Universidad Politécnica
de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Estudio de los Protocolos Empleados por Bluetooth y Desarrollo de un Programa para su Análisis.

Autor: Héctor Borreguero Monleón

Tutor(a): Luis Mengual Galán

Madrid, enero 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática
Título: Estudio de los Protocolos Empleados por Bluetooth y Desarrollo de un Programa para su Análisis.
Enero 2025

Autor: Héctor Borreguero Monleón

Tutor:
Luis Mengual Galán
LENGUAJES Y SISTEMAS INFORMÁTICOS E INGENIERÍA DE SOFTWARE
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

En el desarrollo de este proyecto se pueden diferenciar tres partes claramente, las cuales contribuyen al objetivo final: servir como introducción al mundo de las tecnologías Bluetooth. Estas tres secciones son las siguientes: “Teoría”, “Desarrollo del Programa” y “Casos de Uso”. Este trabajo cuenta con una introducción que describe el enfoque y los objetivos de este.

En la parte de “Teoría”, se expone a que nos referimos con la palabra “Bluetooth” de forma más concreta. Bluetooth hace referencia a dos tecnologías distintas: Bluetooth Classic y Bluetooth Low Energy. A menudo nos referimos a Bluetooth como si fuera una sola tecnología, pero estas dos son incompatibles, aunque compartan ciertas características. Se hace una breve introducción de la función común y características físicas que estas comparten y de las características que diferencian ambas tecnologías. Una vez hecho esto, se estudia en mayor detalle cada una de estas. Estudiamos los protocolos más comunes y componentes más importantes de Classic y LE para sentar unas bases que nos permitan entender la comunicación entre dispositivos conectados por Bluetooth. Se dedican también dos secciones suplementarias al HCI y a los codecs, los cuales son partes fundamentales en la comunicación.

En la sección “Desarrollo del Programa”, se describe paso a paso como se ha desarrollado el programa. Contiene apartados para cada uno de los elementos indispensables en el proyecto, además de una pequeña introducción que indica lo que ha guiado el desarrollo. El apartado de versionamiento explica cómo se ha creado un repositorio GitHub para preservar y actualizar el trabajo. En el siguiente apartado se explica punto por punto cómo obtener las fuentes de los datos Bluetooth que vamos a analizar y en el que viene a continuación, qué programa se usa para tratar los datos y que datos se seleccionan según cada protocolo. Por último, se explica cómo configurar un entorno virtual de python y como se ha programado y el aspecto y funcionalidades que tiene el programa.

Por último, ponemos a prueba los conocimientos obtenidos y el programa mediante 7 casos de prueba con 7 dispositivos diferentes que usan tecnología Bluetooth. La finalidad de esto es mostrar la utilidad que tiene el programa a la hora de analizar tráfico Bluetooth de cualquier tipo y de cómo con una introducción para nada extensa, se puede comenzar a comprender cómo funcionan las tecnologías Bluetooth. Para cada uno de estos casos de prueba, se establecen varios tests que permiten ver los distintos paquetes que se envían relacionados a cada uno de los protocolos usados durante la comunicación. Se analizan los paquetes para intentar averiguar cómo funcionan los dispositivos y cómo utilizan los distintos protocolos para cada situación. En ocasiones se pueden distinguir patrones que nos permiten hacer hipótesis bastante factibles y en otras resulta más difícil, dependiendo en gran medida de cómo los desarrolladores de estos dispositivos hayan decidido implementar los envíos de información.

Como se puede observar, es un proyecto amplio que trata los temas de forma superficial. El objetivo no es una comprensión completa si no conocer los elementos principales. Esto abre la posibilidad de que cada individuo se especialice en las partes que más le interesen.

Abstract

In the development of this project, three sections that contribute to the final objective, which is to serve as an introduction to the world of Bluetooth technologies, can be clearly distinguished. The three sections mentioned before are the following: “Theory”, “Development of the Programme” and “Use Cases”. It also presents an introduction that describes the point of view used to develop the project and the objectives related to it.

In the “Theory” part, we describe what is meant with the term “Bluetooth” in a more precise form. Bluetooth refers to two different technologies: Bluetooth Classic and Bluetooth Low Energy. Although we generally use Bluetooth as it were just a technology, Classic and LE are not compatible even though they share some characteristics. A brief introduction about the common functionality and physical characteristics that they share and about the ones that are different is presented. Afterwards, we study each of them in more detail. A study of the most common protocols and components is performed to allow the reader to comprehend how the communication between two Bluetooth devices is performed. Two additional sections are dedicated to explain the concepts of the HCI and the codecs, which are fundamental elements in Bluetooth communications.

In the section of “Development of the Programme”, it is described step by step how the programme is developed. It contains parts for each of the essential elements needed for this project. Additionally, it also has a small introduction describing the points that have guided it. The versioning section explains how a GitHub repository has been created to preserve and update the project. In the next section, it is described point by point how to obtain the Bluetooth source data that is going to be analyzed. Afterwards, the software that is used to treat the data is presented and the data fields that are selected for each of the protocols. Finally, it is shown how to configure a python virtual environment and how the programme has been coded and the visual aspects and functionalities that it presents.

In the last section, the acquired knowledge and the programme are put to the test with 7 different use cases involving 7 different devices which all use Bluetooth technologies. The purpose of this is to show the usefulness the developed programme has and how it is able to analyze all sorts of Bluetooth traffic. It also serves to demonstrate that it is possible to begin to comprehend how Bluetooth technologies work with just a brief introduction to the theory. For each of the use cases, various tests are performed that allow us to see the different protocols and their data frames that are sent during the communication. The data frames are analyzed to try to discover how the devices work and how they use the different protocols for each situation. Occasionally, some patterns can be distinguished that allow us to formulate reasonably

feasible hypotheses. On other occasions, it is more difficult. It depends on how the manufacturers have decided to implement the delivery of the data.

As it can be noted, it is a broad project that treats themes in a superficial way. The objective is not an in-depth understanding of the technologies, but to 4 know the main elements that are used by them. This gives the opportunity to every individual to specialise in the fields that they find most interesting.

Índice

1.Introducción

La finalidad de este TFG es servir como introducción al mundo Bluetooth. La información que se suele proveer para esta tecnología es sumamente técnica y dificulta la comprensión de las bases de esta tecnología. Para lograr esto, se va a redactar un texto que sirva como una descripción breve del funcionamiento de esta, acompañada de diagramas y demás recursos para facilitar la comprensión. Esta comprensión va a permitir una utilización eficaz de la parte clave desarrollada para este trabajo: un programa que permite analizar el tráfico y filtrar el contenido.

Puesto que existen programas considerablemente potentes que ya permiten hacer esto que se ha descrito como "Wireshark", el enfoque es clave a la hora de entender el porqué del desarrollo de esta herramienta. Si bien Wireshark permite hacer esto y mucho más, resulta un programa muy complejo. Al estar enfocado en el análisis de tantos protocolos distintos, ofrece una cantidad de opciones que puede resultar abrumadora. El objetivo a la hora de desarrollar mi programa ha sido el de crear un sistema más interactivo y atractivo. Las opciones son limitadas, lo que puede capar la cantidad de información que puede ser explorada, pero la información que se puede visualizar ha sido minuciosamente seleccionada para permitir ver los elementos clave que componen estos sistemas de comunicación inalámbrica. Nos interesamos principalmente por el flujo de datos, o sea, por las cosas que suceden, para llevarnos una imagen general. Por ello, no me interesa tanto mostrar los ajustes específicos que cambian, puesto que no suelen tener impacto en esto.

El conocimiento que se busca transmitir de forma concisa mediante este proyecto es el que está más vinculado con las tareas realizadas por los protocolos más relacionados con las capas superiores del modelo OSI. Puesto que es un trabajo de ingeniería informática y no de telecomunicaciones, no parece coherente centrarse en describir cómo se transmiten los datos en el medio físico y las distintas configuraciones que hay disponibles para esto. Lo ideal es que se aprenda cómo funcionan los sistemas a nivel lógico y se traten cuestiones como la forma en que se comunican, gestionan sus roles y solicitan y organizan su información. Nos vamos a centrar en el componente "Host". Otro punto importante respecto a esto es que las fuentes de datos que analizamos del tráfico de Bluetooth no suelen contener paquetes que nos puedan dar información acerca de estos protocolos más ligados con el hardware.

El nombre del programa, "Blueprobe", juega con el nombre en inglés de "Bluetooth". La palabra "tooth" significa "diente" en inglés, y la palabra "probe" en el mismo idioma se refiere al instrumento usado por dentistas para analizar la situación de dientes y encías. El término en español sería "sonda". Este título es un juego de palabras que representa la finalidad del programa de permitir la

exploración de la tecnología Bluetooth. Se puede observar el logo de este proyecto en la Figura 1.

Figura 1: Logo del proyecto Blueprobe.

El proyecto ha sido desarrollado en un entorno Linux, pero las herramientas utilizadas están disponibles por lo menos en los principales sistemas operativos (Windows, iOS, Linux). El programa está escrito en python, por lo que se puede ejecutar en cualquier dispositivo que tenga python y las bibliotecas necesarias instaladas.

El trabajo se realiza en español, pero todo lo contenido en el repositorio está en inglés. Desarrollar los proyectos de software de esta forma permite que un mayor número de personas pueda aportar ideas y colaborar, haciendo que la perspectiva de mejora sea mayor.

1.1. Objetivos

Los objetivos del trabajo son la comprensión de cómo funciona Bluetooth y sus componentes principales y el desarrollo de un programa que permita un análisis personalizado para estas tecnologías, presentando las informaciones más relevantes y permitiendo la comprensión del tráfico que se genera.

El interés de esto es servir de introducción al mundo Bluetooth. Se busca comprender los conceptos que son clave desde el punto de vista de un desarrollador, explicarlos de forma concisa y permitir que el programa desarrollado muestre una versión organizada y visual que permita entender el flujo de datos de forma natural e intuitiva.

Lista de objetivos presentados en el plan de trabajo:

- Búsqueda de documentación de calidad.
- Búsqueda de documentación de calidad.

• Lectura y comprensión de los recursos seleccionados para aprender acerca de las distintas tecnologías que componen Bluetooth y del funcionamiento de los protocolos.
• Diseño del material para la exposición concisa de los tipos de Bluetooth y su
funcionamiento.
• Búsqueda del método más apropiado para la captura de los paquetes involucrados en la comunicación.
• Desarrollo de un programa capaz de captar la información deseada.
• Implementación de una forma de procesar la información captada para su posterior presentación.
• Creación de una representación gráfica que exponer cada etapa y su respectiva información de forma clara y completa.

2.Desarrollo

En esta sección se va a tratar el desarrollo del trabajo. Va a contener una sección de teoría, una que muestra cómo se desarrolló el programa y otra para presentar casos de uso.

2.1. Teoría

En este apartado se presenta una base teórica que tiene la finalidad de permitir la comprensión del resto del trabajo y de servir de introducción al mundo de las tecnologías Bluetooth.

2.1.1. Introducción

El término "Bluetooth" puede llevar a la equivocación con facilidad. Al contrario de lo que se podría pensar por el uso que se le suele dar en el día a día, la palabra "Bluetooth" no describe ninguna tecnología en específico. Se trata de una palabra de marca registrada que engloba un conjunto de tecnologías [1]. Estas están gestionadas por el denominado "Bluetooth Special Interest Group", a menudo referido por sus siglas "SIG". Se trata de un grupo de organizaciones que se ocupan del desarrollo y mantenimiento de estas tecnologías.

Las tecnologías a las que nos podemos referir con esta palabra son dos: "Bluetooth Classic" (a veces se refieren también a esta como "BR/EDR") y "Bluetooth Low Energy" (a menudo acortado a "BLE") [2]. Generalmente, la tecnología que se suele usar en el ámbito de dispositivos de uso personal y que solemos usar en el día a día es "Bluetooth Classic", pero "Bluetooth Low Energy" está viendo un uso cada vez más grande y se espera que esta tendencia siga [3].

Originalmente, solo existía "Bluetooth Classic". Fue ideado en 1990 y el primer prototipo salió en 1994. Con cada nueva versión, se hicieron cambios para mejorar la velocidad de transferencia, la rapidez de conexión y algunas funcionalidades específicas. No fue hasta la versión 4.0 en 2010 que "BLE" fue introducido. Se puede considerar el cambio más importante hasta la fecha. La versión actual de "Bluetooth", la cual afecta a todas las tecnologías relacionadas, es la 6.0 y presenta varias mejoras y novedades respecto de la versión 4.0.

Ambas tecnologías están enfocadas en permitir la comunicación inalámbrica y funcionan con ondas de radio, más concretamente en la franja de 2,4 Ghz. Son usadas principalmente para conexiones en redes "PAN", lo cual es un acrónimo de redes de área personal en inglés, aunque ciertos avances relativamente recientes permiten conexiones de mayor distancia usando BLE.

Si bien las dos cumplen la misma función y operan en el mismo medio físico, cada una de ellas tiene particularidades que les permite ser mejores en ciertos casos concretos y para ciertas aplicaciones. Bluetooth Classic es usado

para aplicaciones que implican una transmisión de datos, como para transmisiones de audio o transferencias de archivos, ya que cuenta con una tasa de transferencia de datos mayor que la de BLE (3Mbps de Classic frente a un máximo de 2Mbps de BLE).

Sin embargo, esta tecnología tiene un alto coste energético, un rango limitado, una única topología (peer-to-peer) y supone un alto coste para el desarrollador, ya que los materiales necesarios son más caros. El SIG le está dando una creciente importancia a BLE, proporcionando numerosas actualizaciones que amplían su funcionalidad, y que en ocasiones plantean la posibilidad de usar esta tecnología en lugar de Bluetooth Classic para ciertas funciones hasta ahora reservadas a ella. Es el caso de LE Audio, un uso de BLE que permite la transmisión de audio. Por el momento, la eficiencia energética, la flexibilidad en cuanto a rango y topologías (topologías peer-to-peer, estrella, difusión y malla) y su accesibilidad para un desarrollo muy asequible a nivel económico, hacen de BLE la tecnología de preferencia para sensores, control de dispositivos y aplicaciones que no necesitan una alta tasa de transferencia. BLE tiene una importante proyección de futuro y todo apunta a que su uso va a incrementar en gran medida.

Cabe decir que estas dos tecnologías no son compatibles por mucho que compartan parte del nombre y el mismo documento de especificaciones. Un dispositivo Bluetooth Classic no puede comunicarse directamente con un dispositivo BLE. Existen dispositivos que implementan el llamado Dual Mode Bluetooth, que les permite establecer conexiones usando tanto Classic como BLE.

Ambas tecnologías clasifican los protocolos de su pila en tres categorías, también llamadas componentes o subsistemas. Estos son “Application” (aplicación), “Host” (huésped o anfitrión) y “Controller” (controlador), mencionados de arriba a abajo en la pila. “Application” a veces no se menciona, pues no tiene tanta relevancia como los otros dos, que son esenciales. El componente controlador se ocupa de efectuar las funcionalidades relacionadas con la parte de radio, la de hardware y las conexiones inalámbricas. El componente Host se encarga de darle un formato a la información intercambiada y de cómo se ofrecen los servicios o información entre los dispositivos. También controla los distintos roles que pueden tomar los dispositivos y cómo interactúan. El componente aplicación se encarga de presentar la información al humano y darle una forma de interactuar.

Libro usado como base para la sección de Introducción: [4].

2.1.2. Bluetooth Classic

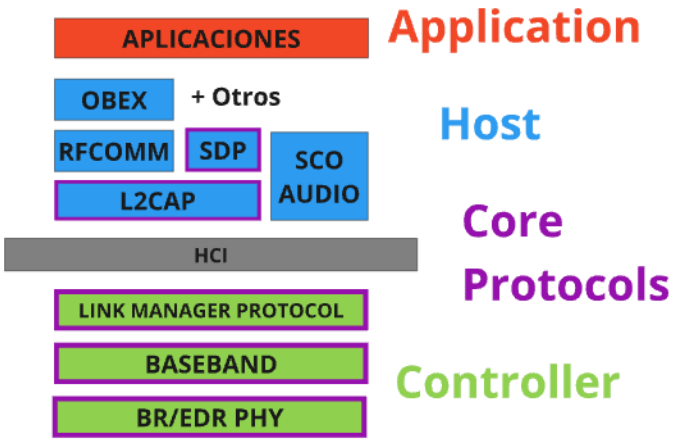

Figura 2: Stack de protocolos de Bluetooth Classic

La Figura 2 presenta el stack de los protocolos más comunes de Bluetooth Classic y los componentes donde se clasifican, destacando en morado los "Core Protocols". Esta figura expone los protocolos de los que se va a hablar. En estas conexiones, los dispositivos pueden tener uno de dos roles: "master" (maestro) o "slave" (esclavo). El maestro puede tener múltiples esclavos conectados y se encarga de establecer el reloj y secuencia de salto entre frecuencias para permitir la sincronización. A este tipo de red PAN formada por un maestro y uno o varios esclavos se le llama "piconet".

En Bluetooth Classic, se utilizan dos tipos de enlace: ACL y SCO. El SCO es el menos usado y generalmente es usado exclusivamente para datos de audio en tiempo real. Para todo lo demás, se usan las conexiones ACL. Los enlaces SCO garantizan una cierta transmisión y tienen preferencia para usar la radio respecto a otros servicios que quieran usarlo. Dos dispositivos sólo pueden tener una conexión SCO al mismo tiempo. Con las conexiones ACL sucede lo mismo: dos dispositivos sólo pueden tener una conexión entre ellos dos como máximo. Al contrario de lo que ocurre con SCO, este tipo de conexiones generalmente se usan para encapsular a otros protocolos y muy rara vez se usan para transportar información directamente. La principal diferencia entre estos dos tipos de conexión es su uso, su capacidad de transmisión de datos (mayor para ACL) y que SCO no permite el reenvío de paquetes. SCO tiene una versión mejorada llamada SCO que ofrece mayor calidad de audio y manejo de errores.

Bluetooth Classic cuenta con una amplia variedad de protocolos que pueden conformar su pila. El uso de unos u otros protocolos depende del uso para la conexión entre los dispositivos; variará según el tipo de información que se envíe y la funcionalidad de los dispositivos [5].

Debido a la gran cantidad de estos, no va a ser posible estudiarlos todos. Aunque haya muchos, solo unos cuantos son necesarios. Estos son los denominados “Bluetooth Core Protocols” y se trata de los protocolos “Baseband”, “Link Manager Protocol (LMP)”, “L2CAP” y “Service Discovery Protocol (SDP)”. “Baseband” se encarga de configurar los enlaces físicos entre dispositivos y de la sincronización. Permite que el medio físico sea aprovechado satisfactoriamente. “LMP” se ocupa del establecimiento y configuración de las conexiones, encargándose de la topología. L2CAP y SDP se tratan en más detalle a continuación, ya que pertenecen al subsistema Host y este nos interesa especialmente en este proyecto.

2.1.2.1. L2CAP

Si bien Bluetooth Classic es independiente de BLE, al ser su antecesor, establece algunas tecnologías importantes para este. L2CAP es encapsulado por ACL y es el protocolo de comunicación más importante de Bluetooth Classic. El protocolo ATT, esencial para la comunicación en BLE, y muchos protocolos de la capa de aplicación de Classic son encapsulados por L2CAP.

L2CAP se ocupa de la segmentación, reensamblado y multiplexación de paquetes. Establece canales lógicos para cada flujo de datos y se asegura de que ciertos flujos tengan prioridad. Soporta tanto datos orientados a conexión como los que no están orientados a estas y se asegura de regular los datos para que no causen problemas como, por ejemplo, desbordamientos. L2CAP se asegura de administrar los datos de la forma apropiada para su retransmisión por el controlador y actúa conforme a lo establecido por “LMP”. Es una etapa esencial para preparar los datos para la mayoría de protocolos de aplicación.

2.1.2.2.SDP

El protocolo SDP es un protocolo “core” que permite conocer los servicios y las características de estos que ofrece un dispositivo. Permite elegir entre unos de esos servicios que se ofrecen. Los servicios están organizados en registros, cada uno de ellos mostrando la información que define cada servicio (nombre, puerto, protocolos usados, ...). Cada uno de estos servicios es identificado por un ID único (UUID). Existe también el concepto de “Service Class ID”, que permite dar un identificador para servicios que cumplen una cierta función. Un uso de este sería, por ejemplo, que todos los servicios que permiten la transmisión de audio para escuchar música tengan el mismo “Service Class ID”.

2.1.2.3. RFCOMM y Protocolos de Aplicación

RFCOMM se trata de un protocolo que simula una conexión por cable usando L2CAP, el cual lo encapsula. No es un protocolo core pero es ampliamente utilizado. Emula los puertos seriales RS-232 y varios protocolos y

perfiles lo usan como base para su comunicación, especialmente las aplicaciones que estaban basadas en comunicación por puerto serial.

Bluetooth Classic tiene protocolos denominados “adoptados” los cuales son protocolos generalmente usados en otros contextos que han sido adaptados a Bluetooth. Esta adaptación permite una fácil interoperabilidad con sistemas que usan estos protocolos y permite una adaptación más sencilla de los sistemas Bluetooth a estos. Un ejemplo serían los protocolos “TCP”, “UDP” e “IP”.

Puesto que no está dentro del ámbito el tratar todos los protocolos de la capa de aplicación, se va a hacer un repaso rápido de ellos para dar una idea de cuantos existen y de la función de cada uno:

- BNEP: permite la transmisión de datos de red entre dispositivos.
- TCS: diseñado para emular un sistema de teléfonos y permitir llamadas y su establecimiento.
- AVCTP: permite enviar contenido de audio y video.
- AVDTP: permite controlar cómo se envía el contenido de audio y video.
- PPP: se trata de un protocolo adoptado encapsulado en RFCOMM que permite la comunicación punta a punta (point-to-point).
- TCP, UDP y IP: adaptación de los famosos protocolos de comunicación.
- OBEX: funcionalidad parecida a HTTP, pero más ligero. Permite el intercambio de objetos y un listado en base a un sistema de carpetas. Basado en TCP/IP o RFCOMM.
- WAP y WAE: trabajan conjuntamente para permitir el acceso a recursos de internet con dispositivos con recursos limitados.

2.1.2.4. Perfiles

Un perfil describe los protocolos y sus ajustes y las particularidades técnicas usadas para un modelo de uso concreto. Permiten establecer una base para desarrollar aplicaciones en base a una funcionalidad cuyos protocolos ya se conocen. Algunos de los perfiles más usados son:

- **GAP**: más que un perfil, sirve como marco para los demás perfiles, estableciendo una base para que los dispositivos interactúen.
- A2DP: usado para transmisión de alta calidad de audio.
- HFP: permite usar un teléfono móvil junto con un sistema manos libres, permitiendo transmitir audio o controlar de forma remota el teléfono.
- AVRCP permite el control remoto de reproducciones de video y audio.
- PBAP: permite intercambiar contactos (objetos de agenda telefónica) entre dispositivos.
- HID: para conectar dispositivos de interfaz humana, o sea, periféricos como teclados, ratones o mandos.

[6]

Libros usados como base para la sección de Bluetooth Classic: [7], [8] y [9].

2.1.3. Bluetooth Low Energy

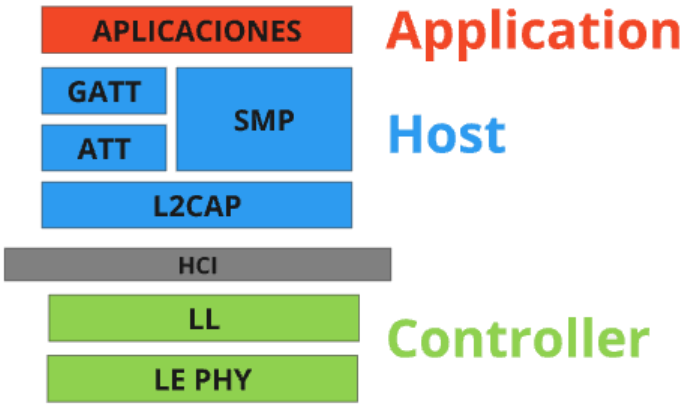

Figura 3: Stack de protocolos de Bluetooth Low Energy

La Figura 3 ilustra el stack de protocolos de Bluetooth Low Energy y su clasificación por componentes. Al contrario que con Bluetooth Classic, BLE permite usar controladores que solo tengan capacidad de recibir o solo de transmitir. Esto permite desarrollar dispositivos más compactos, puesto que solo se desarrolla una de las funcionalidades y permite no usar elementos innecesarios.

La pila Bluetooth Low Energy cambia casi por completo en relación con la de Classic. Los protocolos del controlador son diferentes, pero como ya se ha explicado anteriormente, no vamos a profundizar en este apartado. Está compuesto por dos capas: LE PHY y LL. Hay distintas configuraciones de LE PHY que se ocupan de la parte física, definiendo cómo se va a usar la radio para enviar los datos por el medio físico (el aire en este caso). LL se ocupa de los enlaces y está basado en una máquina de estados, transicionando entre estos dependiendo de las conexiones que se establezcan y del tipo de estas.

En cuanto al Host, todo cambia salvo por el uso de L2CAP. Para BLE, L2CAP es usado para presentar la información de las capas superiores en paquetes BLE estándar y para fragmentar y recombinar paquetes. Se usa como protocolo de multiplexación.

El protocolo SM (“Security Manager” o “Gestor de Seguridad” en español) se encarga del emparejamiento e intercambio de claves para la encriptación de los datos. En un principio no había intención de cubrirlo puesto que lo relacionado con la seguridad de las tecnologías Bluetooth no se considera relevante según el enfoque adoptado para el proyecto. Sin embargo, después de analizar los casos de uso, se decidió dedicarle una sección, ya que su uso parece bastante común.

Una característica importante de BLE es que los enlaces no son SCO, son solo ACL, pero una versión especial adaptada a LE. L2CAP funciona de la misma

forma que con los enlaces ACL de Classic, solo que cambian los canales reservados. El canal de control es otro en este caso, por ejemplo.

2.1.3.1.GAP

Define las funciones y comportamiento de la pila y sus diferentes capas y cómo interactúan entre sí. Establece los requisitos básicos que un sistema Bluetooth debe cumplir. GAP plantea diferentes roles que describen distintas formas de comunicarse entre dispositivos. Hay cuatro posibilidades:

1. Broadcaster(emisor): para transmitir sin crear conexiones.
2. Observer(observador): para recibir sin crear conexiones.
3. Peripheral(periférico): para anunciar presencia; tienen una única conexión.
4. Central: para iniciar conexión con dispositivos que se anuncian; mantienen múltiples conexiones.

También define una serie de modos y procedimientos más concretos:

1. Broadcast mode and Observation procedure: para permitir a un observador encontrar emisores en situaciones sin conexiones.
2. Discovery modes and procedures: define las formas en que un dispositivo se puede anunciar (emisor o periférico) y como un dispositivo puede descubrirlos (observador o central).
3. Connection modes and procedures: define el tipo de conexiones que se pueden establecer.
4. Bonding modes and procedures: para intercambiar y almacenar información para establecer una relación de confianza entre dispositivos.
5. Periodic advertising modes and procedures: para permitir a dispositivos comunicarse sin crear una conexión usando eventos de anuncio (“advertising events”).
6. Isochronous broadcast modes and procedures: para comunicación sin conexión y unidireccional.
7. Channel sounding procedures: para permitir a dos dispositivos intercambiar información que permite aproximar la distancia entre ellos.

Estos procedimientos se pueden usar o no dependiendo del rol que tenga el dispositivo

2.1.3.2. GATT

Se trata de un estándar que establece una jerarquía para organizar la información que se envía. Como vamos a ver en el siguiente apartado, los datos se almacenan y transfieren por medio de atributos. GATT usa los roles de cliente y servidor y ordena atributos y gestiona estos elementos compuestos. Se pueden agrupar en características, las cuales forman servicios que, a su vez, son parte

de perfiles. Para entender esta organización, debemos entender cada elemento más a fondo:

- Característica: se trata de una característica concreta que queremos ofrecer. En ocasiones, vamos a necesitar múltiples atributos para describirla satisfactoriamente.
- Servicio: agrupación de características y atributos que permite obtener información acerca de una funcionalidad específica del servidor.
- Perfil: conjunto de servicios que ofrecen ciertos dispositivos. Sucede que hay dispositivos que tienen el mismo uso o uno similar entre ellos, por lo que resulta razonable establecer perfiles para este conjunto de funcionalidades que se repiten. Definen el comportamiento entre cliente y servidor tomando como base los servicios ofrecidos.

2.1.3.3.ATT

Define cómo se van a comunicar dos dispositivos de la forma más básica, describiendo los datos en concreto que se van a enviar, sin gran énfasis en la organización de estos, ya que de esto se ocupa el servicio GATT. Hay dos roles importantes:

- Servidor: acepta comandos y provee la información requerida o anuncia la información por defecto de acuerdo con un comportamiento interno.
- Cliente: solicita información y recibe información captada, siendo esta solicitada o no.

La representación más básica en la que se puede presentar la información que se transmite son los atributos. Es un término que sirve para definir los datos que se exponen. Los atributos de se componen de:

- Attribute Handle: valor de 2 bytes usado por el cliente para referirse a un atributo concreto.
- Attribute Type(tipo): UUID de 2 bytes en el caso de atributos adoptados por el SIG o de 16 bytes en caso de que sean personalizados por el fabricante (a veces se les llama vendor-specific). Identificador único del atributo.
- Attribute Value(valor): valor de los datos que el servidor ofrece.
- Attribute Permissions(permisos): indica las operaciones que se pueden realizar con este atributo y el nivel de seguridad requerido para ello.

El formato de estos atributos se puede ver representado en la Figura 4.

Figura 4: Formato de los atributos

Se pueden realizar seis tipos de operaciones diferentes con estos atributos:

- Comando: enviados por el cliente al servidor; no requieren respuestas.
- Petición: enviados por el cliente al servidor; requieren respuestas.
- Respuesta: enviado del servidor al cliente para responder a una petición.
- Notificaciones: enviado del servidor al cliente para hacerle saber que un atributo ha cambiado.
- Indicaciones: como las notificaciones, pero requieren de una confirmación por parte del cliente.
- Confirmaciones: para responder a las indicaciones.

Cada tipo de operación tiene varias opciones que permiten usar los atributos para comunicarse. Estas operaciones se transmiten usando el protocolo ATT, basado en L2CAP.

2.1.3.4.SMP

Si bien inicialmente no se le iba a dedicar una sección a este protocolo, después de hacer los casos de uso se ha reconsiderado la situación. Los dos dispositivos con los cuales se puede presenciar el proceso de la creación de la primera conexión, usan el protocolo SMP en esa etapa. Debido a que, por lo visto, su uso resulta común, se le va a dedicar una breve sección.

Este protocolo, como ATT, es encapsulado por L2CAP. SMP se ocupa de permitir mantener unas ciertas medidas de seguridad basadas en facetas esenciales: confidencialidad, autenticación, privacidad e integridad. Para permitir esto, los dispositivos deben pasar por un proceso de “pairing” (emparejamiento) en algún momento de la conexión. El proceso de “bonding” (unión) es opcional y permite evitar el proceso de emparejamiento en conexiones futuras entre dos dispositivos.

Para comenzar el proceso de emparejamiento, el dispositivo central debe enviar un paquete de tipo “Pairing Request”. Esto no tiene por qué suceder cuando se establece la conexión y puede suceder como respuesta a una petición del periférico, aunque el central tiene la capacidad de decidir si lo considera o no. En este paquete se envían las distintas características que el dispositivo soporta y sus requisitos de seguridad. Estas características se dividen en si

tiene o no componentes de entrada y salida, si posee métodos OOB (Out-of-Band), los requisitos de autenticación (protección a MITM, si se requiere "bonding" y otros más), la longitud de la clave de cifrado y las claves que el dispositivo quiere usar. El periférico responde con un mensaje de tipo "Pairing Response" con las mismas características. En base a las características que cada dispositivo establezca, se acuerdan los parámetros de acuerdo con unas ciertas reglas.

Una vez se ha enviado esta información, se pueden usar dos métodos para intercambiar las claves: LE Legacy Pairing o LE Secure Connections. LE Legacy solo es fiable en ciertas situaciones bastante concretas y se recomienda el uso de LE Secure Connections. El problema que tiene este último método es que solo se puede usar a partir de la versión 4.2. Las claves que se intercambian dependen del método y permiten que la conexión sea segura de acuerdo con las facetas mencionadas al principio. El intercambio de claves puede darse usando medios físicos externos a Bluetooth, lo que puede mejorar la seguridad de la conexión.

Después de esto sucede la fase de "bonding", en la cual los dispositivos almacenan las claves para uso futuro. Esto también permite la autenticación de los dispositivos en el futuro.

Libros usados como base para la sección de Bluetooth Low Energy: [10] y [11].

2.1.4. Protocolo HCI

Como ya se ha discutido en secciones previas, los protocolos pueden ser divididos en tres subsistemas: "Application", "Host" y "Controller". El subsistema "Controller" está compuesto por los protocolos bajos de la pila, que se encargan de controlar la radio física, transmitiendo las señales e interpretandolas y estableciendo un control en las conexiones. Por otra parte, el subsistema "Host" se ocupa de multiplexar esos datos y de la posible encriptación y de organizar y definir como se expone y se solicita la información. También se va a ocupar de definir el comportamiento de cada dispositivo.

"HCI" significa "Host Controller Interface" (Interfaz Host Controller). Como su nombre bien indica, es una interfaz que va a conectar estos dos subsistemas. Este elemento no es estrictamente necesario y hay implementaciones de la pila de Bluetooth Low Energy que implementan el "Host" y el "Controller" en un mismo microchip, evitando así este método de conexión entre los dos subsistemas y obteniendo una conexión más directa y rápida, ya que se evita la utilización del protocolo HCI.

Si bien esto puede dar la impresión de que es un componente inutil, no es así. Es un componente importante de abstracción, permitiendo el desarrollo independiente de los dos subsistemas. Su uso asegura que diferentes dispositivos y compañías sean compatibles y que los controladores puedan ser cambiados unos por otros sin problema. Tiene claras ventajas y, además de eso,

es el elemento que nos permite obtener información de lo que sucede en nuestros dispositivos.

HCI es un protocolo que puede ser implementado mediante distintos tipos de conexión física (USB o DUART hasta donde se ha investigado) y que provee un método de comandos y recepción de eventos uniforme. También provee una forma de encapsular los datos recibidos o enviados para transmitirlos al "Host" o al "Controller" respectivamente. Los archivos de los que vamos a hablar más tarde en el proyecto y que contienen los distintos paquetes que vamos a analizar, contienen muchos paquetes de tipo "HCI_CMD" (Comandos HCI) y "HCI_EVT" (Eventos HCI).

Estos son muy útiles a la hora de ver las señales que el controlador recibe y comunica al "Host" o la información interna que el controlador le envía y los comandos que el "Host" hace que el controlador ejecute y las respuestas que recibe. Aporta mucha información de interés para conocer cómo interactúan estos elementos. Es importante remarcar que los comandos solo se pueden enviar de Host a controlador y los eventos solo de controlador a Host.

Dado que este sirve como enlace entre los dos componentes, todos los paquetes que el componente de Host va a recibir del controlador van a estar encapsulados en forma de paquetes HCI, y los mensajes que el Host envíe al controlador también. Una de las secciones del paquete permite identificar qué tipo de información ha sido encapsulada. Todos los paquetes que va a tratar el programa desarrollado van a ser paquetes HCI, encapsulando cualquier tipo de información que se esté transmitiendo o recibiendo. Los monitoreos realizados ocurren a nivel del HCI.

Los paquetes del protocolo HCI se usan tanto para Classic como para BLE, pero es fácil diferenciar si un paquete se está usando para uno u otro. Los paquetes de comando de este protocolo tienen dos elementos importantes: el OGF y el OCF. El OGF sirve para diferenciar a qué grupo de comandos pertenece y permite clasificar los comandos y el OCF sirve para diferenciar de qué comando específico se trata. Tenemos la siguiente lista de grupos de comandos:

1. Link Control commands: permite a un controlador controlar conexiones con otros controladores.
2. Link Policy commands: permite al "Host" controlar cómo funciona la piconet.
3. Controller and Baseband commands: permite acceso y control a varias de las funciones del hardware Bluetooth.
4. Informational parameters: permite obtener información de las características de un controlador BR/EDR, sin tener capacidad de modificar ninguna de ellas.
5. Status parameters: permite ver el estado del controlador. No se puede modificar el estado más que para reiniciar ciertos parámetros específicos.
6. Testing Commands: permiten hacer pruebas para distintas funcionalidades del hardware.
7. Eventos.

8. LE Controller Commands: permite acceder y controlar varias funcionalidades del hardware Bluetooth y gestionar el funcionamiento del Link Layer. Estos comandos son los que están relacionados con el controlador de Bluetooth Low Energy.

Los eventos permiten que el Host reciba información del controlador. Los eventos pueden suceder como resultado de un comando o por que el controlador registra actividad que sucede en el entorno.

Continuando con lo que se comentó antes, voy a explicar cómo se puede saber si un paquete está relacionado con Classic o con BLE. Si un comando pertenece al grupo de "LE Controller Commands" con OGF 0x08, tenemos certeza de que ese comando solo afecta al controlador BLE. Con los eventos que comienzan por "LE", sucede lo mismo. Hay comandos y eventos específicos que, a pesar de no cumplir estas reglas, pueden ser usados para BLE. Sin embargo, son casos muy específicos; se da con pocos comandos y eventos. Los comandos y eventos que no son de LE o comunes a Classic y LE, solo afectan a Classic. Adjunto una lista de los comandos y eventos que Classic y LE comparten en el anexo A.

Antes de acabar esta sección, debo hablar de los "Vendor Specific Commands". Se trata de un grupo especial de comandos de depuración definidos por el fabricante y tienen OGF 0x3F.
[12]

2.1.5. Codecs

Los codecs son un elemento esencial en la transmisión de datos. Se encargan de comprimir y descomprimir los datos que se envíen para optimizar el rendimiento. Son usados a menudo en transmisiones más pesadas, como las relacionadas con audio, imágenes o vídeos. Los datos a enviar se comprimen en el dispositivo emisor y se descomprimen en el receptor.

Estos tienen tres parámetros importantes:

- Tasa de muestreo (en Hz): relacionado con el número de muestras que se toman por segundo de una señal de radio. Cuanto más alta sea, más similar va a ser al archivo original, pero más grande va a ser el archivo recibido.
- Profundidad de bits (bits): cantidad de bits por muestra.
- Tasa de bits (kbps): número de bits codificados por segundo, lo cual tiene efecto en la calidad de los datos.

Tanto Classic como BLE tienen sus propios codecs. Classic cuenta con una amplia variedad (SBC, AAC, aptX, LDAC, …), siendo SBC el único que vamos a analizar, al ser el códec estándar de esta tecnología [13]. BLE también cuenta con un códec estándar llamado LC3 [14]. Por desgracia, este códec es relativamente novedoso y aún son pocos los dispositivos que lo implementan, por lo que no ha sido posible obtener uno para poder estudiarlo.

Libros usados como base para la sección de Codecs: [15].

2.2. Desarrollo del Programa

En este apartado vamos a presentar como se ha enfocado el programa, los pasos que se han tomado para su realización y las herramientas utilizadas.

2.2.1. Enfoque del programa

Si bien se ha tratado ya en cierta forma el enfoque que se busca en el proyecto en general, voy a marcar algunos puntos clave que han guiado el desarrollo del programa en concreto:

- Se busca mostrar los distintos protocolos que usa Bluetooth y los datos que transmite cada uno de ellos. Para esto, se busca hacer lo más claro posible el flujo de datos y la secuencia de paquetes para ver paso a paso lo que sucede y entender el comportamiento de los protocolos. También se procura mostrar los datos y metadatos enviados para dar una idea de que se transmite con cada protocolo.
- No se indaga en temas de seguridad o encriptación. No nos interesa saber cómo se protege la información que se envía si no qué información se envía. Se considera que para saber cómo se protege algo, primero hay que saber de qué se trata. La comprensión de eso es compleja y está fuera del alcance de este proyecto.
- Por medio del programa vamos a poder presentar la información que se envía entre dispositivos. Estos datos no tienen por qué ser comprensibles a simple vista y el análisis de estos también está fuera del alcance de este programa. Sería interesante comprender la información concreta que se está enviando para comprender aún mejor el flujo de información y los protocolos, pero puede resultar muy problemático. Si esos datos han sido encriptados antes del envío por las aplicaciones, el estudio de estos mensajes es un asunto de desencriptación, lo cual nos aleja del estudio de Bluetooth.

2.2.2. Versionamiento (GitHub)

Para asegurarme de que no hubiera ningún problema a nivel de hardware que pudiera poner en riesgo el trabajo y retrasar o imposibilitar su entrega, decidí usar un sistema de versionamiento. Decidí usar GitHub puesto que es perfecto para gestionar un proyecto individual que permite subir las actualizaciones a la nube. Además, puesto que se plantea este proyecto como un programa con impacto y continuidad en el tiempo, GitHub permite

disponibilizar este programa para que los usuarios puedan descargarlo y usarlo fácilmente.

Para esto, creé el repositorio “Blueprobe” para almacenar tanto el programa como los archivos de soporte a este como archivos snoop con tráfico Bluetooth, el perfil personalizado de Wireshark y los archivos CSV producidos mediante Wireshark.

Pasos para la creación y actualización del repositorio:

1. Creación del repositorio: usé la página web de GitHub para crear el repositorio y establecer la configuración inicial. Se puede observar esta etapa en la Figura 5.

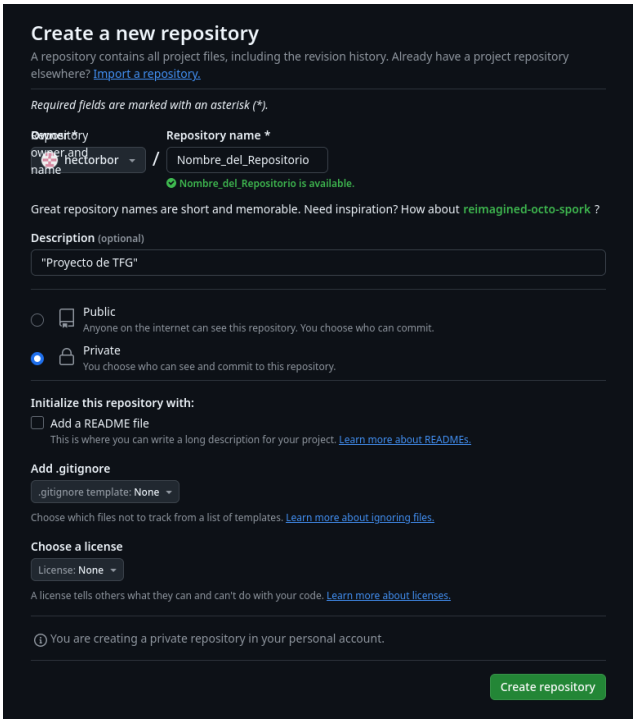

Figura 5: imágen de la página `https://github.com/new`

2. Clonaje del repositorio: `git clone <enlace_del_repositorio>`
   El comando descarga el directorio del repositorio en la carpeta en la que estemos ubicados en el momento de ejecutar el comando.

3. Actualizar el repositorio:
   ```
   git add ./*
   git commit -m “Mensaje informando de los cambios realizados”
   git push
   ```
   Estos comandos se ejecutan desde dentro de la carpeta del repositorio.

Puesto que el proyecto es privado por el momento, ya que creo que es lo correcto durante el desarrollo, hace falta un nombre usuario y un token de identificación

(el cual se puede crear en la página web) para ejecutar los comandos clone y push.

El repositorio contiene cuatro carpetas y un archivo README.md sin rellenar, en el que sería interesante escribir las distintas etapas para preparar la información y poder usar el programa en el momento en el que se decidiese hacer el proyecto público. La carpeta "images" contiene el logo del proyecto, "logs" contiene los archivos usados en los casos de uso y en el testeo del programa (los archivos snoop y las disecciones de estos obtenidas con Wireshark), "personalProfileWireshark" contiene el perfil personal de Wireshark y versiones previas de este y la carpeta "program" contiene el programa con nombre "GUI.py" y versiones antiguas de este.

2.2.3. Fuente de Datos de Bluetooth

Por lo que se ha podido averiguar, hay dos formas de obtener archivos que contengan los paquetes de los diferentes protocolos involucrados en una comunicación Bluetooth. La primera forma sería usar un dispositivo externo a esa conexión Bluetooth que tiene la función de escuchar y captar esos datos que están siendo enviados por el medio físico (se refiere comúnmente a esto como "sniff" en inglés). La otra forma sería usar uno de los dispositivos involucrados en esa conexión para monitorear lo que sucede y generar un archivo con el tráfico.

La primera opción supone ciertos problemas destacables. Generalmente, los dispositivos que usamos en el día a día como teléfonos android o apple y portátiles u ordenadores de sobremesa no tienen un controlador Bluetooth que permita la captura de paquetes. Debido a esto, es necesario la compra de un dispositivo adicional para realizar esto y esto supone un problema a la hora de hacer el programa accesible a todo el mundo.

Si bien algunos de estos dispositivos son asequibles, como los ofrecidos por Nordic Semiconductors, su compra involucra una inversión económica que no todo el mundo tiene por qué estar interesado en hacer y realizar una compra online, ya que no se van a encontrar en tiendas de informática al uso, que puede suponer una molestia. Otro problema importante es que los dispositivos de esta marca no suelen permitir la captura de paquetes de Bluetooth Classic, ya que se especializan en BLE, y las alternativas que permiten esto suelen ser considerablemente más caras.

La idea es que todas las personas que usan el programa lo hagan con datos generados por ellos mismos. Si bien voy a disponibilizar algunos archivos de ejemplo para permitir a los usuarios hacer pruebas, lo que busco es que ellos puedan hacer uso del programa con tráfico generado por ellos mismo con sus propios dispositivos. Esto no solo le añade interactividad y hace el programa

más estimulante para las personas que buscan iniciarse en Bluetooth al manipular información directamente relacionada con su vida diaria, si no que permite la evolución del programa.

Si solo se usaran los mismos archivos de prueba siempre, el programa siempre trataría los mismos datos y no tendría margen de mejora. Al ser usado constantemente con datos de diversas fuentes, se puede buscar mejorar el programa adaptándolo a cada una de las diferentes situaciones encontradas.

Por todas estas razones, he decidido desarrollar el programa usando datos obtenidos de la segunda forma. Este monitoreo puede ser realizado con cualquier dispositivo Android. Aunque me he limitado a estudiar en profundidad exclusivamente los archivos generados por Android, por lo poco que se ha consultado, los dispositivos iOS tienen la capacidad de generar un contenido similar [16]. De cualquier forma, la amplia mayoría de la población usa android y aunque no sean propietarios de un dispositivo de este tipo, lo más probable es que puedan tener acceso con relativa facilidad a uno [17].

El punto de vista que se va a tener va a ser el de un master conectando con slaves o el de un central comunicándose con periféricos generalmente, puesto que los dispositivos android suelen tener esas funciones.

El archivo que se va a generar es un archivo de tipo snoop, con extensión ".log". Se trata de un formato que permite almacenar paquetes [18]. Este tipo de archivo puede ser leído sin problema por Wireshark. Estos archivos snoop contienen, como se indicó en el capítulo de HCI en la parte de teoría, paquetes HCI que encapsulan todo tipo de información.

Aunque la obtención de estos registros es un proceso relativamente sencillo que está documentado en varias páginas web, voy a hacer un breve resumen de los pasos que yo tomé para obtenerlos. La función de esto es dejar claro la forma en la que he obtenido los archivos en caso de que otros usuarios la obtengan de forma diferente y eso resulte en variaciones en los datos finales y para que sirva como guía y garantía de que este método es perfectamente funcional ya que me ha servido para obtenerlos satisfactoriamente.

Lista de pasos a seguir (siendo el teléfono un Motorola G04s):

1. Asegurarse de que la versión de android es la 4.4 o superior, puesto que esta funcionalidad fue añadida en esa versión [19]. Un ejemplo de cómo se puede comprobar la versión se puede ver en la Figura 6.

Figura 6: representación de las etapas del paso 1.

2. Activar el modo de desarrollador de android. Para esto, debemos presionar 7 veces en el campo de texto reservado para el número de compilación. La ubicación de este depende de la marca y modelo del

dispositivo. Se puede ver un ejemplo de esto en la Figura 7. Una vez hecho esto, se van a desbloquear los ajustes de desarrollador, cuya ubicación también depende de lo antes mencionado [20]. Debemos activar la opción “Usar opciones para desarrolladores”, como descrito en la Figura 8.

Figura 7: representación de las etapas de la primera parte del paso 2.

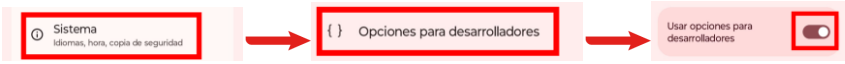

Figura 8: representación de las etapas de la segunda parte del paso 2.

3. En estos ajustes, debemos cambiar a “Habilitado” el ajuste “Registro de búsqueda HCI Bluetooth”. Podemos ver como hace en la Figura 9.

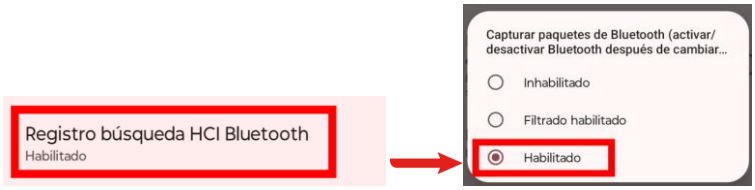

Figura 9: representación de las etapas del paso 3.

4. Si Bluetooth está activo, debemos desactivarlo y activarlo de nuevo, como se observa en la Figura 10, para que este cambio surta efecto.

Figura 10: representación de las etapas del paso 4.

5. Generar el tráfico que queremos analizar. Una vez en esta etapa, la comunicación ya está siendo registrada, por lo que podemos generar lo que vamos a analizar más tarde.

6. Una vez estemos satisfechos con la comunicación generada, debemos conectar el dispositivo a un ordenador mediante USB. Para poder conectarlo, debemos activar la opción “Depuración por USB” en las opciones para desarrolladores. Está opción se puede observar en la Figura 11. Una vez conectado, deberemos seleccionar el modo

"Transferencia de archivos" en el dispositivo android, lo cual se puede ver en la Figura 12.

Figura 11: representación de la activación de la depuración por USB del paso 6.

Figura 12: imágen de la activación de transferencia de archivos del paso 6.

7. Cuando esté conectado, vamos a usar la herramienta "adb" por terminal. Si la instalación de esta herramienta aún no ha sido hecha, deberá hacerse. Deberemos ejecutar el siguiente comando para generar un archivo zip con los registros del dispositivo, el cual será generado en el directorio en el que nos encontramos al ejecutar el comando:
   adb bugreport <nombre_archivo_zip>
   Es posible que, a la hora de ejecutarlo, aparezca una pantalla en el dispositivo android con el mensaje "Permitir depuración por USB" y una huella digital RSA. Deberemos presionar en "permitir" para que el comando funcione. Esta ventana se puede ver en la Figura 13.

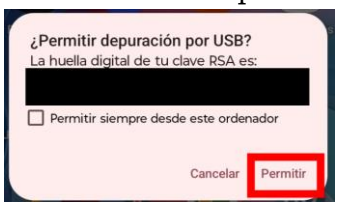

Figura 13: imágen de la última etapa del paso 7.

8. Debemos descomprimir el archivo zip e ir a la carpeta "FS/data/misc/bluetooth/logs"
   Allí nos vamos a encontrar un archivo con el nombre "btsnoop_hci.log". Este es el archivo snoop que vamos a analizar [21].

9. Si queremos limpiar el registro y borrar todos los paquetes, basta con reiniciar el dispositivo.

2.2.4. Tratamiento de datos

Para el tratamiento de datos, he usado el programa Wireshark. Es el programa por excelencia para analizar paquetes de todo tipo de protocolos de comunicación. Lo escogí porque tiene soporte para una gran cantidad de protocolos, incluyendo todos los relacionados con Bluetooth. Otra de las cosas que lo hacen especialmente atractivo es que es un programa muy potente. Permite filtrar la información de acuerdo con una muy amplia variedad de parámetros y presentar los paquetes de acuerdo con esos parámetros. No solo tiene soporte para muchos protocolos, si no que analiza en profundidad todos los paquetes y ofrece una gran cantidad de propiedades especialmente útiles. Los archivos que vamos a manipular con Wireshark son los archivos snoop, acabados en .log que mencionamos en el apartado anterior.

Se barajó la posibilidad de usar bibliotecas de python para analizar los paquetes de forma más personalizada, pero hubiera supuesto un esfuerzo mayor sin obtener un beneficio significativo. Hay una gran cantidad de trabajo de clasificación y análisis ya realizado por el equipo de Wireshark.

Para ejemplificar esto, voy a tomar como ejemplo los paquetes relacionados con el protocolo “HCI_CMD”. Como expliqué previamente en la sección de teoría, hay una muy amplia variedad de comandos que están clasificados en distintos grupos. Wireshark traduce cada uno de los códigos de operación de cada comando de bytes a texto, dando el nombre de cada comando sin necesidad de procesamiento. Además de esto, también indica en texto a cuál de los 7 grupos diferentes pertenece. Esto hubiera supuesto un esfuerzo enorme, al tener que hacer esta traducción de bytes a texto cientos de códigos diferentes. Lo visto en este ejemplo, es aplicable a los demás protocolos. Lo más inteligente es hacer uso de esta gran herramienta.

La ventana principal de Wireshark que se puede ver al abrir el programa junto con la división por secciones de la que se va a hablar se puede observar en la Figura 14. El programa de Wireshark presenta tres secciones en su configuración predeterminada cuando abrimos un archivo. Se trata de una superior, con una columna para cada atributo y una línea para cada paquete, y dos inferiores, la de la izquierda permitiendo ver cada apartado del paquete de forma detallada y la de la derecha mostrando una versión en hexadecimal o ASCII. Las columnas son modificables y se pueden eliminar o añadir nuevas, pudiendo elegir entre una enorme cantidad de atributos para mostrar lo que más nos interese. También hay una gran variedad de filtros que se pueden aplicar para los paquetes, pero esa función va a ser realizada dentro de mi programa.

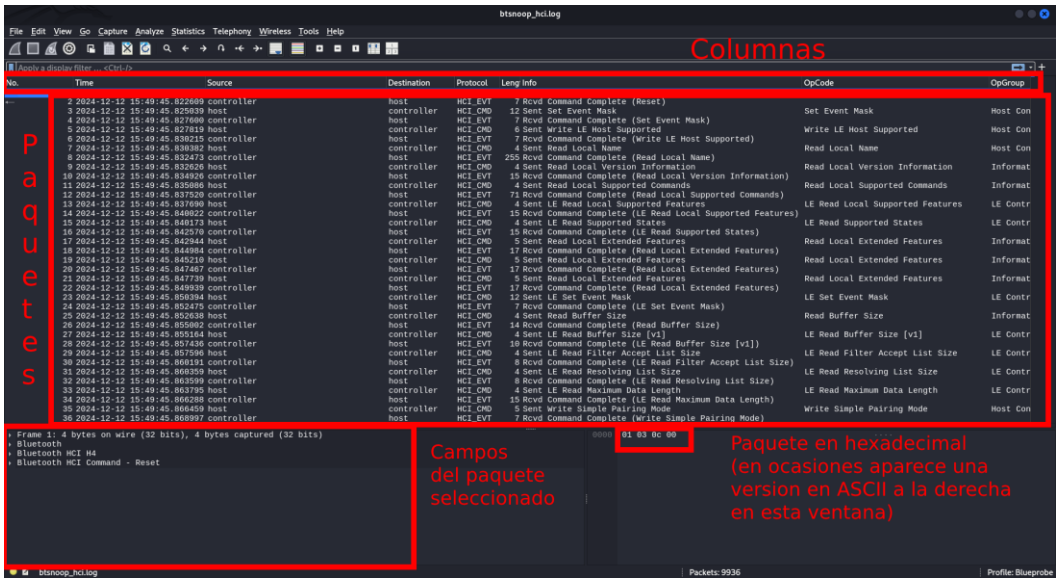

Figura 14: Ventana principal de Wireshark

Esta sección superior es la que nos va a permitir realizar el tratamiento de datos y sacar de cada paquete la información que vamos a presentar más tarde. Wireshark tiene una funcionalidad llamada “Export Packet Dissections” en el menú “File”, que permite exportar los valores de cada columna para cada paquete en el formato deseado. La ubicación de esta opción se muestra en la figura 15. Para el tratamiento de datos, lo que hice fue descartar las columnas que no fueran útiles para mi programa y añadir las que iba a necesitar presentar o analizar más tarde.

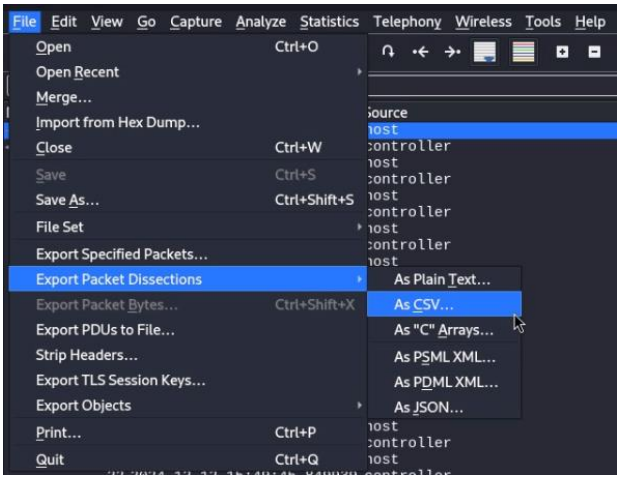

Figura 15: Ubicación de la opción “Export Packet Dissections”

Para presentar ciertos atributos que son específicos de cada protocolo es necesario crear columnas “Custom” (personalizadas) en “Column Preferences” (preferencias para columnas). Esta opción se puede encontrar al hacer clic derecho en la zona de las columnas y podemos ver una imágen con su ubicación

en la Figura 16. La ventana de “Column Preferences…” se puede observar en la Figura 17. En el campo “fields” (campos) hay que poner el atributo en concreto que queremos mostrar ahí. Este se puede encontrar en la ventana inferior izquierda, encontrando una aparición y haciendo clic derecho encima y seleccionando “Prepare as Filter” (preparar como filtro) y luego “Selected” (seleccionado). Hecho eso, el nombre del atributo aparecerá en la barra de filtros.

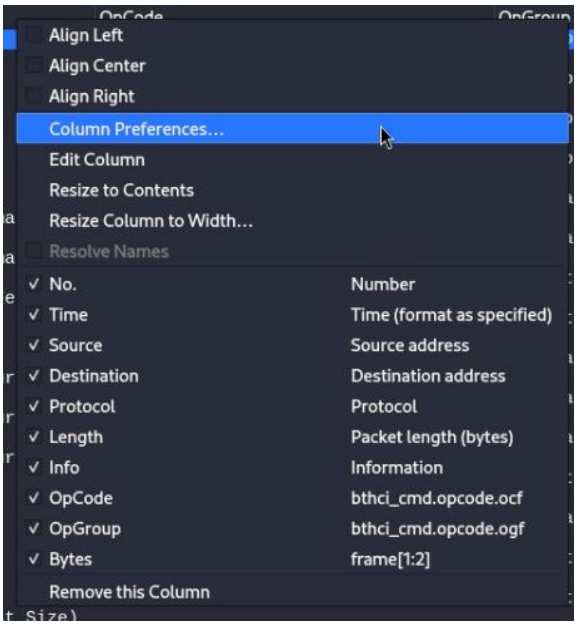

Figura 16: Ubicación de la opción “Column Preferences”

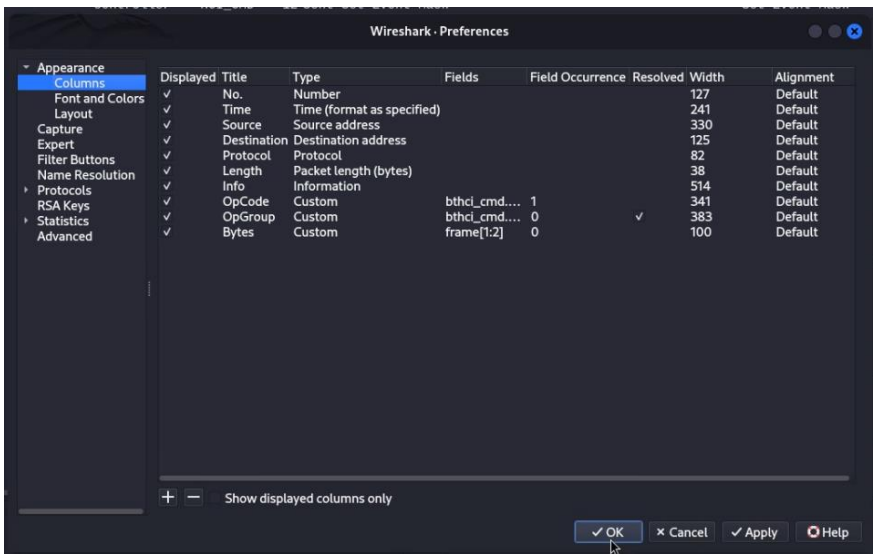

Figura 17: Ventana de “Column Preferences”

En caso de que no encontremos ninguna aparición, podemos ir a la página web “Display Filter Reference“ [22], donde podemos encontrar cada atributo posible para cada protocolo y su código para poder ponerlo como campo para la columna personalizada.

Para facilitar esta etapa a otros usuarios y preservar la configuración utilizada, he decidido guardarla. Al conjunto de configuraciones que se establecen en el entorno de Wireshark se le llama “Profile”, y cuando son personalizadas y generadas para el usuario se llaman “Personal Profile”. Wireshark presenta la opción de exportar estos perfiles personalizados, por lo que exporté el mío y lo guardé en una carpeta en el repositorio de GitHub para que cualquiera pueda usarlo para generar los archivos necesarios por el programa sin necesidad de configurar el “Profile” a mano. Bastaría con importarlo en la ventana de perfiles. La ubicación para abrir la ventana con los “Profile” se puede observar en la Figura 18 y la ventana que se crea y los botones para importar y exportar en la Figura 19.

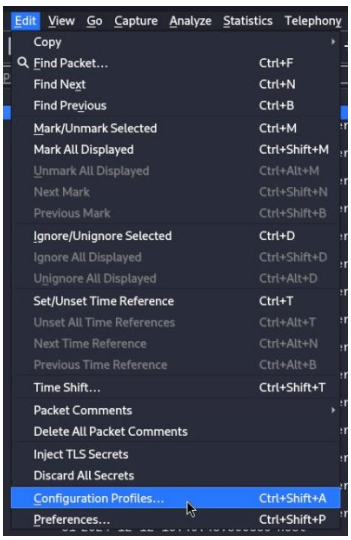

Figura 18: Ubicación de la opción “Configuration Profiles…”

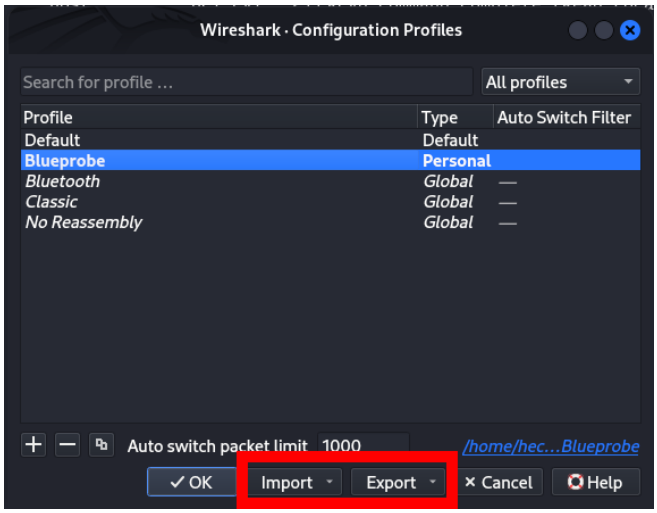

Figura 19: Ventana de "Configuration Profiles…" con botones para importar y exportar.

El formato escogido para exportar la información escogida de los paquetes fue "CSV". Las razones para esto son que no necesito tratar con jerarquías entre los datos en cada paquete y que no voy a trabajar con javascript. La información que voy a tratar es tabular y "CSV" es un formato compatible con una amplia variedad de programas y es simple y eficaz. Puesto que voy a trabajar con python y que su biblioteca para tratar datos en este formato es sencilla y robusta, parece adecuado para este contexto. Es importante indicar que antes de crear el .csv, los paquetes deben estar organizados por tiempo y en orden creciente de arriba a abajo en Wireshark.

2.2.4.1. Datos Escogidos

Para escoger los valores de interés, se ha estudiado la función de cada campo de cada tipo de paquete en la especificación de Bluetooth y se han seleccionado los más relevantes. Los protocolos, perfiles y codecs estudiados se limitan a los que se han encontrado en los casos de estudio, puesto que resulta imposible estudiarlos todos. De esta forma, tenemos ejemplos reales para explicar la utilidad de estas informaciones que se han seleccionado para cada tipo de Bluetooth.

Generales:

- Número del paquete: Para poder comparar que ocurre antes y después. No se usa por el momento, pero podría ser útil en el futuro.
- Tiempo: en el formato yyyy-mm-dd hh:mm:ss.ms (siendo ms representado con 6 digitos). Este formato es similar al de la interfaz gráfica usada, lo que permite una integración más fácil. Esta columna nos permite filtrar por tiempo, permitiéndonos estudiar de forma precisa los intercambios de datos que más nos interesa estudiar.

- Source(origen): permite saber quién envía el paquete.
- Destination(destino): permite saber quién recibe el paquete.
- Protocolo: Nos permite saber de qué protocolo se trata.

HCI:

Los comandos y eventos HCI no necesitan origen y destinatario porque los comandos siempre van del host al controlador y los eventos siempre en la dirección contraria.

Puesto que Wireshark separa los distintos tipos de paquete HCI (Command, ACL, Synchronous, Event y ISO Data), no tenemos por qué hacer esa tarea de filtraje. Esa información está contenida en el primer byte y ya es interpretada por Wireshark.

HCI CMD:

- OGF: Permite clasificar los distintos comandos en grupos, facilitando su organización y visualización. Además, va a ser usado para diferenciar los comandos de BLE de los de classic.
- OCF: Permite identificar cada comando, lo que permite saber qué es lo que el componente de Host está haciendo en cada momento. Al usar este valor como columna, Wireshark solo muestra la descripción y no el código numérico en hexadecimal.
- Bytes de OGF y OCF: Son los bytes dos y tres del paquete. Permiten diferenciar cada comando en específico de forma concreta mediante el código numérico y nos da la posibilidad de separar los comandos relacionados con Classic de los relacionados con BLE.

HCI_EVT:

- Event code(código de evento): Permite identificar qué evento ha sucedido.
- Command Opcode(código de operación de comando): Permite identificar el comando en caso de que el evento sea “Command status” o “Command complete”.
- Byte de Event Code: Sirve la misma función que para HCI_CMD.

CLASSIC:

L2CAP:

- CID: El CID es el identificador de canal lógico. Uno de los principales usos de L2CAP es la multiplexación, ya que permite crear distintos canales lógicos para una misma conexión ACL o LE. El CID es esencial a la hora de entender que finalidad van a tener los datos, permitiéndonos entender mejor su destino. Además, algunos CID están reservados para servicios o para el llamado “signaling channel” (0x0001), un canal que permite configurar las conexiones L2CAP. Este es diferente dependiendo de si la conexión es ACL o LE.

- Code: Permite diferenciar qué comando se ha usado por el "signaling channel".

Puesto que los siguientes protocolos o codecs están basados en L2CAP, vamos a dejar el CID como campo de todos ellos para dar continuidad a lo averiguado leyendo la información de paquetes puramente L2CAP (o sea, que no encapsulan otros protocolos).

RFCOMM:

- DLCI: similar al CID en L2CAP. El DLCI permite identificar cada canal lógico, lo que nos da la opción de multiplexar la información y direccionarla. De forma similar a L2CAP también, el canal 0 está reservado para el "control channel" (canal de control), a través del cual se puede configurar la conexión RFCOMM. Esto permite una multiplexación que es especialmente interesante en el caso de aplicaciones basadas en los puertos que este protocolo simula.
- Frame Type(tipo de frame): permite saber qué tipo de información se está enviando.
- MCC Command Type(tipo de comando MCC): Permite identificar el comando que se ejecuta usando el control channel.

AVCTP:

Se trata de un protocolo que no tiene funcionalidad por sí mismo y se encarga de encapsular información que se interpreta mediante perfiles. De acorde a lo hablado en el capítulo de Bluetooth Classic, AVCTP se encarga de la parte técnica, o sea, del transporte, mientras que el perfil (que en este caso va a ser AVRCP) se encarga de darle un significado concreto a esos datos transportados de acuerdo a un caso de uso específico. Los perfiles usan los protocolos que más les convengan según su funcionalidad.

- Transaction Label: Permite identificar una transacción comando-respuesta y es esencial para poder asociar un comando a su respuesta.
- C/R: Indica si se trata de un comando o de una respuesta.
- Profile Identifier: Permite saber con qué perfil se está usando AVCTP.

Puesto que AVRCP está basado en AVCTP, estos paquetes van a mostrar la información relacionada con AVCTP.

AVRCP:

- Info: se trata de un campo preprocesado de Wireshark. Puesto que este perfil puede cubrir una amplia variedad de comandos, no resulta fácil conocer los parámetros usados en los comandos u obtener los valores recibidos al ejecutar cada comando puesto que son numerosos y habría que dedicar una o múltiples columnas para cada tipo. Wireshark ya hace el trabajo de procesar cada tipo de comando y muestra la información más relevante. Usamos este campo para mostrar la finalidad de un paquete de forma clara.

- Ctype: Indica el tipo de mensaje.
- Opcode: Permite conocer la acción que ejecuta el mensaje.
- PDU ID u Operation ID: Da más información acerca del Opcode cuando no aporta la información suficiente (Cuando Opcode es "Vendor dependent" o "Pass Through" respectivamente).

AVDTP:

- Info: Se usa por las mismas razones que en AVRCP.
- Transaction Label: Permite identificar una transacción comando-respuesta y es esencial para poder asociar un comando a su respuesta.
- Message Type: Indica si se trata de un comando o de una respuesta e indica si el comando se ha aceptado o no.
- Signal: Indica el comando ejecutado.

A2DP:

- ACP SEID: Permite identificar el punto de acceso para los datos en el dispositivo fuente.
- INT SEID: Permite identificar el punto de acceso para los datos en el dispositivo receptor.
- Códec: Va implícito, Wireshark lo procesa de base y nos da paquetes del tipo del códec directamente (en este caso, paquetes SBC). Indica el códec utilizado.

A2DP usa AVDTP para establecer y controlar sesiones de audio. No hay paquetes A2DP, pero los paquetes SBC usan A2DP.
[Usa AVDTP para establecer y controlar sesiones de audio]
SBC:

- Expected data speed(tasa de bits): Es el único campo esencial para la descompresión que no se indica cuando se establece una sesión AVDTP.
- ❖ Datos: Wireshark no permite extraer los datos enviados usando SBC usando columnas personalizadas, por lo que es necesario extraer manualmente esa información.

BLE:

Para BLE, el CID del "signaling channel" va a cambiar puesto que L2CAP está implementado sobre un enlace LE ACL. En este caso, el CID es el 0x0005.
ATT:
El CID reservado que L2CAP usa para ATT es el 0x0004, por tanto, no es necesario indicar el CID.

- Opcode: permite saber qué operación ha ocurrido con relación a un atributo, el cual es identificado mediante un handle.
- Handle: permite saber con qué atributo se está interactuando, aunque no se tenga por qué saber la función que tiene ese atributo.
- Value(valor): permite saber los datos relacionados con el atributo, ya sean los que se escriben, los que se leen o los relacionados con otras acciones.

SMP: Se decide añadir soporte para este protocolo después de realizar los casos de uso. Está construido en base a L2CAP y siempre se usa el CID 0x0006 para el tráfico de este protocolo.

- Opcode: permite saber qué acción se está realizando en ese momento y aporta información acerca de la etapa en la que está el protocolo SMP.
- Info: Se usa únicamente para "Pairing Request" y "Pairing Response" para conocer las características de cada dispositivo.

En el anexo B se mapea cada campo de cada protocolo a una columna de Wireshark.

Se puede observar que hay protocolos que están dedicados a establecer una conexión y controlar la multiplexación y ajustes de esta y otros protocolos que envían la información usando estos canales establecidos. Es especialmente interesante diferenciarlos para poder presentarlos de distinta forma.

Se usan las mismas referencias que se mencionan para el apartado "Casos de Uso".

2.2.5. Entorno Virtual del Programa

Un entorno virtual de python nos permite evitar conflictos de dependencias al mismo tiempo que nos aseguramos de que el proyecto va a tener una mejor portabilidad y compatibilidad. De hecho, mi sistema operativo ni siquiera me permitía instalar bibliotecas sin forzarlo (sin usar sudo), principalmente porque se pueden generar conflictos en el sistema.

Para esto, cree un entorno virtual con el nombre del programa: "Blueprobe". Una vez creado debemos usar el siguiente comando para activar el entorno virtual:

`source Blueprobe/bin/activate` (Se debe ejecutar en la carpeta donde esté el entorno virtual)

Una vez hecho esto, debí instalar la biblioteca que necesito para el desarrollo de la interfaz gráfica:

```
pip install PySimpleGUI
```

o

`./Blueprobe/bin/python3 -m pip install PySimpleGUI` (esta es la que funcionó para mí, pero la otra debería funcionar)

Después de esto, definí el intérprete para el código de python en la primera línea del programa:

`#!/home/hector/Desktop/Bluetooth/Blueprobe/program/Blueprobe/bin/python3` (siendo "/home/hector/Desktop/Bluetooth/Blueprobe/program/Blueprobe" el PATH al entorno virtual)

Hecho esto, basta con ejecutar el siguiente comando para usar el programa:

`./GUI.py` (En la carpeta donde esté el programa)

2.2.6. Programa

La librería utilizada para desarrollar la interfaz gráfica ha sido la librería "PySimpleGUI". Es una librería muy simple, como su nombre indica, que permite un desarrollo rápido y claro. Es una opción ideal para el desarrollo de un prototipo y permite conceptualizar sin costes elevados de tiempo la forma que queremos que el programa tenga.

2.2.6.1. Librerías

import PySimpleGUI as sg (Para la interfaz gráfica)

import csv (Para los datos de entrada)

from datetime import datetime (Para manipulación de fechas)

import webbrowser (Para apertura de páginas web)

2.2.6.2. Ventanas

El programa se compone de 2 ventanas principales y de una secundaria.

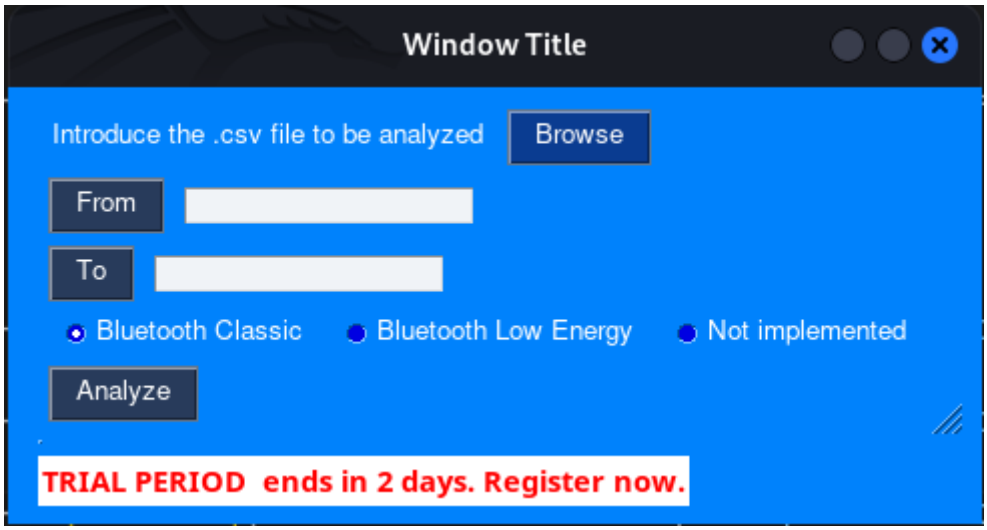

Figura 20: Ventana Principal número 1

Ventana principal 1: Se puede observar en la Figura 20. Permite seleccionar la tecnología que vamos a analizar y hacer un filtrado temporal. Resulta especialmente útil darle una importancia adicional a este tipo de filtrado en una primera instancia por que en ocasiones los registros acaban siendo

notablemente extensos si no reiniciamos el teléfono (lo cual es comprensible, ya que puede resultar engorroso). El programa tiene dificultades para procesar cantidades muy grandes de información, por lo que no se trata tan sólo de un problema a la hora de la visualización al haber demasiados paquetes, sino que también puede afectar al desempeño técnico del programa.Este primer filtrado nos permite centrarnos en una ventana temporal más concreta, pudiendo leer mejor el tráfico. Los paquetes que se presentan para una tecnología son filtrados por protocolo y por el código que los identifica para los protocolos comunes. Para HCI_CMD y HCI_EVT, se han usado las pautas presentes en el anexo A para separar los comandos y eventos. La opción "Not implemented" nos permite ver información básica de protocolos que aún no han sido implementados. Bluetooth tiene una gran variedad de protocolos, por lo que no ha resultado posible implementarlos todos de forma personalizada.

Al hacer clic en el botón de 'Browse', accedemos a un explorador de archivos, el cual se muestra en la Figura 21, que nos permite seleccionar el archivo .csv del que vamos a extraer la información requerida. Los botones 'From' y 'To' van a abrir una ventana de tipo calendario, que se muestra en la Figura 22, donde vamos a poder introducir la fecha. Al seleccionarla, vamos a obtener el formato para las fechas y los horarios, el cual va a deber ser respetado. El botón "Analyze" no os va a permitir analizar los paquetes contenidos, separándolos en distintos protocolos y llevándonos a la siguiente ventana

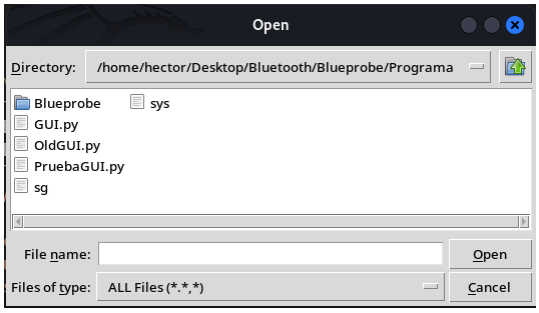

Figura 21: Ventana de búsqueda de archivos.

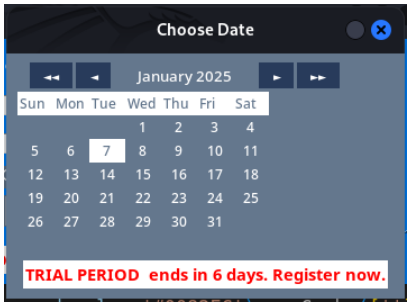

Figura 22: Ventana de calendario.

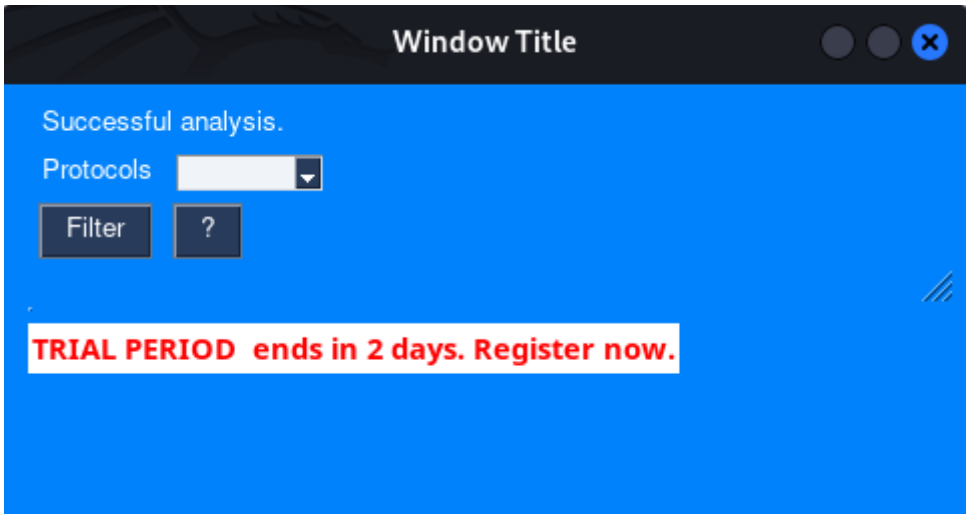

Figura 23: Ventana Principal número 2

Ventana 2: Se puede observar en la Figura 23. Esta ventana nos va a permitir generar una ventana por cada filtrado. Podemos filtrar en base a protocolo. El botón de filtrar nos permite generar un resumen de los mensajes según los filtros, mostrando un mensaje por paquete. El botón “Filter” genera una ventana nueva según lo especificado y el botón “?” nos permite crear una ventana secundaria de ayuda. Ejemplos de pestañas generadas tras pulsar en el botón “Filter” se pueden observar en la Figura 24(con solo una pestaña) y en la Figura 25(con más de una pestaña).

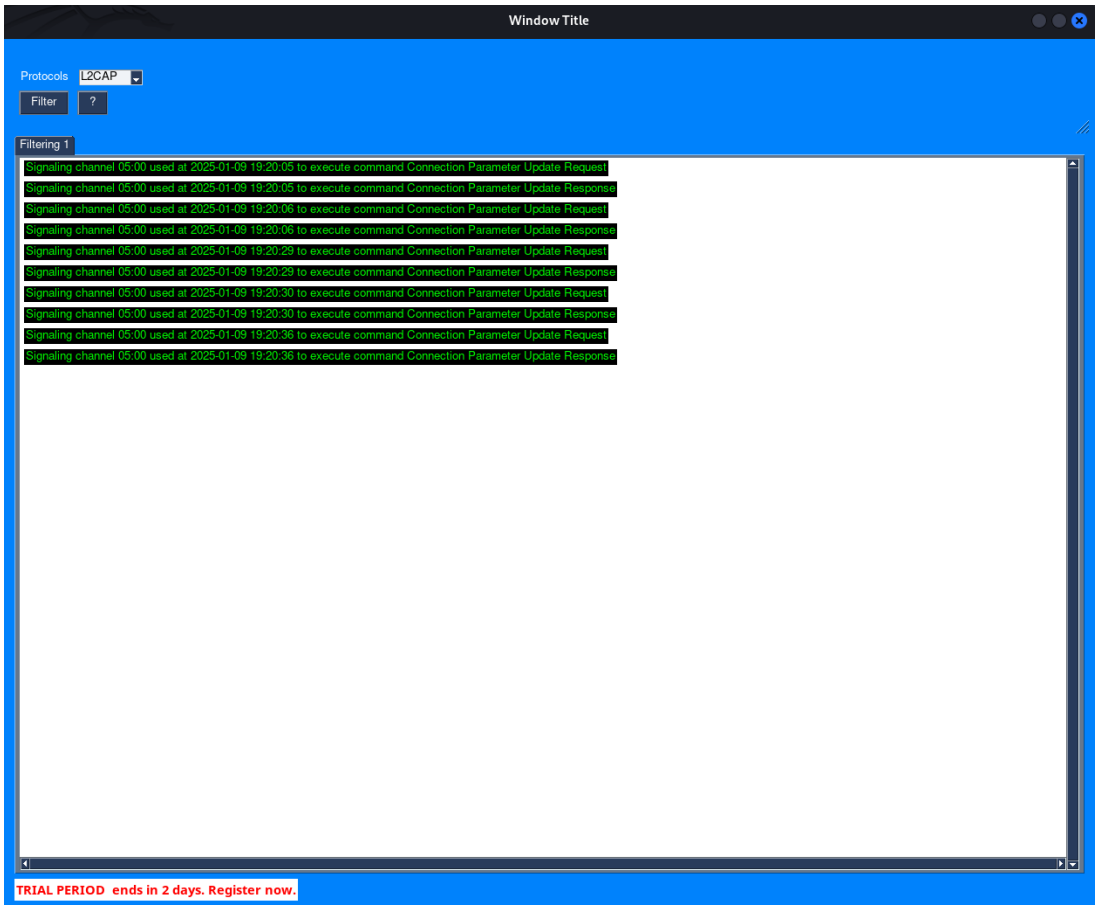

Figura 24: Ventana con una pestaña tras hacer un filtrado.

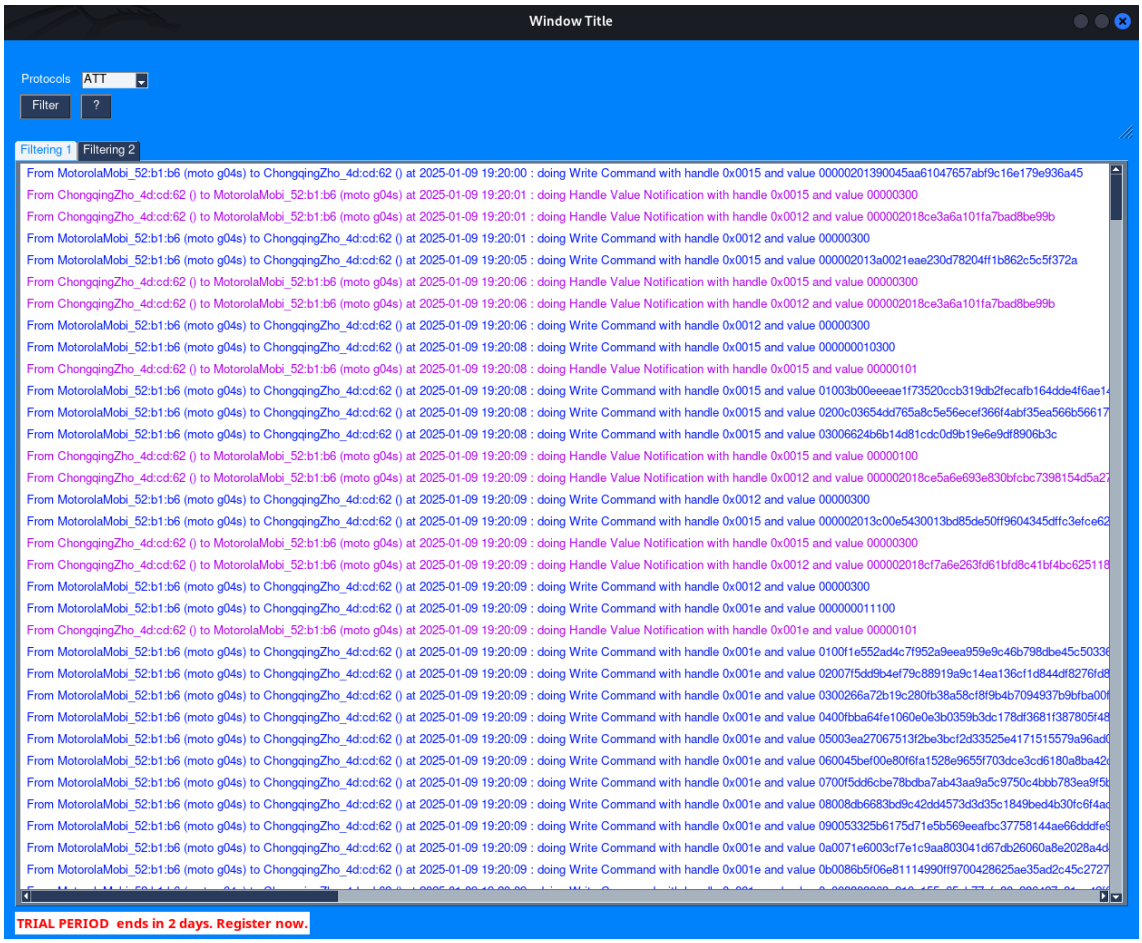

Figura 25: Ventana con más de una pestaña tras hacer más de un filtrado.

Ventana secundaria de ayuda: Hay dos versiones de esta ventana. En la Figura 26, aparece la versión para el análisis de Bluetooth Classic o de "Not Implemented" y en la Figura 27 la ventana para el análisis de BLE. En los dos casos, al hacer clic en los botones, se abre el navegador, dirigiéndonos a la sección procedente de la página de especificaciones oficial de Bluetooth del protocolo en cuestión. Esta ventana es bloqueante y no permite utilizar la ventana principal hasta que se cierre.

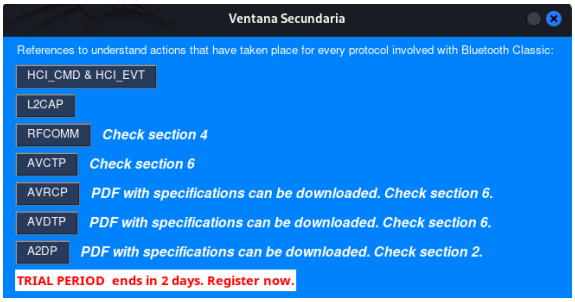

Figura 26: Ventana secundaria de ayuda para Classic.

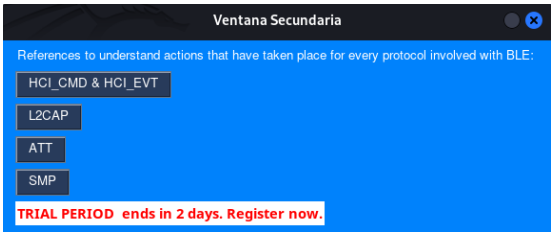

Figura 27: Ventana secundaria de ayuda para Low Energy.

2.2.6.3. Código

El código desarrollado se puede dividir en distintas secciones. En primer lugar, tenemos las bibliotecas usadas, después las constantes, luego las funciones auxiliares, la organización del layout del programa y por último las acciones que suceden según el evento que ocurra (o, en otras palabras, el botón que se pulse). Las bibliotecas ya se han discutido previamente.

Las constantes que se definen se usan para hacer listas de los comandos y eventos que son comunes a Classic y Low Energy para poder presentarlos en las dos distintas configuraciones. También contiene una lista de protocolos implementados.

La sección de funciones auxiliares nos permite organizar el código mejor, permitiéndonos desarrollar ciertos procesos más complejos de forma separada. Las funciones son las siguientes:

`convert_csv_array(csv_address)`

Sirve para obtener una lista bidimensional a partir de un archivo csv. El parámetro es el PATH al archivo.

`filter_by_time(start_date,end_date, values)`

Nos permite filtrar los paquetes según los tiempos escogidos en la interfaz gráfica. Los parámetros son los tiempos usados como referencia para filtrar y la lista obtenida a partir del csv.

`find_protocols(values_timed)`

Genera un diccionario con los protocolos detectados y una lista vacía para cada uno de ellos para almacenar los paquetes más tarde. El parámetro usado es la lista del csv tras ser filtrada por tiempo.

`format_packets_Classic(values_timed,protocols_list)`

Permite sacar los campos de interés de cada paquete según el protocolo que sea en formato diccionario, de tal forma que nos podemos referir a cada campo de forma más clara según el nombre del campo. Solo se ocupa de los protocolos de Classic. Los parámetros usados son la lista del csv tras ser filtrada por tiempo y el diccionario obtenido con `find_protocols`.

`format_packets_LE(values_timed,protocols_list)`

Permite sacar los campos de interés de cada paquete según el protocolo que sea en formato diccionario, de tal forma que nos podemos referir a cada campo de forma más clara según el nombre del campo. Solo se ocupa de los protocolos de Low Energy. Los parámetros usados son la lista del csv tras ser filtrada por tiempo y el diccionario obtenido con `find_protocols`.

`format_packets_Not_Implemented(values_timed, protocols_list)`

Permite sacar los campos de interés de cada paquete según el protocolo que sea en formato diccionario, de tal forma que nos podemos referir a cada campo de forma más clara según el nombre del campo. Solo se ocupa de los protocolos no implementados, guardando campos generales. Los parámetros usados son la lista del csv tras ser filtrada por tiempo y el diccionario obtenido con `find_protocols`.

`present_packets(protocol_list,protocol,column,LE)`

Se ocupa de generar los elementos a presentar según el protocolo seleccionado y el tipo de análisis que estuviéramos haciendo (LE o cualquier otro, "LE" se usa como booleano). Se usan de parámetros la lista de parámetros, tras haberse eliminado los protocolos que no nos interesen según el modo de análisis, el protocolo a analizar, el elemento de la interfaz gráfica al que se le añade cada fila y la booleana ya mencionada.

La organización del layout se puede ver representada en la sección de "Ventanas". Se establecen usando los elementos de la biblioteca de la interfaz gráfica. Las ventanas principales se establecen como columnas que se muestran u ocultan según la ventana que queramos mostrar, mientras que la secundaria funciona como una ventana a parte. También se incluye el grupo de pestañas para los filtrados en la ventana principal.

Los eventos se detectan haciendo un bucle `while True`, que está constantemente funcionando y analizando qué botón es presionado. Los eventos son los siguientes:

Principales:

"Analyze": Se ocupa de excluir los paquetes fuera de la ventana temporal seleccionada tras procesar el .csv y de formatear los paquetes según el modo seleccionado en la primera ventana. También se ocupa de hacer la transición entre las dos ventanas principales. Se modifica ligeramente el formato horario de Wireshark para coincidir con el de PySimpleGUI.

"Filter": Permite presentar la información según los filtros seleccionados.

"Help": Contiene el comportamiento de la ventana secundaria.

Secundaria:

Cada uno de los eventos ocurre cuando uno de los botones relacionados a cada protocolo es presionado. Según el botón presionado, se nos redirige a una página distinta en el navegador, relacionada al protocolo del botón presionado.

2.2.6.4. Presentación de Datos

Para desarrollar el programa de acuerdo con el enfoque deseado y proveer una interfaz gráfica que presente la información de forma más visual, se ha escogido presentar algunos paquetes de especial relevancia de forma diferente, para hacerlos destacar y señalar su importancia e impacto en el contexto general del tráfico.

HCI EVT: Si bien las decisiones tomadas por el host y los comandos relacionados con ellas se encapsulan en paquetes con el protocolo HCI CMD, la confirmación de las cosas que realmente llegan a suceder a partir de esos comandos se puede ver reflejado en paquetes HCI EVT. Cuando una conexión se crea, el fondo se hace verde En caso de desconexiones, sucede lo mismo solo que con fondo rojo. En caso de que se detecten dispositivos a los que sería posible conectarse, el fondo se vuelve amarillo. Estos mensajes con fondo varían entre Classic y LE, pues Classic detecta otros dispositivos al recibir respuestas a una petición de escaneo mientras que los dispositivos BLE que se quieren anunciar envían mensajes constantemente, sin necesidad de recibir una petición. Un ejemplo de los mensajes que se presentan de forma especial se puede observar en la Figura 28.

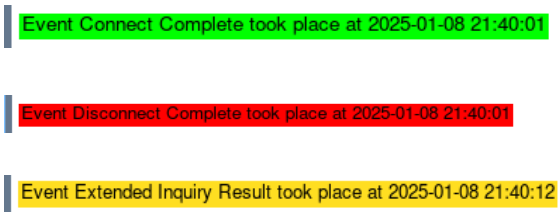

Figura 28: Ejemplo de la presentación de paquetes especiales de información HCI_EVT

CLASSIC y BLE:

L2CAP: Cuando se usa el "signaling channel", sabemos que se está usando para configurar la conexión L2CAP de alguna forma por medio de comandos. Para representar cuando sucede esto, he decidido darle un aspecto de terminal antigua, con los colores negro y verde para asociar esa información a comandos. En la Figura 29 se puede observar un ejemplo de cómo se presenta la información para L2CAP.

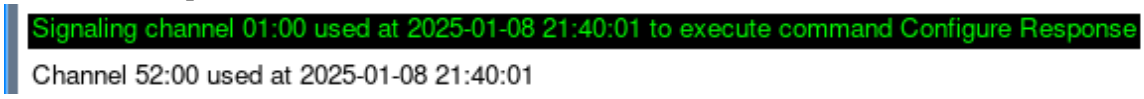

Figura 29: Ejemplo de la presentación de información L2CAP

CLASSIC:

RFCOMM: En el caso de RFCOMM, el canal de control en ocasiones se usa para ejecutar comandos y en otras para mandar otro tipo de mensajes definidos por el campo "Frame Type". Cuando se usa el canal de control, las letras son blancas y el fondo negro, y si además se trata de un comando, el color de las letras

cambia a verde y se da el mismo estilo que con los comandos de L2CAP. En la Figura 30 se puede observar un ejemplo de cómo se presenta la información para L2CAP.

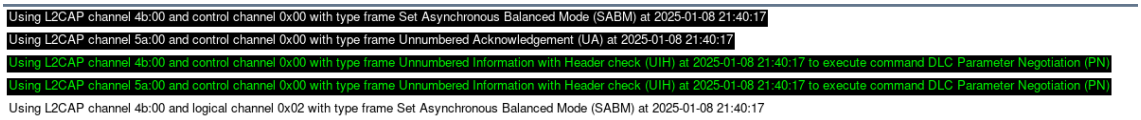

Figura 30: Ejemplo de la presentación de información RFCOMM

BLE:

ATT: Se pone el color de texto en azul si es un mensaje de tipo “Write Command” ya que es el que se suele usar, por lo analizado, al enviar información del central al periférico. En morado se ponen los de tipo “Notification Value” porque son con los que el central suele recibir información del periférico. Se puede observar un ejemplo de estas representaciones especiales en la Figura 31.

Figura 31: Ejemplo de la presentación de información ATT para escritura y recepción.

SMP: Se le añade fondo verde a los paquetes de tipo “Pairing Request” y “Pairing Response” para darles especial importancia, ya que son los que establecen la configuración para la seguridad en BLE. Se puede ver un ejemplo de cómo se representa la información relacionada con SMP en la Figura 32.

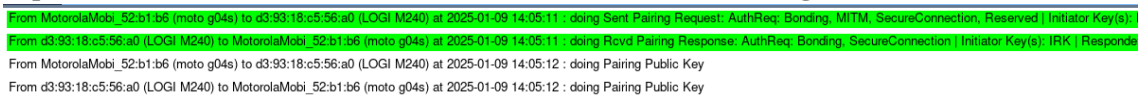

Figura 32: Ejemplo de la presentación de información SMP

2.3. Casos de Uso

Para mostrar la funcionalidad del programa y para poder tener archivos con tráfico para el desarrollo, se ha usado una amplia variedad de dispositivos. Varios dispositivos de las tecnologías Classic y Low Energy han sido utilizados.

Los resultados de los análisis efectuados para el desarrollo del proyecto están expuestos tanto en la sección de teoría como en las secciones de tratamiento de datos y del programa, por lo que en esta parte nos centramos en ver el funcionamiento del programa, a la vez que se corrigen pequeños detalles o implementan nuevas funcionalidades. El proyecto se podría dejar tal y como está en este punto, pero hacer esto sirve para demostrar que Bluetooth es un conjunto de tecnologías muy amplio y que el programa se puede nutrir de nuevos análisis y modos de uso para seguir evolucionando. Además, se complementa este trabajo de presentación con el de desarrollo, puesto que en

la práctica el desarrollo de un programa no es estático, sino que puede ser mejorado de forma dinámica.

Los casos de uso se dividen en dispositivos Classic y Low Energy y por dispositivos. Cada dispositivo puede tener uno o varios usos. Se procura que los usos sean cortos y muy concretos, permitiendo relacionar la información mostrada en el programa con las acciones realizadas con los dispositivos. También es importante registrar el tiempo en el que suceden esas acciones, puesto que va a ser especialmente útil para el filtrado más tarde.

2.3.1. Classic

En esta sección se van a analizar los dispositivos que funcionan utilizando la tecnología Bluetooth Classic.

2.3.1.1. Auriculares inalámbricos: JBL Wave Buds

Figura 33: Fotografía de los auriculares JBL Wave Buds.

El dispositivo que se trata en este apartado se puede observar en la Figura 33.

Primera Conexión:

Si bien ya se había realizado una conexión previa, se olvida la conexión para poder simular una primera conexión. La conexión se realiza entre las 19:10 y las 19:12 del día 8 de enero de 2025.

Los protocolos detectados con el análisis son los protocolos que se podrían esperar tratándose de un dispositivo Classic para reproducción de audio. Al analizar con el modo “Bluetooth Classic”, se obtienen paquetes de los siguientes protocolos: HCI_EVT, HCI_CMD, L2CAP, AVDTP, AVRCP, RFCOMM. Al ejecutar un análisis con el modo “Not Implemented”, no se encuentran protocolos, por lo que todos los protocolos usados por el dispositivo se han tratado en este trabajo.

Lo primero es encontrar los tiempos en los que se conecta y desconecta. Puesto que he procurado desconectar el dispositivo nada más pasar las 19:10, a las 19:11, uso la pestaña de filtrado del protocolo para buscar esa desconexión

y la conexión previa. Encuentro que la desconexión sucede a las 19:11:07 y que la conexión anterior es a las 19:10:40, por lo que el tráfico que nos interesa está entre las 19:10:40 y las 19:11:07. También se puede observar un gran número de eventos "Completed Packets", por lo que sabemos que ha habido un intercambio de datos en este periodo.

En este caso, al negociarse el establecimiento de la conexión por primera vez, hay una intención de encriptado, como se puede deducir por el comando "Set Connection Encryption".

El análisis de L2CAP nos muestra la creación de varias sesiones al analizar los comandos de "signalling channel". Antes de usar un canal para enviar datos, se establece una conexión mediante este. Por lo que se puede observar, los canales RFCOMM no actúan de esta forma, cerrando los canales sólo al final de la conexión.

En cuanto a AVDTP, se puede observar que se configura un canal de audio y el códec a utilizar (MPEG-2,4), pero no se encuentran paquetes de ese códec porque no se envían datos al no reproducir contenido. Se establecen las configuraciones iniciales, al igual que con AVRCP. Con AVRCP, podemos ver una breve comunicación informando del estado actual del dispositivo (volumen, status de reproducción, etc).

Reproducción de audio:

Se establece una conexión entre las 20:10 y las 20:11 del mismo día en la cual se activa la reproducción de una canción, se mantiene la reproducción durante un breve periodo de tiempo (alrededor de 10 segundos), se para la reproducción y se desconecta Bluetooth. Abrimos el programa y filtramos en esta ventana temporal. Vamos primero a hacer el análisis en "Not Implemented".

Al hacer el análisis en este modo, podemos observar un protocolo no implementado llamado "RTP" y varios codecs con nombres similares entre ellos: "MPEG-1", "MPEG-2" y "MPEG-2.5". Después de una breve búsqueda se descubre que se trata de un protocolo para transporte en tiempo real que se usa junto con A2DP.

En cuanto al análisis en modo "Bluetooth Classic", podemos encontrar rápidamente los momentos precisos de conexión y desconexión mediante una pestaña de filtrado HCI_EVT, que suceden a las 20:10:09 y a las 20:10:36. Los protocolos encontrados son los mismos que en la situación anterior, solo que algunos muestran más información. Se puede observar que, de nuevo, hay un comando "Set Command Encryption" en la pestaña HCI CMD.

En la pestaña dedicada al protocolo AVRCP, podemos encontrar los momentos precisos en los que se mandan los comandos para reproducir la música y para pausarla de nuevo. Lo primero ocurre a las 20:10:17 y se pausa exactamente 10 segundos después, a las 20:10:27. En AVDTP, el número de paquetes es mayor, indicando no sólo la apertura del canal si no su uso.

Recepción de llamada de Whatsapp:

Se establece una conexión entre las 21:00 y las 21:03 del mismo día en la cual se conecta al dispositivo para recibir una llamada que dura exactamente 21 segundos, pasados los cuales cuelgo la llamada y desconecto el dispositivo.

Al hacer un análisis “Not Implemented”, obtenemos un resultado similar al anterior, solo que sin el códec “MPEG-2.5”. Al contrario de lo que se podría pensar, las llamadas no son tratadas mediante una conexión SCO si no usando protocolos basados en L2CAP, el cual usa una conexión de tipo ACL. Solo hay un paquete en cada ventana de cada códec, por lo que ninguno de estos dos se ha usado para la transmisión de la llamada. Por lo que se observa, parece que RTP ha sido usado para transmitir la información sin hacer uso de los otros codecs y utilizando un códec “MPEG” no registrado por Wireshark. Sin embargo, los paquetes RTP no parecen ser suficientes, y el tiempo durante el cual se envían tampoco.

El análisis usando el modo “Bluetooth Classic” no aporta mucha información nueva de interés. Se puede observar el establecimiento de la conexión mediante el filtro HCI EVT y los procedimientos de HCI CMD. También se puede observar actividad de los protocolos RFCOMM y L2CAP. AVCTP establece los canales de audio y AVRCP no registra el establecimiento de llamada, puesto que no tiene soporte para llamadas (al menos, no en este caso) y no registra las acciones ocurridas.

2.3.1.2. Altavoz inalámbrico: JBL Clip 3

Figura 34: Fotografía del altavoz JBL Clip 3.

El dispositivo que se trata en este apartado se puede observar en la Figura 34.

Primera Conexión:

De forma similar a lo ocurrido con los auriculares, se olvida la conexión para poder mostrar el tráfico de una primera conexión entre los dos dispositivos. La conexión se realiza entre las 21:40 y las 21:41 del día 8 de enero de 2025. En este caso, se desconecta el dispositivo periférico.

El análisis “Not Implemented” no muestra ningún protocolo en este caso. El modo de análisis Classic muestra los protocolos esperados: HCI_EVT, HCI_CMD, L2CAP, AVDTP, AVRCP, RFCOMM. La actividad mostrada es similar a la de los auriculares inalámbricos. Las mayores diferencias se pueden ver en los ajustes de los canales de audio mostrados por la ventana de filtrado AVDTP, que ya anticipa que las configuraciones van a ser distintas.

Reproducción de audio:

Se realiza una retransmisión de audio el mismo día de las 22:10 hasta las 22:12. De nuevo, solo se reproducen 10 segundos de una canción, solo que en este caso antes de iniciar la reproducción se reinicia la canción. Se plantea un segundo caso en de las 22:16 a las 22:17 en el cual se usan los botones del dispositivo para pasar a la siguiente canción y parar la reproducción en lugar de los del móvil.

El análisis de protocolos no implementados no produce ningún resultado en ninguno de los casos, mostrando que el dispositivo usa la tecnología estándar de Bluetooth. Al analizar en el modo Classic, podemos observar el uso del códec SBC además de los protocolos ya vistos en la anterior situación.

Comparando los dos casos de reproducción de audio, podemos ver diferencias en las ventanas dedicadas al protocolo AVRCP. En el primer caso, en el cual la reproducción se controlaba por medio del móvil, se ve que los comandos son de tipo “vendor dependent”, mientras que en el segundo caso los comandos son de tipo “Pass Through”, mostrando distintos tipos de controlar la reproducción mediante comandos.

Se puede observar un elevado número de paquetes SBC en ambos casos, indicando que el códec usado es este. Esto ya se esperaba en el anterior caso debido a que en el protocolo AVDTP se ve que se establece el canal de audio para ser usado con este códec. En cuanto a los demás protocolos, no hay cambios destacables.

Recepción de llamada de Whatsapp:

La conexión sucede entre las 02:00 y las 02:02 del día 9 de enero de 2025. Durante la conexión, se recibe una llamada que dura 17 segundos. Se pueden ver claramente los tiempos de conexión al filtrar por HCI EVT.

El problema que se encuentra, el cual ya sucedía con el anterior dispositivo, es que no hay ningún protocolo que coincida con la función de la llamada. No hay ningún protocolo de los que detecta el programa y, por tanto, de los que detecta Wireshark, que coincida con los tiempos de la llamada. No se encuentran paquetes de ningún tipo que se envíen durante un periodo de 17 segundos. A partir de esto, asumo que es un protocolo que Wireshark no detecta. También se podría suponer que los datos relacionados con las llamadas se envían usando una tecnología inalámbrica ajena a Bluetooth, pero esto resulta una posibilidad extraña. Con la información con la que contamos, se pueden establecer varias teorías, pero habría que profundizar en ellas para llegar a una conclusión.

2.3.1.3. Dispositivo Manos Libres: Amazon Echo Auto

Figura 35: Fotografía del dispositivo manos libres Amazon Echo Auto.

El dispositivo que se trata en este apartado se puede observar en la Figura 35.

Añadir elementos una lista:

La conexión sucede entre las 3:10 y las 3:11 del día 9 de enero de 2025. Este dispositivo funciona junto con la aplicación de Alexa. Se utiliza la funcionalidad de Alexa para añadir tres elementos a la lista de la compra. Para ello, se presiona el botón de acción en el dispositivo y se dice la frase *"Alexa, añade <elemento a añadir> a la lista de la compra"*. En este caso, los elementos a añadir han sido "plátanos", "ajo" y "mangos" y se añaden uno por uno.

Como de costumbre, se hace un análisis en el modo "no implementados" para ver si se usa algún protocolo que no conozcamos en profundidad. Al no encontrar ningún protocolo, concluimos que todos los protocolos usados son los que ya tenemos en cuenta.

El análisis en modo Classic nos muestra los protocolos más comunes: HCI CMD, HCI EVT, L2CAP y RFCOMM. Analizando L2CAP y RFCOMM brevemente, nos damos cuenta de que hay muchos más paquetes del protocolo RFCOMM. Podemos concluir que se usa RFCOMM para enviar los datos, los cuales se reciben por medio de la aplicación "Amazon Alexa". El "Type Frame" de la mayoría de los paquetes RFCOMM son de tipo "UIH", por lo que este es el tipo de mensaje que se usa para mandar datos. Se usa un único canal RFCOMM, el "0x16", y varios canales L2CAP para conectar con el mismo canal. Sería interesante desarrollar una forma de ver los datos enviados cuando RFCOMM no se usa para encapsular sino para enviar datos directamente.

El estudio de los protocolos HCI CMD y HCI EVT no provee información especialmente interesante. Lo más destacable es que la conexión ocurre sin detección de paquetes de tipo "Inquiry Result", los cuales permiten a un dispositivo anunciar su presencia, por lo que se debe usar un método alternativo. Mirando en detalle la ventana de HCI CMD, se puede observar que se ejecuta el comando "Accept Command Request", lo que sugiere que en este caso es el dispositivo periférico el que ha iniciado la conexión.

Establecimiento de una ruta para la navegación:

En esta situación, se le pide al dispositivo por voz que nos lleve a la "Avenida Paulista". Después de pedirle esto, requiere de una confirmación. Tuve que pedirlo 2 veces en la conexión porque la primera vez no funcionó y falló en el momento de la confirmación. La segunda vez fue exitosa y se abrió la aplicación de Google Maps en el modo ruta con las direcciones para llegar a la "Avenida Paulista". Esta conexión se realiza entre las 04:00 y las 04:03 del mismo día.

En este caso, de nuevo, podemos ver los mismos protocolos y sacar las mismas conclusiones. No se ha encontrado ninguna diferencia significativa. Destaca el gran número de paquetes RFCOMM que se utilizan para enviar unas pocas frases.

Reproducción de una canción:

Se establece una conexión entre las 04:42 y las 04:44 el mismo día y se da el comando "Reproduce en Spotify la canción Fusil contra Fusil de Silvio Rodríguez". Después de eso, el dispositivo informa de que se va a reproducir y procede a hacerlo.

Como se podía prever, los protocolos son los mismos que en las otras situaciones con este dispositivo. Al parecer, este dispositivo únicamente envía los datos usando el protocolo RFCOMM.

2.3.2.BLE

En esta sección se van a analizar los dispositivos que funcionan utilizando la tecnología Bluetooth Classic.

2.3.2.1. Ratón inalámbrico: Logitech M240 Silent

Figura 36: Fotografía del ratón Logitech M240 Silent.

El dispositivo que se trata en este apartado se puede observar en la Figura 36.

Primera conexión:

Se conecta el Bluetooth del móvil y se conecta con el periférico por primera vez a las 14:05, desconectando el periférico en el mismo minuto y el Bluetooth del móvil acto seguido, por lo que la conexión sucede en la ventana 14:05-14:06. Esto sucede el día 9 de julio de 2025.

Al analizar con el modo de protocolos no implementados, encontramos dos protocolos: SMP y HCI ACL. HCI ACL solo contiene dos paquetes y por lo que se puede observar parece tener una labor de transporte. No parece especialmente interesante. SMP, sin embargo, contiene un número mayor de paquetes (13) y su labor es claramente importante. Mediante esta visualización básica, podemos ver las distintas etapas que ocurren a la hora de establecer una cierta seguridad para la tecnología Bluetooth. Aunque no se le había dado mucha importancia hasta ahora en el proyecto al no considerarla como esencial para el transporte de los datos, se puede observar que es un protocolo usado y se postula como un posible protocolo a ser implementado en el futuro.

En los paquetes SMP se puede observar cómo se intercambian informaciones relacionadas a los ajustes de seguridad de cada uno de los dispositivos y se pueden ver varios intercambios de información y claves que permiten el cifrado de la información

En cuanto al análisis en modo Low Energy, observamos los protocolos esperados: HCI CMD, HCI EVT, L2CAP y ATT. Al analizar los paquetes HCI EVT, se pueden observar claramente los paquetes “LE Advertising Report” que llegan al controlador por parte de dispositivos Low Energy para informar de su presencia. También se pueden observar los paquetes de conexión y desconexión. HCI CMD se ocupa de establecer ajustes varios. L2CAP presenta tan solo dos paquetes que se usan para actualizar una conexión. El tráfico ATT cuenta con muchos paquetes, contando con muchos handles diferentes. Esto suele suceder

cuando se establece la conexión ya que se están intentando obtener los distintos atributos del dispositivo al que se conecta, por lo que en esta etapa de "descubrimiento", hay un importante tráfico de información. Se pueden ver varias "Read by Type Request", utilizadas con la finalidad de descubrir los distintos servicios y handles relacionados.

Pruebas de clics:

El mismo día a las 14:10, se vuelve a conectar el móvil con el ratón para hacer diferentes pruebas. Cada una de ellas se ejecuta en los periodos que se indican más adelante, al principio del minuto puesto que son tareas cortas. Vamos principalmente a analizar las ventanas generadas con el filtrado ATT, ya que nos permite ver los datos que se envían y no como se establece la conexión. Solo vamos a usar el modo LE, puesto que un análisis con el modo "no implementados" permite ver que todos los protocolos en juego son tomados en cuenta por el modo de análisis LE.

- Cinco clics derechos: 14:10-14:11

  El único handle que se repite las veces suficientes como para ser el relacionado con la información enviada de los clics tan solo puede ser el 0x0028. Ahora, debemos buscar un valor que se repita 5 veces. Resulta complicado porque al ser la primera prueba, aún hay algo de ruido causado por la etapa de establecimiento. El único valor que se repite 5 veces es el valor "02000000000000", por lo que parece razonable asociarlo con el evento del clic derecho.

- Cinco izquierdos: 14:11-14:12

  Se toma el mismo enfoque que con la prueba anterior. Esta vez, solo vemos un handle, el cual es el 0x0028 de nuevo. Parece claro que este es el que usa para comunicar, por lo menos, la información relacionada con los clics. En este caso el tráfico es más reducido y podemos encontrar rápidamente el valor: es el "01000000000000". Parece interesante que, si le sumamos uno al único dígito que no es cero en el valor, obtenemos el valor del clic derecho.

- Cinco clics con rueda: 14:12-14:13

  De nuevo, el handle es 0x0028, haciendo aún más factible que este sea el único handle usado para la comunicación de los clics. El valor se encuentra fácilmente: es el "04000000000000". Hay un patrón claro entre los distintos valores de los clics: solo el segundo dígito empezando por la derecha es diferente de 0. Imaginaba que sería un 3, pero se trata de un 4. Tal vez solo se usan valores que son potencia de 2.

- Tres izquierdos, uno derecho, tres izquierdos: 14:13-14:14

  Estas dos últimas pruebas nos permiten confirmar que los valores que creemos que están relacionados con los clics son los correctos. Y, efectivamente, parece que estaba en lo cierto. Se puede encontrar una sucesión del valor "01000000000000" 3 veces, el "02000000000000" una vez y el "01000000000000" 3 veces de nuevo.

- Tres derechos, uno con rueda, tres derechos: 14:14-14:15

Realizamos lo mismo que con la prueba anterior, obteniendo el resultado esperado: 3 veces “02000000000000”, una “04000000000000” y tres “02000000000000” de nuevo.

El valor “00000000000000” se puede encontrar varias veces a lo largo de las pruebas. Parece que se trata de un valor que se usa para separar los eventos enviados por ATT.

Pruebas de desplazamiento:

Estas pruebas pueden aportar información interesante, sin embargo, son más complicadas de analizar que las anteriores, porque modificaciones imperceptibles en la dirección del movimiento del ratón pueden modificar los datos. Como humano, no puedo mover el ratón de forma exacta en una dirección concreta. Los desplazamientos se realizan de forma continua durante todo el minuto. El enfoque es el mismo que el de las pruebas anteriores.

- Desplazamientos pequeños hacia arriba: 16:25-16:26

  Lo primero que se puede observar es que hay muchos más paquetes que en las pruebas de clics. De nuevo, el handle principal es el 0x0028, aunque también hay otros que se usan. Podemos suponer que los otros están relacionados con el proceso de descubrimiento del dispositivo y que más tarde no serán usados como sucede con las pruebas de clics. Podremos confirmarlo en la siguiente prueba. De entre los valores presentes, destaca uno, el “000000f0ff0000”. Otro valor también destaca: “0000ffffff0000”. Puesto que es el que más se repite, podría tratarse del valor asociado a un desplazamiento hacia arriba de una cierta magnitud.

- Desplazamientos pequeños hacia abajo: 16:26-16:27

  Solo se usa el handle 0x0028, dando más fuerza a lo ya sospechado. En este caso, la mayoría de los valores siguen el formato “000000X0000000”, siendo X un 1 o un 2. Puede ser que él sea 1 o 2 dependiendo de la magnitud del movimiento, y que el séptimo dígito por la izquierda sea el relacionado con un movimiento hacia abajo.

- Desplazamientos pequeños hacia la derecha: 16:27-16:28

  Sucede algo similar a lo ocurrido en la prueba anterior, solo que ahora el patrón es “00000X00000000”, tomando X diferentes valores, como 1, 2, 3 o 4.

- Desplazamientos pequeños hacia la izquierda: 16:28-16:29

  Destaca el valor “0000ff0f000000”. Parece similar a lo sucedido con los desplazamientos hacia arriba. Parece que los movimientos hacia la derecha y hacia abajo comparten comportamiento, mientras que los desplazamientos hacia arriba y hacia la izquierda comparten otro comportamiento diferente. Haría falta un análisis más profundo para sacar más conclusiones.

2.3.2.2. Pulsera inteligente: Xiaomi Smart Band 8 Active

Figura 37: Fotografía de la pulsera Xiaomi Smart Band 8 Active.

El dispositivo que se trata en este apartado se puede observar en la Figura 37.

Cambio de aspecto:

Es especialmente difícil conectar este dispositivo con el móvil, por lo que las tres situaciones se han dado en la misma conexión, así que no se va a encontrar eventos de conexión o desconexión. Esta primera prueba se realiza de las 19:20-19-21 del día 9 de enero de 2025. Se usa la aplicación "Mi Fitness" de Xiaomi para enviar un diseño nuevo a la pulsera inteligente.

El análisis de protocolos no implementados no presenta ningún resultado. La ventana HCI EVT muestra principalmente eventos del tipo "Number of Completed Packets", lo que indica mucho tráfico de datos. Los únicos paquetes HCI CMD son 4 y son del tipo "LE Connection Update". L2CAP actualiza 5 veces los parámetros de conexión usando el "signaling channel".

El tráfico del protocolo ATT presenta muchos paquetes y varios handles distintos. Se trata de una operación compleja ya que se están enviando archivos como imágenes. Se usan 3 handles: 0x0015, 0x0012 y 0x001e. El que destaca es el handle 0x0001e ya que se usa varias veces para enviar 13 paquetes seguidos con valores muy largos. Podemos imaginar que es usando este handle que se envían los archivos pesados. Posiblemente los otros handles están relacionados con los metadatos de estos archivos o con los detalles del envío de estos.

Envío de información de salud (ritmo cardiaco):

Esta prueba se realiza el mismo día que la anterior. En la misma aplicación mencionada antes, hay una opción para recibir actualizaciones del ritmo cardíaco cada minuto. Después de habilitar esa opción y dejarla funcionando unos minutos (alrededor de 7), llegan dos actualizaciones, una de las 19:30 y otra de las 19:31. Se deberían haber recibido más, pero solo se reciben dos. No conozco la razón de esto. No hay ningún protocolo no

implementado en este tráfico. Analizamos el tráfico en modo LE entre las 19:30 y las 19:32.

El tráfico es muy similar al estudiado anteriormente. De nuevo la gran mayoría de paquetes HCI EVT son de tipo “Number of Completed Packets”, por lo que se prevé de nuevo bastante tráfico. En cuanto a los paquetes HCI CMD, solo podemos observar 5 veces el comando “LE Connection Update”. Estos coinciden con 5 actualizaciones de la conexión que se pueden ver en el “signalling channel” de L2CAP.

En esta ocasión, el handle que se usa para enviar grandes cantidades de información es el 0x0018. Podemos imaginar que este handle está relacionado con el envío de información relacionada con el ritmo cardiaco. De nuevo podemos ver los handles 0x0012 y 0x0015, por lo que nos podemos imaginar que estos handles son importantes para cualquier comunicación entre el móvil y la pulsera. Podrían tratarse de handles con una función similar a los canales de control de otros protocolos o dar información acerca de qué información se va a enviar o recibir.

Transmisión de información del tiempo:

El mismo día se hace una última prueba con el mismo dispositivo entre las 19:45 y las 19:46. La pulsera presenta una función para hacer que el móvil suene pulsando un botón en la pulsera, para permitirnos encontrarlo. Esta prueba consiste en pulsar el botón para hacerlo sonar y luego pulsar otro botón para detener este sonido. Este proceso se realiza un total de tres veces. El análisis de protocolos no implementados no produce ningún resultado.

En este caso, en la ventana HCI EVT podemos observar varios eventos de tipo “Number of Completed Packets” pero no tantos como en las anteriores pruebas. Esto tiene sentido ya que está vez no hace falta transmitir datos más extensos y solo se trata de informar de que se ha presionado un botón. Esta vez podemos encontrar dos comandos de actualización de la conexión LE y también dos actualizaciones de la conexión en L2CAP. La idea de que estos dos eventos están relacionados gana fuerza. También se pueden observar cuatro comandos de tipo “LE Set Extended Scan Enable”.

En cuanto al protocolo ATT, podemos ver que solo hay dos handles está vez y son dos de los que ya vimos anteriormente: el 0x0012 y el 0x0015. Al contrario de lo hipotetizado anteriormente, parece que estos dos handles no solo se usan para control o metadatos, si no que se puede usar para enviar información. También podría interpretarse como que al presionar el botón para encontrar el móvil estamos ejecutando un comando, por lo que aún serían handles de “control”. Lo que es relevante es que estos handles son capaces de comunicar algo sucedido sin necesidad de otros. Hay varios valores asociados a estos handles que se repiten exactamente tres veces, pero son valores complejos y resulta complicado sacar asociaciones entre estos valores y su función. Podemos observar que sucede lo mismo tres veces: se envía un valor que termina por “db”, luego otro que acaba por “3a”, uno que varía y uno que acaba por “f0”. Se puede identificar un patrón, pero la complejidad de los valores requeriría un análisis más profundo.

2.3.2.3. Teclado inalámbrico: C3Tech K-BT40BK Preto Mini

Figura 38: Fotografía del teclado C3TECH K-BT40BK Preto Mini.

El dispositivo que se trata en este apartado se puede observar en la Figura 38.

Primera Conexión:

El día 9 de enero en el periodo 16:40-16:41 se realiza una conexión con el teclado. Este dispositivo tiene la particularidad de que el móvil no lo recuerda, así que siempre es como si fuera la primera vez que se conecta. La desconexión se realiza apagando el periférico, después de lo cual se apaga el Bluetooth en el teléfono.

Al hacer un análisis de los protocolos no implementados, encontramos el protocolo SMP de nuevo, siendo el único encontrado en este análisis. En este punto, se decide implementar en el programa este protocolo y dedicarle una sección en la teoría, ya que es más común de lo inicialmente pensado. Esto se hace después del análisis, para aportar facilidades en el análisis de este protocolo en el futuro.

En el análisis LE, los protocolos encontrados son los típicos para este tipo de tecnología: HCI CMD, HCI EVT, L2CAP y ATT. Al igual que en los otros casos, podemos ver cuando se establece y se desconecta la conexión mediante la pestaña HCI EVT. Como sucede en el caso del ratón, el único tráfico L2CAP son dos paquetes con comandos enviados por el "signalling channel". HCI CMD contiene una variedad de comandos para controlar el flujo de conexión y descubrimiento de dispositivos, al igual que comandos para configurar el controlador. En cuanto a ATT, hay un intercambio significativo de paquetes, que permite al dispositivo central conocer la funcionalidad del periférico.

Pruebas de tecleo:

En el punto en el que empiezan estas pruebas, el teclado ya estaba conectado, por lo que no se va a estudiar el proceso de conexión. Nos vamos a interesar por el comportamiento del protocolo ATT al presionar ciertas teclas. Nos interesa ver los valores escritos usando handles. Las pruebas empiezan a las 18:45 del mismo día de la situación anterior y acaban a las 18:49, momento en el cual se desconecta el Bluetooth del móvil. Todos los protocolos presentes en esta conexión están implementados.

- "1234567890": 18:45-18:46

Se pueden observar dos handles, “0x002c” y “0x0015”. De estos, solo el 0x0015 se utiliza para enviar información mediante la escritura del valor del handle. Todos los mensajes enviados siguen el siguiente formato: “0000XY0000000000”, siendo X un dígito y Y un dígito o una letra. Podemos suponer que X también podría llegar a ser una letra, pero en este caso no se da. Se envían 10 mensajes con este formato, con un mensaje “0000000000000000” después de cada uno de ellos.

- “qqq<Backspace><Backspace><Backspace>”: 18:46-18:47

  Ocurre lo mismo que en la prueba anterior, se envían mensajes con el mismo formato “0000XY0000000000” descrito antes separados por mensajes “0000000000000000”. En este caso, como se podría esperar, se envía tres veces el mismo código y tres veces un código diferente. Resulta razonable asumir que el primer código representa la “q” y el segundo la tecla <Backspace>. No se puede saber con seguridad aún si el valor está relacionado con la tecla o con el valor enviado.

- “aaa<BloqMayus>A<BloqMayus>aaa”: 18:47-18:48

  En esta prueba sucede algo que no ocurre en las otras. Entra en juego un nuevo handle, el 0x0021, al que se escriben valores de dos dígitos, siendo el primero de los dígitos 0 en los dos mensajes vistos. Se envía tres veces el mismo valor con el formato ya visto al handle 0x0015, como en las dos anteriores pruebas (este valor está asociado a la “a”). Después de esto se envía un valor diferente con el formato ya visto al mismo handle, y, acto seguido, se envía el mensaje “02” al handle 0x0021, posiblemente para indicar que ahora las letras son mayúsculas. Parece un handle relacionado con el estado del teclado y no con las teclas presionadas. Lo siguiente que sucede es que se envía el mismo mensaje que para la “a” antes, por lo que los valores parecen estar asociados con las teclas y no con los valores enviados. Vuelve a aparecer el valor que parece estar asociado a la tecla de bloqueo de mayúsculas, se envía 00 al handle 0x0021 y de nuevo se envían tres mensajes iguales (para las “a” de nuevo). Después de cada envío al handle 0x0015, se envía otro mensaje “0000000000000000”.

- “<Shift Apoyado>12345->!@#$%”: 18:48-18:49

  Sucede algo muy interesante en este caso. Aparece de nuevo el handle 0x002c, pero no tiene valor, por lo que no es de interés. Todos los demás mensajes están relacionados con el handle 0x0015. El formato de los mensajes es diferente en este caso, puesto que el primer mensaje es “0200000000000000”. Después de este se van intercalando mensajes con el formato “0200XY0000000000” con el mensaje “0200000000000000”, enviando un mensaje “0000000000000000” al final de esta sucesión. Parece que los mensajes con el formato “0200XY0000000000” quieren decir que se ha presionado una tecla mientras que el shift estaba presionado.

2.3.2.4. Lámpara inteligente: ELG SHLL100

Figura 39: Fotografía de la lámpara ELG SHLL 100.

El dispositivo que se trata en este apartado se puede observar en la Figura 39.

Apagado y encendido:

Esta prueba fue realizada el 9 de enero de 2025. Mediante el uso de la aplicación ELG Connect podemos modificar la luz de una bombilla inteligente, pudiendo cambiar la intensidad de la luz, al igual que su color y otras funcionalidades varias. Esta prueba comienza a las 19:55 y acaba a las 19:56 y la bombilla empieza conectada por BLE y acaba el minuto aún conectada. Durante este minuto se apaga y enciende (en este orden) la bombilla cinco veces. Mantenemos la conexión durante todo el minuto para comprobar si hay un intercambio de información sin importar si se realiza una acción o no. Dicho de otra forma, se quiere ver si hay un intercambio constante de información con un cierto intervalo y si una acción como apagar la luz o encenderla puede afectar a este cambio. Todos los protocolos están incorporados.

No se encuentra ningún paquete HCI CMD relacionado con LE. Todos los eventos encontrados son del tipo “Number of Completed Packets” hasta un cierto punto (19:55:20). Después, hay varios del tipo “LE Advertising Report”, uno de tipo “Command Complete” (analizando en el modo Classic se puede observar que está relacionado a un comando ajeno a LE, de tipo “Vendor Command”) y un último de tipo “Disconnect Complete”. Esto último es extraño, ya que yo no desconecté el Bluetooth. Puede ser que se desconecte automáticamente pasado un tiempo o que al hacer ciertas acciones en la aplicación se desconecte.

En cuanto al protocolo ATT, hay un intercambio de mensajes de tamaño medio usando dos handles, el “0x0012” y el “0x0014”. El primero siempre lo usa el móvil para escribir y el segundo lo usa la bombilla para notificar al móvil. El tráfico no es constante y sucede del segundo 8 al 20, por lo que suponemos que solo hay un intercambio de datos cuando sucede algo. Las escrituras y notificaciones se suceden de tal forma que no me permiten encontrar un patrón. Los valores enviados y recibidos están compuestos por dígitos y letras sin un

orden aparente y una longitud variable. No consigo encontrar de qué forma se comunica la información.

Vinculación con música:

Se hace una segunda prueba con este dispositivo. Hay una funcionalidad de la bombilla que permite ponerla en "modo música". Si se reproduce música en el móvil a la vez que se está en este modo, la luz cambia de acorde a la música que esté siendo reproducida. Se activa este modo y se reproduce una canción durante alrededor de 15 segundos. Durante el minuto 20:12-20:13, se habilita Bluetooth, se realiza lo explicado y se vuelve a desconectar. No hay ningún protocolo no implementado involucrado en este proceso.

Se puede observar que hay un tráfico inusual en la ventana de HCI EVT. Se realizan varias conexiones y se reciben muchos paquetes de Advertising en algunos periodos. Al abrir una ventana ATT, se puede observar que durante este minuto también se comunica con la pulsera inteligente. Hay ciertos dispositivos LE que pueden tener un comportamiento "invasivo", conectando sin requerir de una confirmación o alguna acción clara. Posiblemente las anomalías vistas en la ventana HCI EVT hayan sido causadas por la pulsera. En la ventana HCI CMD se pueden ver varios comandos modificando los ajustes del controlador.

Si nos centramos en el tráfico entre el móvil y la bombilla en la pestaña ATT, podemos observar dos handles que no se habían visto antes: el 0x000f y el 0x0015. Los valores no aportan información acerca de su uso, pero parece que están relacionados con el proceso de conexión inicial, ya que aparecen al principio y no vuelven a aparecer más tarde. Sucede lo mismo que en la anterior prueba: se usan los handles 0x0012 y 0x0014 para escritura y notificaciones respectivamente pero no consigo encontrar ningún tipo de patrón. Los últimos paquetes enviados en este protocolo usan el handle 0x0003. Dado que solo aparece al final, posiblemente esté relacionado con un proceso de desconexión.

Cambios de color:

Se hace una última prueba en el periodo 04:55-04:56 del día siguiente, el 10 de enero de 2025. Se comienza con el dispositivo ya conectado y se desconecta el Bluetooth después de finalizar la prueba. Consiste en hacer la siguiente transición de colores en la bombilla usando el siguiente orden: Verde, Rojo, Azul, Amarillo y por último Morado. Todos los protocolos relacionados están implementados.

En la ventana HCI CMD se pueden observar un tipo de comandos "LE Remove Device From Filter Accept List" que ya se había visto antes. Investigando este tipo de comando, se descubrió que elimina un dispositivo de la "Filter Accept List", que permite filtrar la de que dispositivos se recibe información. La ventana HCI EVT muestra mensajes "Command Complete" y "Number of Completed Packets".

En cuanto al protocolo ATT, sucede lo mismo, solo que sin los handles involucrados con la etapa inicial (0x000f y 0x0015). Se usan de nuevo los handles 0x0012 y 0x0014 de la misma forma y el handle 0x0003 al final de la conexión. Los valores enviados no me permiten formular ninguna hipótesis.

Como idea general en estos casos de usos de BLE, podemos destacar que suele haber un número reducido de handles para la comunicación relacionada con la funcionalidad principal del dispositivo, sin contar inicialización o terminación. Esto se puede ver en este caso y en los del ratón y el teclado.

Implementación SMP

Se ha implementado el protocolo SMP para el modo BLE y se han generado el perfil de Wireshark adaptado y disecciones para las primeras conexiones del ratón y el teclado inalámbricos. Se puede comprobar que el programa funciona como se esperaba y la implementación ha sido exitosa. Se pueden observar las etapas del establecimiento de las claves para la seguridad con SMP.

Documentos de especificación usados como base para la selección de datos de interés en Wireshark, para el tratamiento de los datos en el programa y para el análisis de los casos de uso: [23], [24], [25], [26], [27], [28], [29], [30] y [31].

3.Resultados y conclusiones

Los resultados del trabajo se pueden ver bien reflejados en los casos de uso, puesto que combina lo estudiado en la teoría y el uso del programa desarrollado. Los resultados son generalmente positivos. Con lo aprendido, se puede entender el tipo de mensajes que se envían y que función general tienen. Sin embargo, hay que entender una cuestión: la especificación "Core" de Bluetooth, la cual es la más importante pero no la única para estas tecnologías, cuenta con 3816 páginas en su versión más reciente, la 6.0. Es sencillamente imposible abarcar toda esa información en un TFG. Lo que sí es posible es generar interés y conducir el aprendizaje.

El resultado de este proyecto es satisfactorio por lo que se puede ver en casos de uso como el del teclado inalámbrico. En este se puede ver que se puede comprender un tráfico sencillo con el programa y lo aprendido. Sin embargo, no todos los casos de uso son tan sencillos ni todas las etapas de la conexión se entienden de la misma forma. Destaca que, en muchos casos de uso, se muestra incertidumbre respecto a ciertos paquetes o grupos de paquetes. Esto puede dar la sensación de que el proyecto no ha sido satisfactorio pero, de nuevo, resulta inabarcable cubrir estas tecnologías por completo. Cada vez que se indica que no se sabe cómo funciona algo, no debería ser visto como una carencia sino como una oportunidad de aprendizaje. De esta forma se puede ver que el objetivo didáctico principal de servir como introducción se cumple: tras leer la teoría, usamos el programa y podemos encontrar lagunas de conocimiento dentro de esa comprensión general de lo que está sucediendo. Podemos concluir que, como un comienzo, es satisfactorio y prometedor, teniendo potencial para llegar a ser más.

Después de usar Wireshark extensamente, he podido descubrir que es un programa muy completo que puede proveer la funcionalidad que el programa desarrollado en este trabajo ofrece. Sin embargo, es más complejo de usar y en ocasiones resulta lento, ya que la gran cantidad de opciones que ofrece puede hacer que la información más importante pase desapercibida. Si bien ya estaba cómodo usando Wireshark, me sorprendí varias veces usando mi programa en vez de Wireshark para encontrar ciertas informaciones. El programa que he desarrollado presenta ventanas para cada protocolo que se pueden generar con un simple clic y a las que se puede volver con solo un clic también, sin necesidad de más complejidad. Además, el flujo de datos se presenta de una forma sencilla, lo que permite hacerse una idea de lo que sucede más rápido.

Sin duda, no se plantea este programa como un competidor de Wireshark, ya que este es un programa muy potente. Sin embargo, el enfoque tomado por el programa aquí desarrollado es interesante y ese trato personalizado para Bluetooth resulta esencial a la hora de encontrar informaciones de forma rápida. Con más trabajo e investigación, podría convertirse en un programa verdaderamente útil. Como se menciona en el capítulo de Bluetooth Classic en la sección de Teoría, Classic tiene muchos protocolos, y cada uno de ellos funciona de una forma diferente. Desarrollar un programa capaz de organizar la información esencial de cada protocolo y presentarla de forma interactiva, visual y comprensible con opción de exportar la información de más interés, es una tarea titánica. Hacer un programa de estas características exclusivamente para BLE parece una tarea mucho más abarcable, e interesante a futuro dado que es la tecnología que más se está desarrollando. Idealmente este programa también contendría pequeños tutoriales y guías para entender las tecnologías Bluetooth.

Yo, después de usar el programa que se ha desarrollado, tengo la certeza de que un programa de estas características, pero con más desarrollo podría resultar increíblemente útil para entender Bluetooth y desarrollar con estas tecnologías. La conclusión más importante que saco de este TFG es que este trabajo puede servir como prueba de concepto de una idea prometedora. Las tecnologías Bluetooth están más presentes que nunca y no siempre se enseña cómo funcionan. Cualquier método que facilite la comprensión de estas puede ser muy valioso.

3.1. Resultados objetivo a objetivo

• Búsqueda de documentación de calidad: Se han encontrado recursos de calidad que han permitido el desarrollo satisfactorio del trabajo. Hay una gran cantidad de libros dedicados a Bluetooth Low Energy que están muy actualizados y cuyo enfoque era ideal. En cuanto a Bluetooth Classic, los libros eran más antiguos y demasiado técnicos (excepto el de Gordon Colbach), explicando los temas en gran detalle, lo que en ocasiones complicó su comprensión. Sin embargo, se pudo extraer la información clave y resumir el contenido. También, se aprendió a usar los documentos de especificaciones oficiales del Bluetooth SIG, lo que resultó muy útil para el desarrollo del programa, ya que proveía información más específica.

• Lectura y comprensión de los recursos seleccionados para aprender acerca de las distintas tecnologías que componen Bluetooth y del funcionamiento de los protocolos: Se seleccionaron y comprendieron satisfactoriamente las secciones más útiles para el desarrollo del trabajo. Como ya se ha expuesto en este trabajo en varias ocasiones, los apartados de seguridad y los relacionados con la parte física de Bluetooth solo se han analizado de forma superficial, mientras que los que estaban más ligado con el componente Host se han comprendido y analizado en profundidad.

• Diseño del material para la exposición concisa de los tipos de Bluetooth y su funcionamiento: Se han desarrollado varios diagramas para establecer una base visual de lo que se iba a exponer o ya se había expuesto en el texto. Si bien considero que los diagramas esenciales están presentes en la sección de teoría, se podrían haber añadido más recursos visuales. Esta parte del objetivo se satisface, pero se podría haber hecho mejor. En cuanto al texto, la información está bien resumida y solo se dan los elementos esenciales, por lo no se destaca ningún problema en cuanto a esto.

• Búsqueda del método más apropiado para la captura de los paquetes involucrados en la comunicación: Tras barajar las distintas posibilidades para captar el tráfico, se selecciona la que aporta más beneficios a la hora de desarrollar este proyecto. La justificación de por qué es el mejor método se trata de forma extensa en el apartado 2.2.3 de este trabajo.

• Desarrollo de un programa capaz de captar la información deseada: Este objetivo se puede dar por cumplido puesto que ni siquiera ha sido necesario su desarrollo. Se ha encontrado una alternativa para obtener los datos deseados usando una funcionalidad ya desarrollada por el equipo de desarrolladores de Android.

• Implementación de una forma de procesar la información captada para su posterior presentación: Se cumple satisfactoriamente. Se selecciona en Wireshark toda la información a tratar y se usa el programa desarrollado para clasificar la información seleccionada según el protocolo.

• Creación de una representación gráfica que presente cada etapa y su respectiva información de forma clara y completa: El programa desarrollado se ocupa de presentar la información de forma personalizada para cada protocolo, presentándola de una forma u otra dependiendo del tipo de información que se presente. Como se explica al comienzo del apartado 3 de este trabajo, la

cantidad de protocolos e información a presentar es muy grande. Teniendo esto en cuenta considero que el resultado es positivo.

4.Análisis de Impacto

En un ámbito general, no se puede considerar que el proyecto en sí tenga un impacto negativo. Se puede argumentar que la proliferación de los dispositivos con Bluetooth incorporado puede promover la creación de dispositivos de baja calidad para usos cotidianos o dispositivos de usar y tirar. Dado que Bluetooth es muy práctico y los controladores LE son baratos, podrían generarse muchos desechos si proliferan este tipo de dispositivos de baja calidad. Sin embargo, eso no se puede atribuir a la tecnología en sí, si no al uso que se le da.

El impacto positivo que puede tener el proyecto es significativamente mayor al negativo. Bluetooth es una de las formas más claras en las que las tecnologías de la información y la comunicación se hacen presentes en nuestras vidas. Existen todo tipo de sensores que hacen mediciones de características físicas que tienen la capacidad de conectarse por Bluetooth. De esta forma se pueden monitorizar eventos que suceden en la vida real, obteniendo datos de estos. Estos datos pueden ser usados por toda una variedad de herramientas informáticas para ayudarnos a tomar las mejores decisiones posibles. También hay todo tipo de actuadores capaces de conectarse por Bluetooth, por lo que no solo se puede obtener información, si no que se pueden tomar decisiones y realizar acciones que tienen impacto en la vida real.

La capacidad de Bluetooth de conectarse sin usar cables la hace especialmente interesante en situaciones en las que resulta complicado establecer un cableado ethernet o cualquier otro tipo de método de comunicación por cable. Esto abre nuevas posibilidades. El continuo desarrollo de Low Energy también trae novedades que pueden tener un impacto en las tecnologías del futuro. Comprender bien cómo funciona es esencial para su desarrollo y para asegurarnos de que se está usando correctamente.

Voy a hablar de cómo creo que el proyecto puede tener un impacto positivo en los siguientes ámbitos.

- Personal:

A nivel personal, este proyecto puede ayudar a los individuos a aprender nuevas habilidades que pueden marcar la diferencia en el mercado laboral. Conocer bien las tecnologías Bluetooth puede ser positivo a la hora de trabajar en el desarrollo de aplicaciones relacionadas con estas. Puesto que son tecnologías que cada vez se usan más, resulta muy interesante introducirse en este mundo.

- Empresarial:

A nivel empresarial puede suponer una herramienta útil para formar a los empleados. Si se busca desarrollar algún tipo de producto Bluetooth, es importante saber cómo funcionan estas tecnologías. El uso de este trabajo puede acelerar el proceso de aprendizaje.

- Social:

El desarrollo de las tecnologías Bluetooth puede hacer que se haga más barato su uso, permitiendo que sea más generalizado. Formar a profesionales en esta tecnología es esencial para que se hagan avances, lo cual es esencial para que cada vez más gente tenga acceso a estas tecnologías.

- Económico:

Los controladores BLE son considerablemente más baratos que los de Bluetooth Classic. Para desarrollar las tecnologías, es imprescindible conocerlas a fondo. El desarrollo de BLE puede tener un impacto económico significativo, ya que reduce de forma importante los gastos relacionados con tecnologías inalámbricas. Cuanto más se desarrolle BLE y más usos tenga, mayor impacto tendrá Bluetooth en la economía. Siempre que se encuentra una alternativa más barata, puede haber empresas que quiebren al no poder adaptarse, lo que puede ser negativo. Sin embargo, creo que es razonable decir que es positivo para la economía que algo se abarate. De nuevo, la formación de los profesionales encargados de desarrollar BLE pueden usar este proyecto como base para su aprendizaje.

- Medioambiental:

Bluetooth Low Energy es una tecnología que usa muy poca energía. El desarrollo de esta tecnología puede suponer encontrar alternativas de menor consumo para algunas tecnologías que son usadas hoy en día. Para que BLE se desarrolle apropiadamente, es necesario que haya profesionales formados adecuadamente que conozcan la tecnología en profundidad. Mi proyecto puede ayudar a esos profesionales a dar los primeros pasos.

- Cultural:

A nivel cultural puede ayudar a generar una mayor comprensión del mundo que nos rodea. Las tecnologías Bluetooth son muy populares, pero muy poca gente sabe cómo funcionan realmente. A pesar de su presencia en nuestra vida diaria, otras tecnologías como HTTP son

mucho mejor conocidas. Este proyecto puede ayudar a que la gente conozca mejor cómo funciona Bluetooth, ampliando el conocimiento general de las TIC, las cuales se hacen más importantes con cada día que pasa.

4.1. Objetivos 2030

- Objetivo 3: Salud y Bienestar

Los dispositivos como "smartbands" utilizan bluetooth para conectarse con el móvil y enviar los datos monitorizados acerca del estado de un individuo. Estos pueden estar relacionados con el ritmo cardiaco, la calidad del sueño, el nivel de SpO2, etc. Proveer formación relacionada con Bluetooth puede hacer que el desarrollo de estas tecnologías mejore, y por lo tanto que estos dispositivos también lo hagan.

- Objetivo 6: Agua Limpia y Saneamiento

Sensores que midan la calidad del agua pueden ser conectados usando Bluetooth (BLE posiblemente). Estos pueden aportar información acerca del estado del agua.

- Objetivo 7: Energía Asequible y No Contaminante

Bluetooth Low Energy tiene ese nombre debido a su bajo consumo de energía. El Bluetooth SIG está centrado en desarrollar esta tecnología, por lo que se puede esperar que no haya demasiadas actualizaciones relacionadas con Bluetooth Classic en el futuro. Este proyecto ayuda con la formación de profesionales en BLE, lo que puede suponer un menor consumo de energía en el futuro.

- Objetivo 8: Trabajo Decente y Crecimiento Económico

Puesto que las tecnologías Bluetooth se usan cada vez más, formar a profesionales para que puedan desarrollar las tecnologías Bluetooth o aplicaciones que las usen puede suponer un impacto económico. Desarrollar este mercado creciente puede suponer un impulso a la economía en el terreno de las TIC.

- Objetivo 11 Ciudades y Comunidades Sostenibles

Bluetooth, y particularmente BLE, es especialmente interesante relacionado con la idea de "smart cities". Sensores como sensores ambientales, medidores de energía y agua y dispositivos de movilidad permiten obtener

información del estado de la ciudad. También se pueden instalar luces inteligentes y sensores de gestión de residuos para mejorar estos servicios. Muchos dispositivos diferentes pueden ser usados para mejorar la gestión de la ciudad, muchos de los cuales posiblemente usen alguna tecnología BLE.

- Objetivo 13: Acción por el Clima

Las tecnologías Bluetooth permiten comunicarse con sensores que pueden monitorizar varias características físicas. Esto puede facilitar la recolección de datos relacionados con el cambio climático, ayudando a tomar las decisiones correctas y proveyendo datos para hacer conocido cómo está la situación actual.

- Objetivos 14 y 15: Vida Submarina y Vida de Ecosistemas Terrestres

El uso de sensores inteligentes que usen Bluetooth puede ayudar a monitorizar la actividad de seres vivos. Está información puede ser útil para conocer el comportamiento de los animales y saber cómo reaccionan a ciertas circunstancias.

5.Bibliografía

[1]https://es.wikipedia.org/wiki/Marca_(registro)

[2]https://www.bluetooth.com/develop-with-bluetooth/join/
[3]https://www.businessresearchinsights.com/market-reports/bluetooth-low-energy-ble-ic-market-107175
[4] Mohammad Afaneh, "Capítulos 1 a 3" de Intro to Bluetooth Low Energy: Learn Bluetooth Low Energy in a Single Weekend, segunda ed. edición Kindle, editorial Novel Bits.

[5] https://www.mathworks.com/help/bluetooth/ug/bluetooth-protocol-stack.html
[6] https://www.bluetooth.com/Specifications/Profiles-Overview/
[7] Nathan J. Muller, "Capítulos 1 a 7" de Bluetooth Demystified, primera ed., editorial McGraw-Hill Professional.
[8] Gordon Colbach, "Capítulos 8 a 13" de Wireless Networking: Introduction to Bluetooth and WiFi, segunda ed., edición Kindle, publicación independiente.
[9] Albert S. Huang y Larry Rudolph, "Capítulo 1" de Bluetooth Essentials for Programmers, primera ed., edición Kindle, Cambridge University Press.
[10] Mohammad Afaneh, "Capítulos 2 a 4" de Intro to Bluetooth Low Energy: Learn Bluetooth Low Energy in a Single Weekend, segunda ed. edición Kindle, editorial Novel Bits.

[11] Robin Heydon, "Capítulos 1,2,3,8,9,10,11 y 12" de Bluetooth Low Energy: The Developer's Handbook, primera ed., edición Kindle, editorial Pearson.

[12]https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host-controller-interface/host-controller-interface-functional-specification.html
[13]https://www.pcmag.com/how-to/what-are-bluetooth-codecs-a-guide-to-everything-from-aac-to-sbc
[14]https://www.bluetooth.com/blog/a-technical-overview-of-lc3/
[15] Mohammad Afaneh, "Capítulo 5" de Intro to Bluetooth Low Energy: Learn Bluetooth Low Energy in a Single Weekend, segunda ed. edición Kindle, editorial Novel Bits.

[16] https://www.bluetooth.com/blog/a-new-way-to-debug-iosbluetooth-applications/
[17] https://www.cnmc.es/prensa/panel-hogares-usos-internet-20231103
[18] https://datatracker.ietf.org/doc/html/rfc1761

[19]
https://source.android.com/docs/core/connect/bluetooth/verifying_debuggin
g?hl=es
[20] https://developer.android.com/studio/debug/dev-options?hl=es-419
[21] https://developer.android.com/studio/debug/bug-report?hl=es-419
[22] https://www.wireshark.org/docs/dfref/

[23]https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host-controller-interface/host-controller-interface-functional-specification.html#UUID-5ab4cace-d0bc-38ea-2675-598d57905d3d

[24]https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/logical-link-control-and-adaptation-protocol-specification.html

[25]https://www.bluetooth.com/specifications/specs/rfcomm-1-1-html/

[26]https://www.bluetooth.com/specifications/specs/avctp-1-4/

[27]https://www.bluetooth.com/specifications/specs/a-v-remote-control-profile-1-6-2/

[28]https://www.bluetooth.com/specifications/specs/a-v-distribution-transport-protocol-1-3/

[29]https://www.bluetooth.com/specifications/specs/advanced-audio-distribution-profile-1-4/

[30]https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/attribute-protocol--att-.html#UUID-44501232-7370-3ff0-be57-3d458b542a5b

[31]https://www.bluetooth.com/wp-content/uploads/Files/Specification/HTML/Core-54/out/en/host/security-manager-specification.html#UUID-357f5fc6-5b55-6d33-e459-f42c929137f8

6.Anexos

Anexo A: Comandos y eventos HCI comunes entre Bluetooth Classic y Bluetooth Low Energy

Aunque a primera impresión pueda parecer que son muchos, comparativamente no lo son. Con relación al número total de eventos y comandos que hay, representan una pequeña porción.

COMANDOS

Obligatorios para los dos:
-Read BD_ADDR command
-Read Local Supported Commands command
-Read Local Supported Features command
-Read Local Version Information command
-Reset command
-Set Event Mask command

Dependiendo de condiciones:
-Configure Data Path command
-Disconnect command
-Get MWS Transport Layer Configuration command
-Host Buffer Size command
-Host Number Of Completed Packets command
-Read AFH Channel Assessment Mode command
-Read Local Supported Codec Capabilities command
-Read Local Supported Codecs command
-Read Local Supported Controller Delay command
-Read Remote Version Information command
-Read Authenticated Payload Timeout command
-Read Connection Accept Timeout command
-Read Remote Version Information Complete event
-Read RSSI command
-Read Transmit Power Level command
-Set Controller To Host Flow Control command
-Set Event Mask Page 2 command
-Set MWS Transport Layer command
-Write AFH Channel Assessment Mode command

-Write Authenticated Payload Timeout command
-Write Connection Accept Timeout command

Opcionales:
-Set Ecosystem Base Interval command
-Set External Frame Configuration command
-Set MWS Channel Parameters command
-Set MWS Scan Frequency Table command
-Set MWS Signaling command

EVENTOS

Obligatorios para los dos:
-Command Complete event
-Command Status event

Según condiciones:
-Authenticated Payload Timeout Expired event
-Disconnection Complete event
-Encryption Change event
-Encryption Key Refresh Complete event
-Number Of Completed Packets event

Opcionales:
-Data Buffer Overflow event
-Hardware Error event

[12]

Anexo B: Relación entre campos y columnas de Wireshark

PROTOCOLO	**CAMPO**	**ÍNDICE**
General	Número del paquete	0
	Tiempo	1
	Origen	2
	Destino	3
	Protocolo	4
	Tamaño	5
	Info	6
HCI_CMD	OGF	7
	OCF	8
	Bytes	9
HCI_EVT	Código de evento	10
	Sub Evento	11
	Código de operación de comando	12
	Bytes	13
L2CAP	CID	14
	Código de comando	15
RFCOMM	DLCI	16
	Tipo de frame	17
	Tipo de comando MCC	18

AVCTP	Transaction Label	19
	C/R	20
AVRCP	CType	21
	Opcode	22
	PDU ID	23
	Operation ID	24
AVDTP	Transaction Label	25
	Message Type	26
	Signal	27
A2DP	ACP SEID	28
	INT SEID	29
SBC	Expected Data Speed	30
ATT	Opcode	31
	Handle	32
	Value	33
SMP	Opcode	34

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jan 14 21:20:17 CET 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)