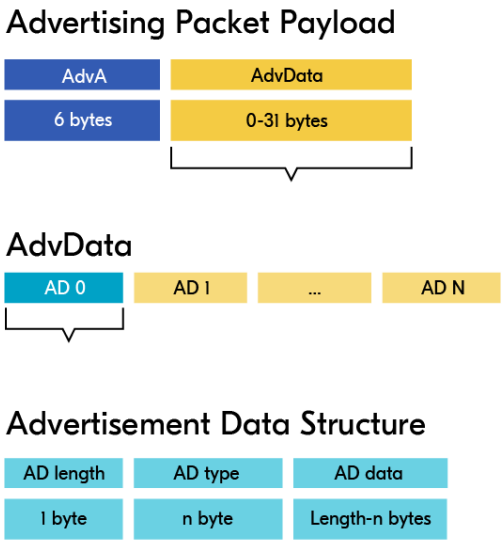

Figura 2.10: AdvData

– **Manufacturer Specific Data (BT_DATA_MANUFACTURER_DATA):** Este es un tipo popular que permite a las empresas definir sus propios datos de publicidad personalizados, como en el caso de iBeacon.

– **Flags:** Variables de 1 bit que pueden señalar una cierta propiedad o modo operativo de ese dispositivo. Son banderas de un bit encapsuladas en un byte, lo que significa que hay hasta 8 banderas que se pueden establecer.

1. BT_LE_AD_LIMITED: Establece el LE Limited Discoverable Mode, utilizado con publicidad conectable para indicar a un central que el dispositivo está disponible solo durante un cierto período de tiempo antes de que la publicidad caduque.

2. BT_LE_AD_GENERAL: Establece el LE General Discoverable Mode, utilizado en publicidad conectable para indicar que la publicidad está disponible durante un largo período de tiempo (timeout = 0). Tanto BT_LE_AD_LIMITED como BT_LE_AD_GENERAL están destinados a un dispositivo en un rol periférico.

3. BT_LE_AD_NO_BREDR: Indica que Bluetooth clásico (BR/EDR) no está soportado.

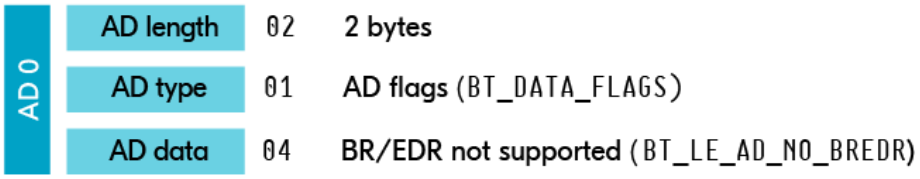

Figura 2.11: Estructura de datos en funcion de bandera BT_LE_AD_NO_BREDR

2.10 Proceso de Conexión:

2.10.1 Conexión

Establecer una conexión requiere dos dispositivos, uno actuando como periférico que está anunciando, y otro actuando como central que está escaneando. Cuando un dispositivo central detecta un paquete de publicidad de un dispositivo periférico, puede iniciar una conexión. Por lo general, esto implica escanear el contenido del paquete de publicidad y luego decidir si iniciar o no una conexión en función de eso. Cuando el central envía una solicitud de conexión, el periférico y el central han establecido un canal de conexión bidireccional (orientado a la conexión).

Como podemos ver en la figura anterior, un periférico que envía publicidades que son conectables siempre tendrá una ventana de recepción (RX) corta después de cada publicidad, que se utiliza para escuchar las solicitudes de conexión entrantes. Lo llamamos una solicitud de conexión, pero en realidad, el periférico no puede elegir si aceptar o rechazar la solicitud de conexión. Siempre tiene que aceptar la solicitud de conexión, a menos que esté utilizando un filtro de lista de aceptación. Más tarde, en cualquier momento, puede optar por desconectarse del central si no desea permanecer conectado.

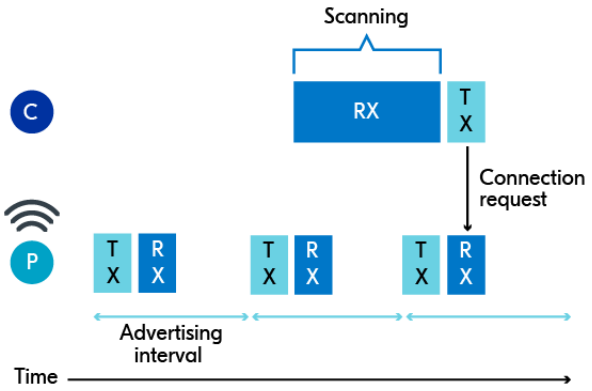

Figura 2.12: Proceso de conexión

2.10.2 Durante la conexión

Después de que el periférico recibe con éxito el paquete de solicitud de conexión, los dos dispositivos están en una conexión. Cuando se inicia una conexión, los dispositivos dejarán de usar los canales de publicidad (canales 37, 38 y 39), y comenzarán a usar los canales de datos (canales 0 a 36). Para reducir la interferencia y mejorar el rendimiento durante una conexión, Bluetooth LE utiliza el salto de canal, lo que significa que el canal utilizado para la transmisión de datos se cambia con frecuencia. De esta manera, si están ubicados en un entorno que tiene mucho ruido en algunos canales, los mensajes se retransmitirán en otro canal en el siguiente intervalo de conexión. Para garantizar la integridad de los datos, todos los paquetes transmitidos por Bluetooth LE se reintentarán infinitamente hasta que se reciba un acuse de recibo o se termine la conexión.

La naturaleza de una conexión Bluetooth LE es un factor principal en cómo los dispositivos logran un consumo de energía tan bajo. En una conexión, ambos dispositivos pasan la mayor parte de su tiempo en reposo. Para lograr esto, acuerdan con qué frecuencia se despertarán para hablar. De lo contrario, apagan la radio, establecen un temporizador y se van a dormir. El tiempo en el que acuerdan dormir se conoce como el intervalo de conexión, y se establece en la conexión inicial, mientras que el evento de conexión ocurre en cada intervalo de conexión cuando se despiertan para hablar.

La siguiente figura muestra cómo se ve típicamente una conexión. Tanto el dispositivo central como el periférico se despiertan en cada intervalo de conexión para los eventos de conexión y transmiten datos.

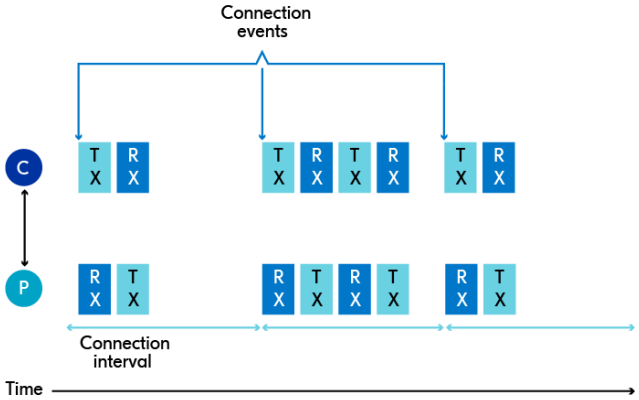

Figura 2.13: Proceso de conexión

El intervalo de conexión se establece inicialmente por el dispositivo central en el paquete de solicitud de conexión, pero puede ser modificado más tarde durante la conexión. Los dos dispositivos pueden enviar muchos paquetes en cada intervalo de conexión si necesitan enviar mucha información, pero cuando dejan de enviar datos, deben esperar al próximo evento de conexión para enviar más datos. Incluso si no hay datos útiles que enviar, los dispositivos necesitan enviar paquetes vacíos para sincronizar sus relojes. Si deseas enviar más datos de los que hay tiempo para transmitir en un intervalo de conexión, se dividirá en varios intervalos de conexión.

2.10.3 Desconexión

Cuando dos dispositivos están conectados, permanecerán conectados indefinidamente si no ocurre nada. Hay dos formas en que una conexión puede ser terminada, lo que significa que los dispositivos se desconectan:

- **Disconnected by application:** Si cualquiera de los dispositivos así lo desea, puede enviar un paquete de terminación que desconectará el dispositivo. Esto puede hacerse, por ejemplo, porque un dispositivo ya no desea estar conectado al otro dispositivo, pero también ocurrirá si hay algo mal con la conexión. Por ejemplo, si un dispositivo al que estás conectado está reclamando ser un dispositivo conectado previamente, pero no puede firmar los paquetes con las claves de cifrado correctas. El paquete de terminación contendrá un campo llamado "motivo de desconexión"que indicará algo sobre por qué los dispositivos se desconectaron.

Por ejemplo, si fue el usuario quien quiso desconectar, o si hubo un problema con la pila de software.

- **Disconnected by supervision timeout:** La otra razón por la que un dispositivo puede desconectarse es si deja de responder a los paquetes. Puede haber varias razones para esto. Ya sea que la aplicación en el dispositivo conectado se haya bloqueado y reiniciado (lo cual no es demasiado inusual, especialmente durante la fase de desarrollo), que el dispositivo conectado se haya quedado sin batería, o que el dispositivo conectado esté fuera del alcance de radio. El tiempo que transcurre antes de que la conexión se agote se establece mediante el connection supervision timeout parameter.

2.10.4 Parámetros de conexión

Cuando un dispositivo periférico y uno central establecen una conexión, se intercambian un conjunto de parámetros de conexión. Algunos de ellos tienen un valor de inicio estándar, para garantizar la compatibilidad hacia atrás, mientras que otros son dictados por el dispositivo central y se incluyen en el paquete de solicitud de conexión. El intervalo de conexión y el tiempo de supervisión de conexión son establecidos por el central en el paquete de solicitud de conexión, además de la latencia del periférico. La latencia del periférico permite que este omita despertarse para eventos de conexión si no tiene datos que enviar. El modo de radio (1M, 2M o PHY codificado) se establece en 1M de forma predeterminada para garantizar la compatibilidad hacia atrás, pero puede cambiarse durante la conexión. La longitud de datos y la MTU (Unidad Máxima de Transferencia) también se establecen para garantizar la compatibilidad hacia atrás.

- **El intervalo de conexión (Connection interval):** Un dispositivo Bluetooth LE pasa la mayor parte de su tiempo "dormido"(de ahí el "Bajo Consumo de Energía.[en] el nombre). En una conexión, esto se logra al acordar un intervalo de conexión que indica con qué frecuencia los dispositivos se comunicarán entre sí. Cuando terminan de comunicarse, apagarán la radio, configurarán un temporizador y entrarán en modo de espera, y cuando el temporizador se agote, ambos se despertarán y volverán a comunicarse. La implementación de esto la maneja la pila de Bluetooth LE, pero depende de tu aplicación decidir con qué frecuencia deseas que los dispositivos se comuniquen estableciendo el intervalo de conexión.

- **Tiempo de supervisión (Peripheral latency):** Cuando dos dispositivos están conectados, acuerdan un parámetro que determina cuánto tiempo debe transcurrir desde que se recibió correctamente el último paquete hasta que los dispositivos consideren que la conexión se ha perdido. Esto se llama el tiempo de supervisión. Por lo tanto, si uno de los dispositivos se apaga inesperadamente, se queda sin batería, o si los dispositivos están fuera del alcance de la radio, entonces este es el tiempo que transcurre entre recibir correctamente el último paquete antes de que se considere perdida la conexión.

- **Latencia de los periféricos (Peripheral latency):** La peripheral latency permite que el periférico omita despertarse durante un cierto número de eventos de conexión si no tiene ningún dato que enviar. Por lo general, el intervalo de conexión es un compromiso estricto entre el consumo de energía y la baja latencia o retraso en la comunicación. Si deseas reducir la latencia pero mantener un bajo consumo de energía, puedes utilizar la peripheral latency. Esto es particularmente útil en aplicaciones de HID (Dispositivos de Interfaz Humana), como aplicaciones de ratón y teclado de computadora, que generalmente no tienen ningún dato que enviar, pero cuando tienen datos para enviar, queremos tener una latencia muy baja. Utilizando la opción de peripheral latency, podemos mantener una baja latencia pero reducir el consumo de energía al permanecer inactivos durante varios intervalos de conexión.

- **Modo de radio PHY (PHY radio mode):** El Bluetooth LE normal (PHY 1M) transmite a 1 Mbps. Sin embargo, en Bluetooth 5.0, se introdujeron dos modos de radio adicionales: alta velocidad (PHY 2M) y largo alcance (PHY codificado). Esto nos da dos opciones más.

  1. Primero, podemos aumentar el esquema de modulación para usar 2 Mbps para tasas de transmisión más altas. Esto significa que puedes transferir los datos más rápido y volver a dormir más rápido para conservar más energía, o puedes usar ese tiempo extra para enviar aún más datos, prácticamente duplicando el rendimiento de una conexión Bluetooth LE. Sin embargo, esto conlleva el costo de un rango ligeramente más corto.

  2. La otra opción es utilizar PHY codificado, lo que resulta en un aumento significativo en el rango, pero a costa de un menor rendimiento.

- **Longitud de los datos y MTU (Data length and MTU).** La longitud de los datos y la MTU (Unidad Máxima de Transferencia) son dos parámetros diferentes, pero

suelen estar relacionados. El MTU es el número de bytes que pueden ser enviados en una operación GATT (por ejemplo, una operación de envío), mientras que la longitud de los datos es el número de bytes que pueden ser enviados en un paquete Bluetooth LE. El MTU tiene un valor predeterminado de 23 bytes, y la longitud de los datos tiene un valor predeterminado de 27 bytes. Cuando el MTU es mayor que la longitud de los datos, como cuando el MTU es de 140 bytes mientras que la longitud de los datos es de 27 bytes, los datos se segmentarán en trozos del tamaño de la longitud de los datos. Esto significa que, para tu aplicación, parece que se está enviando un mensaje, pero en el aire, los datos en realidad se dividen en segmentos más pequeños.

Idealmente, deseas que todos tus datos sean enviados en un solo paquete para reducir el tiempo que lleva enviar los datos. Por lo tanto, en Bluetooth 4.2, se introdujo la Extensión de Longitud de Datos (DLE, por sus siglas en inglés) para permitir que la longitud de los datos se aumente desde los 27 bytes predeterminados hasta un máximo de 251 bytes. Agrupar todo también reduce el número de bytes que necesitas transmitir por el aire, ya que cada paquete incluye una cabecera de 3 bytes. Esto ahorra tanto tiempo como energía, y a su vez permite un mayor rendimiento en tu conexión Bluetooth LE. La relación entre la longitud de los datos (Data Length) y la MTU (Unidad Máxima de Transferencia) no es de uno a uno. En el aire, la longitud de los datos puede ser de hasta 251 bytes, pero la carga útil real que puedes enviar es de un máximo de 244 bytes. Esto se debe a que la carga útil PDU de datos de 251 bytes necesita una cabecera L2CAP de 4 bytes y una cabecera de atributos de 3 bytes. Esto deja 251 - 4 - 3 = 244 bytes que puedes llenar realmente con datos de carga útil.

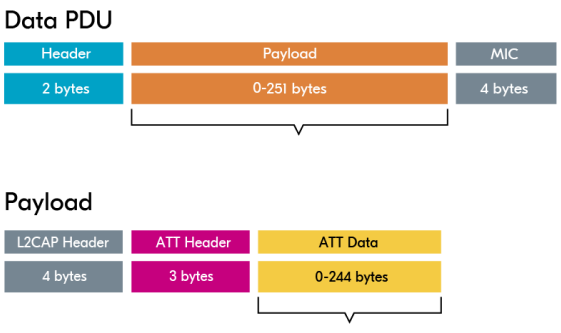

Figura 2.14: PDU de datos

A continuación se muestra una figura que muestra cómo se ve enviar un mensaje con 40 bytes antes y después de cambiar la longitud de datos predeterminada. Es evidente que enviar todos los datos en un solo paquete conduce a un menor tiempo de encendido de la radio.

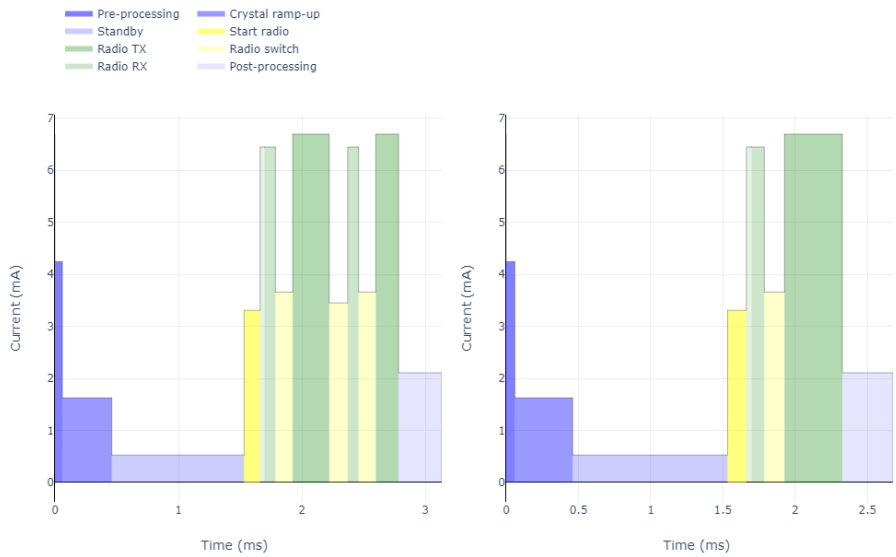

Figura 2.15: Comparación entre longitud de datos predeterminada y ampliada

2.10.5 Actualizar los parámetros de conexión

El intervalo de conexión, el tiempo de supervisión y la latencia periférica son dictados por el central, pero el periférico puede solicitar cambios. Sin embargo, siempre es el central quien tiene la última palabra con estas solicitudes. Por lo tanto, en el caso de que el central sea tu teléfono, es el sistema operativo que se ejecuta en el teléfono el que decide si aceptar o rechazar los nuevos parámetros en la solicitud de parámetros de conexión.En cuanto al modo de radio PHY, la longitud de los datos y la MTU, estos no pueden ser elegidos únicamente por el central. Dado que la capacidad de cambiar estos parámetros se introdujo en versiones posteriores de la Especificación de Bluetooth, siempre se establecen en sus valores predeterminados cuando se establece una conexión por primera vez. Cuando se establece la conexión por primera vez, cualquiera de los dispositivos puede solicitar actualizar estos parámetros con nuevos valores. El otro dispositivo luego enviará sus valores admitidos o indicará que no admite la actualización de uno o más de esos parámetros.Tomando la longitud de los

datos como ejemplo, esta siempre es de 27 bytes cuando se establece la conexión por primera vez. Luego, supongamos que el periférico desea actualizar esto a 200 bytes y envía una solicitud para hacerlo. El central puede responder con un mensaje diciendo que puede manejar 180 bytes, y luego acordarán establecer la longitud de los datos en 180 bytes. El valor predeterminado para el modo de radio PHY es 1M, y la MTU predeterminada es 23.

2.11 Operaciones de la capa GATT

Como hemos visto, la capa GATT define servicios y características, compuestos por atributos, que se almacenan en el servidor GATT. En este apartado se discute sobre el intercambio de datos en Bluetooth LE, que se refiere a las operaciones que se ejecutan entre el servidor y el cliente para conocer los atributos e intercambiar sus valores de acuerdo con los permisos de los atributos. El servidor puede enviar datos directamente al cliente o el cliente puede solicitar los datos del servidor. Pero para que el cliente sepa qué solicitar al servidor, necesita conocer qué servicios y características ofrece el servidor GATT. Por lo tanto, el cliente realizará un descubrimiento de servicios al inicio de la conexión, para aprender sobre los servicios y características del servidor, antes de realizar cualquier operación para acceder a ellos.

2.11.1 Service discovery

Descubrimiento de servicios. El proceso en el cual un cliente GATT descubre los servicios y características en una tabla de atributos alojada por un servidor GATT.

2.11.2 Data access:

Acceso a datos. Recuerda que la comunicación aquí se basa en una arquitectura cliente-servidor, donde el servidor almacena los datos y puede enviarlos directamente al cliente o el cliente puede solicitarlos del servidor. Por lo tanto, las operaciones GATT se clasifican en operaciones iniciadas por el cliente y operaciones iniciadas por el servidor.

- **Operaciones iniciadas por el cliente** Las operaciones iniciadas por el cliente son operaciones GATT en las que el cliente solicita datos del servidor GATT. El cliente puede solicitar leer o escribir en un atributo, y en el caso de escribir, puede elegir si desea recibir una confirmación del servidor.

  1. **Read** Si un cliente desea leer un cierto valor almacenado en un atributo en un servidor GATT, el cliente envía una solicitud de lectura al servidor. El servidor responde devolviendo el valor del atributo.

  2. **Write** Si el cliente desea escribir un cierto valor en un atributo, envía una solicitud de escritura y proporciona datos que coinciden con el mismo formato del atributo objetivo. Si el servidor acepta la operación de escritura, responde con una confirmación.

  3. **Write without response** Si esta operación está habilitada, un cliente puede escribir datos en un atributo sin esperar una confirmación del servidor. Esta es una operación de escritura no reconocida que se puede usar cuando se necesita un intercambio rápido de datos.

- **Operaciones iniciadas por el servidor:** La otra categoría de operaciones GATT son las operaciones iniciadas por el servidor, donde el servidor envía información directamente al cliente, sin recibir primero una solicitud. En este caso, el servidor puede notificar o indicar.

  1. **Notify** Una operación de notificación es utilizada por el servidor para enviar automáticamente el valor de cierto atributo al cliente, sin que el cliente lo solicite. Esto, por ejemplo, puede utilizarse para actualizar al cliente sobre una lectura de sensor específica que ha cambiado recientemente. Las notificaciones no requieren confirmación por parte del cliente.

  2. **Indicate** Similar a la operación de Notificar, Indicar también enviará el valor del atributo directamente al cliente. Sin embargo, en este caso, se requiere una confirmación por parte del cliente. Debido al requisito de confirmación, solo se puede enviar una Indicación por intervalo de conexión, lo que significa que las Indicaciones son más lentas que las notificaciones. Aunque estas operaciones son iniciadas por el servidor, el cliente primero debe habilitarlas suscribiéndose a la característica y activando ya sea las notificaciones o las indicaciones.

2.11.3 Servicios y características

La capa ATT define atributos y cómo se exponen los datos entre un cliente y un servidor. Como tal, una de las funciones principales de GATT es la estructuración jerárquica de atributos almacenados en un servidor GATT en entidades estandarizadas (servicios y características), proporcionando una interoperabilidad fluida entre diferentes dispositivos Bluetooth LE.

Atributos

La capa ATT define cómo se almacenan y acceden los datos en la base de datos de un servidor. Los datos se almacenan en forma de estructuras de datos llamadas Atributos. Los atributos son las unidades de datos fundamentales en las que se basan tanto las capas ATT como GATT. Los atributos contienen datos de usuario, así como metadatos que describen el atributo en sí, su tipo, permisos de seguridad, etc. El intercambio de datos que ocurre entre servidores y clientes ATT, o servidores y clientes GATT, se realiza en forma de atributos. Cuando se habla solo de atributos, se dice que están almacenados en un servidor ATT. Sin embargo, como veremos más adelante en esta lección, cuando comenzamos a clasificar los atributos en servicios y características, nos referimos a esa estructura de datos como un servidor GATT. Un atributo consta de 4 bloques de datos:

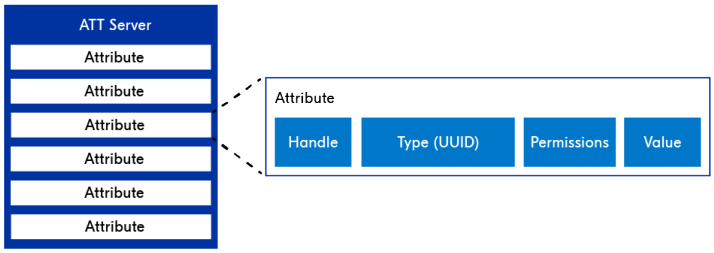

Figura 2.16: Bloques que componen un atributo

1. **Handle:** Identificador. Un índice único de 16 bits para un atributo específico en la tabla de atributos, asignado por la pila (stack). Un atributo se aborda a través

de su identificador. Se puede pensar en él como el número de fila en la tabla de atributos, aunque los identificadores no necesariamente son secuenciales.

2. **Type (UUID):** Tipo. Identificador único universal (UUID), que nos indica el tipo de atributo. Por ejemplo, si este atributo declara una característica, esto se reflejará en su campo de Tipo ya que contendrá un UUID utilizado específicamente para indicar la declaración de una característica.

3. **Permissions:** Permisos. El nivel de seguridad requerido (encriptación y/o autorización) para manejar ese atributo, además de indicar si es un atributo legible y/o escribible.

4. **Value:** Valor. Los datos reales del usuario (por ejemplo, lectura de un sensor) que se almacenan en el atributo. Este campo acepta cualquier tipo de dato. Puede contener un valor de monitor de ritmo cardíaco (pulsaciones por minuto), una lectura de temperatura o incluso una cadena de texto. También puede contener información (metadatos) sobre otro atributo.

Identificador Único Universal (UUID)

UUID es una abreviatura que se ve mucho en el mundo de Bluetooth LE. Es un número único que se utiliza para identificar atributos y nos dice sobre su importancia. Los UUID tienen dos tipos.

El primer tipo es el UUID de 16 bits definido por SIG (Special Interest Group). Por ejemplo, el servicio de ritmo cardíaco definido por SIG tiene el UUID 0x180D y una de sus características incluidas, la característica de Medición de Ritmo Cardíaco, tiene el UUID 0x2A37. El UUID de 16 bits es eficiente en energía y memoria, pero dado que solo proporciona un número relativamente limitado de IDs únicos, hay una necesidad de más UUID para cubrir todos los vendedores, usuarios y casos de uso.

El segundo tipo es un UUID de 128 bits, a veces referido como un UUID específico del proveedor. Este es el tipo de UUID que necesitas usar cuando estás creando tus propios servicios y características personalizados. Se ve algo así: 4A98-xxxx-1CC4-E7C1-C757-F1267DD021E8 y se llama el ÜUID base". Las cuatro x representan un campo donde insertarás tus propios IDs de 16 bits para tus servicios y características personalizados y los utilizarás de la misma manera que un UUID predefinido. De esta manera, puedes almacenar el UUID base una vez en la memoria, olvidarte de él y trabajar con IDs de 16 bits como de costumbre.

Servicios

Comencemos examinando qué constituye un servicio y cómo se estructuran jerárquicamente los atributos en un servicio dado. Como se muestra en la figura a continuación, los atributos son los principales bloques de construcción para los servicios. Una definición de servicio (comúnmente referida como servicio) está compuesta por múltiples atributos dispuestos en un formato especificado por GATT que facilita el intercambio de datos estandarizado entre dispositivos Bluetooth LE.

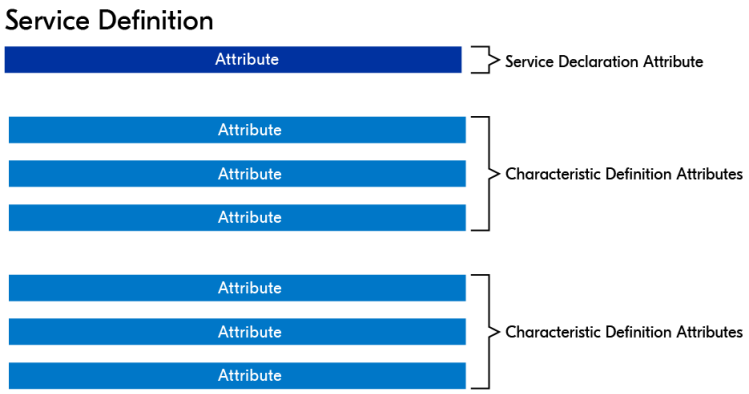

Figura 2.17: Estructura de los servicios

Atributo de declaración de servicios

Las definiciones de servicios siempre comienzan con un atributo de declaración de servicios. Este atributo contiene metadatos sobre el servicio e indica el inicio de un servicio en la secuencia de servicios almacenados en un servidor GATT.

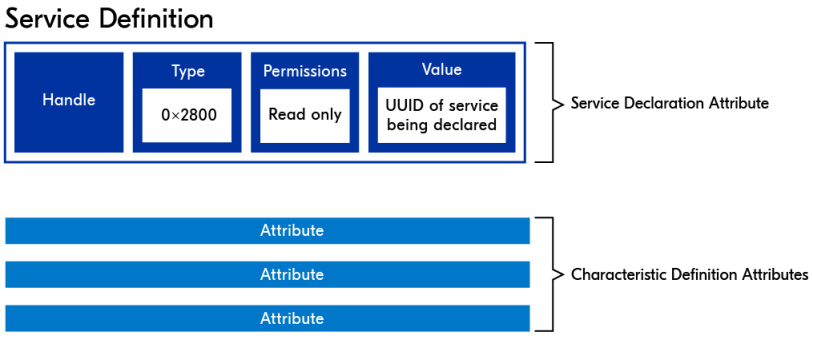

Figura 2.18: Atributo de declaración de servicios

El identificador (Handle) es similar a un número de fila mediante el cual se direcciona el atributo. El campo Tipo (Type) del atributo de declaración de servicio contiene el UUID (0x2800), que es un UUID único definido por SIG utilizado únicamente para indicar el inicio de un servicio. El campo Permisos (Permissions) aquí indica "solo lecturaz no se necesita autenticación. Esto se espera en un atributo de declaración de servicio ya que no hay razón para tener un permiso de escritura para él, ya que solo declara el inicio de un servicio. Por último, el campo Valor (Value) contiene el UUID del servicio que se está declarando. Por ejemplo, el Servicio de Ritmo Cardíaco es un servicio definido por SIG y se referencia mediante el UUID 0x180D, que se almacena en el campo Valor del atributo de declaración de servicio del Servicio de Ritmo Cardíaco.

Caracteríticas

Posteriormente, un servicio puede tener cero o más definiciones de características (comúnmente denominadas características). Una característica está compuesta por al menos dos atributos y opcionalmente más.

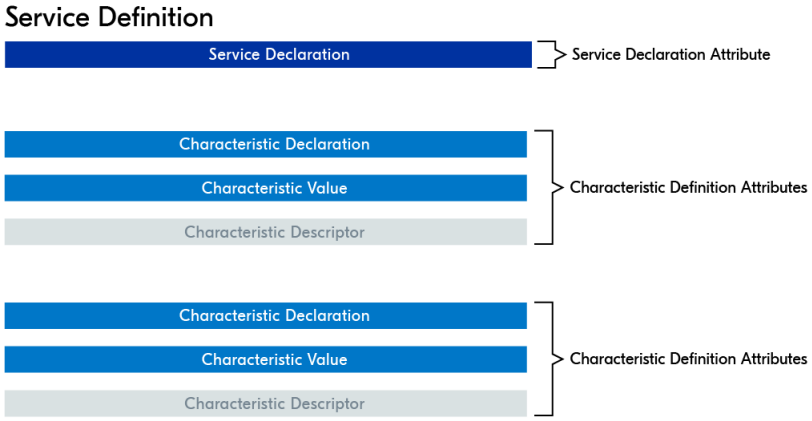

Figura 2.19: Estructura de una característica

Similar a una definición de servicio, una definición de característica comienza con un atributo de declaración, para indicar el inicio de una característica en la secuencia de características en una definición de servicio. Esto es seguido por el atributo de

valor de característica, que contiene los datos de usuario reales. Opcionalmente, una característica también puede tener uno o más atributos descriptores de características.

- **Atributo de declaración de característica.** Contiene metadatos sobre el Atributo de Valor de Característica. Una definición de característica comienza con un atributo de declaración de característica, para indicar el inicio de una característica en la secuencia de características en una definición de servicio. El campo Tipo (Type) del atributo de declaración de característica contiene el UUID (0x2803) utilizado únicamente para declarar una característica. El atributo de declaración tiene permisos de solo lectura, asegurando que los clientes puedan leer el valor pero no escribir en él.

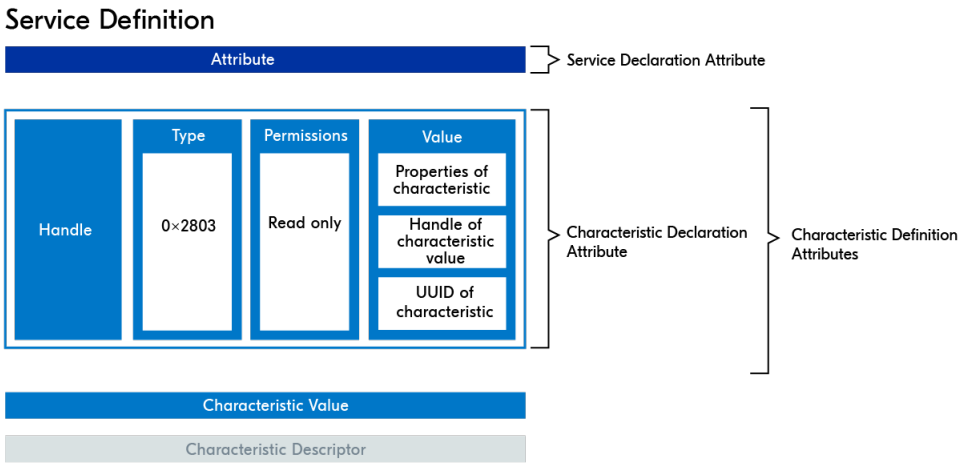

Figura 2.20: Atributo de declaración de característica

El campo Valor (Value) contiene información importante sobre la característica que se está declarando, específicamente tres campos separados:

1. Propiedades de la característica Qué tipo de operaciones GATT están permitidas en esta característica.
2. Identificador (handle) del valor de la característica El identificador (dirección) del atributo que contiene los datos de usuario (valor), es decir, el atributo de valor de la característica.
3. UUID de la característica El UUID de la característica que se está declarando.

- **Atributo de valor de característica:** Contiene los datos de usuario reales. Después del atributo que declara la característica viene el atributo de valor de característica. Aquí es donde se almacenan los datos de usuario reales. Su Identificador (Handle) y Tipo son los que se refieren en el campo Valor del Atributo de Declaración de Característica. Naturalmente, su campo de Valor es donde se almacenan los datos de usuario reales. El campo Permisos indica si el cliente puede leer y/o escribir en este atributo.

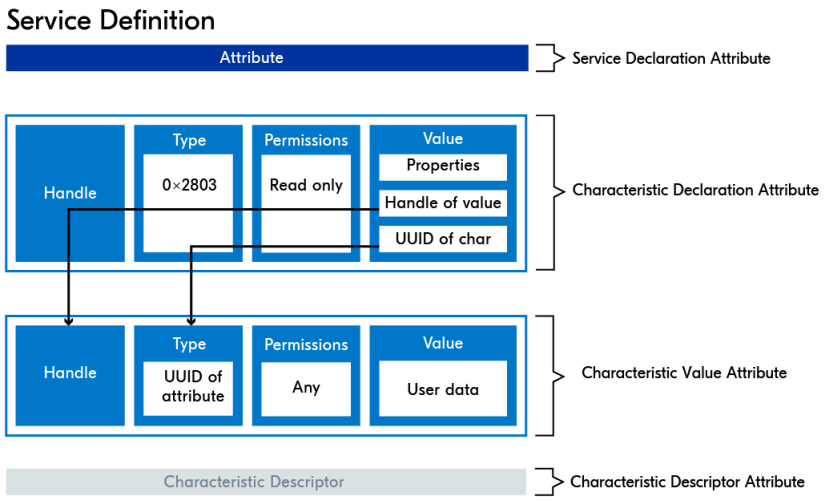

Figura 2.21: Atributo de valor de característica

- **Atributo descriptor de características (opcional):** Contiene más metadatos sobre la característica. Los atributos descriptores de características son opcionales. Contienen metadatos adicionales sobre la característica, proporcionando al cliente más información sobre la naturaleza de la característica. Hay varios tipos de descriptores, pero generalmente se dividen en dos categorías: definidos por GATT y personalizados.

  Los descriptores también permiten al cliente establecer permisos para ciertas operaciones GATT iniciadas por el servidor. El más comúnmente utilizado es el Descriptor de Configuración de Características del Cliente (CCCD, por sus siglas en inglés) definido por GATT.

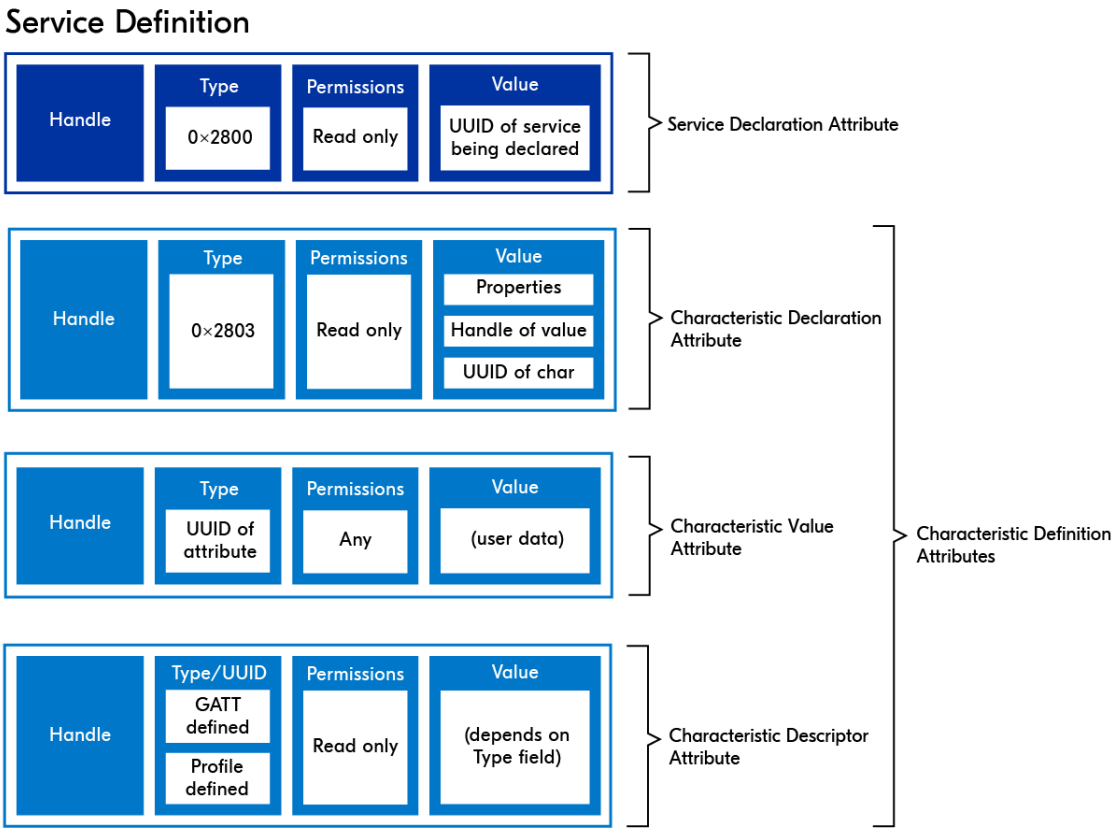

Figura 2.22: Atributo descriptor de características

- **Descriptor de Configuración de Características del Cliente (CCCD):** El Descriptor de Configuración de Características del Cliente (CCCD) es un tipo específico de descriptor de característica que es necesario cuando la característica admite operaciones iniciadas por el servidor (es decir, Notificar e Indicar). Este es un descriptor escribible que permite al cliente GATT habilitar y deshabilitar notificaciones o indicaciones para esa característica. El cliente GATT puede suscribirse a la característica de la que desea recibir actualizaciones, habilitando ya sea Indicaciones o Notificaciones en el CCCD de esa característica específica.

  Por ejemplo, en el Servicio de Ritmo Cardíaco, hay una característica llamada Medición de Ritmo Cardíaco. El cliente GATT (por ejemplo, su teléfono móvil) puede usar el CCCD de esta característica para recibir actualizaciones sobre esta característica. Así que se suscribe a la característica de Medición de Ritmo Cardíaco habilitando ya sea Indicaciones o Notificaciones en el CCCD de dicha característica. Esto significa que el servidor GATT (probablemente un dispositivo de sensor de ritmo cardíaco) enviará estas mediciones a su teléfono, sin que su teléfono tenga que solicitar estas mediciones.

  El formato del atributo CCCD es como se muestra a continuación. El UUID para los CCCD es 0x2902. Un CCCD siempre debe ser legible y escribible. Los descriptores con el Tipo CCCD solo tienen 2 bits en su campo de Valor. El primer bit indica si las Notificaciones están habilitadas, y el segundo bit es para Indicaciones.

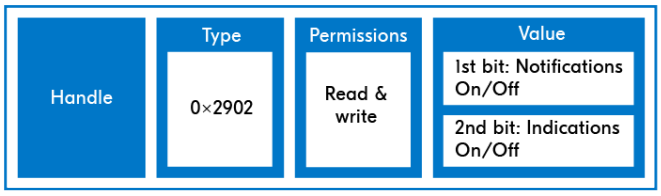

Figura 2.23: Descriptor de Configuración de Características del Cliente

Tabla de atributos

Una tabla de atributos es cómo se almacenan los atributos en el servidor GATT.

2.12 Proceso de emparejamiento

La práctica más común para proteger la comunicación inalámbrica es cifrar la conexión. Para cifrar el enlace, ambos pares necesitan tener las mismas claves. El proceso de generar, distribuir y autenticar estas claves para el cifrado se conoce como el proceso de emparejamiento.El proceso de emparejamiento requiere que los dispositivos repitan el proceso cada vez que deseen cifrar el enlace nuevamente. Además del "emparejamiento", se utiliza el término "vinculación" cuando los pares almacenan la clave de cifrado para que puedan volver a cifrar el enlace en futuras conexiones con el mismo par. Durante la vinculación, también pueden intercambiar y almacenar claves de identidad para que puedan reconocerse en futuras conexiones mediante la dirección privada resoluble aleatoria.

2.12.1 Fase 1. Iniciar el emparejamiento

Para iniciar el proceso de emparejamiento y, en algunos casos, el proceso de vinculación, el dispositivo central necesita enviar una Solicitud de Emparejamiento y el periférico responde con una Respuesta de Emparejamiento. En esta fase, los dos dispositivos intercambian sus características de emparejamiento, que se utilizarán para determinar qué método de emparejamiento usarán en la fase 2 y qué claves se distribuirán en la fase 3. Intercambian las características de seguridad que soportan, si se solicita o no la vinculación, y más. Lo más importante es que los pares intercambian sus capacidades de I/O (entrada/salida), seleccionadas de una de las siguientes:

- DisplayOnly: El par solo tiene una pantalla
- DisplayYesNo: El par tiene una pantalla y la opción de seleccionar "sí" o "no"
- KeyboardOnly: El par solo tiene teclado
- NoInputNoOutput: El par no tiene capacidades de entrada ni salida
- KeyboardDisplay: El par tiene capacidades de teclado y pantalla

2.12.2 Fase 2: Realizar el emparejamiento

En la fase 2, se generan las claves utilizadas para cifrar la conexión. El método de emparejamiento utilizado aquí depende de la información intercambiada en la fase 1.

En el emparejamiento LE Legacy, los pares intercambian una Clave Temporal (TK) que se utiliza para generar una Clave a Corto Plazo (STK) que luego se usa para cifrar el enlace. Sin embargo, dado que la STK puede ser fácilmente descifrada, Bluetooth v4.2 introdujo algo llamado Conexiones Seguras Bluetooth LE. En las Conexiones Seguras LE, los dispositivos generan e intercambian un tipo de clave más segura y la usan para generar una Clave a Largo Plazo (LTK) que se utiliza para cifrar la conexión. El emparejamiento Legacy define tres métodos diferentes para intercambiar la TK, llamados métodos de emparejamiento. Las Conexiones Seguras LE soportan estos tres métodos de emparejamiento pero también un cuarto (comparación numérica) que no es soportado en el emparejamiento Legacy. La seguridad del proceso de emparejamiento depende del método de emparejamiento utilizado en esta fase.

- **Just Works:** Ambos pares generan la STK basada en la información intercambiada en texto plano, y al usuario solo se le pide que acepte la conexión. Este método no está autenticado.
- **Passkey Entry:** Se muestra un número de 6 dígitos en un dispositivo, y debe ser ingresado en el otro dispositivo. Las capacidades de E/S de los dispositivos determinan cuál muestra el número y cuál lo ingresa.
- **Out of Band (OOB):** Las claves de cifrado se intercambian por algún otro medio que no sea Bluetooth LE, por ejemplo, mediante el uso de NFC.
- **Numeric Comparison (LE Secure Connections only)** Ambos dispositivos muestran un número de 6 dígitos y el usuario selecciona “sí” o “no” para confirmar la visualización.

Qué método de emparejamiento utilizar se decide en función de la bandera OOB, la bandera Man-In-The-Middle (MITM) y las capacidades de E/S de los pares, intercambiadas durante la fase 1. Las banderas OOB y MITM primero determinan si utilizar el método de emparejamiento OOB directamente o determinar el método de emparejamiento en función de las capacidades de I/O.

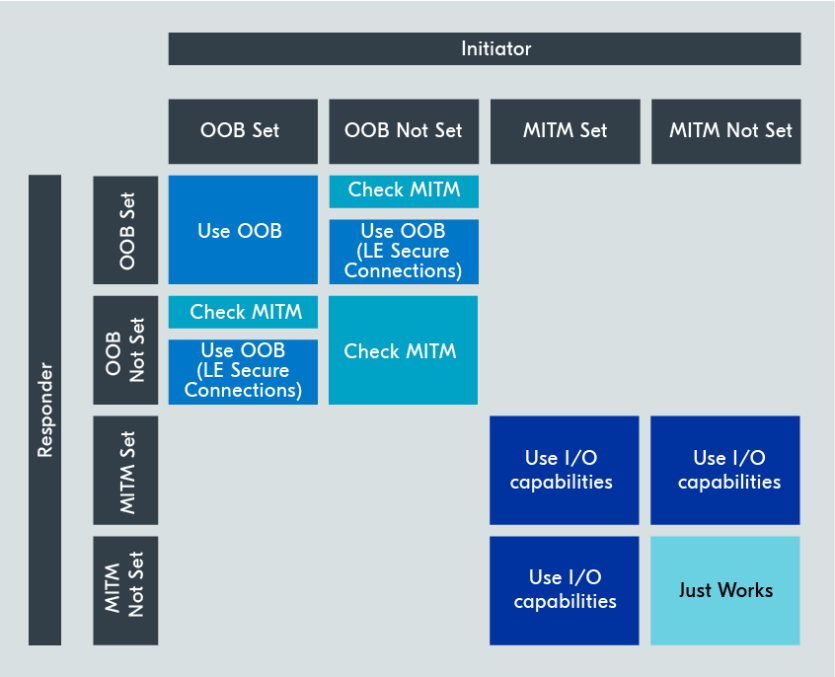

Figura 2.24: Decisión del método de emparejamineto

Es importante tener en cuenta que en Conexiones Seguras LE, solo uno de los pares necesita tener la bandera OOB configurada para que se utilice este método de emparejamiento. Dependiendo de las banderas OOB y MITM, las capacidades de I/O de los pares podrían usarse para determinar el método de emparejamiento. En este caso, se utiliza la figura 2.25

La clave generada en esta fase se utilizará para cifrar el enlace después de la fase 2. Si solo estás realizando el emparejamiento, sin vinculación, entonces solo se llevarán a cabo estas 2 fases y los pares omitirán la fase 3.

2.12.3 Fase 3: Distribución de claves

En esta fase, la Clave a Largo Plazo (LTK) se utiliza para distribuir el resto de las claves. En el emparejamiento legacy, la LTK también se genera en esta fase (en Conexiones Seguras LE, la LTK se genera en la fase 2). Otras claves también se generan e intercambian en esta fase, para identificar a los pares la próxima vez que se reconecten y poder volver a cifrar el enlace utilizando la misma LTK.

Figura 2.25: Mapeo de capacidades I/O al método de generación de claves

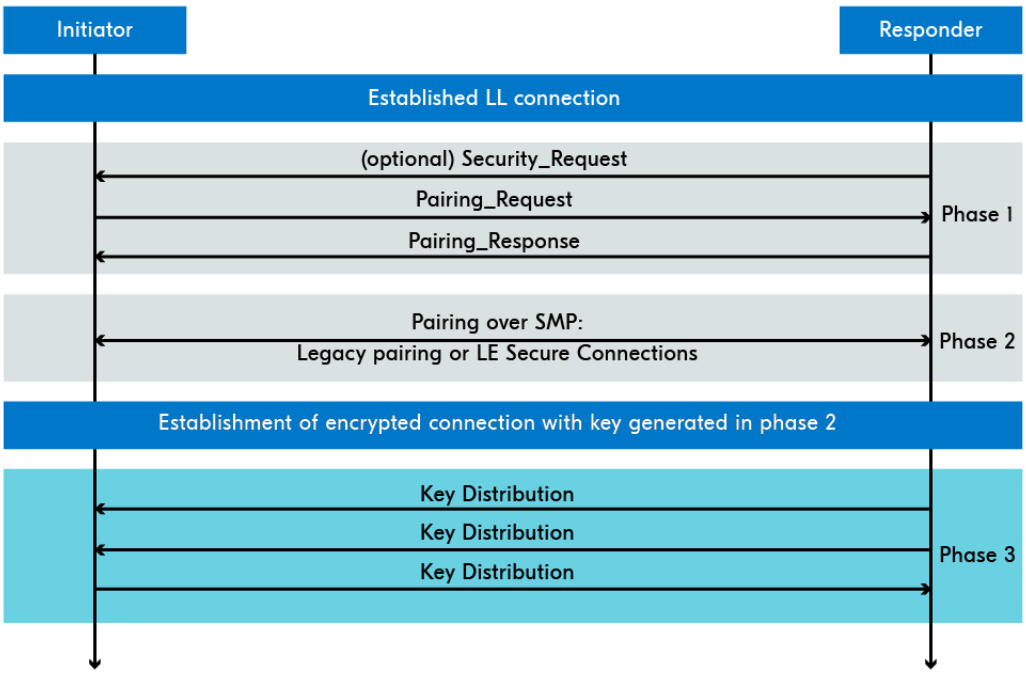

Figura 2.26: Diagrama del proceso de emparejamiento

2.12.4 Legacy pairing vs LE Secure Connections

- **Emparejamiento legacy:** Antes de Bluetooth v4.2, el emparejamiento legacy era el único método de emparejamiento disponible en Bluetooth LE. Es bastante simple y expone un riesgo porque la Clave a Corto Plazo (STK) utilizada para cifrar el enlace puede ser fácilmente descifrada.

  Cuando se utiliza Just Works en el emparejamiento legacy, la TK se establece en 0, lo que no ofrece protección en términos de espionaje o ataques Man-In-The-Middle (MITM). Un atacante puede descifrar fácilmente la STK y espiar la conexión, y tampoco hay forma de verificar los dispositivos.

  Usar la entrada de clave es un poco mejor, ya que la TK es ahora un número de 6 dígitos que es pasado entre los dispositivos por el usuario. Por ejemplo, uno de los dispositivos muestra el número en su pantalla, y el usuario introduce este número en el otro dispositivo usando un teclado. Desafortunadamente, un atacante puede fácilmente interceptar los valores que se están intercambiando, y entonces es muy fácil averiguar la STK y descifrar la conexión. Incluso si no puede determinar directamente la TK, la clave puede ser fácilmente descifrada probando todas las 999999 combinaciones.

  En el emparejamiento Fuera de Banda (OOB), la TK se intercambia por algún otro medio que no sea Bluetooth LE, por ejemplo, utilizando NFC. Este método soporta el uso de una TK de hasta 128 bits, lo que incrementa la seguridad de la conexión. La seguridad del canal OOB utilizado durante el intercambio también determina la protección de la conexión Bluetooth LE. Si el canal OOB está protegido contra escuchas y ataques MITM, también lo estará el enlace Bluetooth LE.

  El emparejamiento legacy no es recomendado por el Bluetooth SIG, pero si debes usarlo, utiliza el emparejamiento OOB. La autenticación Fuera de Banda es el único método que puede considerarse seguro cuando se empareja con el emparejamiento legacy.

- **Conexiones Seguras LE:** Por esta razón, se introdujeron las Conexiones Seguras LE en Bluetooth v4.2. En lugar de usar TK y STK, las Conexiones Seguras LE utilizan criptografía de Elliptic-Curve Diffie-Hellman (ECDH) para generar un par de claves pública-privada. Los dispositivos intercambian sus claves públicas. Utilizarán uno de los cuatro métodos de emparejamiento (Just Works, Entrada

de Clave, OOB o Comparación Numérica) para verificar la autenticidad del dispositivo par y generar la LTK basada en la clave Diffie-Hellman y los datos de autenticación.

Aunque Just Works es más seguro cuando se utiliza Conexiones Seguras LE, todavía no ofrece autenticación y, por lo tanto, no se recomienda como método de emparejamiento. El método de emparejamiento de Entrada de Clave ahora utiliza la clave de paso, junto con la clave pública ECDH y un número arbitrario de 128 bits para autenticar la conexión. Esto significa que es mucho más seguro que lo descrito en el emparejamiento legacy. Utilizar el emparejamiento OOB sigue siendo una opción recomendada, ya que proporciona protecciones siempre que el canal OOB sea seguro, tal como en el emparejamiento legacy.

Además, se introdujo un nuevo método de emparejamiento llamado Comparación Numérica con esta característica. Aunque sigue el mismo procedimiento que el método de emparejamiento Just Works, agrega otro paso al final para proteger contra ataques MITM haciendo que el usuario realice una verificación manual basada en los valores mostrados en ambos dispositivos.

Los únicos datos intercambiados entre los pares son las claves públicas. El uso de la criptografía de clave pública ECDH hace que sea extremadamente difícil descifrar la LTK, lo que representa una mejora significativa en comparación con el emparejamiento legacy.

Aunque la mayoría de los dispositivos son compatibles con Conexiones Seguras LE, todavía hay algunos productos Bluetooth LE que solo soportan el emparejamiento legacy. Por lo tanto, es una buena idea habilitar el soporte para el emparejamiento legacy además de Conexiones Seguras LE para lograr una mejor interoperabilidad en la aplicación.

2.13 Modelos de seguridad

Como de seguro es Bluetooth LE depende en gran medida de cómo se ejecute el proceso de emparejamiento y de las opciones de entrada/salida (I/O) que soporten los dispositivos que se emparejan.

Hay 3 tipos comunes de ataques que la seguridad de Bluetooth LE debe enfrentar:

- **Identity tracking:** El seguimiento de identidad explota la dirección Bluetooth para rastrear un dispositivo. Protegerse contra tales ataques requiere protección de privacidad. Esto se puede lograr utilizando una dirección privada resoluble que cambia aleatoriamente, donde solo los dispositivos vinculados/confiables pueden resolver la dirección privada. La clave de resolución de identidad (IRK, por sus siglas en inglés) se usa para generar y resolver la dirección privada.
- **Intercepción pasiva (sniffing):** La intercepción pasiva permite a un atacante escuchar los datos que se transmiten entre dispositivos. Esto se puede proteger encriptando la comunicación entre los pares. El desafío aquí es cómo los pares generan y/o intercambian las claves para encriptar la conexión de manera segura. Este fue el principal inconveniente que hizo vulnerable el emparejamiento heredado de Bluetooth LE y creó la necesidad de las Conexiones Seguras LE.
- **Intercepción activa (hombre en el medio, o MITM):** En un ataque de intercepción activa (o hombre en el medio), el atacante se hace pasar por dos dispositivos legítimos para engañarlos y que se conecten a él. Para prevenir esto, necesitamos asegurarnos de que el dispositivo con el que estamos comunicándonos es, de hecho, el dispositivo con el que queremos hablar y no un dispositivo no autenticado.

Bluetooth LE define 4 niveles de seguridad en el modo de seguridad 1:

- **Nivel 1:** Sin seguridad (texto abierto, es decir, sin autenticación y sin encriptación)
- **Nivel 2:** Encriptación con emparejamiento no autenticado
- **Nivel 3:** Emparejamiento autenticado con encriptación
- **Nivel 4:** Emparejamiento autenticado con Conexiones Seguras LE y encriptación

Cada conexión comienza en el nivel de seguridad 1 y luego se actualiza a un nivel de seguridad superior según el método de emparejamiento utilizado.

Utilizar Just Works llevará la conexión al nivel de seguridad 2. Este método no protege contra ataques MITM, ya que el enlace solo está encriptado, y no autenticado. Para protegerse contra MITM, la conexión debe estar en el nivel de seguridad 3 o superior. Esto se puede lograr usando ya sea el método de Entrada de Clave o el emparejamiento OOB con el emparejamiento Heredado, ambos de los cuales proporcionan autenticación y elevan la seguridad al nivel 3.

La conexión solo puede alcanzar el nivel de seguridad 4 si ambos pares admiten Conexiones Seguras LE y se usa uno de los métodos de autenticación como Entrada de Clave, Comparación Numérica o OOB.

El campo de Permisos de un atributo determina no solo si el atributo es legible y/o escribible, sino también el nivel de seguridad requerido de esa conexión para que el atributo sea accesible. Por ejemplo, si un enlace está encriptado con el nivel de seguridad 2, encriptación no autenticada, el par no podrá acceder a ninguna característica que requiera un nivel de seguridad superior.

De esta manera, podemos configurar nuestra tabla de atributos para que los datos solo se puedan intercambiar cuando el enlace esté encriptado con un cierto nivel de seguridad.

Bluetooth LE tiene un total de 3 modos de seguridad. El modo de seguridad 2 utiliza la firma de datos para la seguridad y rara vez se usa. El modo de seguridad 3 se refiere a la transmisión isócrona, que se utiliza con Bluetooth LE Audio y es bastante nuevo.

2.13.1 Filtro de Lista de Aceptación

El Filtro de Lista de Aceptación, anteriormente conocido como Listado Blanco, es una forma de limitar el acceso a una lista de dispositivos tanto en publicidad como en escaneo.

Cuando se usa en publicidad, solo los dispositivos en la Lista de Aceptación pueden enviar una solicitud de conexión para establecer una conexión o enviar una solicitud de escaneo para recibir la respuesta de escaneo del anunciante. Si un dispositivo que no está en la lista envía estas solicitudes, el anunciante ignorará la solicitud.

Cuando se usa en escaneo, solo los paquetes de publicidad y respuesta de escaneo de los dispositivos en el filtro serán escaneados y reportados a la aplicación. El escáner filtrará cualquier paquete de otros anunciantes.

Al utilizar la dirección del par y las claves de identidad distribuidas en la fase 3 del proceso de emparejamiento, podemos construir el Filtro de Lista de Aceptación para permitir únicamente que un dispositivo vinculado y autorizado se conecte al dispositivo.

CAPÍTULO 3

CONFIGURACIÓN DEL ENTORNO Y SELECCIÓN DE HARDWARE Y SOFTWARE

3.1 Descarga del entorno Nordic

En primer lugar se debe instalar el gestor de descargas de Nordic: Descargar nRF-Connect-for-Desktop

Se deben instalar Programmer, Serial Terminal y Toolchain Manager.

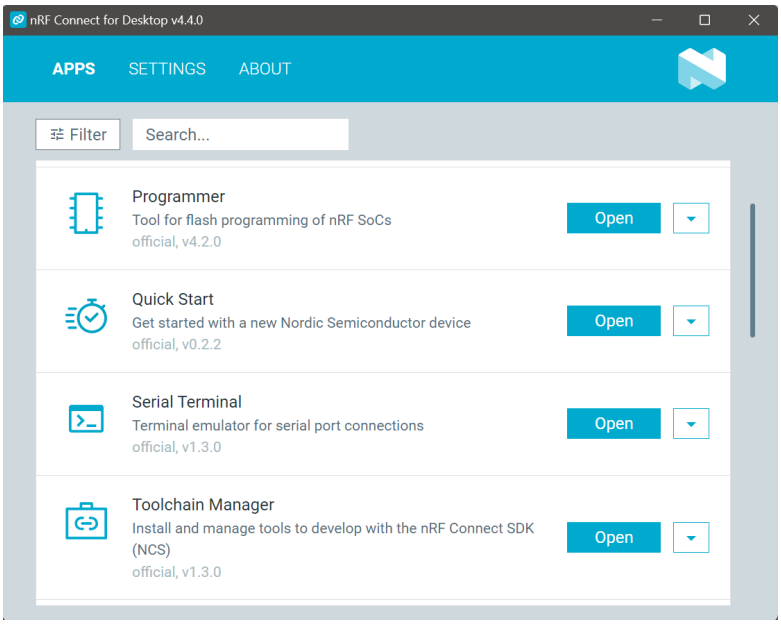

Figura 3.1: Nordick Desktop

En concreto instalamos el Toolchain Manager nRF Connect SDK v2.5.2

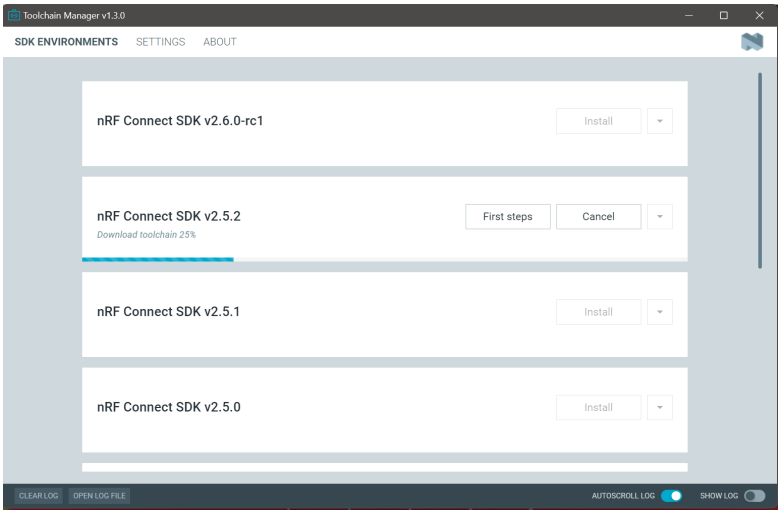

Figura 3.2: Toolchain Manager

Al intentar abrir VS Code te pide instalar un plugin Descargar plugin VS Code

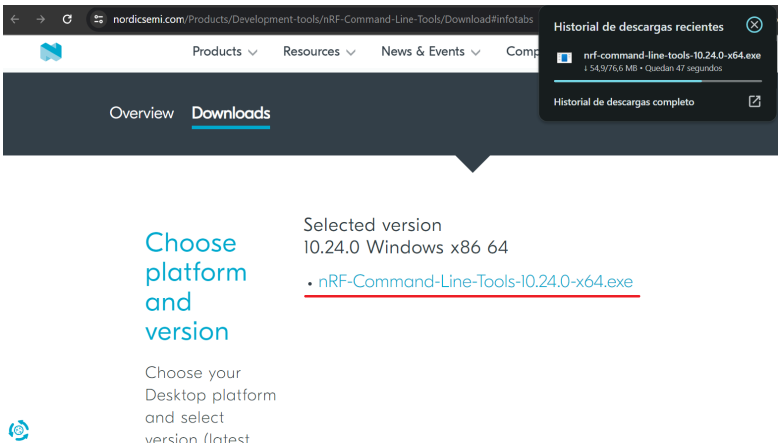

Figura 3.3: Command Line Tools

3.2 Crear y flashear un programa desde VS Code

En este enlace hay un vídeo explicativo sobre todo el proceso inicial.

En primer lugar debes abrir el programa en *Visual Studio Code* (VS) o descargar algún ejemplo desde la opción *Browse samples* del desplegable *WELCOME* en la pestaña *nRF Connect* que se habrá añadido a VS.

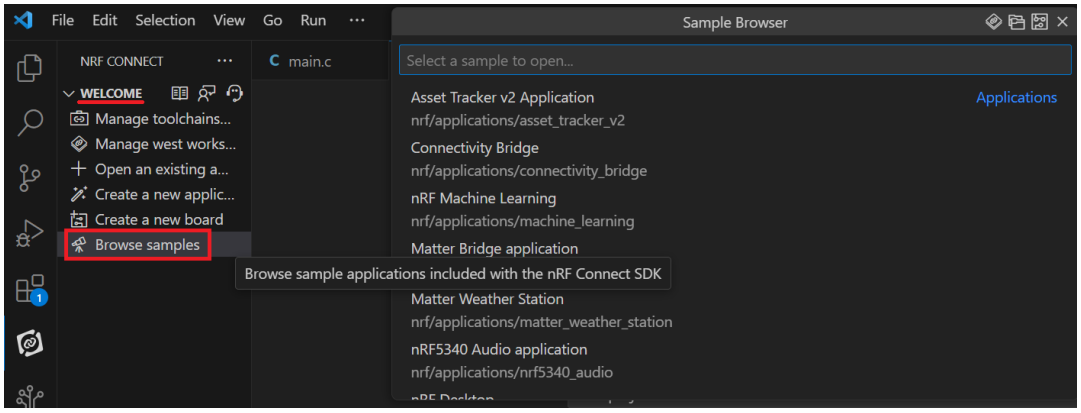

Figura 3.4: Browse samples

Con el programa ya terminado se procede a buildearlo. Se debe añadir una nueva *Build configuration* en el botón *add buil configuration* del desplegable *APPLICATIONS*. Se abrirá una nueva pantalla, donde se elige en el desplegable *Board* la placa que vayas a utilizar en este caso: nrf52840dk_nrf52840. Al final de esta pantalla se debe presionar el botón azul *Build Configuration*.

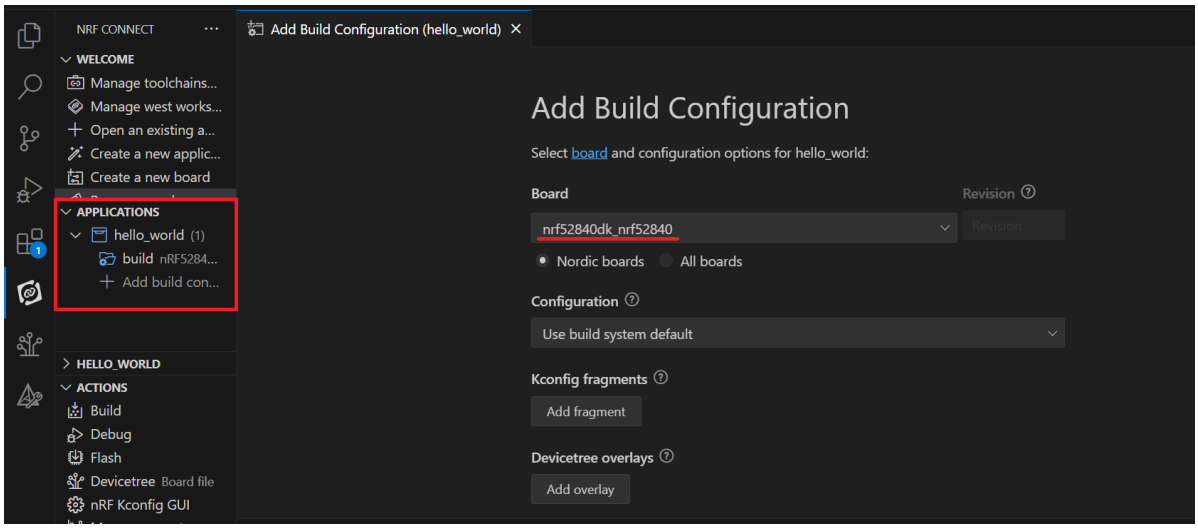

Figura 3.5: Añadir Build Configuration

Por último, se deben pulsar en el desplegable *ACTIONS* los botones *Build* y después *Flash*, en ese orden.

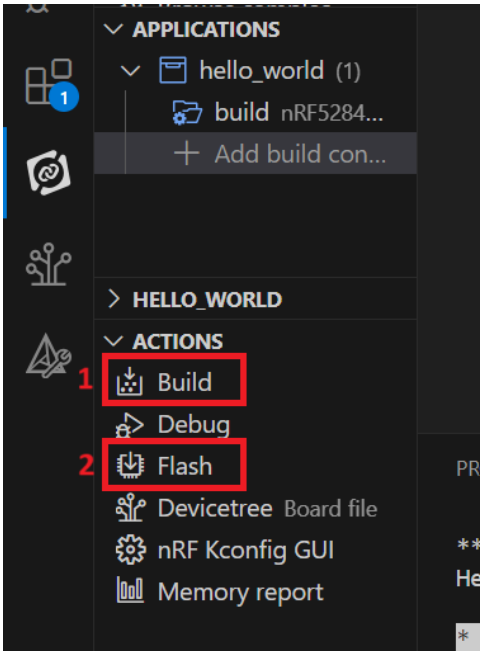

Figura 3.6: Build and Flash

Ahora el programa ya está grabado en la placa. En este caso es *Hello Word*, por lo que, cada vez que se ejecute debe aparecer en el *TERMINAL* la siguiente salida:

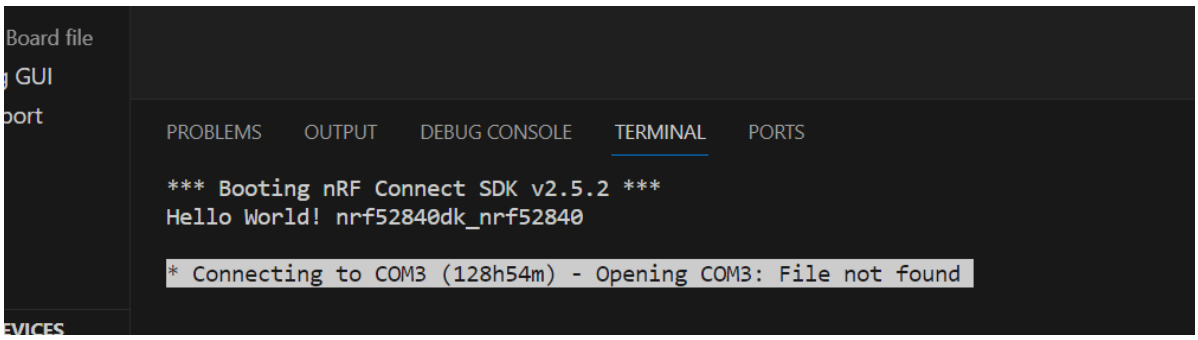

Figura 3.7: Salida por el terminal

3.3 Elección del Kit de Desarrollo

He elegido utilizar el nRF52840 DK y el nRF52840 Dongle, ya que ambos ofrecen una plataforma robusta y flexible para trabajar con Bluetooth Low Energy (BLE), que es fundamental en las mediciones de distancia entre dispositivos.

El nRF52840 DK (Development Kit) es una placa de desarrollo basada en el SoC nRF52840 de Nordic Semiconductor. Este kit soporta BLE y otros protocolos de comunicación inalámbrica como Zigbee y Thread. Es ideal para desarrollar y probar aplicaciones inalámbricas, ya que incluye una amplia gama de interfaces periféricas, una potente CPU Arm Cortex-M4, y suficiente memoria (1 MB Flash y 256 KB RAM) para manejar aplicaciones complejas. Esto lo convierte en una opción sólida para realizar pruebas precisas en entornos reales [5].

El nRF52840 Dongle complementa al DK actuando como un dispositivo conectado al ordenador mediante USB. Esto permite capturar datos en tiempo real y realizar pruebas entre dispositivos BLE. Además, el Dongle es una opción práctica y económica para desplegar y analizar conexiones BLE sin necesidad de una infraestructura adicional [5]. Cabe destacar que la programación del dongle es considerablemente más complicada que la del DK. Mientras que el DK se puede llegar a programar desde una herramienta externa como puede ser VScode, el dongle debe ser programado desde un terminal, realizando con comandos cada paso, lo cual precisa conocimientos mayores.

La combinación de ambos dispositivos permite aprovechar las capacidades del protocolo BLE, específicamente el método MCPD para las mediciones de distancia en diversos escenarios, lo que es esencial para el análisis de tu trabajo.

3.4 Elección del Programa

El programa seleccionado para basar el TFG es el que se presenta como ejemplo de uso de la librería de estimación de distancias **nrf_dm** de Nordic Semiconductor. Aplica el método MCPD (Multi-Carrier Phase Difference) que se basa en técnicas de medición de distancia de alta precisión. Este método permite medir la distancia entre dos dispositivos BLE utilizando las diferencias de fase entre múltiples portadoras de radiofrecuencia. Esto es especialmente útil en aplicaciones que requieren una estimación de distancia precisa y rápida, como en el seguimiento de activos o en aplicaciones de ubicación en interiores.

3.4.1 ¿Cómo funciona MCPD?

MCPD mide la diferencia de fase de las señales de varias frecuencias que se transmiten entre dos dispositivos. La diferencia de fase es luego utilizada para calcular la distancia. El enfoque multi-portadora mejora la precisión al utilizar varias frecuencias, reduciendo errores asociados con los entornos multipath (donde las señales rebotan en superficies) [2].

3.4.2 Limitaciones

- **Acceso restringido:** La librería está distribuida como un binario estático, lo que impide modificar su código internamente. Aunque la documentación proporciona algunos métodos de mejora, las personalizaciones avanzadas requerirían un acceso más profundo a la fuente, algo que no está disponible de manera pública.
- **Precisión variable:** Aunque el método MCPD puede proporcionar alta precisión, los resultados pueden ser inconsistentes en distancias cortas o en entornos complicados. Este enfoque de medición es altamente eficaz para aplicaciones de posicionamiento basado en BLE y es parte de la evolución de la tecnología de

ubicación con Bluetooth, junto con otros métodos como el AoA (Angle of Arrival) y AoD (Angle of Departure).

CAPÍTULO 4

CAPTACIÓN Y PROCESAMIENTO DE MUESTRAS

4.1 Obtención de las muestras

En el proceso de obtención de datos para la estimación de distancia entre dispositivos Bluetooth, se han realizado mediciones en diversos escenarios y condiciones para asegurar la precisión y robustez de los resultados. La metodología seguida es la siguiente:

Para las distancias cortas (1,5, 3 y 6 metros), se utilizó un metro tradicional para obtener la medida real, asegurando precisión en entornos controlados como una habitación y un pasillo. Para las distancias más largas (10 y 20 metros), se empleó una cuerda previamente medida, lo cual permitió obtener mediciones más extensas en entornos como campo abierto y bosque.

Las mediciones se realizaron durante periodos de 30 segundos, capturando un total de 60 muestras en cada escenario. Esto permitió obtener una muestra significativa del comportamiento de la señal de Bluetooth en distintos medios y distancias. Los escenarios seleccionados para el experimento incluyeron:

- **Habitación** a 1,5 y 3 metros.
- **Pasillo** a 3 y 6 metros.
- **Campo abierto** a 6, 10 y 20 metros.
- **Bosque** a 6, 10 y 20 metros.

Esta variedad de entornos fue elegida para observar cómo varía la propagación de la señal Bluetooth en espacios cerrados, semiabiertos y completamente abiertos, con la finalidad de obtener estimaciones de distancia que consideren las particularidades de cada medio.

4.1.1 Zero-distance calibration (Offset)

Para obtener un valor preciso en las mediciones de distancia, ha sido necesario realizar una calibración del offset, tal como se explica en la documentación proporcionada por Nordic Semiconductor en el apartado "Zero-distance calibration". Según el README del programa, las mediciones de distancia incluyen un offset que puede variar dependiendo del diseño del circuito de radio, la antena utilizada o el diseño de la PCB. Para compensar este offset, el procedimiento indica que los dispositivos deben colocarse a una distancia de referencia de 60 cm, obteniendo un conjunto de datos de medición. El offset se calcula como la diferencia entre el valor promedio de las mediciones y la distancia real (en este caso, 60 cm) [2].

4.2 Formato de las muestras

Se utiliza la aplicación nRF Connect para capturar la salida del puerto serie del dispositivo BLE, que es por dónde envía las muestras al ordenador. La aplicación te ofrece la opción de guardarlo en un fichero de texto plano que tiene el formato que se puede ver en el código fuente 4.1.

Código fuente 4.1: Ejemplo del formato de las muestras

```
Measurement result:
Addr: FB:1F:22:AA:3D:32 (random)
Quality: ok
Distance estimates: mcpd: high_precision=0.85 ifft=1.14 phase_slope=2.60 rssi_ope
nspace=0.89 best=1.14
Measurement result:
Addr: FB:1F:22:AA:3D:32 (random)
Quality: ok
Distance estimates: mcpd: high_precision=1.04 ifft=1.25 phase_slope=2.24 rssi_ope
nspace=0.89 best=1.25
```

4.3 Procesamiento de las muestras

Hemos programado un notebook de python para que el procesamiento de los datos sea más cómodo. Para utilizarlo solo es necesario guardar los ficheros, que devuelve la aplicación nRF Connect, en una carpeta (en mi caso la he llamado "Mediciones") y seleccionar ,desde el menú que ofrece el programa, el fichero que se desea procesar.

Automáticamente el programa devuelve los datos graficados y separados por cada método, en una primera figura. En una segunda figura, representa todos los métodos en una sola gráfica y los datos denominados como "Best" (es decir, los que considera como mejores la librería nrf_dm) en otra. Además guarda en otra carpeta ambas figuras, en formato vectorial (.pdf).

4.3.1 Explicación del código

El código comienza cargando las librerías necesarias para visualizar las gráficas y para poder operar con el sistema de ficheros.

Código fuente 4.2: Código para cargar librerías

```
1 # Esto se emplea para que la salida se adapte al formato del notebook
2 %matplotlib notebook
3 # A continuación importamos librerías
4 import numpy as np  # cálculo numérico
5 import matplotlib.pyplot as plt  # gráficos
```

```
import matplotlib.transforms as mtransforms
import os
```

Seguidamente muestra un menú de selección, con todos los ficheros que contiene la carpeta (aquí deben aparecer tus ficheros con las muestras) y guarda en una variable el nombre del fichero correspondiente al número del menú que se selecciona. Muestra el contenido de la variable con el nombre como salida.

Código fuente 4.3: Código para el menú de selección

```
#MENÚ DE SELECCIÓN DEL FICHERO A PROCESAR

#Path hasta la carpeta que tiene las muestras
folder_path = "../../Mediciones/"

# Listar todos los ficheros de la carpeta
files = [f for f in os.listdir(folder_path) if
    os.path.isfile(os.path.join(folder_path, f))]

# Mostrar un menú con los ficheros
print("Selecciona un fichero:")
for i, file in enumerate(files, 1):
    print(f"{i}. {file}")

# Pedir al usuario que elija un fichero
selection = int(input("\n Introduce el número del fichero que deseas
    seleccionar: "))

# Guardar el nombre del fichero seleccionado en la variable file_name
file_name = files[selection - 1]
print(f"Has seleccionado: {file_name}")
```

Elegido ya el fichero, el siguiente fragmento de código extrae, del nombre del fichero, el tipo de entorno en el que se han tomado las muestras y la distancia a la que se encontraban los dispositivos. Guarda cada uno en una variable. El dato almacenado

de la distancia se usará posteriormente para dibujar una línea de referencia en todas las gráficas. Muestra el contenido de ambas variables con salida.

Código fuente 4.4: Código para extraer el entorno y la distancia

```
# EXTRAEMOS EL ENTORNO Y LA DISTANCIA DEL NOMBRE DEL FICHERO

# Diccionario de equivalencias para el entorno
environment_mapping = {
    "PA": "Pasillo",
    "CA": "Campo Abierto",
    "HA": "Habitación",
    "BO": "Bosque",
    "OF": "Offset"
}

# Separar la parte de código
parts = file_name.split('-')

# Obtener el entorno
env_code = parts[0]
env = environment_mapping.get(env_code, "Desconocido")

# Obtener la distancia y convertirla a float
dist_str = parts[1].replace("m", "").replace(",", ".")
dist = float(dist_str)

# Mostramos los resultados
print(f"Entorno: {env}")
print(f"Distancia: {dist} metros")
```

La siguiente parte del código se encarga de juntar en una misma línea todas las muestras. Como se puede ver en el código fuente 4.1, la línea con los datos se corta y se divide en dos. Esto dificulta la extracción de los datos, por lo que el fichero se rectifica y se guarda en otra carpeta. Se le llama con el mismo nombre que el original pero añadiendo "-R" para diferenciarlos. Muestra el nombre del fichero resultante como salida.

Código fuente 4.5: Código para rectificar el fichero

```
# QUITAR LOS SALTOS DE LÍNEA

# Definir el path con el nombre del fichero y el del fichero resultante
input_path = folder_path + file_name
output_path = folder_path + "Rectificados/" + file_name + "-R"

# Abrir el fichero para su lectura
with open(input_path, 'r') as file:
    lines = file.readlines()

# Procesar las líneas
with open(output_path, 'w') as file:
    for line in lines:
        if line.startswith("Distance estimates:"):
            line = line.rstrip()
            if not line.endswith(' '):
                file.write(line)
            else:
                file.write(line + '\n')
        else:
            file.write(line)

# Mostramos el nombre del fichero final
print(f"Fichero procesado guardado como: {file_name + '-R'}")
```

Ahora viene la parte del código que organiza los datos en vectores, separándolos por métodos de estimación. Como resultado muestra los cinco primeros valores, de cada uno de los cinco vectores resultantes, para comprobar que la extracción se ha realizado con éxito.

Código fuente 4.6: Código para organizar los datos en vectores

```
# LEE EL ARCHIVO Y ORGANIZA LAS MUESTRAS EN VECTORES

file_path = output_path
```

```
4
5  # Inicializamos las listas donde guardaremos los datos extraídos
6  high_precision_values = []
7  ifft_values = []
8  phase_slope_values = []
9  rssi_openspace_values = []
10 best_values = []
11
12 # Procesar el fichero
13 with open(file_path, 'r') as file:
14     for line in file:
15         if 'Distance estimates' in line:
16             # Extraer los valores de cada métrica utilizando el método
                ↪ split
17             parts = line.split(' ')
18             for part in parts:
19                 if 'high_precision=' in part:
20                     high_precision_values.append
                        ↪ (float(part.split('=')[1]))
21                 elif 'ifft=' in part:
22                     ifft_values.append(float(part.split('=')[1]))
23                 elif 'phase_slope=' in part:
24                     phase_slope_values.append
                        ↪ (float(part.split('=')[1]))
25                 elif 'rssi_openspace=' in part:
26                     rssi_openspace_values.append
                        ↪ (float(part.split('=')[1]))
27                 elif 'best=' in part:
28                     best_values.append(float(part.split('=')[1]))
29
30 # Verificamos los primeros 5 datos extraídos de cada lista
31 (high_precision_values[:5], ifft_values[:5], phase_slope_values[:5],
   ↪ rssi_openspace_values[:5], best_values[:5])
```

La siguiente línea de código imprime la longitud de los vectores anteriormente creados, que coincide con el número de muestras. Deben ser todas 60 como se indica en la sección 4.1.

Código fuente 4.7: Código para comprobar el número de muestras

```
# Verificamos la longitud de todos los vectores
(len(high_precision_values), len(ifft_values), len(phase_slope_values),
 ↪  len(rssi_openspace_values), len(best_values))
```

Por último, se grafican los resultados y se guardan como imágenes vectoriales. La figura 1 se guarda con el nombre del fichero original añadiendo "-I" y la figura 2 igual pero añadiendo "-C" en este caso.

Código fuente 4.8: Código para graficar y guardar los resultados

```
# GRAFICAMOS LOS RESULTADOS
offset= dist + 0.3

# Índices de medición
indices = list(range(len(high_precision_values)))

# Crear gráficos individuales
plt.figure(figsize=(10, 6))

# HIGH PRECISION
plt.subplot(2, 2, 1)
plt.plot(indices, high_precision_values, 'o-', label='High Precision')
plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
 ↪  ({dist})')
plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
 ↪  ({offset})')
plt.title('Estimaciones High Precision')
plt.xlabel('Índice de Muestra')
plt.ylabel('Distancia Estimada en metros')
plt.legend()

# IFFT
```

```
21 plt.subplot(2, 2, 2)
22 plt.plot(indices, ifft_values, 'o-', label='IFFT')
23 plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
    ↪ ({dist})')
24 plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
    ↪ ({offset})')
25 plt.title('Estimaciones IFFT')
26 plt.xlabel('Índice de Muestra')
27 plt.ylabel('Distancia Estimada en metros')
28 plt.legend()
29
30 # Phase Slope
31 plt.subplot(2, 2, 3)
32 plt.plot(indices, phase_slope_values, 'o-', label='Phase Slope')
33 plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
    ↪ ({dist})')
34 plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
    ↪ ({offset})')
35 plt.title('Estimaciones Phase Slope')
36 plt.xlabel('Índice de Muestra')
37 plt.ylabel('Distancia Estimada en metros')
38 plt.legend()
39
40 # RSSI Open Space
41 plt.subplot(2, 2, 4)
42 plt.plot(indices, rssi_openspace_values, 'o-', label='RSSI Open Space')
43 plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
    ↪ ({dist})')
44 plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
    ↪ ({offset})')
45 plt.title('Estimaciones RSSI Open Space')
46 plt.xlabel('Índice de Muestra')
47 plt.ylabel('Distancia Estimada en metros')
48 plt.legend()
49
```

```
50 plt.tight_layout()
51 plt.show()
52 plt.savefig('../../memoria/figuras/capitulo05/' + file_name + '-I' +
    ↪ '.pdf')
53
54 # Crear gráfico combinado
55 plt.figure(figsize=(10, 6))
56 plt.subplot(1, 2, 1)
57 plt.plot(indices, high_precision_values, 'o-', label='High Precision')
58 plt.plot(indices, ifft_values, 's-', label='IFFT')
59 plt.plot(indices, phase_slope_values, '^-', label='Phase Slope')
60 plt.plot(indices, rssi_openspace_values, 'd-', label='RSSI Open Space')
61 plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
    ↪ ({dist})')
62 plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
    ↪ ({offset})')
63 plt.title('Estimaciones Combinadas')
64 plt.xlabel('Índice de Muestra')
65 plt.ylabel('Distancia Estimada en metros')
66 plt.legend()
67
68 # BEST
69 plt.subplot(1, 2, 2)
70 plt.plot(indices, best_values, 'o-', label='Best')
71 plt.axhline(y=dist, color='r', linestyle='--', label=f'Valor Real
    ↪ ({dist})')
72 plt.axhline(y=offset, color='g', linestyle='--', label=f'Offset
    ↪ ({offset})')
73 plt.title('Estimaciones Best')
74 plt.xlabel('Índice de Muestra')
75 plt.ylabel('Distancia Estimada en metros')
76 plt.legend()
77
78 plt.tight_layout()
```

```
79 plt.show()
80 plt.savefig('../../memoria/figuras/capitulo05/' + file_name + '-C' +
    ↪  '.pdf')
```

Ahora los datos, al estar graficados, tanto por separado como todos juntos, serán mucho más fácilmente interpretables.

CAPÍTULO 5

MÉTODOS DE ESTIMACIÓN DE DISTANCIA Y ANÁLISIS DE RESULTADOS

5.1 Métodos de estimación de la distancia

En el modo MCPD de la librería nrf_dm, cada tipo de dato ofrece una aproximación diferente para estimar la distancia entre los dispositivos Bluetooth.

5.1.1 High Precision (Alta precisión)

- **Descripción:** Este método utiliza información basada en la fase de la señal recibida, lo que permite una mayor exactitud en la estimación de la distancia.[4]
- **Funcionamiento:** Se basa en la medición precisa del cambio de fase entre la señal transmitida y la recibida. Cuanto más precisa sea la medición de fase, mejor será la estimación de la distancia [4].
- **Comportamiento esperable:** Este es el modo más preciso entre todos, y debes esperar estimaciones bastante exactas, especialmente en distancias cortas o medianas (1,5 a 10 metros), aunque podría ser sensible a cambios bruscos en el entorno como reflexiones o interferencias [4].

5.1.2 IFFT (Inverse Fast Fourier Transform)

- **Descripción:** La IFFT es una técnica para obtener información del dominio de la frecuencia y convertirla al dominio del tiempo **[?]**
- **Funcionamiento:** Se basa en la transformación de la señal de RF (Radio Frecuencia), donde se mide el tiempo de vuelo de la señal a partir del análisis de su frecuencia. Esto permite obtener una estimación del retardo de la señal, que se puede convertir en una distancia [4].
- **Comportamiento esperable:** La precisión de este método será menor que la de High Precision, ya que depende del ancho de banda y de cómo se procesan las frecuencias. No obstante, se comporta mejor en entornos con menos interferencia [4].

5.1.3 Phase Slope (Pendiente de Fase)

- **Descripción:** Este método también se basa en mediciones de fase, pero se enfoca en la pendiente de la fase en diferentes frecuencias.[4]
- **Funcionamiento:** El método compara cómo cambia la fase de la señal a medida que varía la frecuencia de la misma, lo que proporciona una estimación de la distancia.[4]

- **Comportamiento esperable:** Aunque es menos preciso que High Precision, es más robusto frente a reflexiones o interferencias en entornos complicados. Se pueden esperar resultados consistentes en situaciones donde hay obstáculos o entornos más complejos.[4]

5.1.4 RSSI Openspace (RSSI en espacio abierto)

- **Descripción:** Este método utiliza el RSSI (Received Signal Strength Indicator), que mide la potencia de la señal recibida.[3]
- **Funcionamiento:** Se basa en la suposición de que la señal se atenúa a medida que aumenta la distancia entre los dispositivos, lo cual se estima utilizando el modelo de propagación de espacio libre (openspace). Es el método más básico y depende del entorno [3].
- **Comportamiento esperable:** Las estimaciones de distancia usando RSSI suelen ser menos precisas y muy sensibles a las condiciones del entorno. Funciona mejor en entornos abiertos y sin obstáculos, pero puede ser menos fiable en interiores o lugares con muchas interferencias. Además, en distancias largas (más de 10 metros), los errores pueden aumentar [3].

5.2 Calcular el valor del Offset

Como ya se comentó anteriormente en la sección 4.1.1, Zero-distance calibration (Offset), es necesario calibrar las muestras, puesto que estas presentan un offset.

En base a los resultados obtenidos que se pueden observar en las figuras 5.1 y 5.2. Hemos determinado el valor del offset en **0,3 metros**. Que se obtiene de la diferencia entre el valor de la distancia real (0,6 metros) y el valor más repetido (0,9 metros).

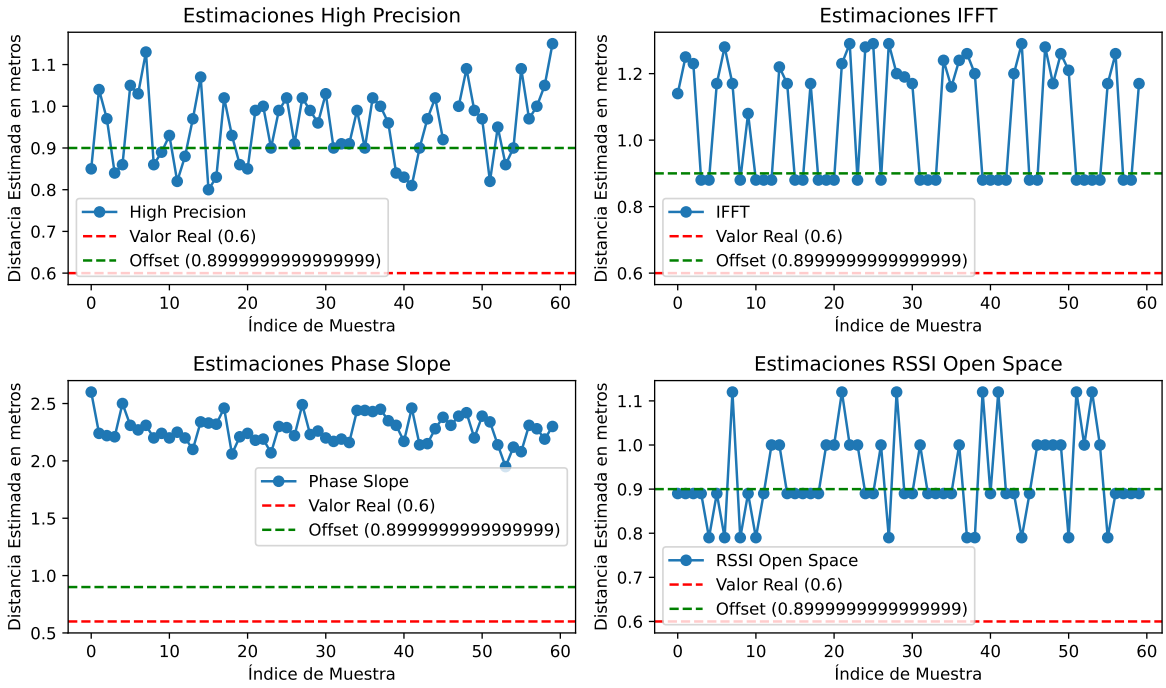

Figura 5.1: Gráficos individuales de Offset

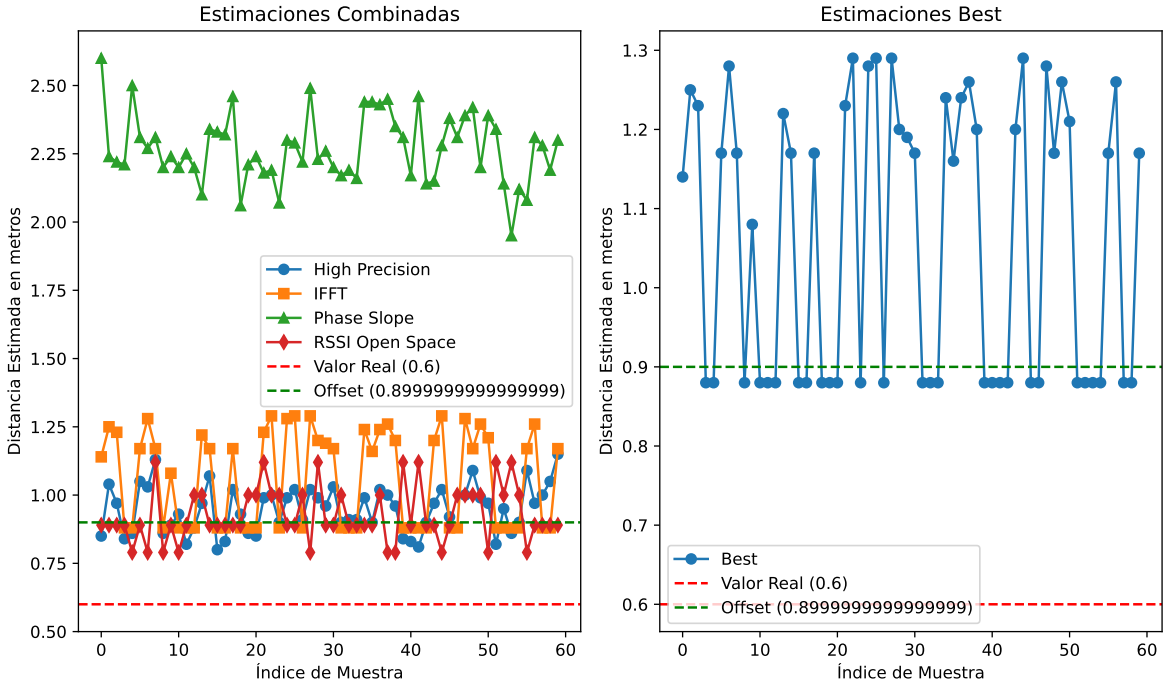

Figura 5.2: Gráfico conjunto y Best de Offset

5.3 Análisis de los Datos

Vamos a analizar los datos separándolos en función de la distancia real a estimar.

5.3.1 Distancia de 3 metros

Como reflejan las figuras 5.3 y 5.4, la mejor precisión, en este caso, se consigue en la habitación. Esto se debe a que la geometría del pasillo provoca mayor error en las mediciones. Cabe destacar que ambos métodos tienen bastante error.

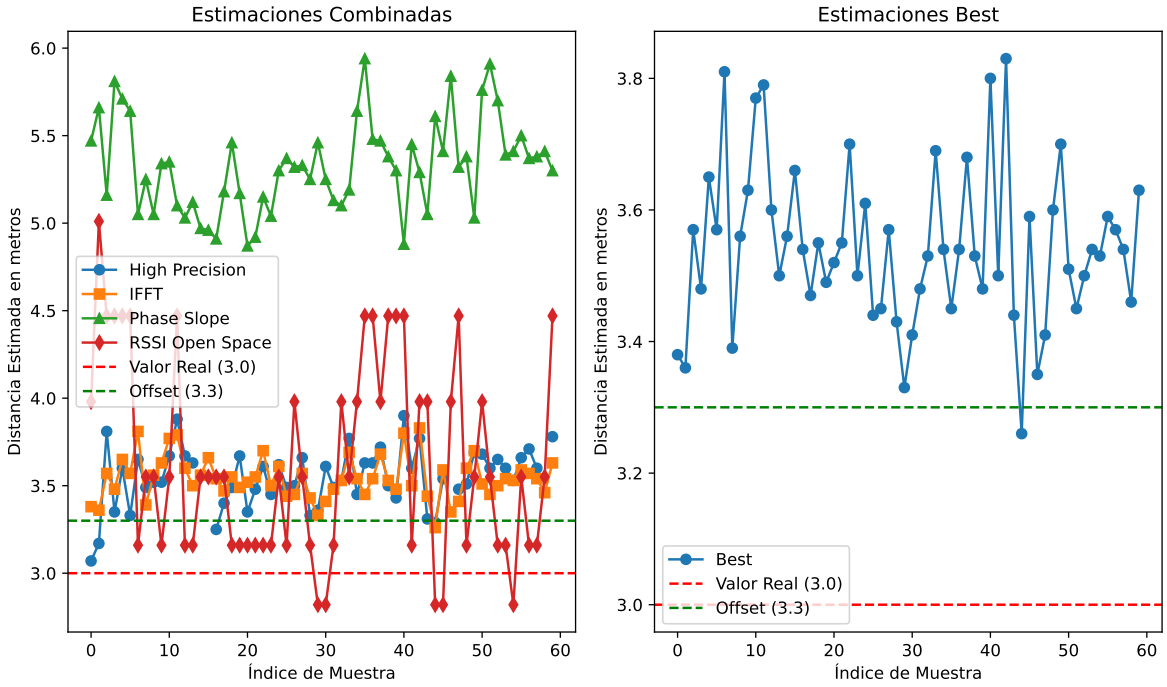

Figura 5.3: Gráfico conjunto y Best en Habitación a 3 metros

- **Habitación:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones (5.1.1 - 5.1.2). Se puede observar en la figura 5.5.
- **Pasillo:** Los mejores resultados los ha proporcionado el IFFT, como era de esperar en este entorno y estas condiciones (5.1.2). En este caso RSSI presenta mucho más error que en las mediciones de la habitación, esto es debido a que funciona mejor en entornos abiertos y sin obstáculos, pero puede ser menos fiable en lugares con muchas interferencias (5.1.4). Se puede observar en la figura 5.6.

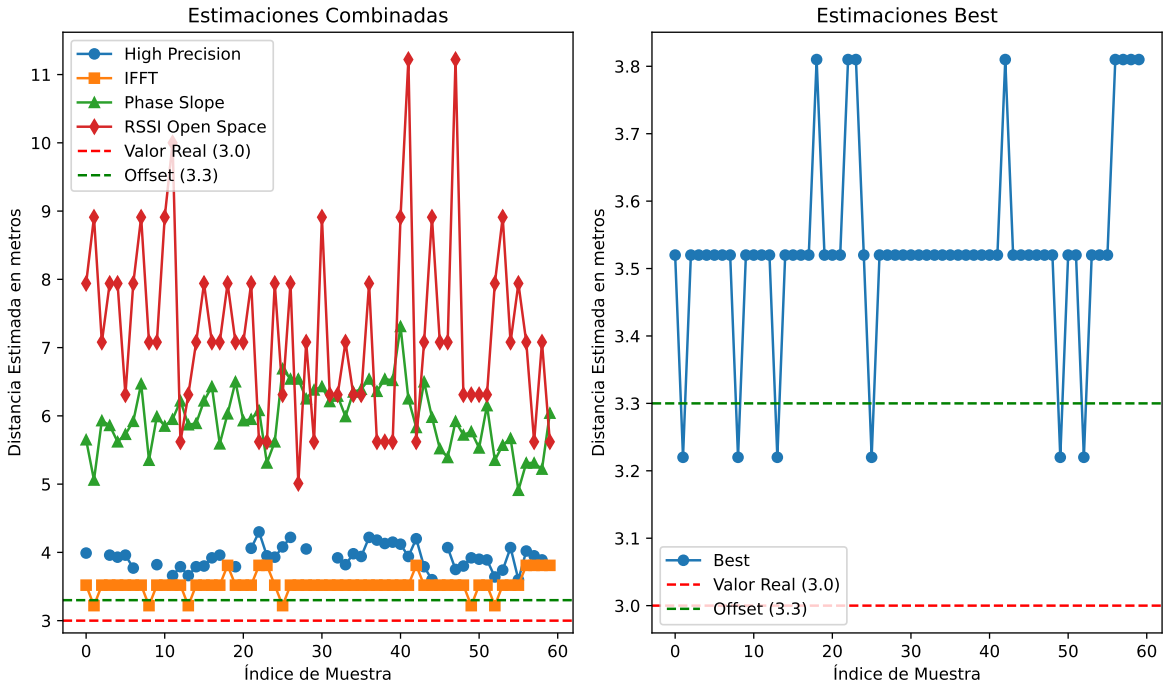

Figura 5.4: Gráfico conjunto y Best en Pasillo a 3 metros

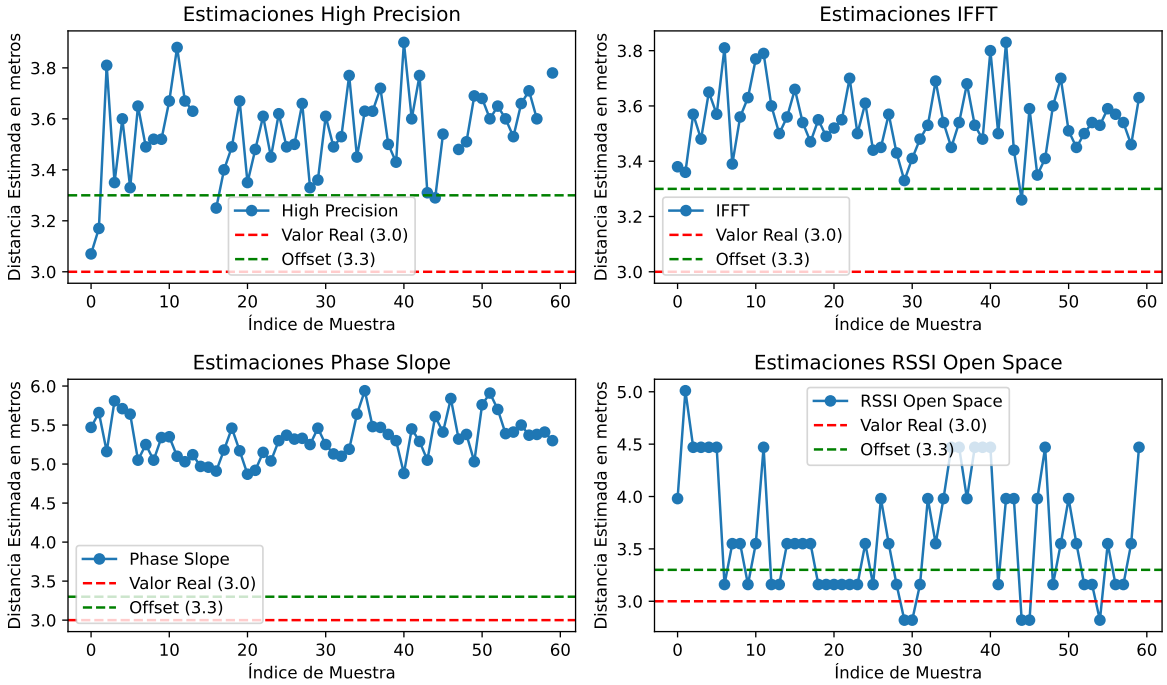

Figura 5.5: Gráficos individuales en Habitación a 3 metros

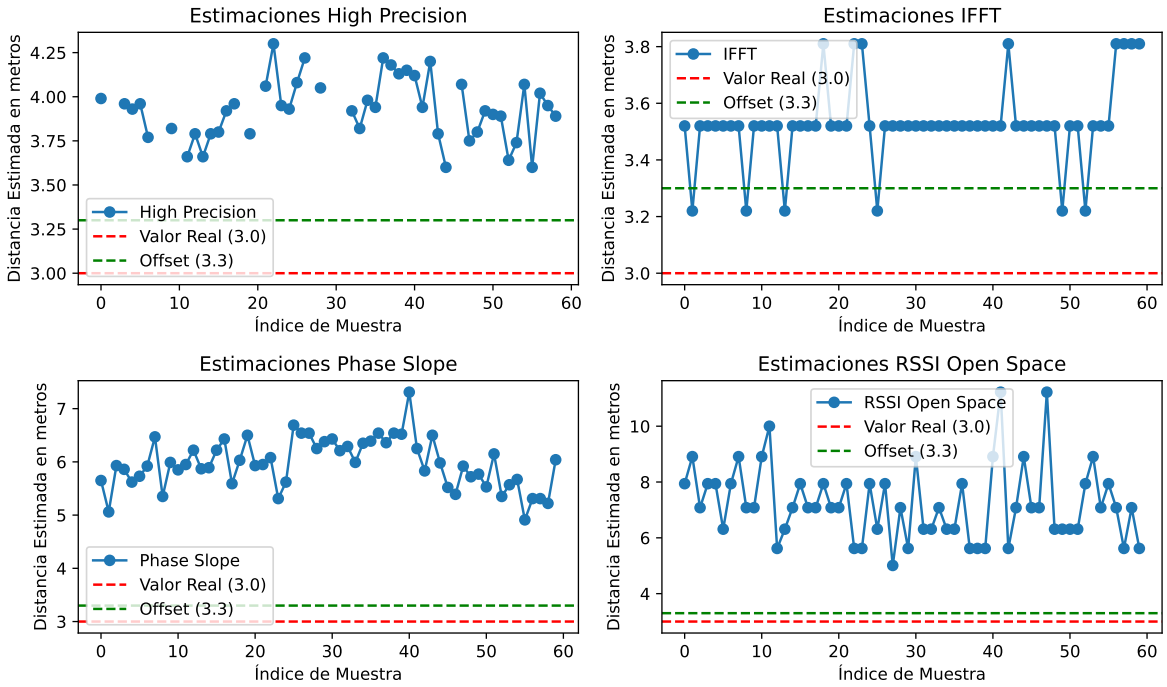

Figura 5.6: Gráficos individuales en Pasillo a 3 metros

5.3.2 Distancia de 6 metros

Como reflejan las figuras 5.7, 5.8 y 5.9, la mejor precisión, en este caso, se consigue en campo abierto. Esto se debe a que en campo abierto hay menos obstáculos que puedan producir errores en las mediciones.

- **Campo abierto:** Todos los métodos proporcionan buenas estimaciones en este caso (5.1.1 - 5.1.2 - 5.1.3), a excepción del RSSI que comienza dando buenos resultados, pero en general, presenta demasiado error. Se puede observar en la figura 5.10.
- **Pasillo:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones (5.1.2). High precision presente algunos valores nan que podrían ser debidos a cambios bruscos en el entorno como reflexiones o interferencias (5.1.1). Se puede observar en la figura 5.11.
- **Bosque:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones, aunque high precision presenta peores estimaciones (5.1.1 - 5.1.2). Por otra se puede observar como,

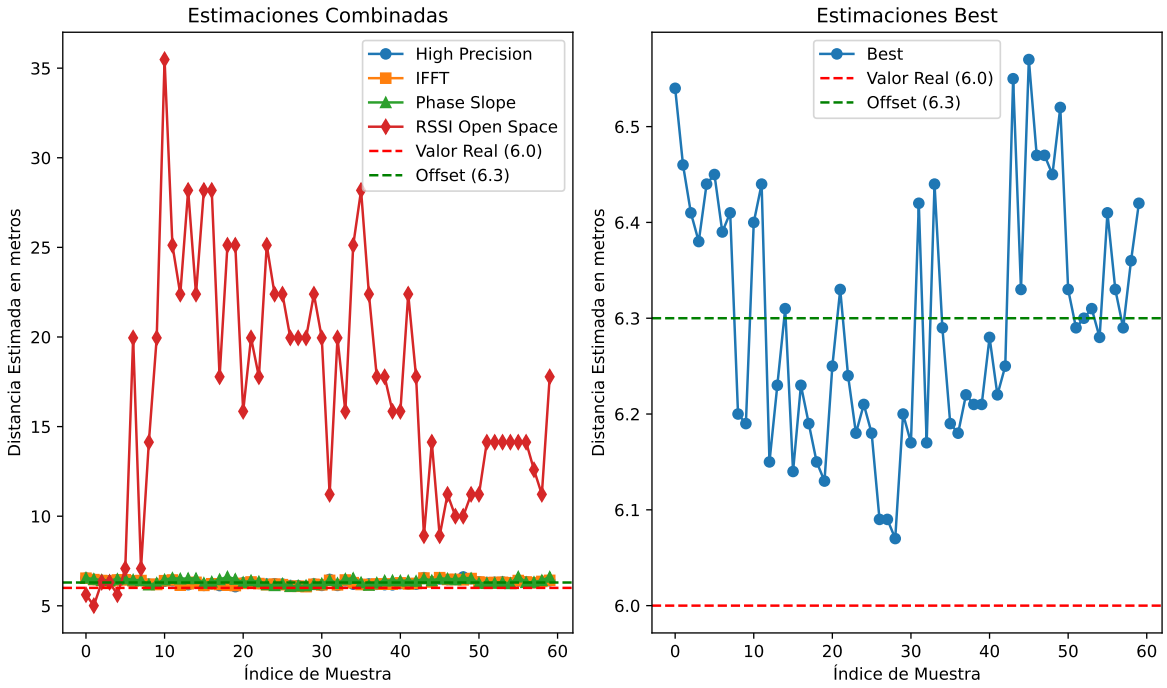

Figura 5.7: Gráfico conjunto y Best en Campo Abierto a 6 metros

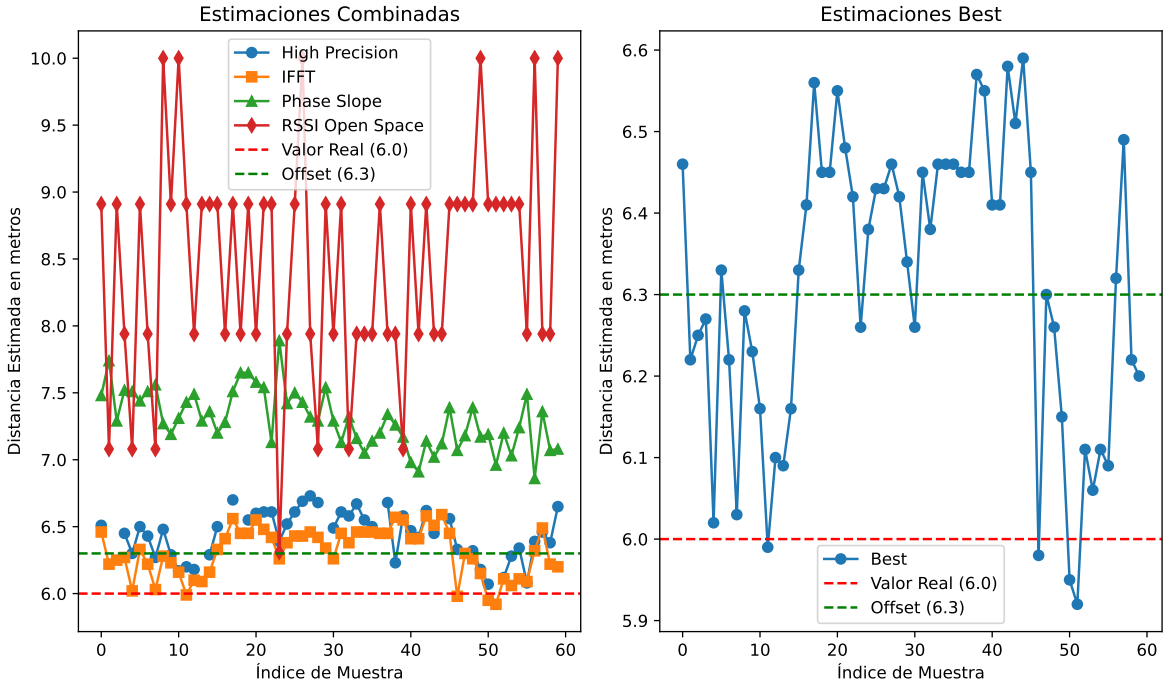

Figura 5.8: Gráfico conjunto y Best en Pasillo a 6 metros

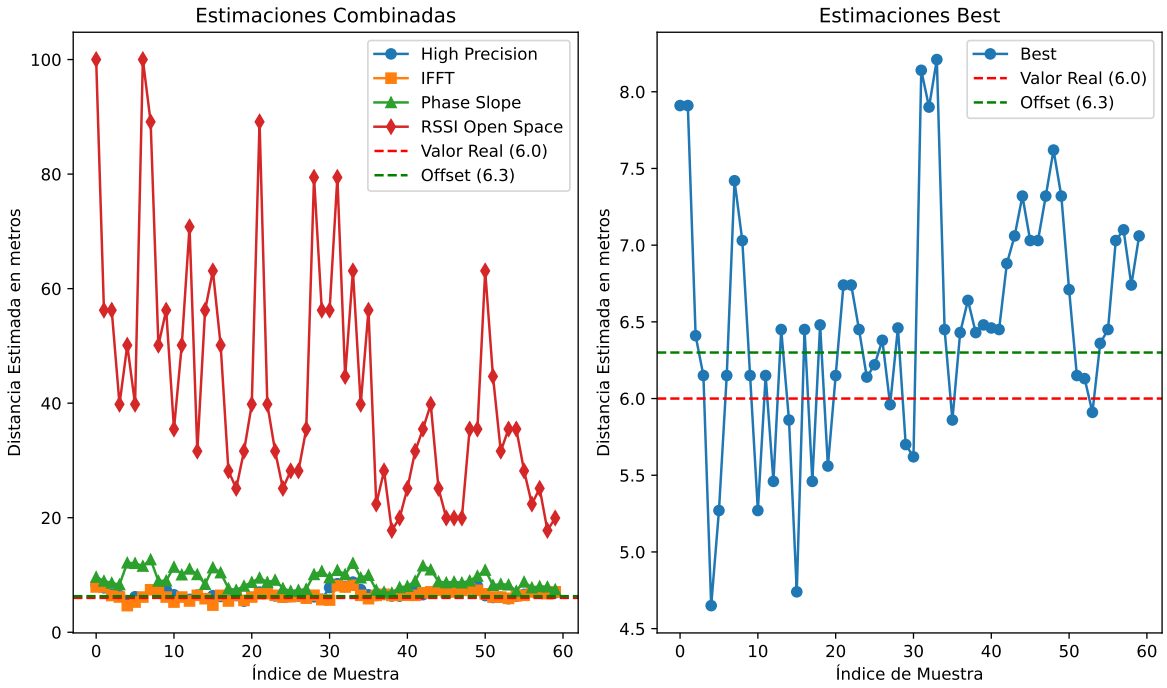

Figura 5.9: Gráfico conjunto y Best en Bosque a 6 metros

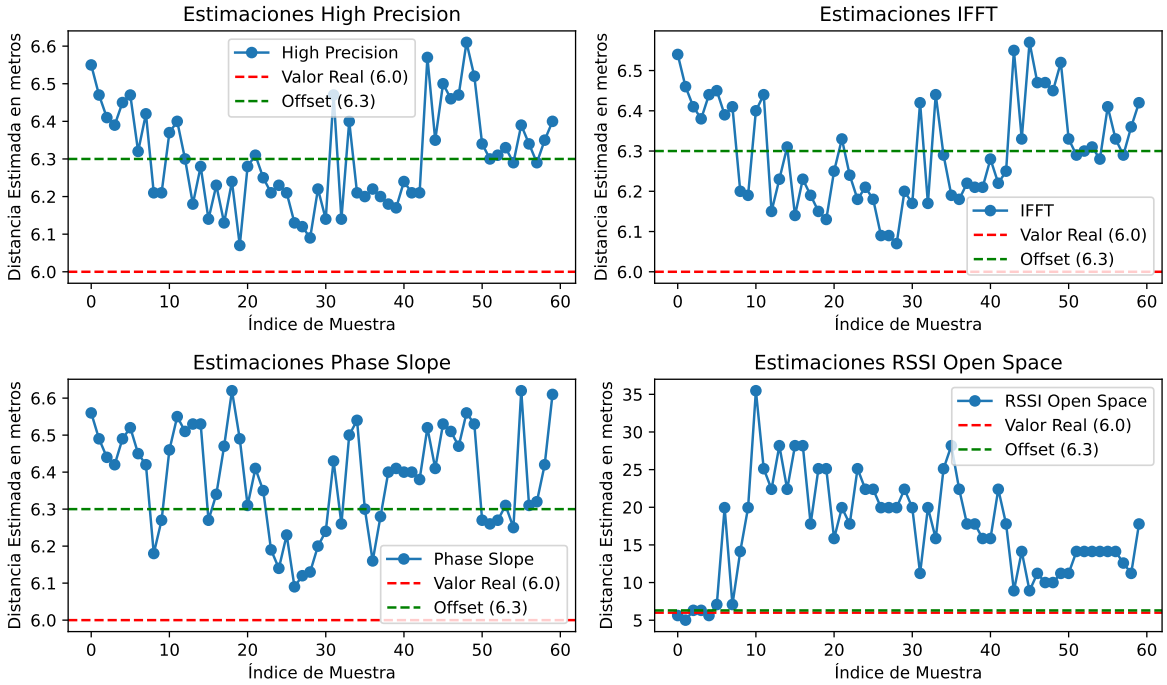

Figura 5.10: Gráficos individuales en Campo Abierto a 6 metros

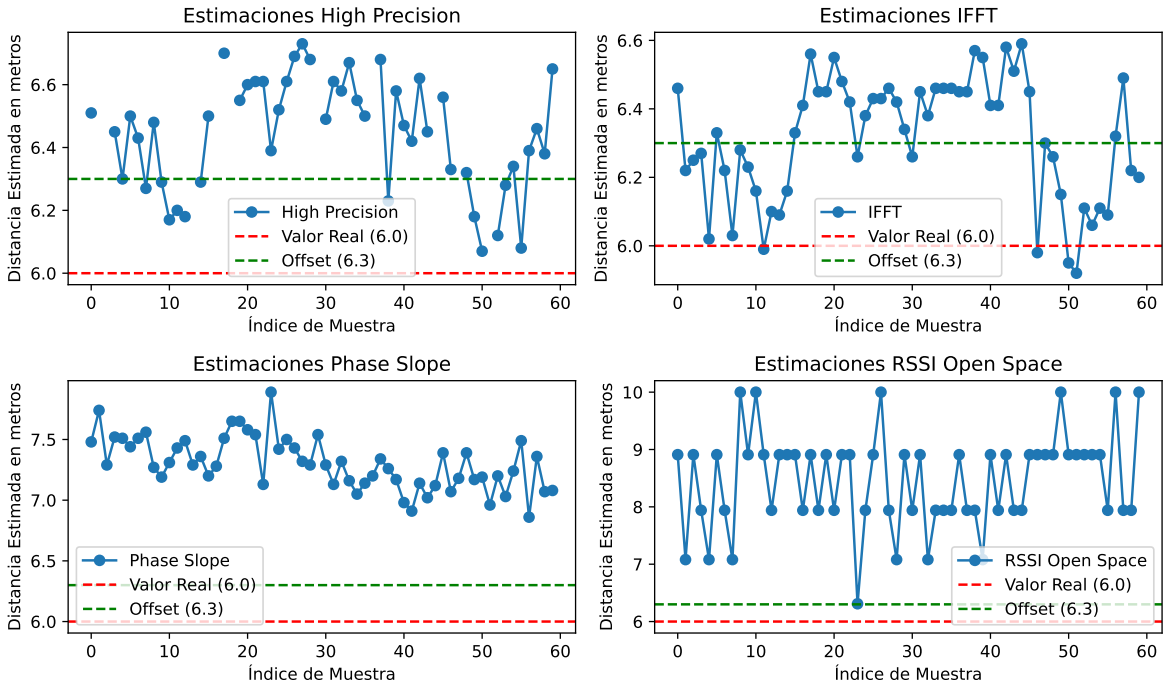

Figura 5.11: Gráficos individuales en Pasillo a 6 metros

debido a las interferencias producidas por el entorno de árboles y arbustos, los datos en general se presentan mucho más dispersos que en otros entornos. El claro reflejo es la cantidad de error en las medidas del método RSSI. Se puede observar en la figura 5.12.

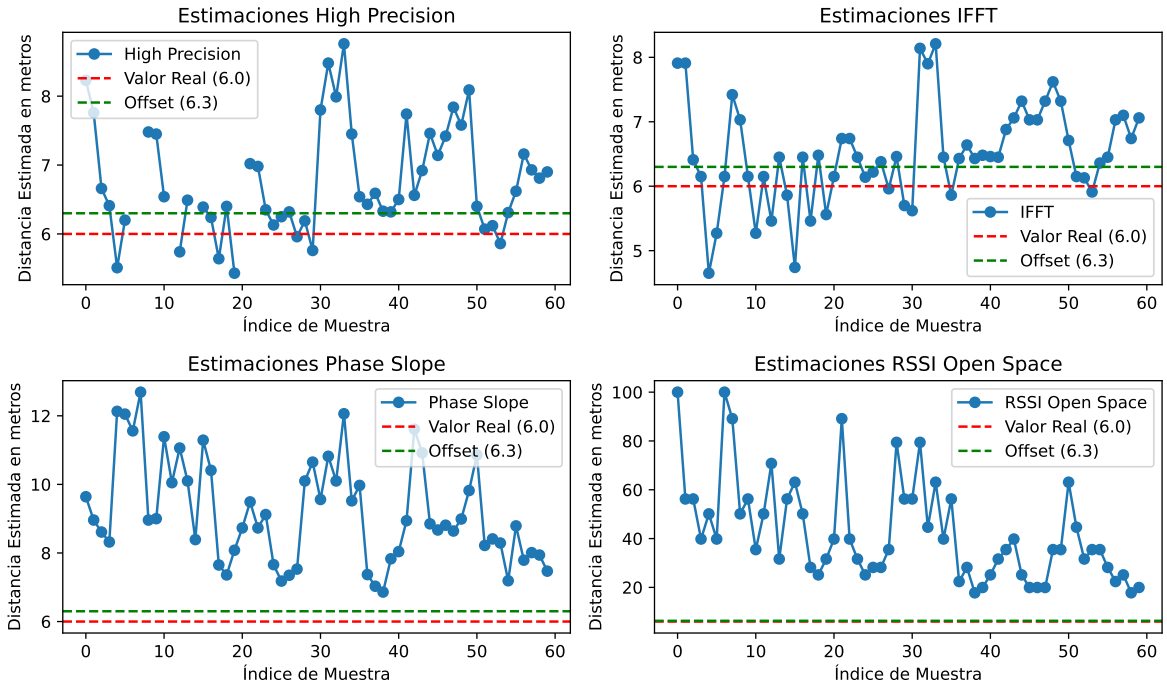

Figura 5.12: Gráficos individuales en Bosque a 6 metros

5.3.3 Distancia de 10 metros

Como reflejan las figuras 5.13 y 5.14, la mejor precisión, en este caso, se consigue también en campo abierto. Se puede observar que ahora el método RSSI presenta mucho más error, esto es debido al aumento de distancia. El resto de métodos siguen teniendo bastante precisión, como comentaremos a continuación.

- **Campo abierto:** Los mejores resultados los ha proporcionado el IFFT, como era de esperar en este entorno y estas condiciones (5.1.2). Son mejores resultados que en el resto de entornos. Cabe destacar que high precision presenta algunos datos con mucho error. Se puede observar en la figura 5.15.
- **Bosque:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones (5.1.1 - 5.1.2). También cabe destacar que en este caso high precision presenta mayor densidad de valores nan. Se puede observar en la figura 5.16.

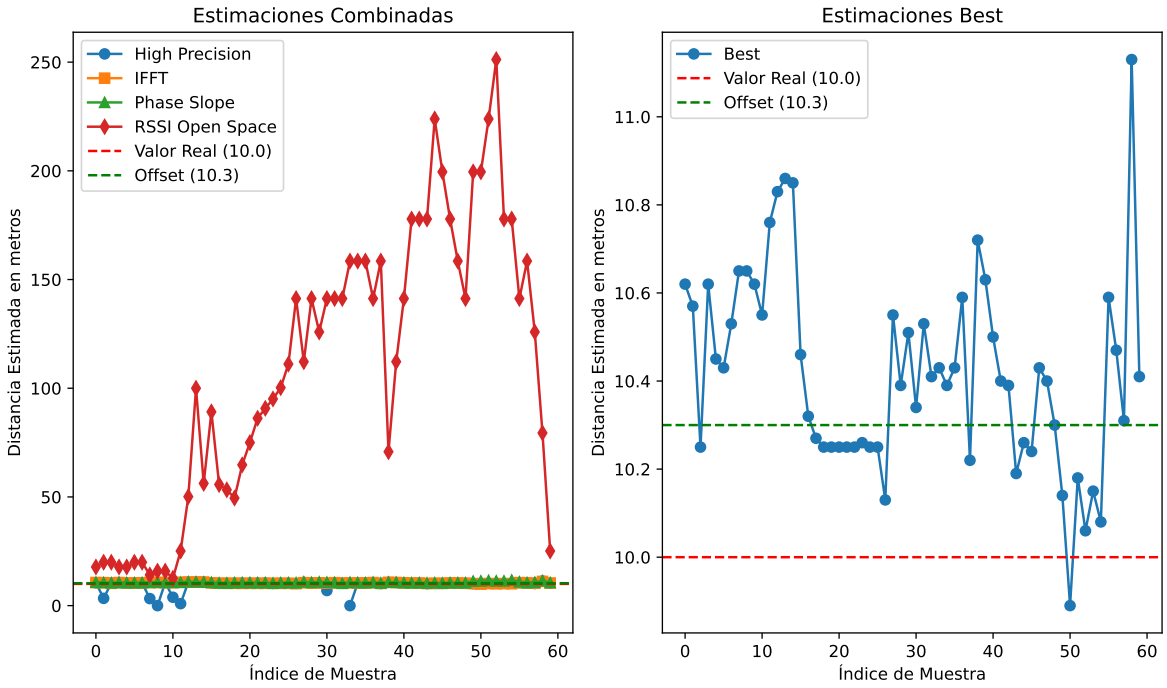

Figura 5.13: Gráfico conjunto y Best en Campo Abierto a 10 metros

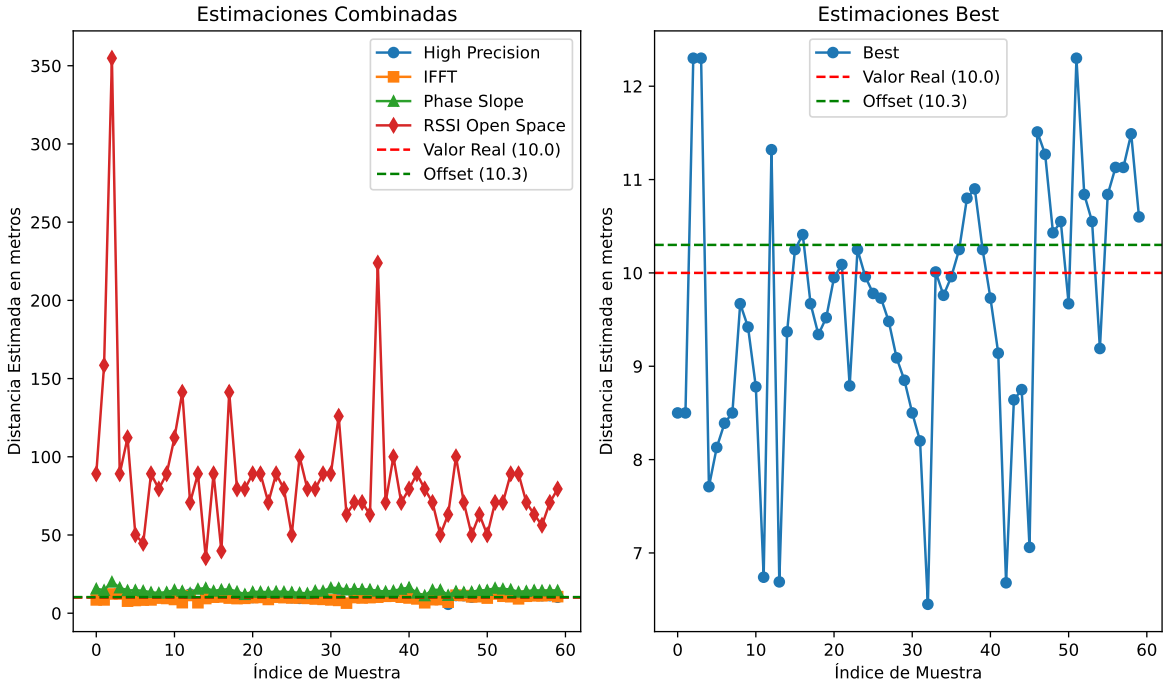

Figura 5.14: Gráfico conjunto y Best en Bosque a 10 metros

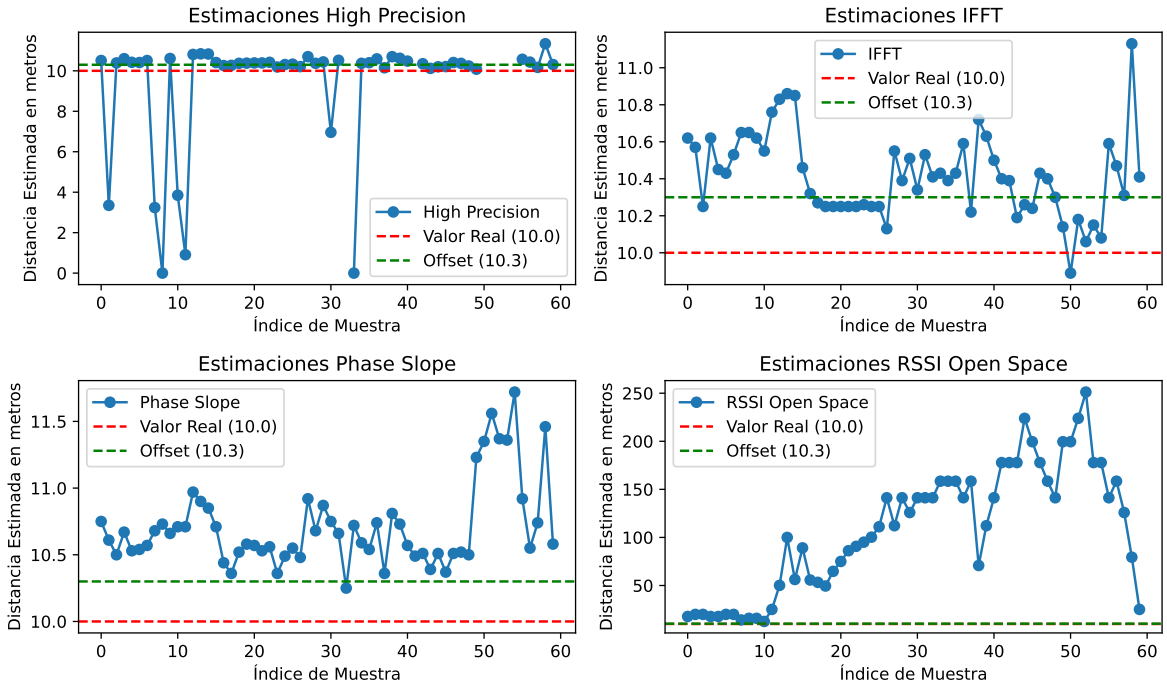

Figura 5.15: Gráficos individuales en Campo Abierto a 10 metros

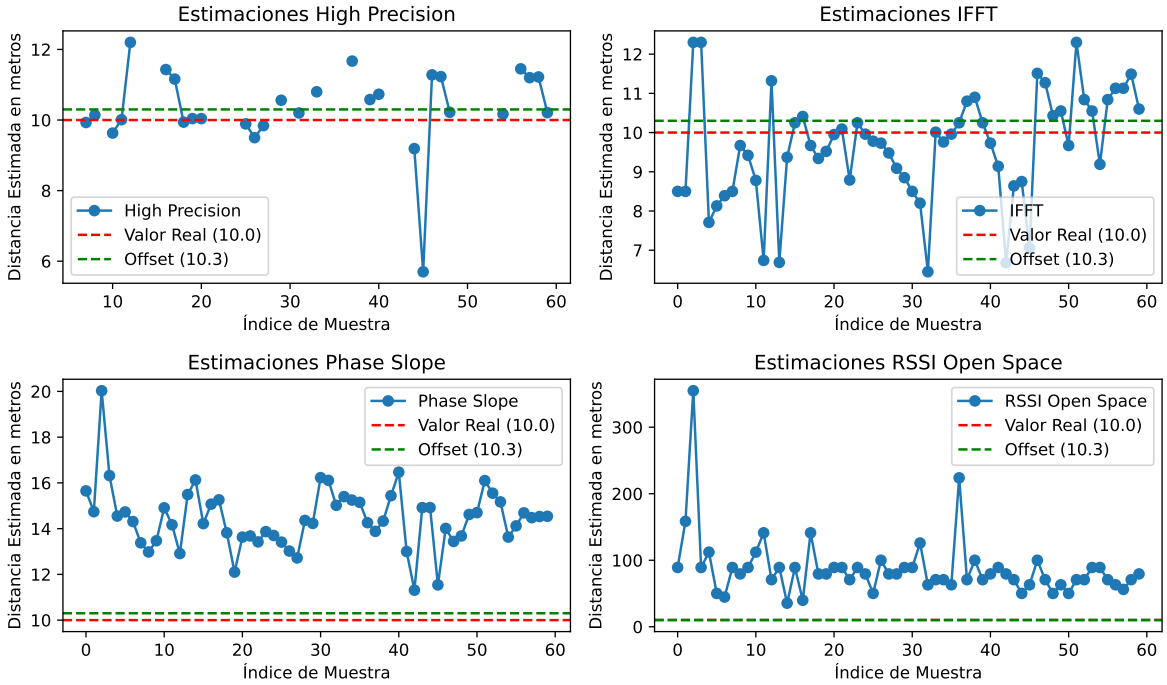

Figura 5.16: Gráficos individuales en Bosque a 10 metros

5.3.4 Distancia de 20 metros

Como reflejan las figuras 5.17 y 5.18, la mejor precisión, en este caso, se consigue en el bosque. Se puede ver reflejado en las muestras que a 20 metros ya nos aproximamos a la distancia máxima que soporta una conexión BLE. Aun así, la gráfica best del bosque tiene gran precisión.

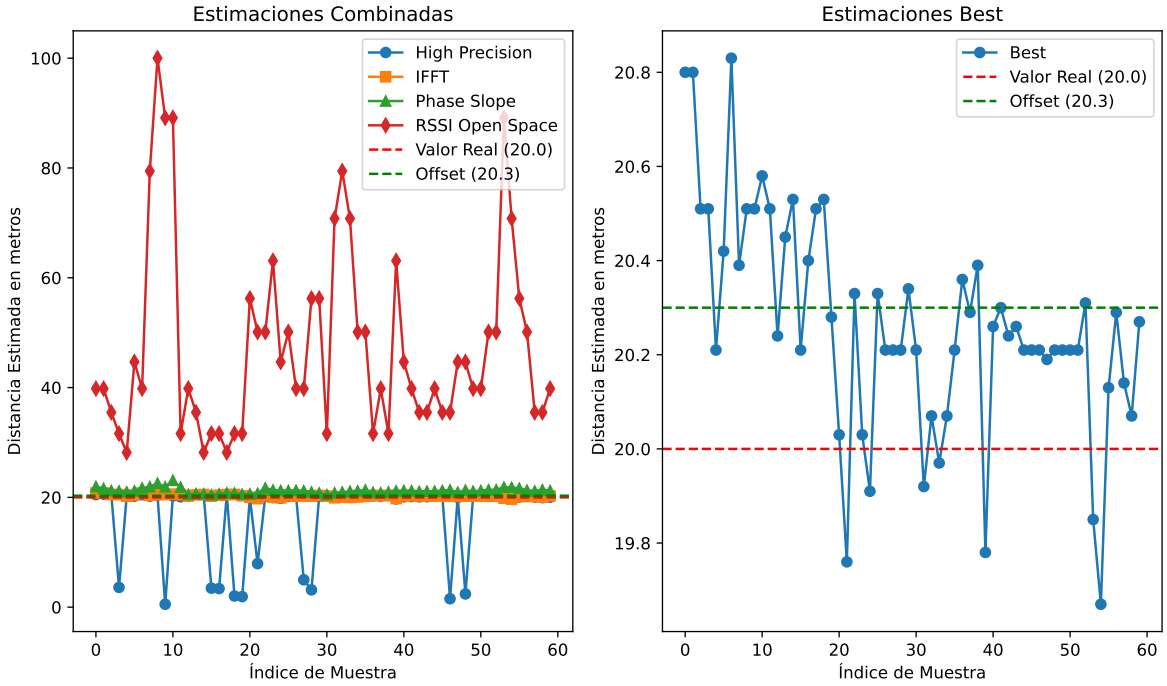

Figura 5.17: Gráfico conjunto y Best en Bosque a 20 metros

- **Bosque:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones (5.1.2). High precision vuelve a presentar algunos datos con mucho error. Se puede observar en la figura 5.19.
- **Campo abierto:** Los mejores resultados los han proporcionado el high precision y el IFFT, como era de esperar en este entorno y estas condiciones (5.1.1 - 5.1.2). High precision vuelve a presenta gran densidad de valores nan. Se puede observar en la figura 5.20.

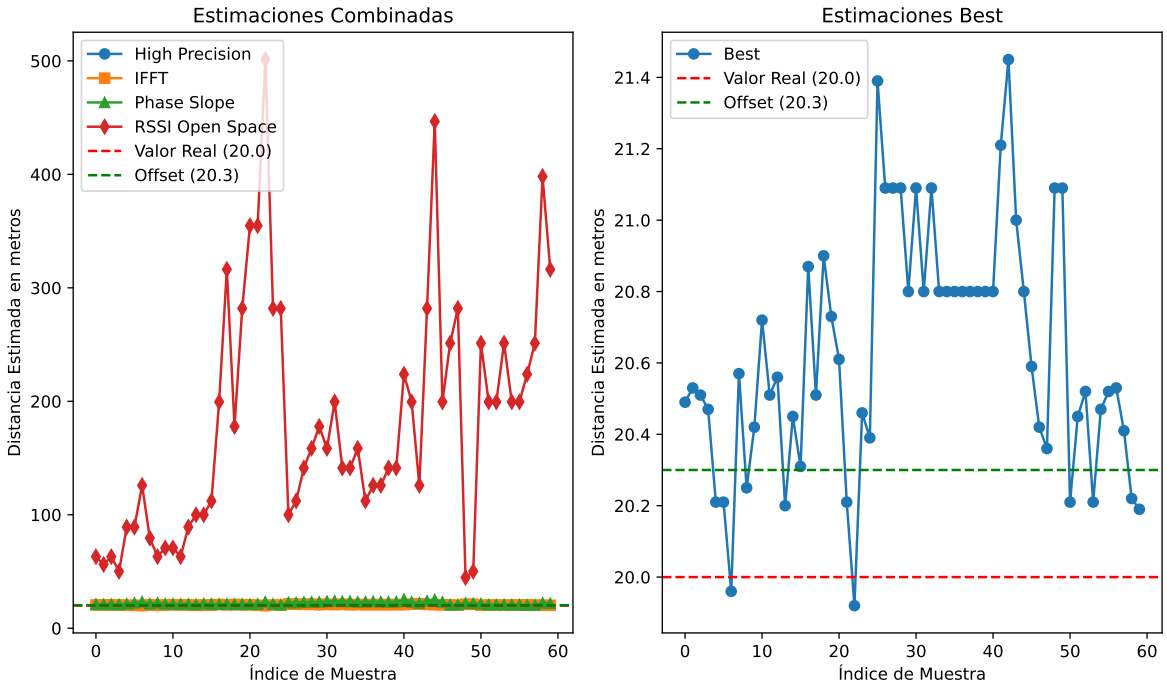

Figura 5.18: Gráfico conjunto y Best en Campo Abierto a 20 metros

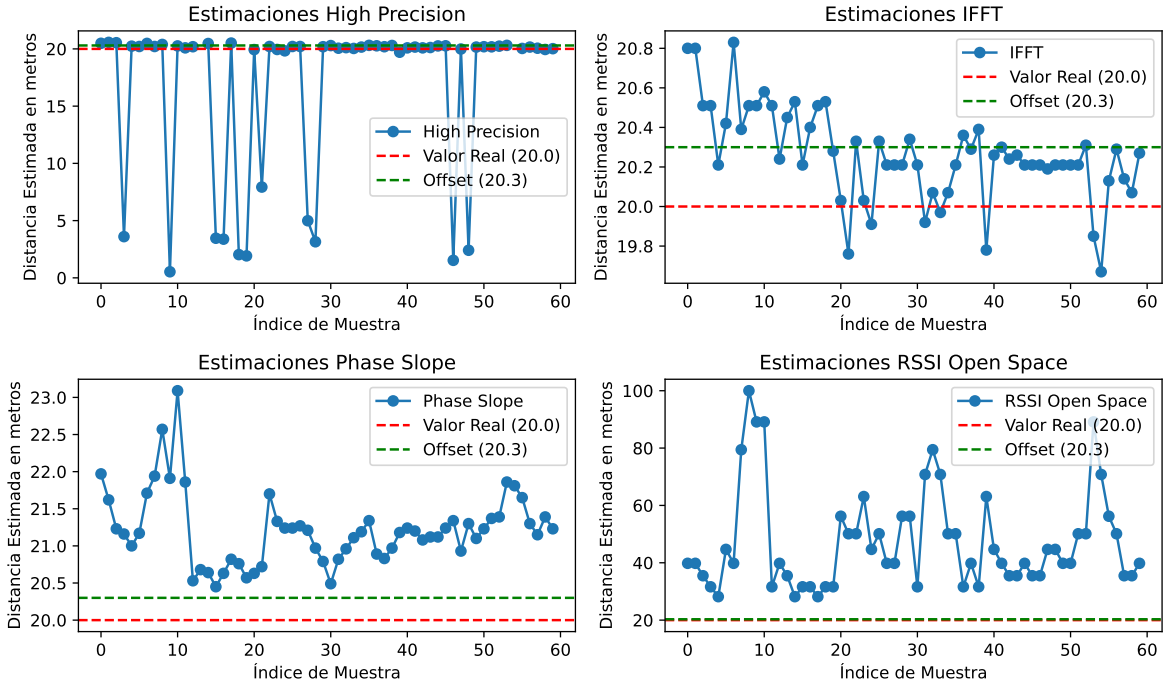

Figura 5.19: Gráficos individuales en Bosque a 20 metros

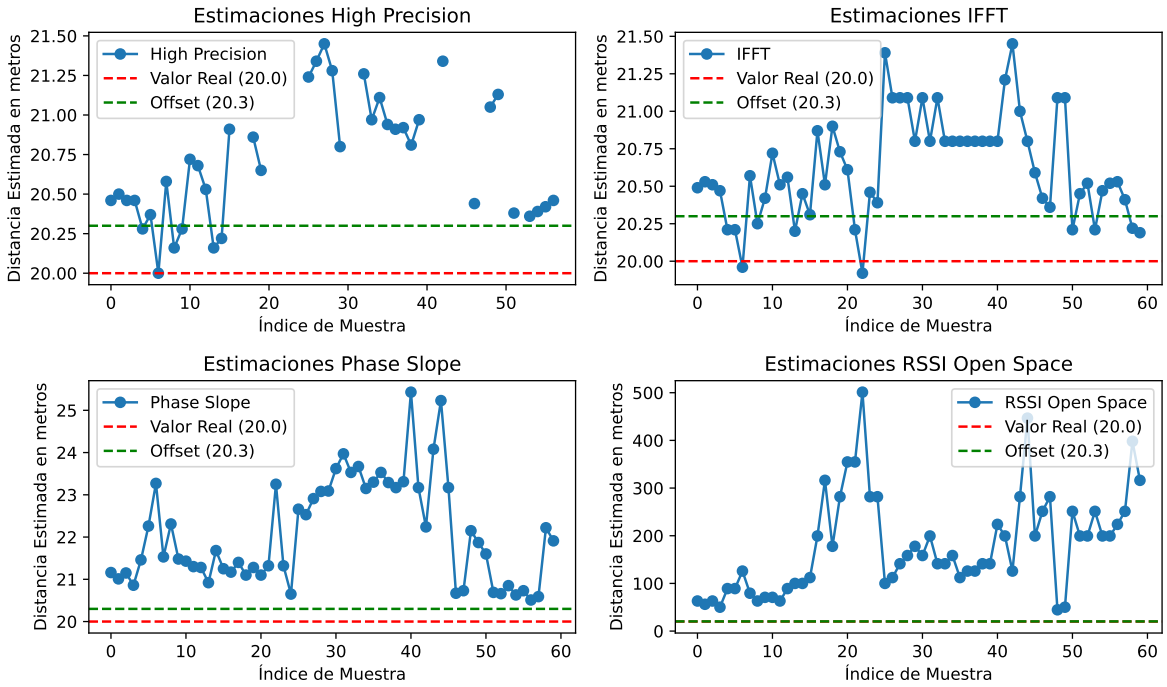

Figura 5.20: Gráficos individuales en Campo Abierto a 20 metros

5.3.5 Distancia de 1,5 metros

De manera adicional hemos recogido muestras en la habitación a una distancia de 1,5 metros. Se pueden observar en las figuras 5.21 y 5.22.

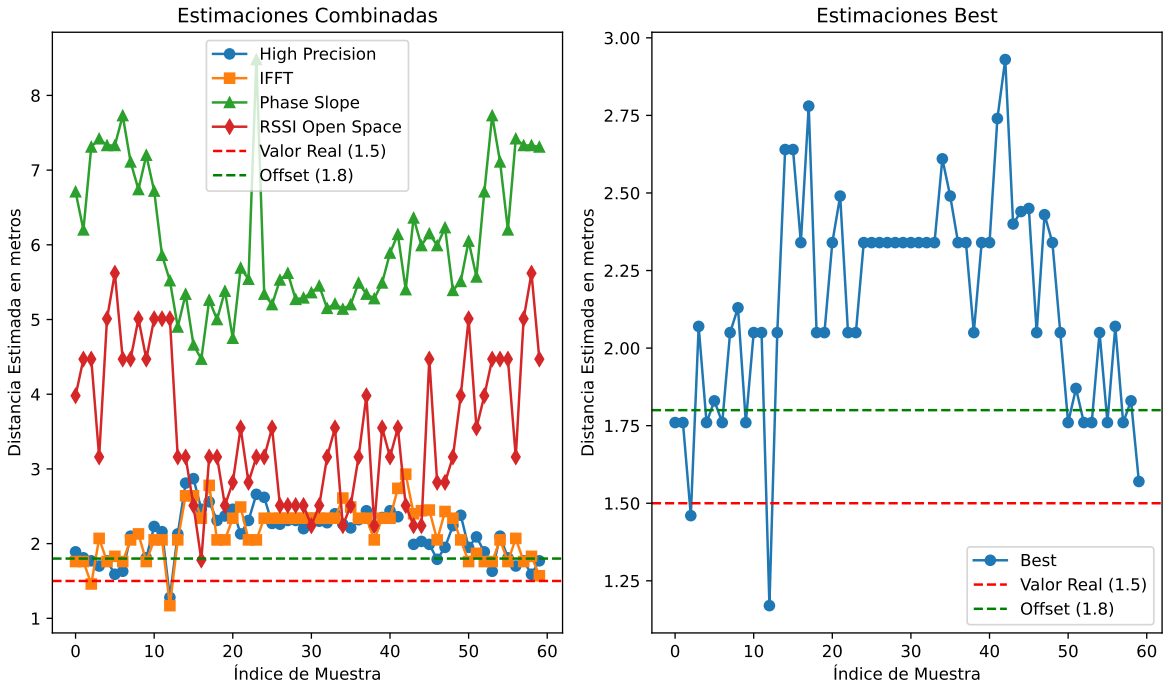

Figura 5.21: Gráfico conjunto y Best en Habitación a 1,5 metros

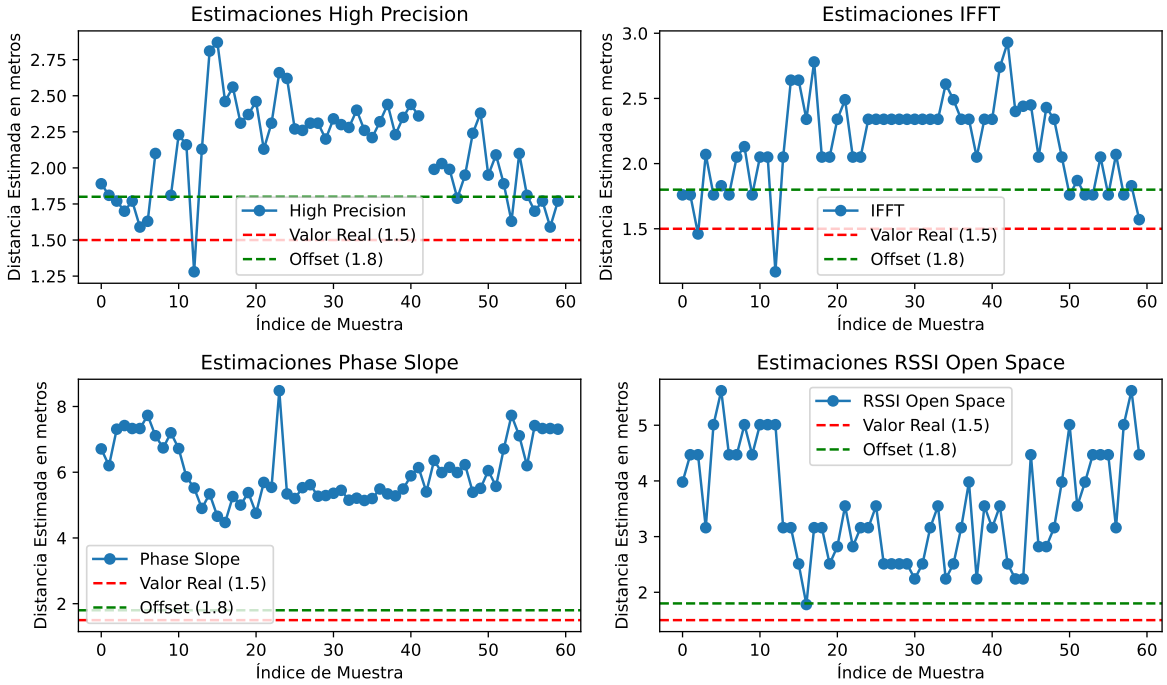

Figura 5.22: Gráficos individuales en Habitación a 1,5 metros

CAPÍTULO 6

CONCLUSIONES

6.1 Conclusiones obtenidas de los datos

El análisis de los datos obtenidos a partir de las mediciones de distancia utilizando la librería nrf_dm ha revelado que, en términos generales, existe un margen de error significativo en las estimaciones. Este error es particularmente notable en algunos métodos específicos, lo cual quedó reflejado en el análisis de las muestras en diferentes entornos y distancias. A pesar de que la tecnología BLE y la librería nrf_dm ofrecen una aproximación viable para la estimación de distancias, la precisión no es constante y varía según el método y las condiciones del entorno.

Uno de los elementos que merece especial atención es el método “Best” proporcionado por la librería nrf_dm. Este método selecciona automáticamente el valor que considera como la mejor estimación de distancia, basado en los cuatro métodos disponibles. Sin embargo, se ha observado que en muchos casos, la elección del “Best"no coincide con el valor más preciso o real, lo que sugiere que el criterio de selección puede mejorarse. Esto abre la puerta a futuras investigaciones enfocadas en optimizar este proceso de selección.

Una posible línea de mejora sería emplear el análisis realizado en este trabajo para ponderar qué método es más adecuado bajo determinadas circunstancias. Por ejemplo, podría ajustarse la selección del "Best" en función del entorno, la distancia o las condiciones de la señal, lo que permitiría un enfoque más dinámico y adaptado a las características de cada escenario. Otra posibilidad, compatible con la anterior, sería realizar la elección del "Best" una vez se hayan recopilado todas las muestras, evaluando de manera global cuál es la mejor estimación en función del conjunto de datos. Esto difiere del enfoque actual de la librería nrf_dm, que realiza una selección en tiempo real para cada estimación individual. Esta adaptación podría ser explorada en futuros proyectos para mejorar la precisión de las estimaciones.

6.2 Conclusiones y líneas futuras

El presente Trabajo de Fin de Grado ha logrado cumplir con los objetivos planteados inicialmente, sentando una base sólida tanto teórica como práctica para futuros proyectos basados en la tecnología Bluetooth Low Energy (BLE). En primer lugar, se ha ofrecido una explicación técnica completa de BLE, proporcionando un marco teórico esencial para aquellos que deseen trabajar en este campo. Asimismo, se ha desarrollado un tutorial práctico que guía al lector en la instalación y configuración del entorno de desarrollo de Nordic Semiconductor, facilitando el inicio en el uso de dispositivos nRF52840 y la librería nrf_dm.

En el aspecto práctico, se ha creado un programa en Python que permite procesar las muestras obtenidas a través de la librería nrf_dm, transformando los datos crudos en información visualizable y fácilmente interpretable mediante gráficos. Este programa, incluido en el TFG, agiliza el análisis de los datos y puede ser utilizado como herramienta directa en proyectos futuros.

El análisis de las muestras obtenidas en diversos entornos y distancias ha permitido obtener conclusiones relevantes sobre el comportamiento de los diferentes métodos de estimación de distancia de la librería nrf_dm. Aunque se ha identificado la presencia de errores significativos en las estimaciones, especialmente en ciertos métodos, el trabajo ofrece sugerencias claras sobre posibles mejoras. Entre ellas, se destaca la necesidad de optimización y de elección apropiada del método "Best" y la posibilidad de ajustar dicha elección en función de las características del entorno o realizarla a posteriori de haber recogido todas las muestras.

En conclusión, este TFG no solo contribuye con conocimientos y herramientas útiles para el análisis de distancias mediante BLE, sino que también propone líneas de trabajo futuras que podrían mejorar la precisión y aplicabilidad de las estimaciones en entornos reales. Por tanto, el trabajo no solo cumple con su propósito inmediato, sino que abre la puerta a nuevas investigaciones que podrían perfeccionar el uso de la tecnología BLE en aplicaciones de localización y conectividad.

Índice de figuras

Índice de tablas

Índice de código fuente

BIBLIOGRAFÍA

[1] DevAcademy: *Bluetooth low energy fundamentals*. Nordik Semiconductor, 2024. https://academy.nordicsemi.com/courses/bluetooth-low-energy-fundamentals/.

[2] DevZone DevAcademy: *Bluetooth: nrf distance measurement with bluetooth le discovery*. Nordik Semiconductor, sep 2024. https://docs.nordicsemi.com/bundle/ncs-latest/page/nrf/samples/bluetooth/nrf_dm/README.html.

[3] Ramsey Faragher y Robert Harle: *Location fingerprinting with bluetooth low energy beacons*. IEEE Journal on Selected Areas in Communications, 33(11):2418–2428, 2015.

[4] Neal Patwari, Joshua N. Ash, Spyros Kyperountas, Alfred O. Hero III, Robert L. Moses y Neal S. Correal: *Locating the nodes: cooperative localization in wireless sensor networks*. IEEE Signal Processing Magazine, 22(4):54–69, 2005.

[5] Nordik Semiconductor: *Broad portfolio of bluetooth low energy socs*. Nordik Semiconductor, 2024. https://www.nordicsemi.com/Products/Wireless/Bluetooth-Low-Energy/Development-hardware?lang=en#infotabs.