Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Desarrollo de un sistema para la gestión del procesado de información y las comunicaciones en sensores biomédicos

Autor: Marcos López García
Tutores: Luis Javier Reina Tosina
David Naranjo Hernández

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2023

Trabajo Fin de Grado
Grado en Ingeniería de las Tecnologías de Telecomunicación

Desarrollo de un sistema para la gestión del procesado de información y las comunicaciones en sensores biomédicos

Autor:
Marcos López García

Tutores:
Luis Javier Reina Tosina
David Naranjo Hernández

Dpto. de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla
Sevilla, 2023

Trabajo Fin de Grado: Desarrollo de un sistema para la gestión del procesado de información y las comunicaciones en sensores biomédicos

Autor: Marcos López García

Tutores: Luis Javier Reina Tosina

David Naranjo Hernández

El tribunal nombrado para juzgar el Trabajo arriba indicado, compuesto por los siguientes miembros:

Presidente:

Vocales:

Secretario:

Acuerdan otorgarle la calificación de:

Sevilla, 2023

El Secretario del Tribunal

A mi familia

A mis maestros

Agradecimientos

En primer lugar, me gustaría agradecer a mis tutores, Luis Javier Reina Tosina y David Naranjo Hernández, por ofrecerme la posibilidad de realizar este trabajo e introducirme en el campo de la Ingeniería Biomédica. Gracias a Javier por ofrecerme su atención y su ayuda siempre que lo necesitara. Gracias a David por explicarme muchas dudas que me surgieron durante el trabajo. Gracias a ambos por ser unos magníficos docentes.

En segundo lugar, me gustaría agradecer a mi familia, por haberme dado la educación que, sin duda, me ha posibilitado acceder y completar la titulación, y por resultar siempre un apoyo fundamental en mi vida. Gracias también a mis amigos por estar conmigo en todas las situaciones.

Finalmente, me gustaría agradecer a mis compañeros de la titulación y a los profesores por su ayuda durante el grado.

Marcos López García

Sevilla, 2023

Resumen

En la era actual de la medicina impulsada por la tecnología, los sensores biomédicos se han convertido en una herramienta crucial para el seguimiento de la salud del paciente, el diagnóstico y la detección temprana de enfermedades. Asimismo, los avances en los sistemas de gestión de datos, la transmisión de la información y la optimización de los sistemas de salud, junto a la interconexión de dispositivos a través de IoT, permite al personal médico el análisis en tiempo real de la información.

Sin embargo, a día de hoy, la interoperabilidad supone un desafío para la interconexión entre dispositivos médicos. A pesar de la existencia de estándares para la realización de una implementación normalizada, ninguno de ellos ha resultado ser ampiamente usado. Algunos de los motivos de la falta de éxito son la falta de infraestructura, la falta de colaboración entre fabricantes y propulsores de estándares, la complejidad de los estándares o la inexistencia de una tecnología de transmisión de información adecuada.

Teniendo en cuenta esta problemática, este trabajo propone una solución para las comunicaciones en sensores biomédicos. Para ello, se hace uso de la familia de estándares IEEE 11073, una serie de estándares enfocados específicamente a la comunicación en sensores biomédicos, ofreciendo una amplia gama de posibilidades. En cuanto a la tecnología de comunicación, el presente trabajo hace uso de Bluetooth Low Energy, tecnología inalámbrica de área personal, de bajo consumo y muy utilizada en entornos IoT. La principal dificultad que entraña la integración de IEEE 11073 y Bluetooth Low Energy es que son incompatibles. Por lo tanto, se ha de buscar la adaptación de sus protocolos para integrar estos estándares.

Este trabajo tiene como objetivo la creación de un sistema de comunicación entre una placa de evaluación Bluetooth Low Energy y un teléfono móvil siguiendo los estándares IEEE 11073. No se usa un sensor biomédico, sino que se usan distintos datos simulados de una medida de bioimpedancia para simular el funcionamiento de un sensor real. El trabajo presenta dos variantes para el envío de información, buscando ofrecer posibilidades y establecer un marco para el envío de datos genéricos. También se buscar detectar algunas mejoras en IEEE 11073, *Bluetooth Low Energy* y en la integración de ambas.

Abstract

In the current era of technology-driven medicine, biomedical sensors have become a crucial tool for patient health monitoring, diagnosis, and early disease detection. Furthermore, advancements in data management systems, data transmission, and healthcare system optimization, along with device interconnectivity through IoT, enable real-time analysis of information by medical personnel.

However, interoperability remains a challenge for interconnecting medical devices. Despite the existence of standards for standardized implementation, none of them have been widely adopted. Some reasons for the lack of success include infrastructure limitations, lack of collaboration among manufacturers and standardization proponents, the complexity of standards, or the absence of suitable information transmission technology.

Taking this issue into account, this work proposes a solution for communication in biomedical sensors. To do so, it leverages the IEEE 11073 family of standards, a series of standards specifically focused on communication in biomedical sensors, offering a wide range of possibilities. As for the communication technology, this work utilizes Bluetooth Low Energy, a wireless personal area technology known for its low power consumption and widespread use in IoT environments. The main challenge lies in the integration of IEEE 11073 and Bluetooth Low Energy as they are incompatible. Therefore, the adaptation of their protocols is required to integrate these standards.

The objective of this work is to create a communication system between a Bluetooth Low Energy evaluation board and a mobile phone following the IEEE 11073 standards. Instead of using a biomedical sensor, simulated data from bioimpedance measurements is used to simulate the operation of a real sensor. The work presents two variants for information transmission, aiming to provide possibilities and establish a framework for sending generic data. Additionally, improvements in IEEE 11073, Bluetooth Low Energy, and their integration are sought to be detected.

Índice

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

1 INTRODUCCIÓN

El futuro no es un regalo, es una conquista.

- Robert Kennedy -

En los últimos años, los avances tecnológicos han sido colosales e incesantes, transformando por completo todos los ámbitos de la vida, y la medicina, es uno de esos campos que ha evolucionado vertiginosamente a través del flujo continuo de nuevas tecnologías dentro de la ingeniería biomédica. Gracias a estas nuevas herramientas, la medicina ha mejorado el diagnóstico, la detección temprana de enfermedades o el tratamiento, entre otros. Sin embargo, no sólo es relevante la mejora respecto a la práctica médica, también ocupan un lugar importante los sistemas de gestión de datos, siendo clave para el seguimiento de las enfermedades y la optimización de los sistemas de salud.

Para la mayoría de aplicaciones de la ingeniería biomédica, se han erigido los sensores biomédicos como parte fundamental. Estos sensores son capaces de medir las magnitudes fisiológicas de interés, procesar la información y transmitirla. Debido a los avances en la transmisión de datos y la interconexión de dispositivos ofrecida por IoT (Internet de las Cosas), los sensores permiten al personal médico el análisis de los datos médicos en tiempo real.

A pesar de la evolución en la interconexión de dispositivos médicos, existe un gran escollo para su implementación total como es la interoperabilidad. A día de hoy, la falta de interoperabilidad en los sistemas de información de salud desperdicia las posibilidades ofrecidas [1][2]. Para el adecuado intercambio de información en el ámbito de la salud, es indispensable el uso de estándares por parte de todos los dispositivos involucrados. Por este motivo, surgieron varios estándares como HL7 o IEEE 11073 [3] (también referido como X73), pero ninguno de ellos ha resultado ser hegemónico y ni siquiera ampliamente usado. Algunos de los motivos que no han propiciado el éxito de estos proyectos pueden ser la falta de infraestructura, la falta de colaboración entre fabricantes y propulsores de los estándares, la complejidad de los estándares o la falta de un canal eficaz, veloz y de bajo consumo para transmitir los datos. En este momento, reina la solución propietaria, sin embargo, el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) continúa avanzando en el desarrollo de la norma 11073 con el fin de ser capaces de interconectar los sensores biomédicos [4].

La norma IEEE 11073 ofrece un marco de interoperabilidad para servicios médicos enfocados en la monitorización remota en tiempo real, por lo que dicha norma nace ligada a los sensores biomédicos. Los datos X73 necesarios para que el sensor establezca una comunicación con otros dispositivos están definidos en [14], proporcionando también los datos del tipo de medición que se va a realizar con el objetivo de conocer el estado del paciente. IEEE 11073 establece la estructura de datos y la comunicación entre dispositivos, pero no es el único sistema que ha de formar parte de la comunicación.

Además de un estándar para el entendimiento de los datos, es necesaria una tecnología para la comunicación,

siendo especialmente interesantes las tecnologías inalámbricas. Una de las tecnologías inalámbricas destacadas es *Bluetooth Low Energy*, también conocido como Bluetooth LE o BLE, fue creada por el grupo Bluetooth SIG y su principal diferencia con el Bluetooth clásico es su bajo consumo, haciéndolo ideal para entornos IoT [5][6] donde se busca que los dispositivos tengan el menor gasto de energía posible. Bluetooth LE es una Red de Área Personal (PAN), lo que lo hace adecuado para la transmisión de datos de un sensor biomédico. Además, se encuentra implementado en todos los móviles y *tablets*, por lo que es la tecnología adecuada por la que transmitir la información.

Sin embargo, IEEE 11073 y *Bluetooh Low Energy* están lejos de ser compatibles. BLE incluye varios perfiles relacionados con la monitorización de la salud, pero ni contienen la cantidad de datos suficientes para ser compatible con X73 ni tampoco son muchos. Por lo tanto, no hay una forma establecida de tratar el problema de la implementación, ha de ser una solución particular que generalmente tendrá que ver con la encapsulación de los datos X73 en BLE. Pese a haber trabajos de la integración de IEEE 11073 sobre BLE [7][8][9], no proporcionan la información suficiente como para poder servir de referencia.

Este trabajo tiene como objetivo la implementación de una comunicación entre sensor biomédico y móvil a través de Bluetooth LE. No se usará un sensor biomédico real, sino se usarán distintos datos simulados con el objetivo de ofrecer datos reales y diversos. Los datos simulados son provenientes de una medida de bioimpedancia [10], técnica utilizada para medir la resistencia eléctrica del cuerpo humano. Se basa en el principio de que diferentes tejidos corporales conducen la electricidad de manera diferente debido a sus características estructurales y composición [11][12]. Esta técnica da lugar a resultados de composición corporal de gran interés para el conocimiento del estado de la salud y el rendimiento físico.

1.1 Objetivos

Este trabajo tiene como objetivo la implementación real de una medida de bioimpedancia para la comunicación entre sensor y móvil a través de Bluetooth LE como se propuso en [13]. Se usa una placa de evaluación BLE que envia los datos X73 al móvil mediante el perfil GATT de BLE. El trabajo también presenta dos variantes acerca del envío de datos X73 a través de BLE buscando ofrecer posibilidades y establecer un marco para el envío de datos genéricos. Otro de los objetivos es identificar, señalar y tratar de solucionar algunas complicaciones derivadas de la integración de los estándares IEEE 11073 y Bluetooth Low Energy.

Por lo tanto, el trabajo se divide en tres objetivos secundarios:

- Análisis de los parámetros generales de una medida de bioimpedancia y su adaptación a la familia de estándares IEEE 11073.
- Adaptación y encapsulación de los datos IEEE 11073 en el sensor Bluetooth LE para su envío a través del perfil GATT.
- Diseño de una aplicación móvil para la recepción y desencapsulación de los datos IEEE 11073.

1.2 Estructura

El trabajo se divide en los siguientes capítulos:

Capítulo 1. Introducción. El primer capítulo señala la importancia de los sensores biomédicos en la medicina y la falta de interoperabilidad entre ellos. Como solución a la problemática, se presenta el estándar IEEE 11073 para paliar el problema de la interoperabilidad y Bluetooth Low Energy como tecnología de transmisión de la información.

Capítulo 2. Situación de los estándares IEEE 11073 y Bluetooth Low Energy. Este capítulo es una revisión del conjunto de estándares IEEE 11073 y de la tecnología Bluetooth Low Energy, centrándose en las partes más relevantes para el trabajo.

Capítulo 3. IEEE 11073 sobre BLE. El tercer capítulo trata la incompatibilidad entre X73 y la tecnología BLE, así como la necesidad de la transcodificación de los datos. Para solventar estos problemas, se hace una revisión de información actual sobre trabajos publicados con la integración de ambas y se proponen soluciones.

Capítulo 4. Diseño e implementación de una medida de bioimpedancia según IEEE 11073 sobre BLE. Este es el capítulo más extenso del trabajo, pues aglutina la descripción de las tecnologías y los entornos de desarrollo usados, el diseño del sistema para la comunicación, la presentación de la medida de bioimpedancia y la implementación de todos ellos.

Capítulo 5. Resultados. Capítulo para la presentación y discusión de las implementaciones del capítulo anterior.

Capítulo 6. Conclusiones. Capítulo final para el análisis de los estándares utilizados y la integración de todos ellos, puntualizando defectos y posibles mejoras.

2 SITUACIÓN DE LOS ESTÁNDARES IEEE 11073 Y BLUETOOTH LOW ENERGY

La sabiduría comienza con asombro.

- Sócrates -

Para la transmisión de datos de los sensores biomédicos es primordial el uso de un estándar para fomentar la interoperabilidad entre dispositivos y una tecnología de comunicación, generalmente inalámbrica. En el caso del presente trabajo, se usará la norma X73 y la tecnología inalámbrica *Bluetooth Low Energy*. A lo largo de este capítulo, se revisará la situación actual de los estándares IEEE 11073 y de la tecnología inalámbrica BLE.

2.1 IEEE 11073

IEEE 11073 es una familia de estándares desarrollados por la Organización Internacional de Normalización (ISO) y el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) para la interoperabilidad entre dispositivos médicos. Están orientados a los dispositivos de salud personal (personal health device, PHD), aunque también pueden servir para otros dispositivos médicos.

Al ser una familia de estándares, IEEE 11073 lo conforman una amalgama de estándares, de los que destacan los siguientes:

IEEE Std 11073-10101-Nomenclature. Este estándar provee la nomenclatura para la comunicación de los demás estándares. [14]

IEEE Std 11073-10201-Domain information model. Define el modelo de información orientado a objetos que especifica la estructura para el intercambio de mensajes. [15]

IEEE Std 11073-20601 - Application profile - Optimized exchange protocol. Este estándar define las especificaciones para el protocolo de comunicación y el intercambio de datos. [16]

Estos tres estándares conforman la base para todo tipo de comunicación de dispositivos de salud personal, sin embargo, no son tan completos como para abarcar la complejidad de cualquier tipo de medición. Para ello, surge el conjunto de estándares IEEE 11073-104zz, siendo zz cualquier número de 01 a 99 y representando lo que se conoce como device specialization, que son varios dispositivos de salud personal típicos, con el fin de establecer de una forma más concreta la forma de los datos x73 para cada tipo de medición. Estos son algunos de los dispositivos más importantes:

- IEEE Std 11073-10404 - Device specialization - Pulse Oximeter. Esta especialización está enfocada en la comunicación para pulsioxímetros. [17]
- IEEE Std 11073-10406 - Device specialization - Basic electrocardiograph (ECG). Esta especialización está enfocada en la comunicación para monitores de electrocardiograma. [18]
- IEEE Std 11073-10407 - Device specialization - Blood Pressure Monitor. Esta especialización está enfocada en la comunicación para dispositivos de presión arterial. [19]
- IEEE Std 11073-10420 - Device specialization - Body composition analyzer. Esta especialización está enfocada en la comunicación para analizadores de composición corporal. [20]
- IEEE Std 11073-10471 - Device specialization - Independent living activity hub. Esta especialización está enfocada en la comunicación en "centros de actividades de vida independiente", lo que significa dispositivos de información de la salud de actividades diarias o deportivas. [21]

Sabiendo los estándares más importantes, se puede definir la arquitectura de los sensores médicos como se puede ver en la Figura 2-1 apareciendo todos los elementos de la capa "host" definida en el modelo osi. El transporte que se elija ocupará la cuarta capa del modelo OSI, IEEE Std 11073-20601 formará la quinta capa, sesión, y la sexta capa, presentación, y la especialización correspondiente ocupará las capas 6, sesión, y 7, aplicación.

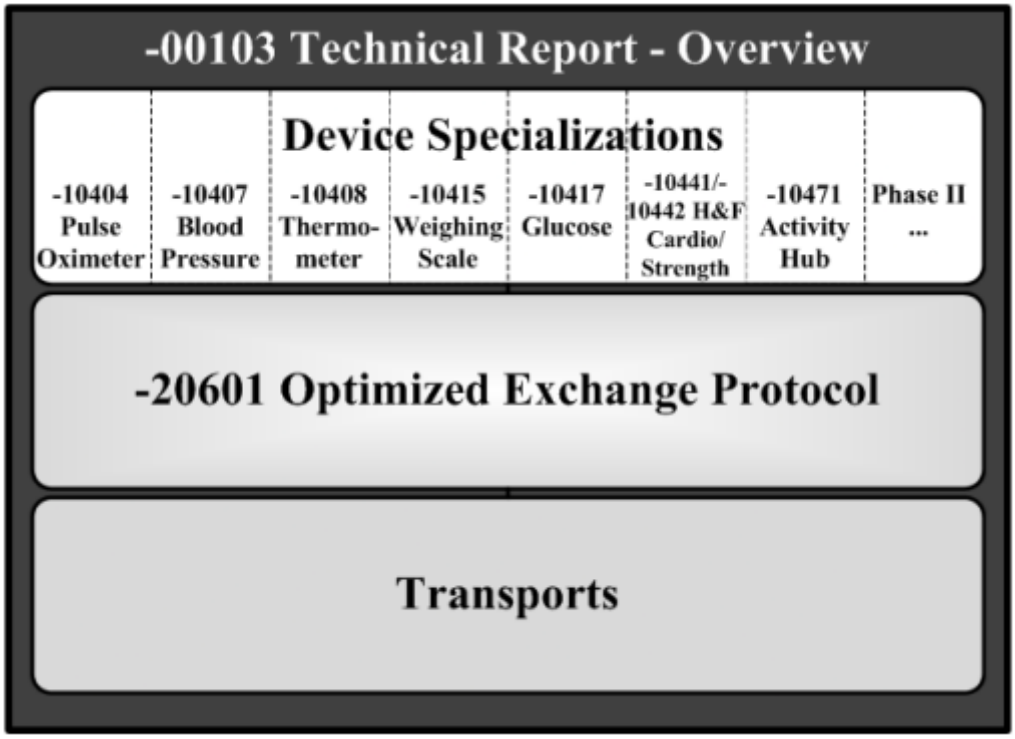

Figura 2-1. Arquitectura de un dispositivo médico siguiendo el estándar IEEE 11073.

Otra de las claves de este conjunto de estándares es la relación agente-gestor o agent-manager, esta es la forma de referirse a los componentes involucrados en la comunicación. El agente es el dispositivo de salud personal, encargado de realizar las mediciones correspondientes y transmitirlas, y el gestor es, generalmente, un ordenador o un smartphone, encargado de recibir la información y procesarla. Ambos dispositivos, como es lógico, han de seguir la arquitectura de la Figura 2-1.

Originalmente, se definieron tres posibles transportes para X73; Bluetooth, USB y Zigbee. Aunque, a día de hoy, se están utilizando otros transportes. Esta temática se desarrollará posteriormente y supone gran parte de la complejidad del estándar y de las dificultades para su implantación.

En términos de notación, se usa la representación de datos Abstract Syntax Notation One (ASN.1). Esta representación convierte la sintaxis médica que se usa, Medical Encoding Rules (MDER), en notación binaria, con el objetivo de facilitar la transmisión de datos. También es especialmente importante para fomentar la interoperabilidad entre dispositivos, porque si la codificación y la decodificación no siguen las mismas normas, no se podría establecer una comunicación adecuada.

Esta serie de estándares destaca por ser orientada a objetos, lo que facilita la estructura de los dispositivos, los servicios y los datos. Los objetos que conforman cada dispositivo se revelan en la configuración del mismo, pudiendo ser configuración estándar en caso de ser un dispositivo que siga las pautas de uno definido previamente, o puede ser configuración extendida en caso de buscar una configuración personalizada. Es importante no confundir el término configuración con device specialization, la diferencia radica en que un dispositivo de especialidad puede tener una configuración estándar si sigue la configuración de uno ya definido en su device specialization o puede tener configuración extendida si a esa definición se le añaden o se le cambian algunos parámetros. En cuanto a la definición de los objetos, se ha de seguir Domain Information Model (DIM). Existe un objeto principal relativo al agente que es el objeto Medical Device System (MDS) y todos los demás objetos derivan de él como se observa en la Figura 2-2. Destaca la clase métrica, clase de la que derivan las diferentes clases de medición como numérica, señal (RT-SA) o enumeración. Los objetos y las clases que se usarán en el trabajo se presentarán en el Capítulo 4.

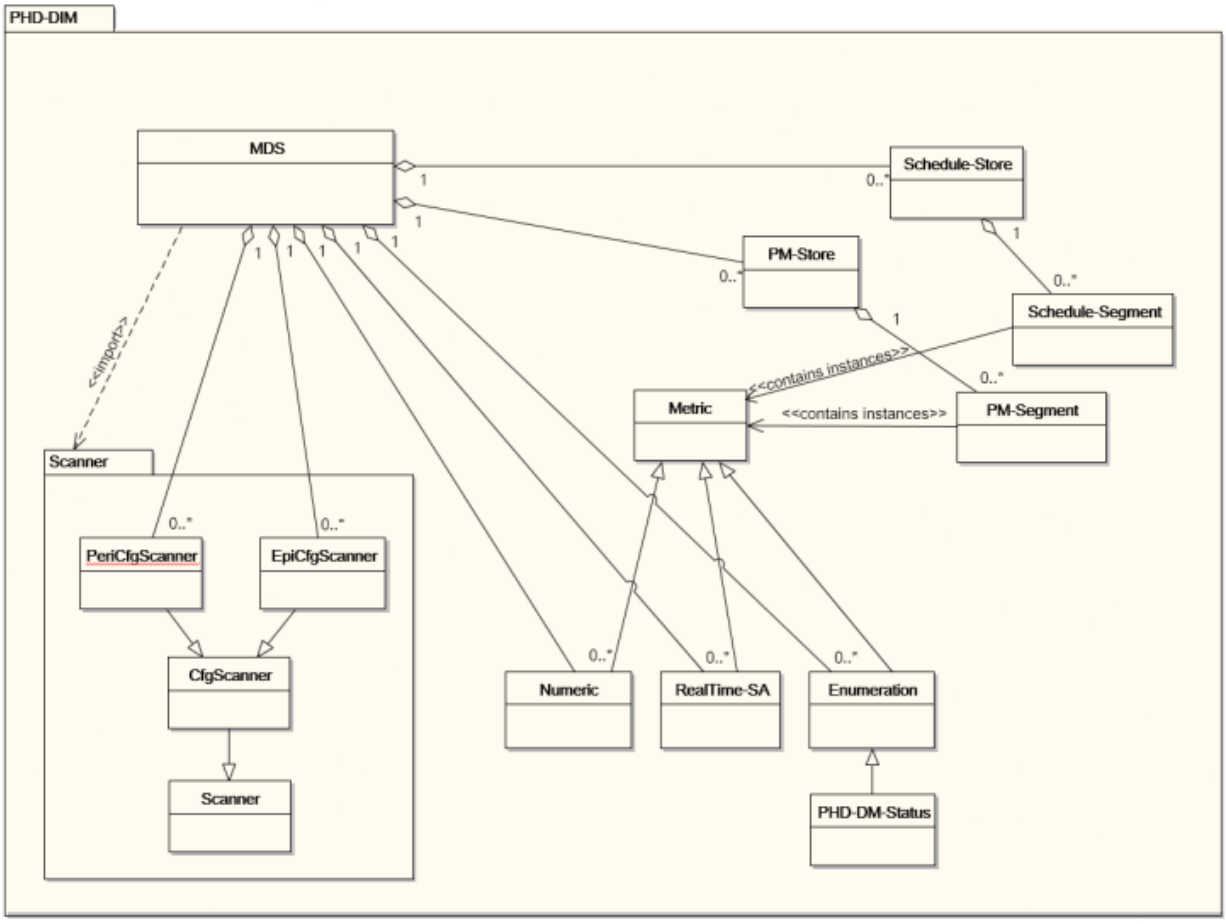

Figura 2-2. Objetos y clases de un dispositivo de salud personal (PHD).

Otra parte crucial del estándar es la definición de la comunicación, representada principalmente por los mensajes entre el agente y el gestor y la máquina de estados que indica el estado de la comunicación. Los mensajes intercambian información entre el agente y el gestor en un mismo estado y la máquina de estados cambia de estados con ciertos mensajes. Por ejemplo, en la Figura 2-3 se aprecia el intercambio de mensajes entre agente

y gestor en la que reporta su configuración y en la Figura 2-4, que es la máquina de estados, se observa que, dependiendo de los mensajes de configuración, se pasa a un estado o a otro. En el Capítulo cuarto se presentarán los estados y los mensajes de agente y gestor, siendo parte principal del presente trabajo.

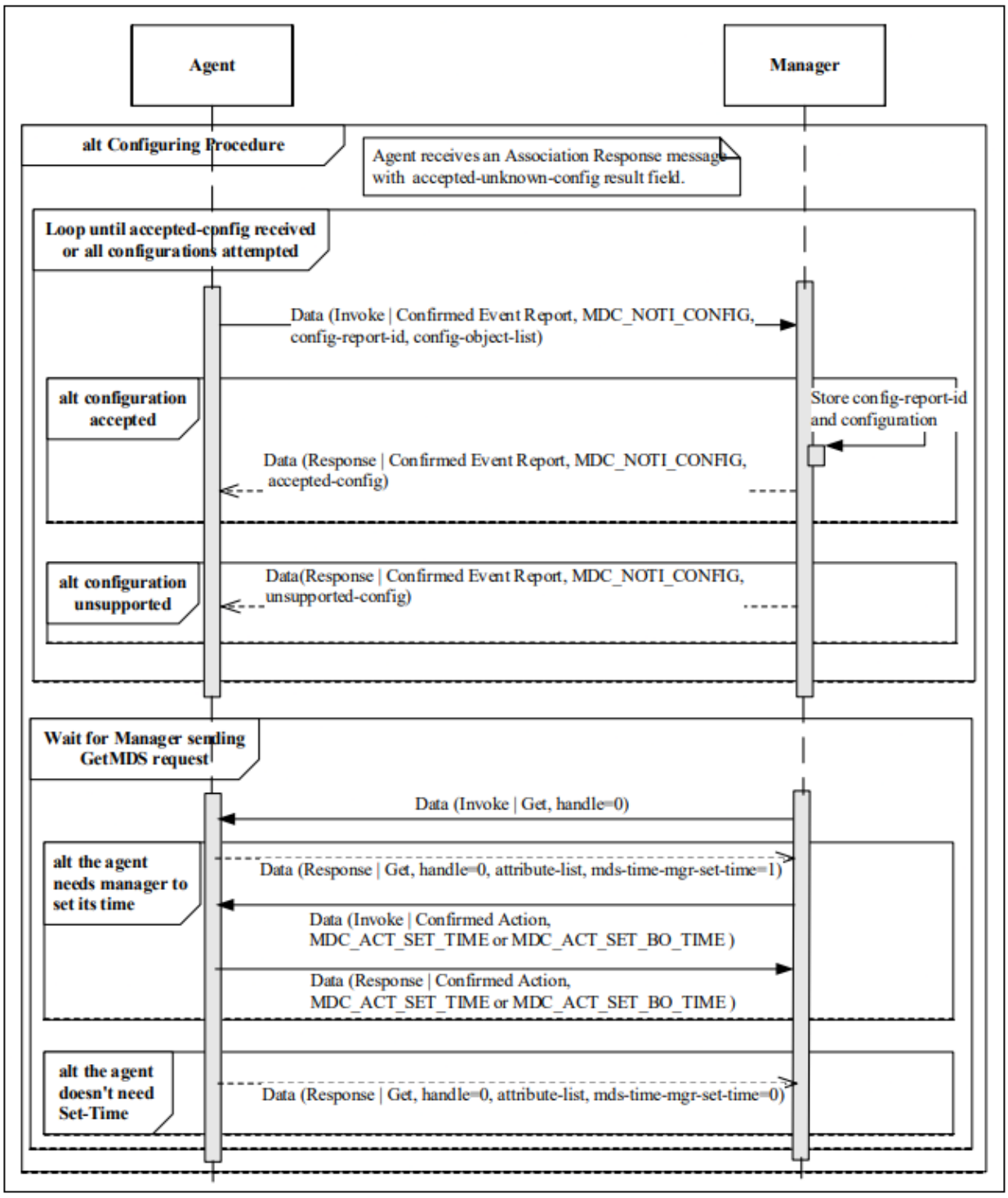

Figura 2-3. Diagrama de paso de mensajes para la configuración.

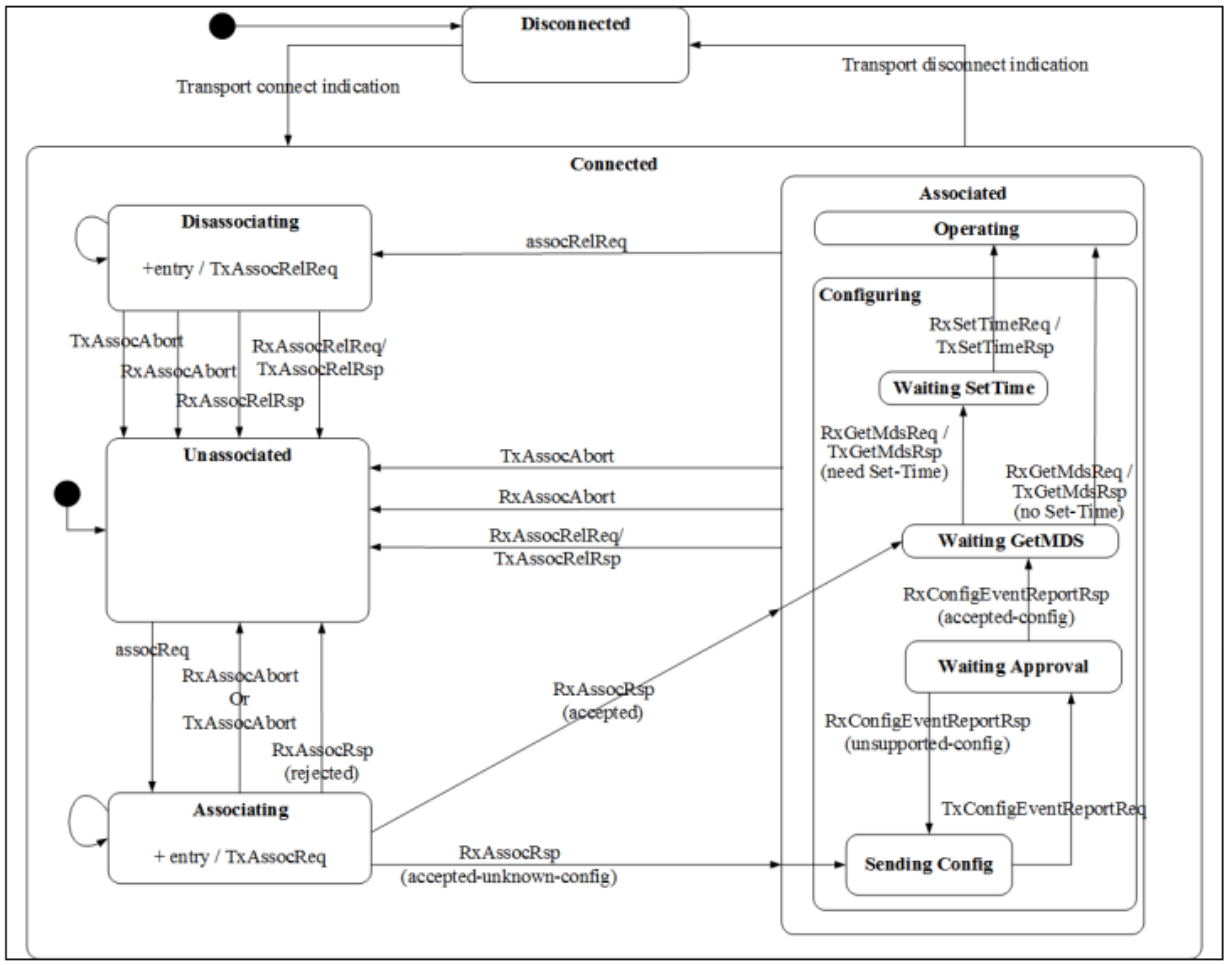

Figura 2-4. Máquina de estados del agente.

2.2 Bluetooth Low Energy

Bluetooth Low Energy es una tecnología de comunicación inalámbrica diseñada para aplicaciones de bajo consumo. Fue creada por Bluetooth SIG [22], organización privada sin ánimo de lucro, como extensión del Bluetooth clásico, pero enfocada a dispositivos portátiles que requerían un menor consumo de energía, pero las mismas cualidades [23] [24]. Por lo tanto, BLE, respecto al Bluetooth original, mantiene la distancia de una Red de Área Personal (PAN), acompañado de una mayor eficiencia energética que no restringe su uso a dispositivos más sencillos.

En el contexto de IoT y las comunicaciones masivas en la Sociedad de la Información, BLE es incorporado en multitud de dispositivos a fin de ofrecer conectividad con toda la red. No solo está implementado en *smartphones* y portátiles, sino también en auriculares inalámbricos, altavoces inteligentes o *wearables*, amplificando su utilidad. Con el crecimiento exponencial del Internet de las Cosas y el avance irrefrenable de la tecnología 5G, se espera que Bluetooth Low Energy sea un elemento principal para aumentar la conectividad entre dispositivos de forma gigantesca.

Para comprender el funcionamiento de la tecnología inalámbrica Bluetooth LE es necesario comprender su arquitectura en la Figura 2-5 que se divide en *Controller*, *Host* y *Application* [25]. La capa *Controller* es la parte referida al enlace físico, que opera en la banda sin licencia industrial, científica y médica (ISM) de 2.4 GHz, como Wi-Fi. El ancho de banda de cada canal es de 2 MHz, habiendo 40 canales y siendo los 37 primeros para los paquetes de transmisión de datos y los tres últimos para paquetes de anuncio (*advertising*). La modulación usada es modulación por desplazamiento de frecuencia gaussiana (GFSK) y utiliza técnicas de espectro ensanchado por salto de frecuencia (FHSS) para reducir el efecto de las interferencias y del desvanecimiento

selectivo en frecuencia. La capa física tiene una tasa binaria de 1 Mbps, como mínimo, pero es reducida en las capas superiores debido a la necesidad de que parte de los datos transmitidos por paquetes sean cabeceras y otros indentificadores del paquete.

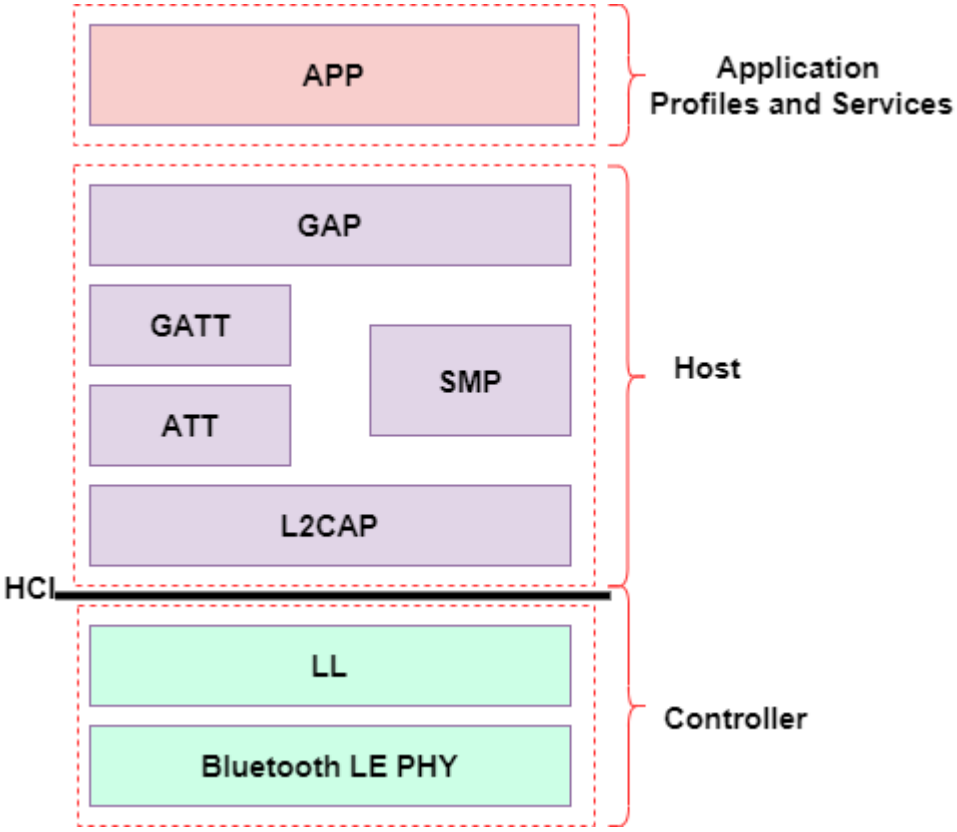

Figura 2-5. Pila de protocolos en BLE. [25]

En la parte del *Host*, L2CAP encapsula (o desencapsula) los datos de las capas superiores (o inferiores) al formato estándar de Bluetooth LE de paquetes. ATT es uno de los protocolos que define la estructura de datos para los perfiles basados en GATT. Como se muestra en la Figura 2-6, la forma del atributo BLE según ATT es *Handle* que ocupa 16 bits y es un identificador del servidor para que el cliente pueda referenciar el atributo. El tipo del atributo, que se basa en un identificador único universal (UUID), de 16 bits en caso de ser de creación de Bluetooth SIG o de 128 bits en caso de ser de iniciativa privada. El valor del atributo es de longitud variable y los permisos del atributo permiten al cliente leer, escribir, recibir notificaciones o indicaciones del atributo según las posibilidades que haya ofrecido el cliente. La comprensión del atributo en BLE es absolutamente fundamental para el uso adecuado del perfil GATT.

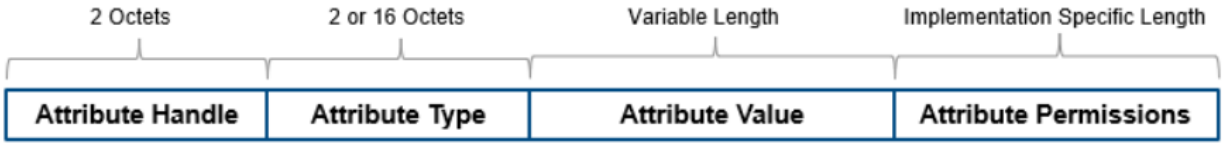

Figura 2-6. Forma del atributo BLE. [25].

En la capa *Host*, también se encuentra el perfil de atributo genérico (GATT), encargado de encapsular el atributo dentro de sus características. Este perfil ha de ser implementado por cliente y servidor y es la principal vía de comunicación entre ambos. El cliente solicita información y el servidor envía respuestas, notificaciones e

indicaciones. La terminología en GATT es la siguiente:

- **Perfil.** Implementación concreta para el que se usa el perfil genérico GATT.
- **Servicio.** Cnjunto de datos agrupado entorno a una misma función.
- **Característica.** Entidad contenedora de los atributos y de los permisos concedidos al cliente.
- **Descriptor.** Descripción particular, en caso de ser necesario, de la característica.

También en la capa *Host* se encuentran el protocolo SMP, encargado de la encriptación y desencriptación de los paquetes; y el protocolo GAP, que especifica roles, modos y y procedimientos de un dispositivo.

Por último, se encuentra la capa *Application Profiles and Services***,** es la parte de la interfaz de aplicación específica que se encontrará y hace uso de todas las capas anteriores. Está mayormente compuesta por elementos creados por el desarrollador y es éste el que ha de buscar la interoperabilidad.

El dispositivo BLE tendrá dos formas de funcionamiento, modo anuncio (*advertising*) y modo conexión. El modo anuncio consiste en emitir periódicamente paquetes de datos con información del dispositivo, con el objetivo de que otros dispositivos, conocidos como observadores, detecten los paquetes y sean capaces de establecer una conexión. Tras esto, se entra en el modo conexión, que consiste en el intercambio de datos de forma bidireccional.

3 IEEE 11073 SOBRE BLE

Solo los muertos han visto el final de la guerra.

- Platón -

Como se comentó en el capítulo anterior, los transportes propuestos para la serie de estándares IEEE 11073 fueron USB, Bluetooth (en su versión clásica) y Zigbee, aun así, se pueden usar muchos otros transportes. En el presente trabajo, se opta por utilizar Bluetooth Low Energy debido a su uso masivo en entornos IoT y en dispositivos de bajo consumo. Entonces, el presente capítulo presenta la situación actual en cuanto a la integración de IEEE 11073 sobre BLE.

Con la aparición de Bluetooth Low Energy se consiguió reducir de forma drástica el consumo del dispositivo, sin embargo, también vino acompañado de una reducción de los servicios ofrecidos por Bluetooth clásico. En el caso que entraña a este trabajo, en Bluetooth se encuentra un perfil llamado Health Device Profile [26], que facilita enormemente la inclusión de X73 en ese entorno para el envío y recepción de datos médicos. Además, hay algunas librerías y algunos proyectos con información suficiente en cuanto a la integración, por lo tanto, se puede decir que Bluetooth es compatible con la serie de estándares IEEE 11073. Sin embargo, en BLE la transmisión de datos es vía GATT, definiendo en esta la manera en la que los servicios pueden comunicarse [27]. En Bluetooth Low Energy existen algunos servicios como Blood Pressure Service, Body Composition Service o Heart Rate Service, que proponen normas para el intercambio de información de sensores de presión sanguínea, composición corporal o frecuencia cardíaca, respectivamente, basados en GATT. Sin embargo, estos servicios están lejos de satisfacer los requisitos mínimos de la familia de estándares IEEE 11073, por lo que, se concluye que X73 y BLE no son compatibles.

Habiendo concluido que IEEE 11073 y Bluetooth LE no son compatibles, es evidente la necesidad de transcodificar los datos. Y, aunque es obvia la necesidad, a día de hoy se carece de una solución o un estándar para la transcodificación de los datos X73 a BLE, además de haber una enorma falta de información acerca de cómo debe ser la transmisión de datos. Los pocos trabajos que abordan el tema no detallan cómo llegan a su solución, entonces la solución a este problema ha de resolverla el fabricante, de forma que llegue a ofrecer interoperabilidad entre dispositivos. En este capítulo se recopilan soluciones de algunos trabajos y se proponen otras nuevas soluciones con el fin de ofrecer información detallada acerca de la integración de X73 en BLE.

3.1 Soluciones y propuestas

El primer problema que surge es el de la transcodificación debido a que los sensores biomédicos no recopilan la información siguiendo las necesidades de IEEE 11073. Los estándares X73 requieren una gran cantidad de información acerca del estado de las mediciones, no solo valor, unidades y fecha y hora como suelen dar la mayoría de sensores. Por este motivo es necesario transcodificar los datos antes de enviarlos. Para ello, es necesario la utilización de un adaptador o transcodificador X73 [28] [29].

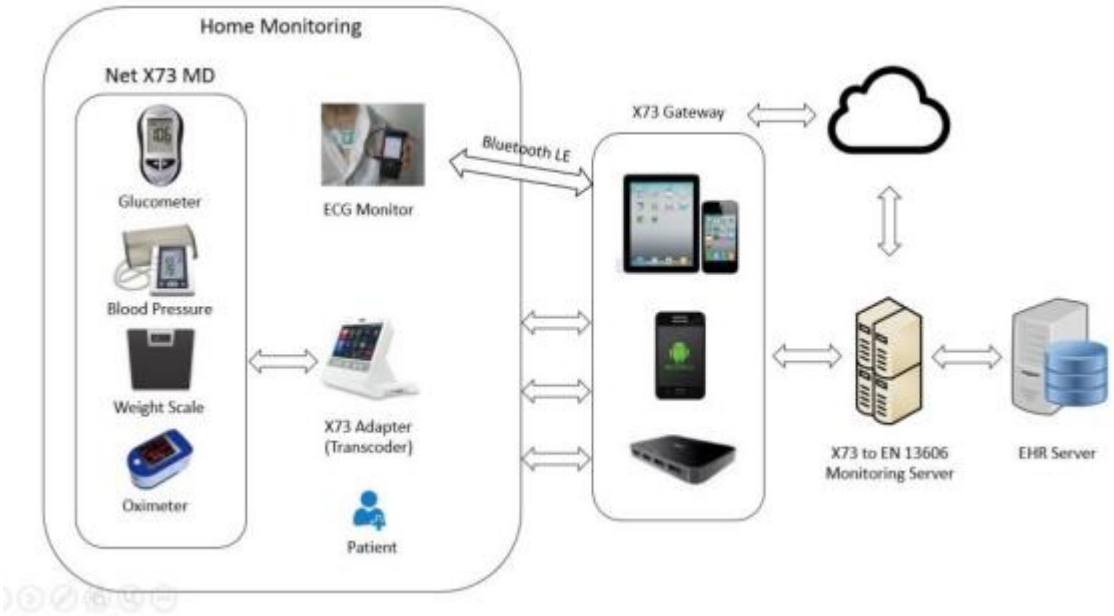

Figura 3-1. Uso del adaptador x73. [30]

La presencia del adaptador X73 en la arquitectura del sistema, como se muestra en la Figura 3-1, es fundamental y debe formar parte del propio agente X73, sin embargo, es un tema poco desarrollado en los trabajos existentes. Tras adaptar los datos recogidos por el sensor a los estándares IEEE 11073, es necesario estudiar la forma en la que se van a enviar. Cabe resaltar que el tamaño de los datos en formato X73 es muy grande y no se puede abarcar con envíos pequeños a través de GATT. Por eso, en [7], la forma para enviar la información es mediante 6LoWPAN, ampliamente utilizado en entornos IoT, como se muestra en la Figura 3-2. El estándar 6LoWPAN sirve para facilitar el uso de el transporte IPv6 en redes que implementen Bluetooth Low Energy. Sin embargo, para este caso concreto, la arquitectura es demasiado avanzada para implementarla en la mayoría de entornos IEEE 11073, requiriendo incluso la recepción de información en un servidor para enviarlo mediante HTTP al móvil posteriormente. La complejidad de esta arquitectura limita las posibilidades de X73 con wearables o dispositivos de salud personal, que son algunos de los principales intereses del estándar.

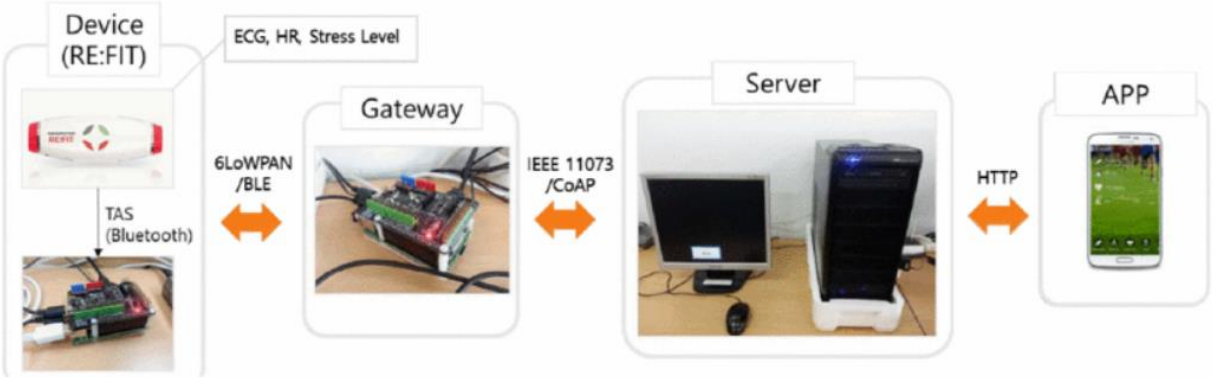

Figura 3-2. Diseño de la red para 6LoWPAN. [7]

Habiendo descartado el uso de 6LoWPAN, la alternativa es el envío de información a través de GATT. Aquí la encapsulación es parte crucial y, por lo tanto, es necesario decidir de qué forma va a ser la encapsulación. En [8], se hace la encapsulación para transcodificar los datos al protocolo *machine to machine* MQTT mediante ficheros JSON. En este artículo, se realiza la transcodificación usando JSON y adaptando la librería Antidote de

Signove [31] que es una implementación de código abierto del estándar IEEE 11073-20601. El problema para la integración junto a MQTT es la falta de información existente, pues en el artículo no se detalla la forma de transcodificar todos los datos, sino solo una pequeña muestra no representativa de los demás, además de existir la duda sobre la fiabilidad de la librería Antidote, cuyas últimas actualizaciones son de hace diez años. Asimismo, esta librería estaba diseñada para Bluetooth clásico, no para BLE, por lo que es necesario adaptar los cambios para GATT, que en este caso lo hace con el uso de la propuesto de envío de datos de Bluetooth SIG [32].

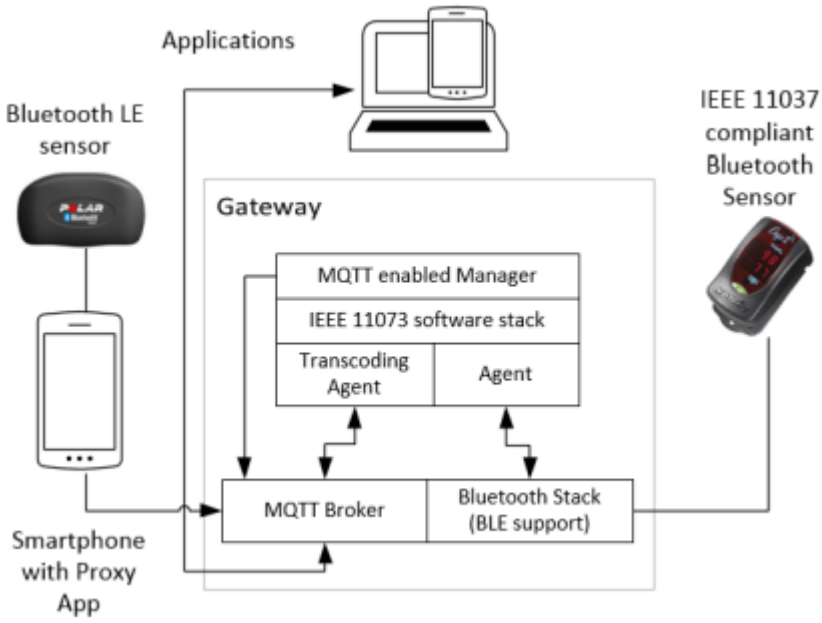

Figura 3-3. Arquitectura para MQTT. [8]

Otra opción es el uso de los actuales servicios de BLE GATT, como Pulse Oximeter Service o Blood Pressure Service, adaptándolos a X73. Para ello, existe un artículo de Bluetooth SIG acerca de su integración [33]. Sin embargo, son tantos los parámetros X73 que no tienen su equivalente en BLE, que no se explica la forma de encapsulación para ellos, siendo entonces, bastante deficiente en este sentido. Podría ser una forma interesante para la implementación de los parámetros compatibles en X73 y BLE, pero es necesario buscar la forma de encapsular el resto de los datos.

La falta de información acerca de una implementación completa de la serie de estándares IEEE 11073 supone tener que encontrar una solución particular recurriendo a algunas de las ideas planteadas en trabajos anteriores. Para el presente trabajo, se van a introducir dos propuestas que hacen uso de BLE GATT comparando su resultado y su rendimiento. Ambas propuestas usan el mismo material, un sensor Bluetooth Low Energy y una aplicación móvil, como se comentó previamente, el señor biomédico es sustituido por datos simulados de un sensor de bioimpedancia multifrecuencia. El sensor BLE, aparte de enviar los datos, funcionará como adaptador X73, con el objetivo de pasar los datos medidos al estándar IEEE 11073 y la recepción de los datos la llevará a cabo una aplicación móvil desarrollada en Android.

La diferencia entre las dos propuestas que se presentan en este trabajo radica en la forma de encapsular (y desencapsular) los datos. Para la primera de las formas se hace uso de ficheros XML y la segunda enviará los datos encapsulados por paquetes. El problema de la primera es la falta de estandarización en ficheros XML, para paliar eso, se adapta la librería de código abierto Antidote a este trabajo. A pesar de ello, va a seguir sin proporcionar interoperabilidad total entre dispositivos, pero sí podría servir como referencia para la creación de unos ficheros XML completamente estandarizados. El principal problema de la segunda implementación es que la cantidad de información a enviar es muy superior, pudiendo generar dificultades en velocidad, consumo y errores. Adicionalmente, para la segunda forma se consigue crear una forma para el envío de cualquier tipo de medición X73 simplemente cambiando los parámetros principales en una lista de configuración. Todas estas cuestiones serán desarrolladas en el siguiente capítulo.

4 DISEÑO E IMPLEMENTACIÓN DE UNA MEDIDA DE BIOIMPEDANCIA IEEE 11073 SOBRE BLE

La virtud está en el punto medio.

- Aristóteles -

El estudio del presente trabajo consiste en desarrollar un sistema para la gestión del procesado de información y las comunicaciones en sensores biomédicos, este capítulo es el epicentro de todo ello. Para este trabajo, ha sido utilizado como referencia un sensor de bioimpedancia, pese a no usarlo, con el objetivo de adaptar una medida de bioimpedancia al entorno IEEE 11073, para posteriormente enviarlo encapsulado por Bluetooth Low Energy y recibirlo y desencapsularlo desde la aplicación móvil.

Este capítulo comienza con la presentación y descripción de todas las tecnologías, dispositivos, lenguajes y entornos de desarrollo utilizados, haciendo especial hincapié en aquellas cuestiones más desconocidas y complicadas de usar.

El segundo apartado se dedica a explicar el diseño de la red y la forma en la que se van a enviar los datos. Se divide en tres objetivos el diseño del sistema: adaptación de la medida de bioimpedancia a X73, encapsulación y envío de los datos X73 a través del sensor Bluetooth Low Energy y recepción y desencapsulación de los datos desde la aplicación móvil. En consecuencia, los siguientes y últimos tres apartados del capítulo desgranan estos tres objetivos.

4.1 Software, tecnologías y lenguajes

Este apartado presenta software, tecnologías y lenguajes agrupándolos en los dos entornos de programación que han sido usados, Simplicity Studio y Android Studio, el primero para generar y enviar los datos, el segundo para recibirlos.

4.1.1 Simplicity Studio

Simplicity Studio [34] es el entorno de desarrollo utilizado para todas las tecnologías de Silicon Labs, en el caso que entraña a este trabajo, la placa BGM220P [35]. La versión 5 de Simplicity Studio, utilizada en este proyecto, se basa en Eclipse y en el entorno de desarrollo C/C++, por lo tanto, ese es el lenguaje utilizado en esta parte del trabajo.

La principal ventaja de Simplicity Studio es la simplicidad para el uso de Bluetooth. Tiene multitud de ejemplos de casos de uso para BLE GATT e incorpora diversas opciones que eliminan la realización de algunas tareas pesadas, como por ejemplo la herramienta Bluetooth GATT Configurator, herramienta que crea el perfil GATT

y los servicios que se van a implementar de forma manual y con un sencillo manual de instrucciones, evitando así tener que crearlo mediante ficheros XML que son menos intuitivos. En este trabajo, Bluetooth GATT Configurator es usado para la creación del perfil GATT.

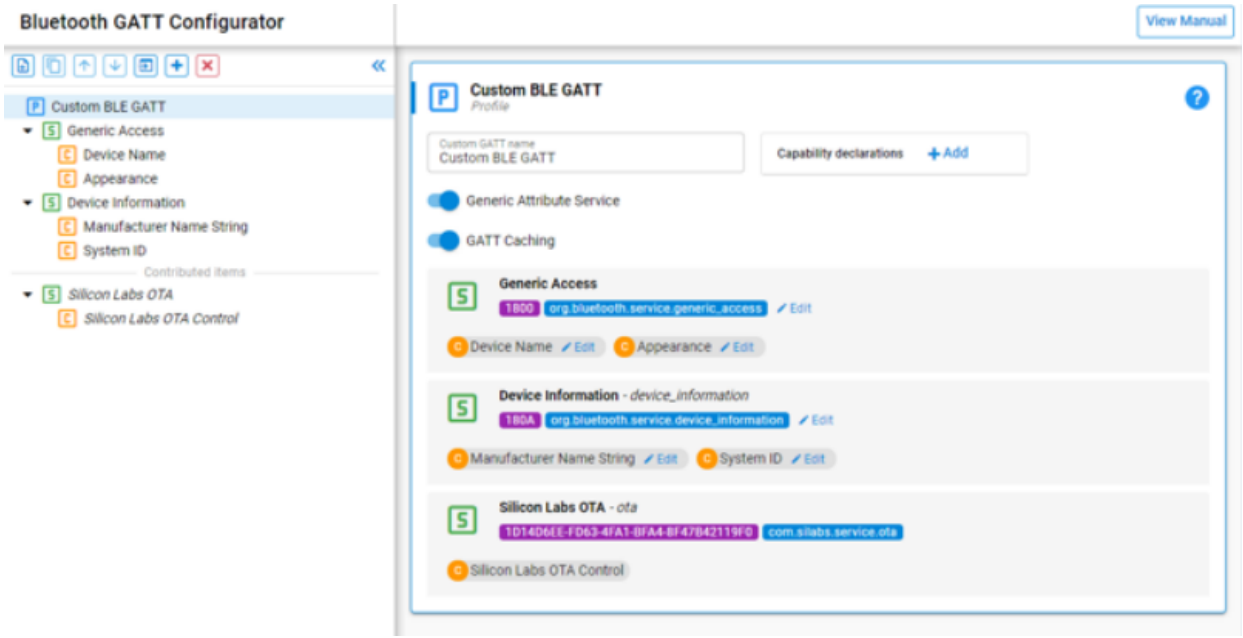

Figura 4-1. Bluetooth GATT Configurator.

Otra herramienta útil es Network Analyzer, que permite el estudio de los paquetes BLE. Ofrece todos los paquetes enviados y recibidos por el sensor BLE, el tiempo de transmisión y la explicación pormenorizada del contenido de los paquetes. Es una herramienta fundamental para el estudio del tiempo de transmisión y de los errores en el envío de paquetes, además de recibir directamente los paquetes enviados por la aplicación para entender posibles fallos.

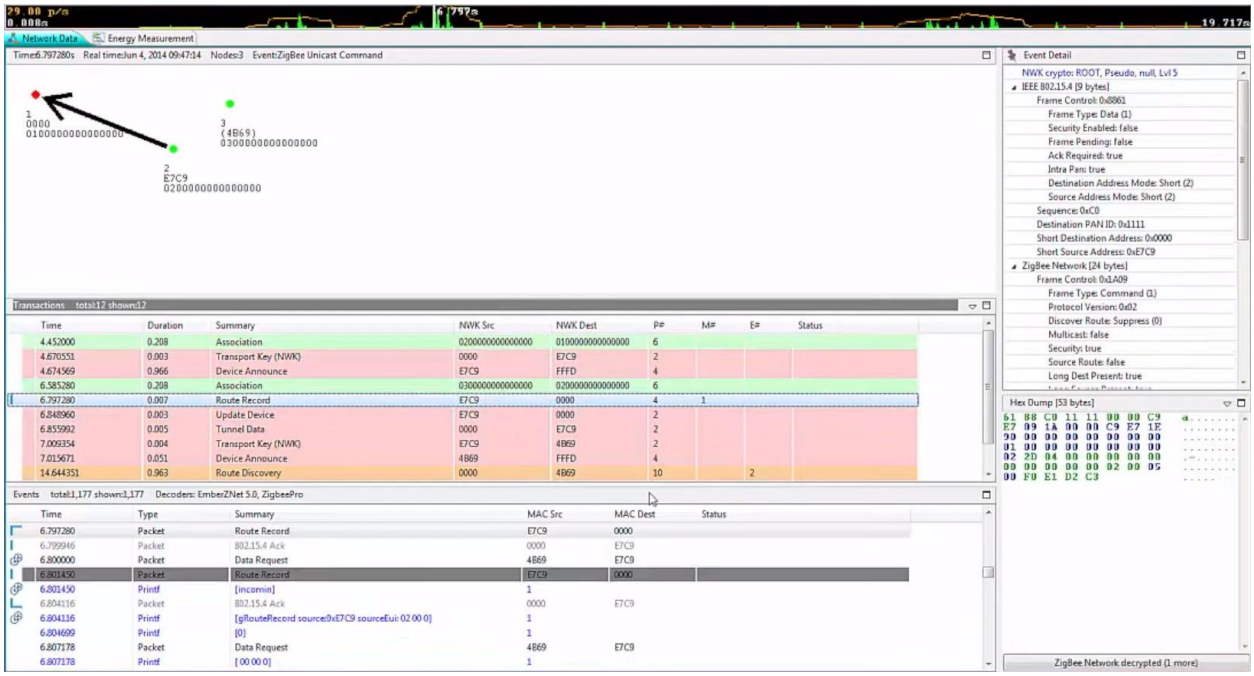

Figura 4-2. Network Analyzer.

La debilidad de Network Analyzer es que no capta los paquetes que van 'por el aire', es decir, que detecta los paquetes transmitidos por el sensor BLE, pero no es capaz de saber si llegan a su destino o no, así que esa parte del estudio de errores deberá estudiarse con otros métodos. Aun así, la herramienta Network Analyzer es fundamental para el análisis y comparación de los resultados de cada una de las dos implementaciones del trabajo.

El principal problema de Simplicity Studio es que es un entorno de desarrollo en constante cambio y dependiente de la tecnología en uso, por lo que algunas de las posibilidades de este entorno no se pueden usar debido al sensor elegido.

En cuanto a la placa de evaluación BLE BGM220P, incorpora Bluetooth Low Energy en su versión Bluetooth 5.2, puede transmitir con hasta 8 dBm de potencia y tiene una sensibilidad de -98.9 dBm (0.1% de BER) a 1 Mbps para la modulación GFSK. La capacidad de poder variar la potencia también resulta muy interesante para la realización de estudios de consumo o de errores en función de la distancia.

4.1.2 Android Studio

Por el otro lado, para la recepción de los datos, se elige el entorno de desarrollo Android Studio para el desarrollo de la aplicación móvil en Android. Pese a ofrecer compatibilidad con varios lenguajes, el elegido para este trabajo es Kotlin. Desde 2019, este lenguaje es el recomendado por Google para el desarrollo de aplicaciones móviles debido a su sencillez y eficacia, habiendo desbancado a Java como el lenguaje más usado en aplicaciones móviles.

Kotlin es un lenguaje de programación orientado a objetos. Una de las ventajas de este lenguaje es que es interoperable con Java, es decir, puede haber ficheros Kotlin y Java en el mismo proyecto y funcionar sin complicaciones. Además, en Kotlin hay suficientes librerías para un uso correcto de Bluetooth Low Energy y del perfil GATT, y se encuentran adecuadamente documentadas. El aprendizaje de este lenguaje ha supuesto un esfuerzo personal debido a un punto de entrada al mismo del desconocimiento total, sin embargo, era la mejor opción para el desarrollo de la aplicación, ya que supone un valor añadido frente a otros trabajos que se limitan a recibir los datos de otras formas más simples.

También se usa XML en algunos ficheros para adaptar IEEE 11073 a Bluetooth LE, cogiendo ideas para ellos de los ficheros estandarizados de Bluetooth SIG para GATT y de la librería de código abierto Antidote de Singove.

Android Studio da la posibilidad de elegir la versión de Android y además ofrece el porcentaje aproximado de dispositivos que soportan esa versión. Para este proyecto se ha elegido Android 9.0 que va a funcionar en el 84.1% de los dispositivos aproximadamente. La forma de exportar la aplicación es mediante la herramienta de Android Studio para la creación de APKs, siendo muy simple la generación de las mismas.

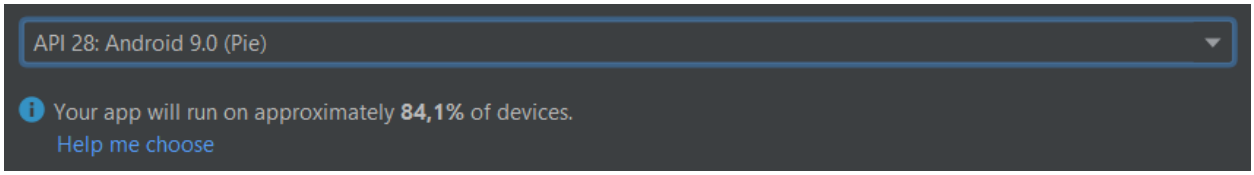

Figura 4-3. Versión de Android y número aproximado de dispositivos que la soportan.

4.2 Diseño del Sistema

Tras finalizar el análisis de todas las tecnologías, estándares y entornos, lo siguiente es dar forma al sistema. Los elementos utilizados son el sensor BLE BGM220P y el móvil. Además, para emular el funcionamiento de un sensor de bioimpedancia, se usan datos simulados provenientes de mediciones reales.

Figura 4-4. Diseño del sistema.

El sistema busca establecer una conexión Bluetooth Low Energy entre el sensor y el móvil a través del perfil GATT. Para ello, es necesario llevar a cabo los tres pasos mencionados anteriormente, adaptación de la medida de bioimpedancia a X73, encapsulación y envío de los datos X73 a través de Bluetooth Low Energy y recepción y desencapsulación de los datos X73 desde la aplicación móvil.

El primero de los pasos se realiza en el agente X73, que, como se mencionó previamente, incorpora al sensor BLE, donde se van a introducir los datos simulados de la medida de bioimpedancia y se van a adaptar a los estándares IEEE 11073 realizando un exhaustivo análisis de documentación de los estándares y añadiendo los parámetros que no estén contemplados en IEEE 11073. En el segundo paso, es donde el trabajo se fragmenta en dos partes, dependiendo del método de encapsulación de los datos. Los datos, en ambos casos, se envían a través de GATT, sin embargo, en un caso se envían los datos directamente y en el otro, se utiliza un fichero XML para el entendimiento de los datos, lo que hace que la información enviada sea mucho menor. El tercer paso consiste en reunir toda la información y procesarla adecuadamente de forma que pueda utilizarse, para ello se debe realizar la desencapsulación siguiendo los pasos de cada una de las implementaciones.

A continuación, se explica detalladamente cada uno de los pasos.

4.3 Adaptación de la medida de bioimpedancia a X73

Para una medida de bioimpedancia no existe una especialidad concreta [36] en los estándares IEEE 11073-104zz, sin embargo, hay una especialidad llamada Body Composition Analyzer (IEEE 11073-10420) que aglutina algunos de los parámetros de una medida de bioimpedancia. Sabiendo la especialidad que queremos usar, ya podemos rellenar los primeros datos que se van a usar, la clase MDS (Medical Device System), cuyos objetos representan el contexto de la medición.

4.3.1 Objeto MDS

Cada dispositivo de salud personal en los estándares IEEE 11073 está definido según un modelo orientado a objetos. Dicho modelo está conformado por todos los atributos medidos y sus características, pero además también lo forman otros atributos que representan el contexto de la medición, incluyendo identificación y estatus del angente, esos atributos conforman el objeto MDS.

La Tabla 4-1 define la clase MDS que se usa en la especialidad Body Composition Analyzer para un agente X73 con el formato de [37]. Los atributos de la Tabla 4-1 son los atributos MDS estrictamente obligatorios, hay más atributos que son opcionales pero que no aportan nada importante para el contexto de la medición, por lo que no son incluidos ni en este trabajo ni en la mayoría de trabajos sobre IEEE 11073.

Tabla 4-1. Objetos MDS.

Atributo MDS	Valor del atributo
Handle	0
System-Type-Spec-List	{MDC_DEV_SPEC_PROFILE_BCA, 2}
System-Model	{"Manufacturer", "Model"}
System-Id	EUI-64
Dev-Configuration-Id	Extended configuration: 0x4000-0x7FFF

Handle. Número de 16 bits que es solamente para el entendimiento local y que identifica a los objetos con el agente. Al objeto MDS siempre se le asigna Handle igual a cero para identificarlos.

System-Type-Spec-List. Presenta dos valores, la especialización de los estándares IEEE 11073-104zz y la versión de la misma. En este caso, la especialización es MDC_DEV_SPEC_PROFILE_BCA, correspondiente con el número 4116 y la versión es la 2. Cabe resaltar que, en X73, hay unos identificadores para todos los atributos como MDC_DEV_SPEC_PROFILE_BCA y un número asociado a ellos como 4116, el número es, lógicamente, lo que se envía en la comunicación. Tanto la especialización como la versión ocupan 16 bits, es decir, System-Type-Spec-List envía 4 Bytes.

System-Model. Información acerca del fabricante y el modelo, en este caso, al no haber sensor biomédico, no hay fabricante ni modelo, por lo que se puede elegir cualquier valor como "Fabricante" y "Bioimpedancias". Los datos son del formato Octet String, lo que quiere decir que cada carácter de la cadena de caracteres se codifica como su correspondiente valor UTF-8, por ejemplo, el valor de "Fabricante" es 0x46, 0x61, 0x62, 0x72, 0x69, 0x63, 0x61, 0x6E, 0x74, 0x65 en hexadecimal; por lo tanto, cada carácter de System-Model equivale con un Byte, en consecuencia, la longitud de este atributo es variable.

System-Id. Este atributo es del tipo IEEE EUI-64, que consiste en 24 bits de identificador organizacional único (OUI) y 40 bits de identificador definidos por el fabricante. El OUI es un valor asignado por IEEE Registration Authority [http://standards.ieee.org/regau th/index.html] y debe ser usado según IEEE Std 802-2014. Para este trabajo, se ha elegido un número de 8 Bytes cualquiera debido a que no hay un fabricante que provea este identificador.

Dev-Configuration-Id. Este atributo identifica la configuración del dispositivo del agente X73. Hay dos posibilidades, Standard configuration o Extended configuration. La primera corresponde a una configuración fiel a una existente en los estándares, es decir, que comprenda solo los parámetros que se recogen en los estándares, y la segunda, sirve para añadir parámetros nuevos. Este atributo es de vital importancia, debido a que, si se encuentra en Standard configuration, el agente y el gestor no deben pasar por una fase de configuración entre ambos dispositivos, ya que se conocen; sin embargo, si se usa Extended configuration, se ha de pasar por una etapa de configuración con el fin de compartir los parámetros de nueva inclusión. En este trabajo, se elige Extended configuration porque se van a añadir nuevos atributos medidos, para el valor de Dev-Configuration-Id, en el caso de Extended configuration, se puede elegir entre los valores del rango 0x4000-0x7FFF en hexadecimal. Para el resto del proyecto se usa el valor 0x4000, lo que significa que este objeto es de 2 Bytes.

Las funciones utilizadas para el cambio de formato, por ejemplo, cadena de caracteres a Byte, se encuentran en el Anexo B y están programadas en C, para Kotlin no se crean funciones porque el lenguaje incluye funciones propias para cambiar de formato.

4.3.2 Creación del DIM

Una vez definida el objeto MDS, es necesario conocer el DIM (Domain Information Model), que es lo que transforma toda la información del agente en un conjunto de objetos. Cada objeto tiene uno o varios atributos, y los atributos son los que informan acerca de los datos de la medida o los elementos de la misma. Una de las claves del DIM es que todo debe seguir la nomenclatura de [14]. La importancia de la nomenclatura es convertir datos que habitualmente pueden ser poco manejables en códigos númericos para mejorar la interoperabilidad

entre dispositivos. Como es lógico, acarrea una mayor dificultad la transmisión de la cadena de caracteres “kg” que enviar el código numérico que representa a los kilogramos (1731). Pues en este apartado, se van a caracterizar los objetos y a asignar los códigos correspondientes a cada atributo.

Lo primero para la creación del DIM es conocer los objetos medidos. Los objetos medidos según la especialización Body Composition Analyzer son los que aparecen en la Figura 4-5. Como se observa en dicha figura, existe una diferenciación entre los objetos, MDS, Numeric y Enumeration. El objeto MDS ya ha sido definido y caracterizado previamente, los objetos Numeric son los que realizan mediciones de tipo numérica, como, por ejemplo, grasa corporal, cuyo resultado es un número indicando el porcentaje de grasa que existe, y el otro tipo de objeto es Enumeration, esto quiere decir que el resultado no es de una medición de tipo numérica, sino un código que representa el valor real del objeto. Por ejemplo, en el caso de la Figura 4-5, Bioimpedance Analysis method (método para el análisis de la bioimpedancia), existen los códigos MDC_BIA_MTD_SINGLE_FREQ de valor 12288 que representa que solo se ha hecho la medida de bioimpedancia a una frecuencia o MDC_BIA_MTD_MULTI_FREQ de valor 12289 que representa que ha sido con varias frecuencias. Por lo que, si el valor del campo Bioimpedance Analysis method es 12288, sabemos que la medida es a una sola frecuencia.

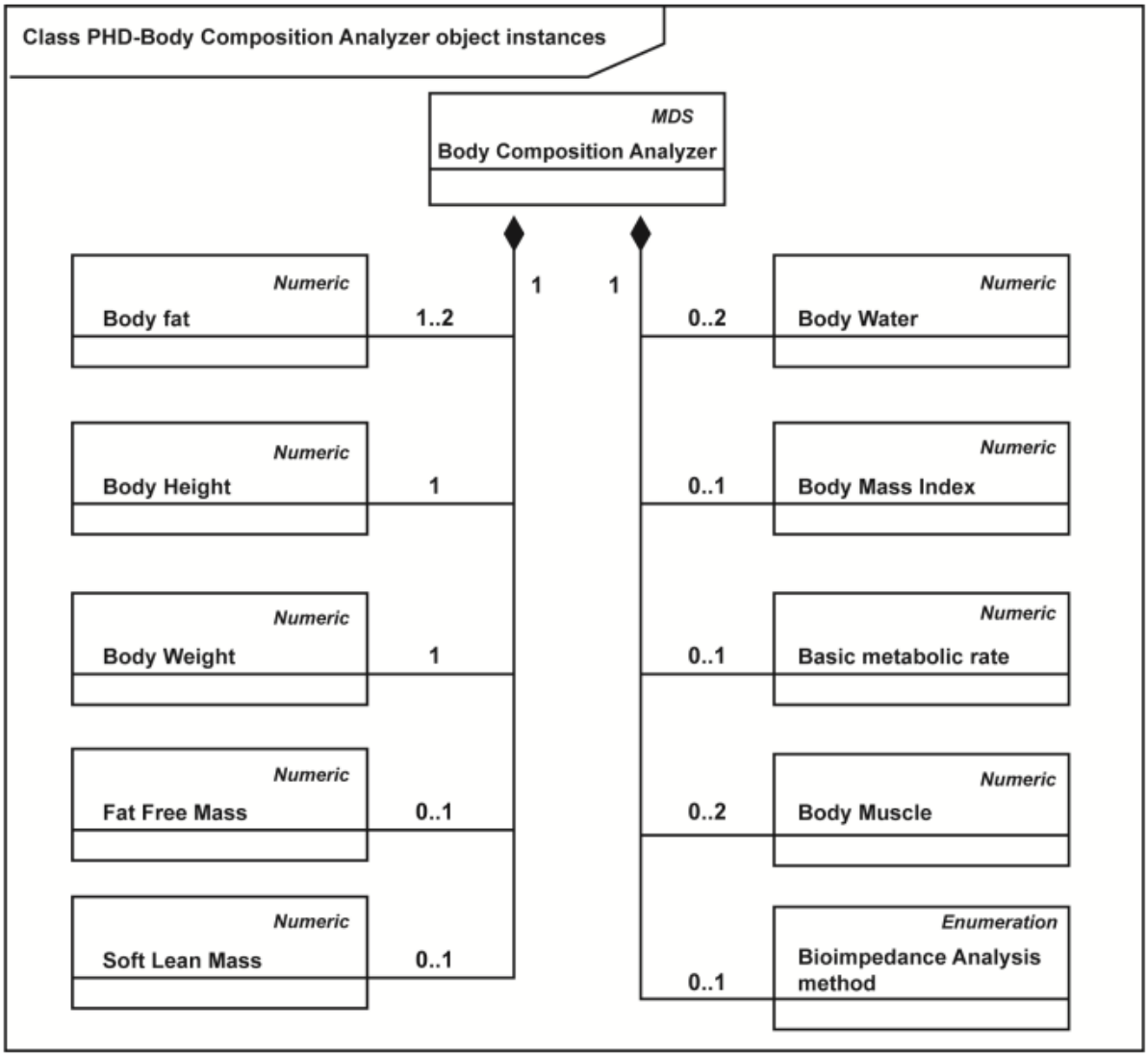

Figura 4-5. Objetos de la clase Body Composition Analyzer.

Como se observa en la Figura 4-5, hay diez objetos definidos, sin embargo, en [13] hay más parámetros medidos, por lo que, se crean nuevos parámetros que se encuentran en la Tabla 4-2. En esta tabla, los objetos vienen

acompañados de su identificador y de su valor. Los parámetros de nueva definición tienen que ser dotados de nuevos identificadores o códigos, estos códigos van de 0 a 65535 (0x0000 a 0xFFFF en hexadecimal) y el estándar IEEE 11073-20601 recoge que se pueden asignar números privados de los valores 61440 a 65535 (0xF000 a 0xFFFF en hexadecimal). Estos identificadores ocupan 2 Bytes.

Tabla 4-2. Objetos y sus códigos.

Objeto	Código	Valor del código
Peso	MDC_MASS_BODY_ACTUAL	57664
Estatura	MDC_LEN_BODY_ACTUAL	57668
Sexo *	MDC_BIA_SEXO	64011
Fecha de nacimiento *	MDC_BIA_FECHA_NACIMIENTO	64012
Masa celular corporal (BCM) *	MDC_BIA_BCM	64030
%BCM *	MDC_BIA_BCM_PORCENTAJE	64014
Volumen de agua extracelular (ECW) *	MDC_BIA_ECW	64015
Volumen de agua intracelular (ICW) *	MDC_BIA_ICW	64016
Agua corporal total (TBW)	MDC_BODY_WATER	57692
Masa libre de grasa (FFM)	MDC_MASS_BODY_FAT_FREE	57684
% FFM *	MDC_BIA_FFM_PORCENTAJE	64018
Índice de masa libre de grasa (FFMI) *	MDC_BIA_FFMI	64019
Grasa corporal (FM)	MDC_BIA_BODY_FAT	57676
% FM *	MDC_BIA_FM_PORCENTAJE	64020
Índice de grasa corporal (FMI)*	MDC_FMI	64021
Resistencia a frecuencia cero *	MDC_BIA_RES_FREC_0	64022
Resistencia a frecuencia infinita *	MDC_BIA_RES_FREC_INF	64023
Resistencia extracelular *	MDC_BIA_RES_EXTRA	64024
Resisitencia intracelular *	MDC_BIA_RES_INTRA	64025
Capacidad asociada a las membranas celulares *	MDC_BIA_BIOIMPED_CAPACIDAD	64026
Parámetro característico de la distribución de las frecuencias de relajación *	MDC_BIA_BIOIMPED_ALPHA	64027
Restraso temporal invariante con la frecuencia *	MDC_BIA_BIOIMPED_RETRASO	64028
Bioimpedance Analysis method	MDC_BIA_METHOD	57704
Frecuencia 1 *	MDC_BIA_FREQ	64029
Frecuencia … *	MDC_BIA_FREQ	64029
Frecuencia N * **	MDC_BIA_FREQ	64029

* Parámetro no definido en Body Composition Analyzer e implementado según las normas de creación de atributos privados.

** Hay una cantidad N de objetos de frecuencia representando cada una de las N frecuencias de la medida de bioimpedancia.

4.3.3 Atributos de los objetos

Tras definir los objetos, hay que asignar los atributos correspondientes a cada objeto. Para ello, vamos a

identificar los atributos más importantes para los objetos.

Handle. Misma definición que en el apartado anterior. Su valor es de 1 a M siendo M el número de objetos presentes en la Tabla 4-2.

Type. Este atributo representa tipo de objeto que es según su definición en la nomenclatura. Tiene dos partes, la primera definiendo el tipo de objeto que es según las particiones en la nomenclatura, y la segunda, el código de la Tabla 4-2. La primera parte de Type de todos los objetos medidos es *Supervisory control and data acquisition* (SCADA) y su valor es 2. Este valor ocupa 16 bits, sumados a los otros 16 bits del código de la Tabla 4-2, el tamaño de Type son 4 Bytes.

Metric-Spec-Small. Este atributo contiene las características de la medida, ofrece muchas posibilidades dependiendo de los valores que se elijan. En este caso, se van a poner los valores recomendados en la clase Body Composition Analyzer debido a que, al no haber sensor biomédico, no se pueden determinar todas las posibilidades de la medición. Este atributo es de 16 bits y cada uno de sus bits representa una característica, la característica toma el valor 1 si puede estar presente en el sistema y 0 en caso contrario. Las características elegidas dependen del objeto, se han usado algunas de las siguientes:

Tabla 4-3. Características de Metric-Spec-Small

Característica	Significado de la característica
mss-avail-intermittent	El valor del objeto está disponible intermitentemente.
mss-avail-stored-data	El agente puede almacenar y enviar datos antiguos.
mss-upd-aperiodic	El valor del objeto es enviado solamente aperiódicamente.
mss-msmt-aperiodic	La medida es aperiódica.
mss-acc-agent-initiated	El valor del objeto es actualizado por el agente.
mss-msmt-phys-ev-id	La medida es solamente fisiológica, es decir, sucede por un evento corporal (por ejemplo, latidos del corazón).
mss-msmt-btb-metric	La medida es 'de latido en latido' o 'de respiración en respiración'.
mss-msmt-manager-initiated-immediate	El agente puede iniciar una nueva medición como respuesta a una petición de datos. Si esta característica no existe, el agente envía la última medición realizada.
mss-acc-manager-initiated	El gestor puede acceder al valor del objeto siempre que pida la transmisión de datos.
mss-cat-manual	El valor del objeto ha sido introducido manualmente en el sensor biomédico.
mss-cat-calculation	El valor del objeto ha sido calculado.
mss-cat-setting	El valor del objeto proviene de las características del dispositivo.

Unit Code. Este atributo representa las unidades de medida del valor del objeto según la nomenclatura y tiene una longitud de 2 Bytes. Toma los valores siguientes:

Tabla 4-4. Valores de Unit Code.

Unidad de medida	Código	Valor del código
Adimensional	MDC_DIM_DIMLESS	512
Kilogramos	MDC_DIM_KILO_G	1731
Centímetros	MDC_DIM_CENTI_M	1297
Fecha (yyy-mm-dd)	MDC_DIM_DATE	2432
Kilohercios*	MDC_X_KHZ	62100
Ohmios	MDC_DIM_X_OHM	4288
Grados	MDC_DIM_ANG_DEG	736
Nanofaradios*	MDC_DIM_NANO_F	62101
Nanosegundos*	MDC_DIM_NANO_S	62102
Porcentaje	MDC_DIM_PERCENT	544
Litros	MDC_DIM_X_L	1600
Kilogramos entre metro cúbico	MDC_DIM_KG_PER_M_SQ	1955

* Parámetro no definido en Body Composition Analyzer e implementado según las normas de creación de atributos privados.

Las unidades de medida kiloHercios, nanoFaradios y nanosegundos han tenido que ser definidas porque no estaban recogidas en los estándares IEEE 11073. Sin embargo, sus correspondientes en el Sistema Internacional; Hercios, Faradios y segundos, respectivamente, sí se encuentran definidos. Pese a existir la definición de estos atributos, se opta por crear los nuevos debido a que, por ejemplo, enviar 3.2 nanoFaradios implica mayor sencillez que enviar 0.0000000032 Faradios. Además, IEEE 11073 no tiene ninguna preferencia por el Sistema Internacional, como se puede observar en la Tabla 4-4, es simplemente que kiloHercios, nanoFaradios y nanosegundos no se encuentran definidos porque no hay ningún atributo definido en X73 que los utilice.

Absolute Time Stamp. Este atributo representa la fecha y hora de la medición con una precisión de centésimas, si fuera posible. Su longitud es de 8 Bytes, donde cada uno de sus Bytes representa en formato INT-U8 (*unsigned integer 8*, entero positivo de 8 bits, es decir, 1 Byte) lo siguiente:

Tabla 4-5. Formato de Absolute Time Stamp para el 15 de junio de 2023 a las 12:11:54,31.

Componente de Absolute Timte Stamp	Formato	Valor hexadecimal para la fecha
Siglo	INT-U8	0x20
Año	INT-U8	0x23
Mes	INT-U8	0x06
Día	INT-U8	0x15
Hora	INT-U8	0x12
Minuto	INT-U8	0x11
Segundo	INT-U8	0x54
Centésimas	INT-U8	0x31

La principal diferencia de este atributo es que el cambio a hexadecimal es directo según los estándares IEEE 11073. Como se observa en la Tabla 4-5, el día 15 se convierte a hexadecimal como 0x15 o el minuto 11 se convierte como 0x11. Esto es posible porque los números que se deben representar no pasan de 99, por lo que, se puede hacer una conversión directa. La función que realiza esta conversión se encuentra en el Anexo B.3.

Adicionalmente, si el sensor biomédico no mide las centésimas, ese campo será 0.

Metric-Structure-Small. Este atributo explica cuántos valores mide el objeto. Cuenta con 2 Bytes, el primero de ellos vale 0 si mide solo un valor y vale 1 en caso de que mida múltiples valores, el segundo Byte representa el número de valores medidos. Este atributo es opcional, por lo que, en este trabajo, solo se va a usar para identificar los valores compuestos, es decir, los objetos con varios valores medidos.

Estos son todos los atributos usados que no tienen que ver con el valor medido, a continuación, se presentan los distintos atributos en función de si el valor es numérico o Enumeration, y si son simples o compuestos. Pero antes de eso, se presenta la Tabla 4-6 para mostrar las características de los valores.

Tabla 4-6. Características del valor medido.

Objeto	Numérico o Enumeration	Simple o compuesto
Peso	Numérico	Simple
Estatura	Numérico	Simple
Sexo	Enumeration	Simple
Fecha de nacimiento	Numérico	Compuesto
Masa celular corporal (BCM)	Numérico	Simple
%BCM	Numérico	Simple
Volumen de agua extracelular (ECW)	Numérico	Simple
Volumen de agua intracelular (ICW)	Numérico	Simple
Agua corporal total (TBW)	Numérico	Simple
Masa libre de grasa (FFM)	Numérico	Simple
% FFM	Numérico	Simple
Índice de masa libre de grasa (FFMI)	Numérico	Simple
Grasa corporal (FM)	Numérico	Simple
% FM	Numérico	Simple
Índice de grasa corporal (FMI)	Numérico	Simple
Resistencia a frecuencia cero	Numérico	Simple
Resistencia a frecuencia infinita	Numérico	Simple
Resistencia extracelular	Numérico	Simple
Resisitencia intracelular	Numérico	Simple
Capacidad asociada a las membranas celulares	Numérico	Simple
Parámetro característico de la distribución de las frecuencias de relajación	Numérico	Simple
Restraso temporal invariante con la frecuencia	Numérico	Simple
Bioimpedance Analysis method	Enumeration	Simple
Frecuencia 1	Numérico	Compuesto
Frecuencia …	Numérico	Compuesto
Frecuencia N	Numérico	Compuesto

El objetivo de la Tabla 4-6 es mostrar que la mayoría de objetos son numéricos y simples, debido a que, posteriormente, se trata al resto de objetos de forma particular por sus especialidades.

Simple-Nu-Observed-Value. Este es el atributo que muestra el valor medido de los objetos numéricos y simples. Es un valor de formato FLOAT-Type con 32 bits, compuesto por 8 bits de exponente y 24 bits de mantisa. La conversión de un valor en punto flotante a FLOAT-Type es complicada de realizar y se encuentra en una función del Anexo B.

Compound-Simple-Nu-Observed-Value. Este atributo representa un vector de Simple-Nu-Observed-Value, es decir, que el valor medido es compuesto. Se trata de un vector de FLOAT-Type de 32 bits cada uno.

Enum-Observed-Value-Simple-OID. Este atributo es para los objetos tipo Enumeration y quiere decir que el valor es OID (Object Identifier), en otras palabras, un identificador como los vistos anteriormente de valor 0 a 65535 (2 Bytes). Por ejemplo, en el caso del objeto Bioimpedance Analysis method, este atributo puede ser MDC_BIA_MTD_MULTI_FREQ de código 12289, y su significado es que la medida es a múltiples frecuencias. A diferencia de los objetos numéricos cuyo valor es fácilmente comprensible al ser un número, los objetos Enumeration deben ser explicados caso por caso. En la Tabla 4-6 se aprecia que hay dos objetos Enumeration, sexo y Bioimpedance Analysis method, el primer caso es un objeto creado en este trabajo, por lo que su valor es asignado arbitrariamente, y el segundo caso, es un objeto que aparece en Body Composition Analyzer, pero al que se le van a añadir algunos valores para ofrecer más posibilidades. Como se comentó anteriormente, los identificadores de creación privada han de valer entre 61440 y 65535.

Tabla 4-7. Valor medido del objeto sexo.

Valor	Identificador	Código del identificador
Desconocido	MDC_SEXO_DESC	65000
Hombre	MDC_SEXO_HOMBRE	65001
Mujer	MDC_SEXO_MUJER	65002
No especificado	MDC_SEXO_NO_ESPEC	65009

Tabla 4-8. Valor medido del objeto Bioimpedance Analysis method.

Valor	Identificador	Código
Bioimpedancia con una frecuencia	MDC_BIA_MTD_SINGLE_FREQ	12288
Bioimpedancia multifrecuencia	MDC_BIA_MTD_MULTI_FREQ	12289
Espectroscopia de bioimpedancia	MDC_BIA_MTD_SPECTROSCOPY	12290
Análisis vectorial de bioimpedancia	MDC_BIA_MTD_VECTOR	12291
Modelo de cuerpo entero (1 cilindro)	MDC_BIA_MDL_WHOLE_BODY	12292
Modelo de 5 cilindros con tronco, extrimidad superior derecha e izquierda y extremidad inferior derecha e izquierda	MDC_BIA_MDL_5CYL_TLREXT	12293
Bioimpedancia multifrecuencia a 2 frecuencias	MDC_BIA_MTD_MULTI_FREQ_2	65102
Bioimpedancia multifrecuencia a 3 frecuencias	MDC_BIA_MTD_MULTI_FREQ_3	65103
Bioimpedancia multifrecuencia a 4 frecuencias	MDC_BIA_MTD_MULTI_FREQ_4	65104

El cambio realizado en la Tabla 4-8 es añadir identificadores que representen el número de frecuencias en caso de que sea un análisis multifrecuencia. En la tabla solo aparece hasta 4 frecuencias, pero se puede asignar hasta el número de frecuencias que sea necesario siguiendo la nomenclatura tomada en este trabajo, de forma que para X frecuencias, identificador MDC_BIA_MTD_MULTI_FREQ_X y código 6510X.

4.3.4 Caracterización de los objetos

Una vez definidos los atributos, es hora de asignar los atributos a los objetos correspondientes y darles valor. Se empieza con los objetos numéricos simples:

Tabla 4-9. Objeto peso.

Atributo	Valor
Handle	1
Type	MDC_PART_SCADA [2] | MDC_MASS_BODY_ACTUAL [57664]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-calculation [0xF0, 0x42]
Unit-Code	MDC_DIM_KILO_G [1731]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Simple-Nu-Observed-Value	Valor medido en formato FLOAT-Type (Anexo B).

Tabla 4-10. Objeto estatura.

Atributo	Valor
Handle	2
Type	MDC_PART_SCADA [2] | MDC_LEN_BODY_ACTUAL [57668]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-manual [0xF0, 0x48]
Unit-Code	MDC_DIM_CENTI_M [1297]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Simple-Nu-Observed-Value	Valor medido en formato FLOAT-Type (Anexo B).

Tabla 4-11. Objeto Masa celular corporal (BCM).

Atributo	Valor
Handle	2
Type	MDC_PART_SCADA [2] | MDC_LEN_BODY_ACTUAL [57668]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-calculation [0xF0, 0x42]
Unit-Code	MDC_DIM_KILO_G [1731]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Simple-Nu-Observed-Value	Valor medido en formato FLOAT-Type (Anexo B).

En las Tablas 4-9, 4-10 y 4-11 se representan los atributos de los objetos numéricos simples de peso, estatura y BCM. Como observaciones a esos valores, el formato MDC_PART_SCADA [2] indica que MDC_PART_SCADA es el identificador y el número encerrado entre los corchetes es el código del identificador. En el caso de Metric-Spec-Small, [0xF0, 0x42] significa que el primer Byte toma valor 0xF0 en hexadecimal y el seguno, 0x42.

Solo se han representado los atributos de los objetos peso, estatura y BCM porque ya son representativos de todos los objetos numéricos simples, los únicos atributos que cambian en las tablas son Hanlde, Type y Unit-Code. Handle recibe un código de 1 a N siguiendo el orden de la Tabla 4-2, Type toma los valores de la Tabla 4-2 y Unit-Code es el único atributo que falta por asignar a cada objeto. Por lo que, en vez de mostrar tablas para cada objeto numérico simple, ya que serían 20 tablas llenas de información reiterativa, se presenta la siguiente tabla asignando Unit-Code a cada objeto numérico simple, teniendo en cuenta los datos de la Tabla 4-2.

Tabla 4-12. Unit-Code de cada objeto numérico simple.

Objeto	Identificador	Valor del identificador
Peso	MDC_DIM_KILO_G	1731
Estatura	MDC_DIM_CENTI_M	1297
Masa celular corporal (BCM)	MDC_DIM_KILO_G	1731
%BCM	MDC_DIM_PERCENT	544
Volumen de agua extracelular (ECW)	MDC_DIM_X_L	1600
Volumen de agua intracelular (ICW)	MDC_DIM_X_L	1600
Agua corporal total (TBW)	MDC_DIM_X_L	1600
Masa libre de grasa (FFM)	MDC_DIM_KILO_G	1731
% FFM	MDC_DIM_PERCENT	544
Índice de masa libre de grasa (FFMI)	MDC_DIM_KG_PER_M_SQ	1955
Grasa corporal (FM)	MDC_DIM_KILO_G	1731
% FM	MDC_DIM_PERCENT	544
Índice de grasa corporal (FMI)	MDC_DIM_KG_PER_M_SQ	1955
Resistencia a frecuencia cero	MDC_DIM_X_OHM	4288
Resistencia a frecuencia infinita	MDC_DIM_X_OHM	4288
Resistencia extracelular	MDC_DIM_X_OHM	4288
Resisitencia intracelular	MDC_DIM_X_OHM	4288
Capacidad asociada a las membranas celulares	MDC_DIM_NANO_F	62101
Parámetro característico de la distribución de las frecuencias de relajación	MDC_DIM_DIMLESS	512
Restraso temporal invariante con la frecuencia	MDC_DIM_NANO_S	62102

Con la información de las Tablas 4-9, 4-10, 4-11 y 4-12, se pueden conocer los atributos de todos los objetos de la Tabla 4-12. Los siguientes objetos van a ser caracterizados uno por uno a continuación:

Tabla 4-13. Objeto fecha de nacimiento.

Atributo	Valor
Handle	4
Type	MDC_PART_SCADA [2] | MDC_FECHA_NACIMIENTO [64012]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-manual [0xF0, 0x48]
Metric-Structure-Small	ms-struct-compound [1]
Unit-Code	MDC_DIM_DATE [2432]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Compound-Simple-Nu-Observed-Value	Son 4 valores codificados en INT-U8 para representar siglo, año, mes y día. Su longitud es 4 Bytes.

Tabla 4-14. Objeto frecuencia.

Atributo	Valor
Handle	24 *
Type	MDC_PART_SCADA [2] | MDC_BIA_FREQ [64029]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-setting [0xF0, 0x44]
Metric-Structure-Small	ms-struct-compound [1]
Unit-Code	MDC_X_KHZ [62100], MDC_DIM_DIMLESS [512], MDC_DIM_ANG_DEG [736]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Compound-Simple-Nu-Observed-Value	Son 3 valores en formato FLOAT-Type, su longitud es 12 Bytes. El primer valor es la frecuencia en kHz; el segundo valor, el módulo; y el tercero, el ángulo en grados.

* El valor de Handle varía respecto a la frecuencia que se mida, la primera frecuencia corresponde al valor 24, la segunda, al valor 25, y así sucesivamente hasta completar todas las frecuencias.

Las Tablas 12 y 13 corresponden a objetos numéricos compuestos, por lo que se añade el atributo Metric-Structure-Small asignándole el valor 1, equivalente a compuesto. Aparte, hay variación en Compound-Simple-Nu-Observed-Value y Unit-Code, explicados en la tabla, y en Metric-Spec-Small. Este último sufre su principal variación en que, en el caso de la fecha de nacimiento, no se trata de un valor calculado, sino introducido manualmente, por lo que varía su valor, y en frecuencia porque es un valor que tampoco proviene de una medida, sino de las características intrínsecas del dispositivo de medida. Metric-Spec-Smal es un atributo muy interesante sobre el contexto del valor medido y supone uno de los puntos fuertes de X73, sin embargo, la parte negativa es que añadir todos estos atributos sobre contexto de medición supone añadir complejidad al estándar.

Por último, queda por mostrar los atributos de los objetos Enumeration simples, debido a que no hay Enumeration compuestos en este trabajo.

Tabla 4-15. Objeto sexo.

Atributo	Valor
Handle	3
Type	MDC_PART_SCADA [2] | MDC_SEXO [64011]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-manual [0xF0, 0x48]
Unit-Code	MDC_DIM_DIMLESS [512]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Enum-Observed-Value-Simple-OID	Valor medido siguiendo la codificación de la Tabla 4-7.

Tabla 4-16. Objeto Bioimpedance Analysis method.

Atributo	Valor
Handle	23
Type	MDC_PART_SCADA [2] | MDC_BIA_METHOD [57704]
Metric-Spec-Small	mss-avail-intermittent, mss-avail-stored-data, mss-upd-aperiodic, mss-msmt-aperiodic, mss-acc-agent-initiated, mss-cat-setting [0xF0, 0x44]
Unit-Code	MDC_DIM_DIMLESS [512]
Absolute-Time-Stamp	Fecha y hora de la medición en el formato de la Tabla 4-5.
Enum-Observed-Value-Simple-OID	Valor medido siguiendo la codificación de la Tabla 4-8.

4.3.5 Modelo de comunicación X73

Tras haber definido los objetos involucrados en la transmisión de la información, la siguiente tarea es crear el modelo de comunicación entre agente y gestor X73. En lo que a comunicación X73 se refiere, no tiene nada que ver con el transporte, en este caso Bluetooth Low Energy, sino con que debe haber unos mensajes de comunicación entre el agente y el gestor para todo el envío de información. Estos mensajes se denominan APDU (Application Protocol Data Unit) y contienen toda la información relativa al tipo de mensaje que se está enviando.

Las APDUs transmitidas del agente al gestor no deben superar los 64512 Bytes y las APDUs enviadas del gestor al agente no deben superar los 8192 Bytes. En APDU debe haber un campo que informe de su longitud, de tal forma que facilite al transporte, Bluetooth Low Energy en este caso, la segmentación y la recomposición de la APDU. La segmentación se estudiará en el apartado 4.4, dedicado al envío y encapsulación de los datos y la recomposición en el apartado 4.5, dedicado a la recepción y desencapsulación de la información.

Es importante no confundir la comunicación X73 con la comunicación del transporte. Como se muestra en la Figura 4-6, existe una comunicación entre los transportes estudiada en el apartado 4.4, y otra comunicación X73 entre niveles de aplicación (Application layer) que se envía encapsulada a través del transporte.

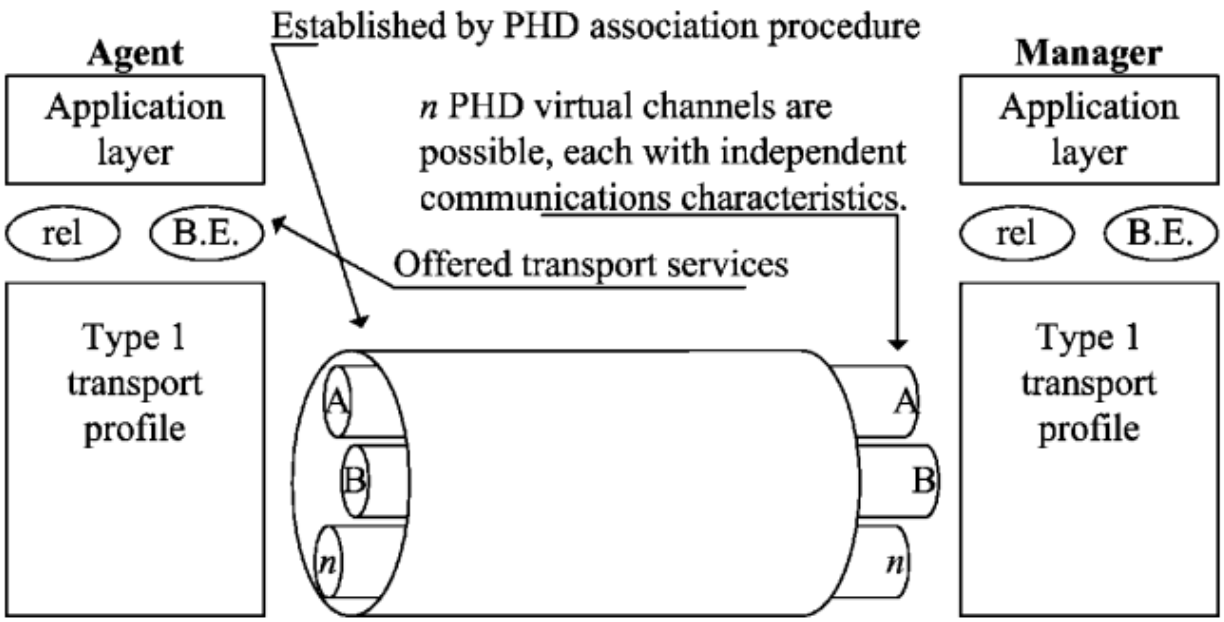

Figura 4-6. Modelo general de comunicación.

Para conocer qué tipo de APDU hay, es necesario conocer los estados del agente y del gestor, pues dependiendo del estado, se van a requerir mensajes para pasar a otro estado.

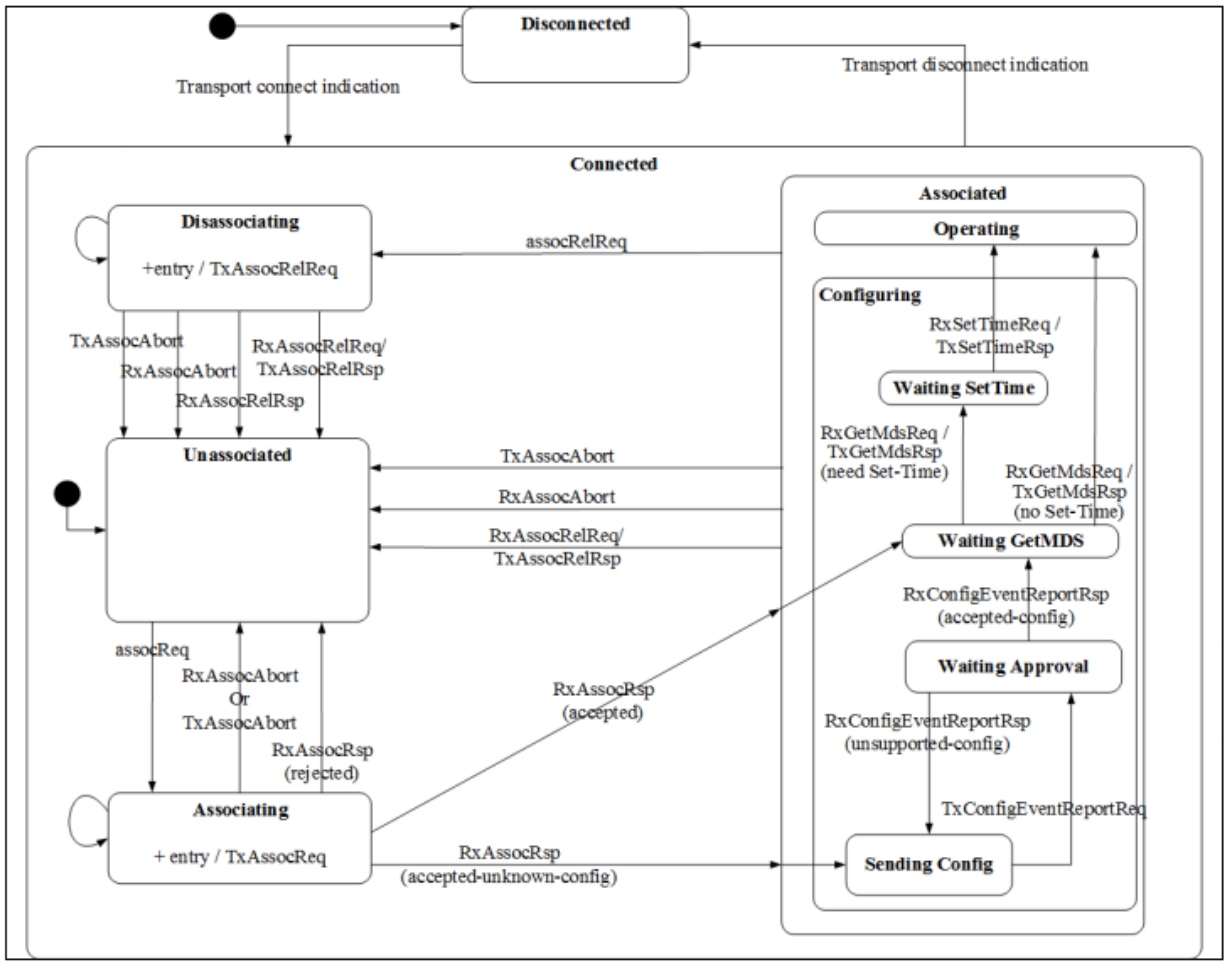

Figura 4-7. Máquina de estados del agente.

Tabla 4-17. Descripción de los estados del agente.

Estado	Descripción
Desconectado	El primer estado es desconectado y es un estado relativo a los dispositivos y al transporte. No contiene nada de información X73, pero no se puede intercambiar datos X73 sin que haya una conexión.
Conectado	Indica cuando el transporte (BLE en este caso) ha establecido una conexión. Todos los estados, a excepción de desconectado, son subestados de conectado, pues la información X73 solo se puede intercambiar si hay una conexión establecida.
No asociado	Estado en el que los dispositivos se encuentran conectados, pero agente y gestor no están asociados para el intercambio de datos X73. Esto se debe a que los dispositivos se acaban de conectar y no se han asociado todavía, o a que el gestor ha rechazado la petición de asociación del agente.
Asociando	A este estado se llega cuando el agente solicita una petición de asociación al gestor, dependiendo de la respuesta del gestor, pasa a asociado o a no asociado.
Asociado	Cuando el agente y el gestor comparten versiones y protocolos comunes, el gestor acepta ela petición de asociación y entran al estado asociado. La comunicación se mantiene en este estado hasta que una de las partes pide desasociarse. Este estado tiene como subestados a operando y configurando.
Configurando	Este ocurre tras la asociación cuando el gestor no reconoce la configuración del agente. Para entender esto, hay que volver al apartado 4.3.1 donde se definía Dev-Configuration-Id. Este atributo podía ser Standard configuration si pertenecía a una configuración existente en el estándar o Extended configuration si era una configuración original (como la de este trabajo). Si tiene la primera configuración, no tiene que pasar por este estado porque el gestor reconoce la configuración, si tiene Extended configuration sí tiene que pasar por este estado dado que la configuración es desconocida para el gestor. Además, con Extended configuration la configuración debe ser aprobada por el gestor para comprobar que esté en el formato adecuado. En caso de Extended configuration, solo tiene que pasar por el estado configurando en la primera comunicación con el gestor, en las siguientes comunicaciones, el gestor ya conocerá su configuración. La configuración es muy importante porque, a pesar de no tener los valores medidos, sí informan acerca de los atributos que se miden, por lo que, sin el conocimiento de la configuración, sería imposible intercambiar información adecuadamente.
Operando	Este es el estado en el que se transmiten los valores medidos del sensor biomédico del agente X73 al gestor. Se mantiene en este estado hasta que la desasociación y se transmite información cada vez que haya una medida, que pueden ser periódicas o aperiódicas dependiendo de la configuración del sensor biomédico y del resto de dispositivos.
Desasociando	Este es el estado en el que la asociación termina y sirve de transición para volver al estado no asociado.

En la Tabla 4-17 se muestran los estados y en la Figura 4-6 los mensajes que debe haber para cambiar de estado. Los mensajes son lo que constituye en sí la comunicación y son bastante numerosos. En la Figura 4-7 se muestra que el agente manda el mensaje Association Request y el gestor responde con el mensaje Association Respone que contiene el parámtero accepted significacndo que acepta la comunicación. La Figura 4-8, es una tabla proveniente del trabajo [29] y que expone los parámetros que conforman el mensaje Association Request empezando por su APDU que es el identificador del mensaje.

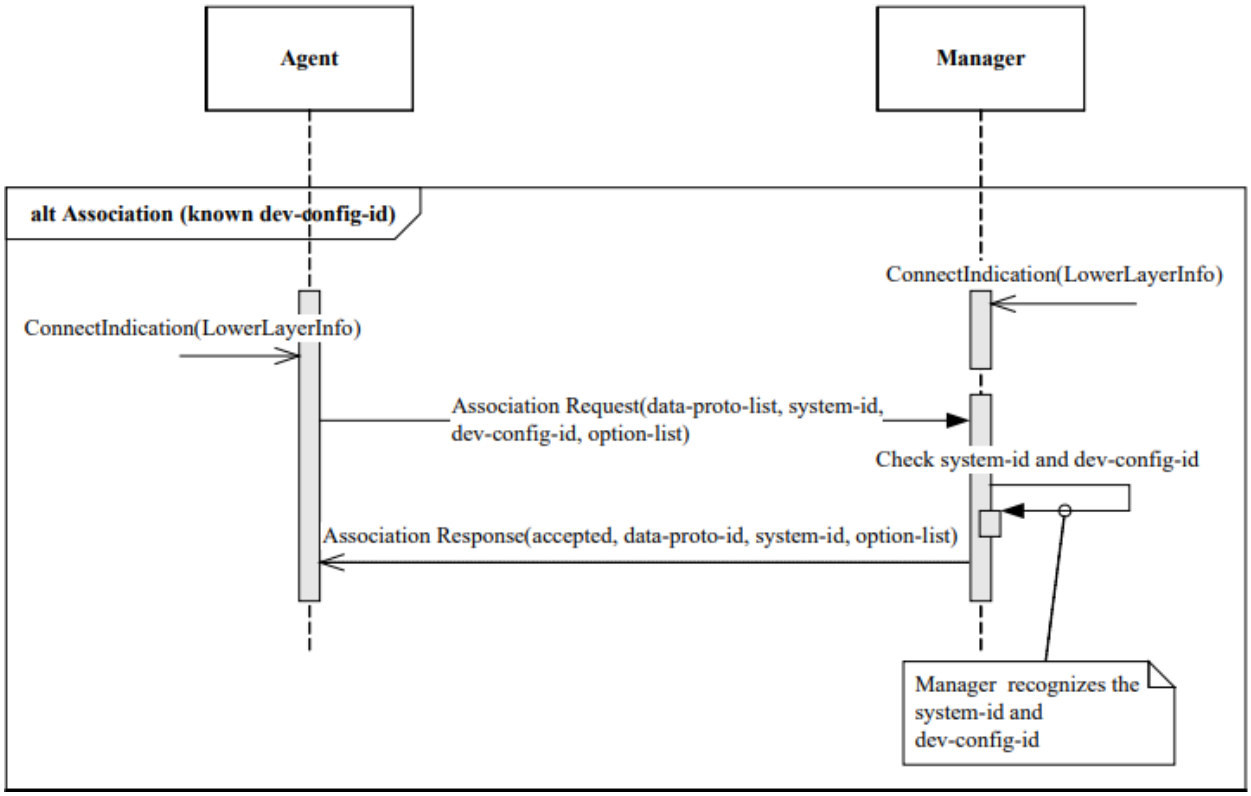

Figura 4-8. Mensajes para la asociación.

BYTES	FIELD	EXPLANTION
2	APDU CHOICE Type	Type of APDU (AARQ_APDU)
2	CHOICE.length	Length of APDU, different devices with the same length of data.
4	assoc-version	The version of optional exchange protocol
2	encoding rules	MEDR or PER
4	systemType	Different types of agents
8	system-id length	The length of system-id
2	dev-config-id	The id of device configuration
4	OptionList	Optional choices of devices

Figura 4-9. Tabla con el contenido del mensaje Association Request. [29]

En la Figura 4-8 se observa la longitud del mensaje y los atributos que componen el mensaje, siendo atributos para facilitar el entendimiento entre dispositivos como la versión del protocolo, las reglas de codificación de los

objetos o Dev-Configuration-Id para saber si se debe configurar el dispositivo o no.

BYTE	FIELD	EXPLANATION
2	APDU CHOICE Type	Types of data unit PRST_APDU
2	CHOICElength	Length of APDU, may not be the same between different devices.
2	OCTET STRING length	The length of OCTET STRING in APDU
2	Invoke-id	start of DataApdu. MDER encoded
2	CHOICE(Remote Operation InvokeEvent Report)	Types of operations
2	CHOICE length	Length of choice
2	MDS object	Object handle =0
2	ScanReportInfoFixed.obs-scan-fixed.count	The number of objects of an APDU.
2	ScanReportInfoFixed.obs-scan-fixed.length	The length of these objects.
2	ScanReportInfoFixed.obs-scan-fixed.value[1].obj-handle	The first object
2	ScanReportInfoFixed.obs-scan-fixed.value[1].obs-val-data.length	The length of this object.
12	Observed-Value	Data of observation
2	ScanReportInfoFixed.obs-scan-fixed.value[2].obj-handle	The first object
2	ScanReportInfoFixed.obs-scan-fixed.value[2].obs-val-data.length	The length of this object.
12	Observed-Value	Data of observation
	.....	

Figura 4-10. Tabla con el contenido del mensaje Data Transmission. [29]

En la Figura 4-9, se aprecia el contenido del mensaje Data Transmission, que se encuentra en el estado Operando. Lo compone una cabecera que comienza con la APDU y luego se encuentran los valores observados acompañados de su Handle, el resto de atributos asociados a los objetos se comparten en los mensajes de configuración, pues no son variables respecto a las nuevas mediciones.

Los mensajes entre agente y gestor son numerosos y con una cantidad enorme de parámetros, en el caso del presente trabajo, todos los mensajes ocupan más de 1500 Bytes, por lo que solo se muestran unos mensajes genéricos en las Figuras 4-8 y 4-9, y el resto de mensajes y sus parámetros se encuentran en el Anexo C, con el objetivo de no sobrecargar este apartado de información superflua y redundante.

Con esto último explicado, ya se puede adaptar completamente la medida de bioimpedancia a X73. Lo siguente es lo que entraña mayor dificultad, encontrar un método para la encapsulación y el envío de los datos en Bluetooth Low Energy.

4.4 Encapsulación y envío de los datos X73 a través de Bluetooth Low Energy

En este apartado comienza el uso de las tecnologías y herramientas Software, usando el entorno de desarrollo Simplicity Studio. Para ello, lo primero es adaptar los datos X73 vistos en el apartado 4.3 al lenguaje de Simplicity Studio, C/C++, para posteriormente enviarlo haciendo uso del perfil GATT. Además, se proponen dos variantes para el envío de los datos que enriquecen los resultados, la primera variante enviando los datos directamente y necesitando de encapsulación, añadiendo una forma de envío de datos genérica, y la segunda variante usando ficheros XML como apoyo para mandar no mandar la información invariable.

Antes de explicar cada variante, es necesario ofrecer las similitudes de ambas. Lo primero, dado que se hace uso de datos simulados, es establecer esos datos simulados. En este caso, como se aprecia en la Tabla 4-18, hay 4 usuarios que realizan una medida de todos los parámetros de la medida de bioimpedancia. En la tabla, aparecen los valores medidos y sus unidades, entonces, como se ha visto en el apartado 4.3, hay que adaptar esos datos a X73.

Tabla 4-18. Medida de bioimpedancia para 4 usuarios diferentes.

Objeto	Usuario 1	Usuario 2	Usuario 3	Usuario 4
Peso	59.8 kg	85.2 kg	77.9 kg	62.3 kg
Estatura	160.4 cm	191.2 cm	175.6 cm	169.0 cm
Sexo	Mujer	Hombre	Hombre	Mujer
Fecha de nacimiento	16/01/2002	08/02/2001	15/05/1960	26/11/1962
Masa celular corporal (BCM)	26.2 kg	40.3 kg	37.1 kg	29.3 kg
%BCM	43.81 %	47.30 %	47.62 %	47.03%
Volumen de agua extracelular (ECW)	16.2 L	19.2 L	15.8 L	16.0 L
Volumen de agua intracelular (ICW)	23.8 L	27.3 L	24.9 L	24.1 L
Agua corporal total (TBW)	40.0 L	46.5 L	41.7 L	40.1 L
Masa libre de grasa (FFM)	46.8 kg	71.6 kg	62.4 kg	48.0 kg
% FFM	78.26 %	84.03 %	80.10 %	77.05 %
Índice de masa libre de grasa (FFMI)	18.9 kg/m^2	20.5 kg/m^2	20.4 kg/m^2	17.1 kg/m^2
Grasa corporal (FM)	13.0 kg	13.6 kg	15.5 kg	14.3 kg
% FM	21.73%	15.96 %	19.90 %	22.95 %
Índice de grasa corporal (FMI)	8.12 kg/m^2	7.12 kg/m^2	8.82 kg/m^2	8.46 kg/m^2
Resistencia a frecuencia cero	600 ohm	650 ohmios	620 ohmios	600 ohmios
Resistencia a frecuencia infinita	350 ohm	400 ohmios	380 ohmios	360 ohmios
Resistencia extracelular	600 ohm	650 ohmios	620 ohmios	600 ohmios
Resisitencia intracelular	250 ohm	250 ohmios	240 ohmios	240 ohmios
Capacidad asociada a las membranas celulares	3.4 nF	4.1 nF	3.6 nF	4.0 nF
Parámetro característico de la distribución de las frecuencias de relajación	0.57	0.54	0.51	0.55
Restraso temporal invariante con la frecuencia	3.2 ns	2.3 ns	2.9 ns	3.7 ns
Bioimpedance Analysis method	3 frecuencias	3 frecuencias	3 frecuencias	3 frecuencias
Frecuencia 1	15 khz, módulo 7, fase 180º	15 khz, módulo 7, fase 180º	50 khz, módulo 50, fase 0º	50 khz, módulo 50, fase 0º
Frecuencia 2	150 khz, módulo 100, fase 90º	150 khz, módulo 100, fase 90º	50 khz, módulo 50, fase 0º	50 khz, módulo 50, fase 0º
Frecuencia 3	50 khz, módulo 50, fase 0º	50 khz, módulo 50, fase 0º	50 khz, módulo 50, fase 0º	50 khz, módulo 50, fase 0º

Para la creación de los datos de la Tabla 4-18, se crea el fichero 'medida.c', contenedor de la función 'medida()' que será la encargada de adaptar todos los datos a X73 y enviarlos. Dentro de esa función, existe la función 'usuarios()', que es la que crea los datos simulados de la tabla. Para ello, hace uso de una estructura de C, que se encuentra en el Anexo B.6, compuesta por todos los objetos de la Tabla 4-18.

Además, para adaptar los datos de la tabla a IEEE 11073, es necesario la creación de un fichero en C que contenga todos los códigos y sus correspondientes valores, en este caso se nombra 'biblio.h'. Contiene la instrucción '#define' para asignar los valores de las Tablas 4-2, 4-4, 4-7 y 4-8, los valores de los APDU o los identificadores del cuerpo de los mensajes de la siguiente forma:

```
#define MDC_DIM_DIMLESS        512
#define MDC_DIM_KILO_G         1731
#define MDC_DIM_CENTI_M        1297
#define MDC_DIM_TOD            2400    /*hh:mm:ss*/
#define MDC_DIM_DATE           2432    /*yyyy-mm-dd*/
#define MDC_X_KHZ              62100   /*Existe Hz pero no kHz*/
```

Estos son solo algunos de los valores de las unidades que se definen, este fichero se encuentra definido en el Anexo C.

Otra similitud es el fichero 'app.c', es un fichero del proyecto en uso en Simplicity Studio, con todo el proceso para el anuncio y el establecimiento de conexión entre dispositivos. Hay varias cuestiones fundamentales para el entendimiento de este apartado en 'app.c'. La primera, el anuncio del dispositivo con las siguientes líneas de código:

```
// Set advertising interval to 100ms.
sc = sl_bt_advertiser_set_timing(
  advertising_set_handle,
  160, // min. adv. interval (milliseconds * 1.6)
  160, // max. adv. interval (milliseconds * 1.6)
  0,   // adv. duration
  0);  // max. num. adv. Events
```

La función 'sl_bt_advertiser_set_timing' pertenece a las librerías Bluetooth de Silicon Labs e indica el tiempo entre diferentes anuncios (advertising Interval [35]), esto es la frecuencia con la que el dispositivo BLE indica a los dispositivos cercanos sobre su frecuencia. En este ejemplo, el tiempo entre anuncios es de 100ms, pues los parámetros mínimo tiempo entre anuncios y máximo tiempo entre anuncio valen 160 y, según esta función, ese valor se multiplica por 0.625, dando lugar a 100ms. Este parámetro es muy importante debido a que afecta al tiempo para el descrubrimiento de los dispositivos, cuanto más pequeño sea el intervalo entre anuncios, más rápido se descubre al dispositivo; a la latencia de la conexión, pues en un intervalo largo puede llevar más tiempo establecer una conexión; y en el consumo de energía, los intervalos más cortos consumen una mayor energía.

Otra parte importante de 'app.c' son los diferentes casos en la conexión. Por ejemplo, cuando se inicia:

```
case sl_bt_evt_connection_opened_id:
  connection_id = evt->data.evt_connection_opened.connection;
  app_log_info("Connection opened.\n");
```

```
    sl_led_turn_on(SL_SIMPLE_LED_INSTANCE(0));
  break;
```

Ese código indica qué es lo que sucede cuando se establece una conexión, en este caso, se ha programado que se encienda el LED del sensor BLE para que se mantenga encendido durante la conexión. Posteriormente, será en esta sección del código donde se añada la petición de asociación X73. Asimismo, también hay otra sección para cuando se termina la conexión y es donde se apaga el LED y donde se desasocian agente y gestor X73. También se encuentra contemplado el caso en el que alguna característica del perfil GATT varíe, por lo que, ese caso se usa para recibir los mensajes que mande el gestor X73 y actuar en consecuencia mandando los mensajes correspondientes.

Por último, otra funcionalidad que se añade en 'app.c' es que cuando se pulse el botón del sensor, se realice la medida.

```
switch (sl_button_get_state(SL_SIMPLE_BUTTON_INSTANCE(0))) {
  case SL_SIMPLE_BUTTON_PRESSED:
    medida();
```

En la función 'medida()' es donde se va a encontrar el envío de todos los datos. Se usa esta función en ambas implementaciones, pero en cada caso, siguiendo los pasos que se van a explicar en los apartados 4.4.1 y 4.4.2.

4.4.1 Envío de datos directo y genérico

Para enviar todos los datos de forma directa sin el apoyo de ningún tipo de fichero como XML o JSON, es necesario ser conscientes de la cantidad de información que se envía y encapsularla de forma que sea posible la segmentación. Además, aprovechando que se implementa de forma directa, se ha conseguido enviar los datos de forma genérica, esto quiere decir que, solo cambiando los datos de las medidas de una lista de configuración, es posible enviar cualquier medida siguiendo los estándares IEEE 11073, siempre que se traten de objetos Enumeration o numéricos, los tratados en este trabajo, aunque se podría ampliar a otros tipos de objetos.

Por lo tanto, lo primero que se va a hacer es crear la lista de configuración para la medida X73, en este caso una medida de bioimpedancia, la forma es el seguimiento de las Tablas 4-1, 4-9, 4-10, 4-11, 4-15 y 4-16.

```
/*MDS ATTRIBUTES*/
Handle = 0;
system_type_spec_list = MDC_DEV_SPEC_PROFILE_BCA;
system_type_spec_list_version = 2;
model = "Bioimpedancias";
manufacturer = "Fabricante";
system_id_24bits = "123456";
system_id_40bits = "7890ABCDEF";
dev_config_id = 16384;
```

Esta sección corresponde a la configuración de los atributos MDS explicados en 4.3.1.

```
/*Peso*/
```

```
object_class[0] = MDC_MOC_VMO_METRIC_NU;
handle[0] = 1;
type[0] = MDC_MASS_BODY_ACTUAL;
metric_spec_small[2] = 61506;                    /*    {0xF0,0x42}    */
unitcode[0] = MDC_DIM_KILO_G;
NuObsValue[0] = Usuarios[x].peso;

/*Estatura*/
object_class[1] = MDC_MOC_VMO_METRIC_NU;
handle[1] = 2;
type[1] = MDC_LEN_BODY_ACTUAL;
metric_spec_small[2] = 61512;                    /*    {0xF0,0x48}    */
unitcode[1] = MDC_DIM_CENTI_M;
NuObsValue[1] = Usuarios [x].estatura;

/*Sexo*/
object_class[2] = MDC_MOC_VMO_METRIC_ENUM;
handle[2] = 3;
type[2] = MDC_SEXO;
metric_spec_small[2] = 61512;                    /*    {0xF0,0x48}    */
unitcode[2] = MDC_DIM_DIMLESS;
NuObsValue[2] = Usuarios [x].sexo;

/*Fecha de Nacimiento*/
object_class[3] = MDC_MOC_VMO_METRIC_NU;
handle[3] = 4;
type[3] = MDC_FECHA_NACIMIENTO;
metric_spec_small[2] = 61512;                    /*    {0xF0,0x48}    */
unitcode[3] = MDC_DIM_DATE;
NuObsValueCmp[3][0] = Pacientes[x].fecha.year;
NuObsValueCmp[3][1] = Pacientes[x].fecha.month;
NuObsValueCmp[3][2] = Pacientes[x].fecha.day;
```

Esta sección corresponde a la lista de configuración de los objetos, mostrando solo 3 objetos que conforman una muestra representativa de todos los demás, donde aparece el tipo de objeto, el atributo Handle, el atributo type, metric-spec-small, unit code y el valor observado, es decir, los parámetros de las Tablas 4-9, 4-10 y 4-11. Para esta lista de configuración, se hace uso de vectores para representar los atributos para que se puedan enviar los mensajes de forma genérica con el uso de bucles. En cuanto al valor observado, NuObsValue, se iguala al parámetro correspondiente de la estructura Usuarios, que se explicó anteriormente, y la variable x es el valor que

representa cuál de los 4 usuarios es sobre el que se está realizando la medida. El atributo NuObsValueCmp es el usado para los objetos compuestos, creando una tabla de valores que también va a ser accedida posteriormente por los bucles. Es de importancia tener en cuenta que los atributos no son ahora mismo datos X73, pues no están en el formato correcto (se realiza la conversión con las funciones del Anexo B), ni tampoco forman parte de la estructura de mensajes.

Después de la creación de los objetos en C, lo siguiente es darles uso. La transmisión de los datos es a través de mensajes, en concreto, los mensajes del Anexo C. Si se observa dicho anexo, se puede comprobar la longitud y complejidad de los mensajes, teniendo muchos más parámetros que simplemente los objetos. Una de las partes más complicadas del trabajo es, por lo tanto, la creación de estos mensajes. Los mensajes necesarios para la comunicación son asociación (Anexo C.1), configuración (Anexo C.2), envío de atributos MDS (Anexo C.3), envío de los datos (Anexo C.4) y desasociación (Anexo C.5). Los mensajes de asociación ny desasociación no entrañan mucha dificultad porque no tienen nada que ver ni con la medida ni, en consecuencia, con los objetos. En cambio, el resto de mensajes son realmente complicados de generar y han requeridos miles de líneas de código para su implementación de forma genérica. Por su extensión, no se puede mostrar la creación de los mensajes, pero sí se pueden mostrar algunas partes:

```
while(handle[i] != 0){

if(object_class[i]==MDC_MOC_VMO_METRIC_NU){

 int obs_scan_fixed_value_handle = i + 1;
 enteroAHexadecimal(obs_scan_fixed_value_handle, e_t, 4);
 dividirCadena(e_t, &parte1, &parte2);
 vector_data[30 + y] = (uint8_t) convertirHexAEntero(parte1);
 vector_data[31 + y] = (uint8_t) convertirHexAEntero(parte2);

 int obs_scan_fixed_value_length = 12;
 enteroAHexadecimal(obs_scan_fixed_value_length, e_t, 4);
 dividirCadena(e_t, &parte1, &parte2);
 vector_data[32 + y] = (uint8_t) convertirHexAEntero(parte1);
 vector_data[33 + y] = (uint8_t) convertirHexAEntero(parte2);

 float observed_value = NuObsValue[i];
 uint32_t float_type = convertToFloatType(observed_value);
 uint8_t byteArray[4];
 uint32ToByteArray(float_type, byteArray);
 vector_data[34 + y] = byteArray[0];
 vector_data[35 + y] = byteArray[1];
 vector_data[36 + y] = byteArray[2];
 vector_data[37 + y] = byteArray[3];

 vector_data[38 + y] = (uint8_t) obtenerHexadecimal(century);
```

```
    vector_data[39 + y] = (uint8_t) obtenerHexadecimal(yy);
    vector_data[40 + y] = (uint8_t) obtenerHexadecimal(mm);
    vector_data[41 + y] = (uint8_t) obtenerHexadecimal(dd);
    vector_data[42 + y] = (uint8_t) obtenerHexadecimal(hour);
    vector_data[43 + y] = (uint8_t) obtenerHexadecimal(min);
    vector_data[44 + y] = (uint8_t) obtenerHexadecimal(sec);
    vector_data[45 + y] = (uint8_t) obtenerHexadecimal(cen);

    y+=16;
    i++;
  }
  }
```

Esta sección de muestra la parte más importante, el envío de los datos, y lo hace de forma genérica recorriendo un bucle que pasa por cada uno de los 26 objetos de esta medida. Además, podría servir para cualquier otra medida cambiando la lista de configuración mentada anteriormente.

En cuanto a la explicación de este código en sí, se trata de un bucle que se recorre mientras el atributo Handle del objeto sea distinto de 0, es decir, se recorren los 26 objetos de la medida. Esta sección es la encargada del envío de los datos numéricos simples, no envía el resto de datos porque los valores observados son totalmente diferentes. El primer valor que se define es el atributo Handle del objeto, y mediante las funciones enteroAHexadecimal y dividirCadena del Anexo B.1, se convierte el número en dos Bytes con las que se rellena el vector de información. A continuación, se hace lo mismo con el valor de la longitud, este valor se mantiene a 12 porque es la longitud de los valores numéricos simples. Para añadir el valor observado, se usa NuObsValue, definida en la lista de configuración, se pasa a formato FLOAT_Type con la función del Anexo B.5 y sus valores se asignan al vector de datos con otra función del Anexo B.4. Por último, se realizan las conversiones especiales de los números de Time Stamp (fecha y hora) de la forma particular de la Tabla 4-5, haciendo uso de la función del Anexo B.3. Las funciones auxiliares han sido relegadas al Anexo B porque ocuparían demasiado espacio en el cuerpo del trabajo de forma innecesaria.

La clave de esta sección del código es el vector de información ‘vector_data’. Este vector contiene toda la información de la medida encapsulada en formato Byte, sin embargo, es un vector muy extenso, su longitud se acerca a los 500 Bytes, en otros mensajes se envían más Bytes todavía. Por lo que, es necesaria la segmentación, pero para explicar el método de segmentación, antes va a ser necesario explicar el perfil GATT, sus características y los permisos de envío, de forma detallada.

Como se explicó en el capítulo 2, el perfil GATT contiene servicios, y esos servicios contienen características. Son las características donde sucede el intercambio de información en BLE GATT. Las características tienen una serie de permisos que se pueden activar o desactivar según quiera el creador del perfil. Dichos permisos aparecen en la Figura 4-10.

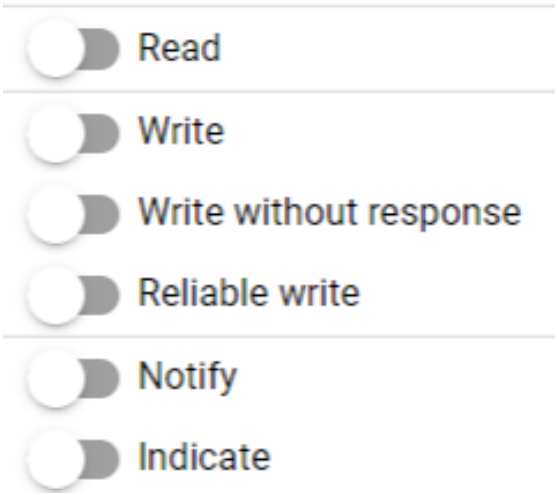

Figura 4-11. Permisos de una característica del perfil GATT.

Hay permisos para leer y escribir en la característica, pero además también hay notificaciones e indicaciones. Estas dos últimas permiten al dispositivo remoto, el móvil en este caso, recibir los valores cuando cambian. Para transmisión de datos pequeños, es más útil escribir directamente en GATT, sin embargo, para transmisiones grandes de información, como en este trabajo, es mejor el uso de las notificaciones. Esto se debe a que, para escribir, es necesario enviar una petición de escritura y recibir una respuesta, sin embargo, las notificaciones solo envían la información necesaria.

Las notificaciones son la forma de enviar paquetes, que es lo requerido al tener tanta información. Para el tamaño del paquete es necesario conocer la longitud de PDU (Protocol Data Unitd), en el caso de este sensor BLE de Silicon Labs es 251 bytes. No confundir PDU con la APDU mentada anteriormente, son conceptos similares, pero uno es el correspondiente a Bluetooth Low Enery y el otro a IEEE 11073. A los 251 bytes de PDU, hay que restarles la cabecera de L2CAP, 4 bytes, y la cabecera de GATT, 3 bytes, haciendo un total de paquetes de 244 bytes.

Los paquetes se transmiten de la siguiente forma:

```
int x=0;


for (int n=0; n< data_length/244+1; n++) {
   for(int i=0; i<244; i++ ){
      vector_config[i] = vector_config[i+x];
   }
   sl_bt_gatt_server_notify_all(gattdb_mds,
                        244,
                        vector_config);
   sl_sleeptimer_delay_millisecond(10);
   x+=244;
}
```

En esta sección de código, se utiliza un bucle que divide el tamaño de los datos de transmisión entre 244 para conocer el número de paquetes a enviar. Después de eso, para cada paquete, se realiza la asignación de los

valores y se usa la función 'sl_bt_gatt_server_notify_all' de la librería Bluetooth de Simplicity Studio. Esta función tiene como primer parámetro la característica que se notifica, el segundo parámetros es la longitud del paquete, establecida a 244 en este caso, si fuera superior, la cantidad enviada seguiría siendo 244, y el tercer parámetro es el vector de datos.

Después de esa función, es necesario usar la función 'sl_sleeptimer_delay_millisecond(10)' de las librerías de Simplicity Studio, que para el sistema durante 10 ms, con el objetivo de no enviar el siguiente paquete antes de terminar de enviar el primero, se tarda aproximadamente 9 ms en enviar el paquete [37]. Se intentó reducir el tiempo de espera por debajo de los 10 ms, pero produjo errores.

Tras finalizar el envío de todos los paquetes de datos, ya se ha finalizado el envío de datos de forma directa y genérica. La principal ventaja de esta implementación es la existencia de una cómoda lista de configuración, creada gracias a muchas funciones auxiliares en este trabajo, que puede evitar tener que adentrarse en todas las dificultades que entraña el conocimiento de los estándares IEEE 11073. Las ventajas y desventajas de cada implementación se comentan detalladamente en el capítulo 5, dedicado a los resultados.

4.4.2 Envío de datos usando XML´

La otra forma de enviar los datos es mediante ficheros auxiliares tipo XML o JSON, para que mediante código se tengan los parámetros invariables y solo se envíe a través del perfil GATT los datos variables. En este caso, se eligen los ficheros porque son más fáciles de implementar con GATT. En el entorno de desarrollo Simplicity Studio y en Android Studio, dicho fichero estará asociado a un UUID elegido arbitrariamente y siendo el mismo en ambos entornos, para poder identificarlo adecuadamente

Para la creación del fichero, se usa como referencia un fichero XML de la antigua librería de código abierto Antidote de Signove, aunque es necesario hacer muchos cambios para adaptarla al GATT. A continuación, se encuentra una pequeña muestra de unos de los ficheros XML creados para mensajes.

```
<Value>
<Field name="System-Type-Type">
   <InformativeText>El tipo de especialización.</InformativeText>
   <Requirement>Mandatory</Requirement>
   <Format>uint16</Format>
</Field>
<Field name="System-Type-Version">
   <InformativeText>La versión de la especialización.</InformativeText>
   <Requirement>Mandatory</Requirement>
   <Format>uint16</Format>
</Field>
<Field name="System-Id">
   <InformativeText>System-ID en formato EUI-64.</InformativeText>
   <Requirement>Mandatory</Requirement>
   <Format>uint64</Format>
</Field>
   <Field name="Dev-Configuration-Id">
   <InformativeText>Configuración del dispositivo.</InformativeText>
   <Requirement>Mandatory</Requirement>
   <Format>uint16</Format>
```

```
</Field>
```

En esta sección de código se encuentran algunos de los atributos MDS que se encuentran en el fichero MDS para el intercambio de los atributos MDS. Se ha optado por no presentar en el cuerpo del trabajo los ficheros XML al completo para no ocupar espacio innecesario y han sido relegados a los anexos.

Lo más importante a destacar en estos ficheros XML, es que solo se intercambia la información presente en ellos. En el caso de este ejemplo, solo se envía el tipo de especialización (2 bytes), la versión de la especialización (2 bytes), System -Id (8 bytes) y configuración del dispositivo (2 bytes). Enviando toda esta información de forma directa, haría falta enviar los identificadores de todos los atributos, la longitud de todos los atributos y las cabeceras de los mensajes, lo que haría necesario un número de bytes mucho mayor. Por lo que, esta es la principal ventaja de esta forma de envío de datos. Sin embargo, su debilidad es que todos los dispositivos involucrados deben conocer la configuración del fichero XML, en caso contrario, no se entendería la información transmitida. Esto último es necesario para lograr la interoperabilidad total entre dispositivos, quizá ayudaría la creción unos ficheros XML generales y de referencia, y su inclusión en los estándares IEEE 11073, para que la creación de ficheros XML fuera homogénea entre todos los fabricantes. En este trabajo, se proponen los XML en los anexos.

Tras crear el fichero XML, es necesario enviar la información siguiendo el formato que se pide. Para ello, se usan los mismos parámetros que anteriormente se incluyeron en la función medida() y se envían los datos usando la propiedad de escribir, que es mucho más cómoda cuando no se envían paquetes.

```
uint8_t System_Type_Spec = (uint16_t) MDC_DEV_SPEC_PROFILE_BCA;
sl_bt_gatt_server_write_attribute_value(caracteristica,
                                        0,
                                        sizeof(System_Type_Spec),
                                        &System_Type_Spec);
```

Para el envío de la especialidad, se define su valor y se envía por el perfil GATT haciendo uso de la función ‘sl_bt_gatt_server_write_attribute_value()’ de la librería Bluetooth de Simplicity Studio. Está función tiene como primer parámetro la característica del perfil GATT por donde se va a enviar el atributo (cada mensaje se envía por una característica diferente), el segundo parámetros es la posición del atributo, en este caso, la especialidad ocupa el primer lugar del fichero XML, por lo que está en la posición 0, el tercer atributo es el tamaño del atributo y el último parámetro es el valor del atributo. Con esta sencilla función, se puede escribir en las características del perfil GATT, de atributo en atributo de un mismo mensaje. No se muestra la transmisión de todos los atributos en este trabajo porque es numerosa y una muestra es representativa de todas las demás.

Las ventajas y desventajas de esta forma de envío de datos y su comparación frente a la otra implementación, se encuentran en el capítulo 5, dedicado a los resultados.

4.5 Recepción y desencapsulación de los datos X73 desde la aplicación móvil

Tras todas las explicaciones anteriores y tras haber completado el envío, falta la recepción de los datos. En cuanto a la aplicación móvil, se ha usado la aplicación EFR Connect de Silicon Labs, que cuenta con el perfil GATT y es de código abierto. Entonces, la recepción de los datos es inmediata con el uso de esta aplicación (que hace uso de las librerías Bluetooth Low Energy de Kotlin), y lo que falta es la desencapsulación y el procesamiento de la información, que se puede realizar gracias a que es una aplicación de código abierto.

Para ello, es necesario sumergirse en la complejidad de la aplicación EFR Connect e identificar dónde y cómo se reciben los datos del perfil GATT. Al usarse las librerías BLE de Kotlin, el funcionamiento y el código de esta aplicación son similares a otras aplicaciones de BLE.

Lo principal en este tipo de aplicaciones es identificar (o crear) las funciones 'onCharacteristicRead()', 'onCharacteristicWrite()' y 'onCharacteristicChanged()', pertenecientes a la clase BluetoothGattCallback, dedicada a gestionar los eventos que suceden en GATT. Estas funciones son comunes para todas las aplicaciones con BLE GATT programadas en Kotlin y de estas funciones hacen uso las clases originales de cada aplicación. Para la escritura desde la aplicación móvil, se puede programar directamente desde la función 'onCharacteristicWrite()'o 'onCharacteristicChanged()', pero para la recepción de los datos vamos a usar otras clases que hagan uso de 'onCharacteristicRead()', para que se puedan mostrar los datos procesados.

4.5.1 Recepción de datos directa

Debido a la existencia de dos implementaciones diferentes en este trabajo, también hay dos formas de recibir los datos. La primera de ellas, para la forma directa, recoge todos los datos y los analiza mediante lógica computacional, para transformarlos en la información. Esto supone el uso de muchas instrucciones para transformar los valores recibidos en los identificadores de los atributos X73 o transformarlos en la información.

```
fun convertirType(valorX73: Int): String {
    var texto = "";
    if (valorX73 == 57664)
        texto= "MDC_MASS_BODY_ACTUAL"
    if (valorX73 == 57668)
        texto= "MDC_LEN_BODY_ACTUAL"
    if (valorX73 == 64011)
        texto= "MDC_SEXO"
    if (valorX73 == 64012)
        texto= "MDC_FECHA_NACIMIENTO"

   …

    return texto
}
```

Esta función llamada convertirType(), recibe como parámetro el código del atributo Type y tiene que devolver el identificador, es decir, recibe el valor 57664 y lo convierte en MDC_MASS_BODY_ACTUAL. Esta función es muy larga dado que contiene los tipos de los 26 objetos en este trabajo (no se han añadido más por simplicidad), por lo que se corta con el uso de los puntos suspensivos.

Para conseguir desencapsular y procesar la información correctamente, se hace uso de multitud de funciones en Kotlin como convertirType(). Por este motivo, no se pueden adjuntar todas, pero se va a mostrar la función convertirFloatType(), que es la función que posibilita la desencapsulación usando el método requerido en IEEE 11073. Esta función es meritoria debido a que es complicado encontrar información correcta acerca del tipo FLOAT-Type requerido en X73. La función convertirFloatType() desencapsula los datos que encapsula la función del Anexo B.5.

```
fun convertirFloatType(fieldValue: ByteArray){
```

```
var exponent = 0
var cadena0 = "00"
val bytes1 = cadena0.toByte(16)
var bytes2 = Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[1].toByte(16)
var bytes3 = Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[2].toByte(16)
var bytes4 = Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[3].toByte(16)

var bytesCombinados = byteArrayOf(bytes1,bytes2, bytes3, bytes4)

var entero = bytesCombinados.fold(0) { acc, byte ->
   (acc shl 8) or (byte.toInt() and 0xFF)
}

if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "00")
   exponent = 0
else if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "FF")
   exponent = -1
else if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "FE")
   exponent = -2
else if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "FD")
   exponent = -3
else if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "FC")
   exponent = -4
else if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "FB")
   exponent = -5
if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "01")
   exponent = 1
if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "02")
   exponent = 2
if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "03")
   exponent = 3
if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "04")
   exponent = 4
if (Converters.bytesToHexWhitespaceDelimited(fieldValue).split(" ")[0] == "05")
   exponent = 5

var obsValue = entero * 10.0.pow(exponent)
```

```
}
```

4.5.2 Recepción de datos usando XML

Al igual que para el envío de datos, la recepción es mucho más simple usando XML.Teniendo los ficheros XML en los anexos, solo hace falta referenciar en el código los atributos de la característica para leer la información y procesarla.

```
            characteristicFieldValue.text = StringBuilder().apply {
                var currentValue = readValue()
                if (characteristicFieldName.text == "Dev-Configuration-Id")
                append(showValue())
            }.toString()

    private fun showValue() : String {
        var currentValue = readValue()
        currentValue = if (field.isStringFormat()) {
            currentValue.replace("\u0000", "")
        } else {
            calculateModifiers(currentValue)
        }
        return currentValue
    }
```

En esta sección de código, se hace referencia al atributo Dev-Configuration-Id, que se mostraba en el fichero XML de 4.4.2. Con esto, se elimina toda la información repetitiva presente en el envío directo de datos, ya que se encuentra presente en el fichero XML en vez de tener que ser transmitida.

5 RESULTADOS

La excelencia no es un acto, sino un hábito.

- Aristóteles -

Las implementaciones del capítulo anterior dan como resultado una serie de mensajes que intercambian el dispositivo BLE y el teléfono móvil para la asociación, configuración, envío de datos y desasociación X73. Este capítulo presenta los resultados obtenidos y estudia varias características de los mensajes como la potencia recibida, el tamaño y el throughput y resultados genereales sobre la interoperabilidad y las dificultades.

A continuación, se muestran algunos de esos mensajes capturados desde la aplicación móvil y el procesamiento de esos mensajes.

ASSOCIATION REQUEST:
E2 00 00 32 80 00 00 00 00 01 00 2A 50 79
00 26 70 00 00 00 80 00 E0 00 00 00 00 00
00 00 00 80 00 00 00 08 12 34 56 78 90 AB
CD EF 40 00 00 01 01 00 00 00 00 00 00

APDU de petición (AarqAPDU)
Soporta la versión 1 del procedimiento de asociación
Reglas de codificación: MDER
Soporta las versiones v2, v3 y v4 del protocolo
System-Id: 1234567890ABCDEF
Dev-Configuration-Id: Extended Configuration

El dispositivo necesita pasar por la etapa de configuración

Figura 5-1. Mensaje de petición de asociación.

La Figura 5-1 muestra la petición de asociación del dispositivo BLE, siendo la serie de bytes el mensaje encapsulado y el resto es la información que contiene el mensaje tras desencapsularlo y procesarlo. En este caso, al contar con configuración extendida, el móvil informa de la necesidad de pasar por la etapa de configuración,

es decir, que el próximo mensaje es el de configuración. La explicación extendida del mensaje de la Figura 5-1 se encuentra en el Anexo C.1.1.

El siguiente mensaje que envía el dispositivo BLE es el de configuración, que se encuentra en el Anexo C.2.1. Después de la configuración, se envía el mensaje que informa sobre los objetos MDS. Para este caso, se van a comparar los mensajes que envían las dos implementaciones del trabajo en la Figura 5-2, implementación directa y en la Figura 5-3, implementación con XML.

```
E7 00 00 5C 00 5A 12 34 02 03 00 54 00 00
00 05 00 4E 0A 5A 00 08 00 01 00 04 10 14
00 02 09 28 00 18 00 00 42 69 6F 69 6D 70
65 64 61 6E 63 69 61 73 00 00 46 61 62 72
69 63 61 6E 74 65 00 00 00 0A 00 08 12 34
56 78 90 AB CD EF 00 00 00 02 00 00 00 00
00 08 20 23 06 16 20 55 44 11

APDU de presentación (PrstAPDU)
System-Type-Spec-List:
{MDC_DEV_SPEC_PROFILE_BCA, 2}
Sytem-Model: {Bioimpedancias, Fabricante}
System-Id: 1234567890ABCDEF
Dev-Configuration-Id: Extended Configuration
Absolute Time Stamp: 16 de junio del 2023 a
las 20:55:34,11
```

Figura 5-2. Objetos MDS en la implementación directa.

```
10 14
Tipo: MDC_DEV_SPEC_PROFILE_BCA


00 02
Versión: 2


12 34 56 78 90 AB CD EF
System-Id: 1234567890ABCDEF


40 00
Dev-Configuration-Id: Extended Configuration


42 69 6F 69 6D 70 65 65
Fabricante: Fabricante


61 6E 63 69 61 73 46 61 62 72 69 63 61 6E 74 65
Modelo: Bioimpedancias
```

Figura 5-3. Objetos MDS en la implementación con XML

La información en ambos casos es la misma, la diferencia principal es que, con el uso de XML, la reducción del tamaño es notable. En el caso de los objetos MDS, la diferencia de tamaño no es tan abismal como en los mensajes de configuración o de envío de datos, ocupando el primero de ellos más de 1000 bytes y el segundo más de 500 bytes. La reducción del tamaño sucede principalmente porque el código XML elimina el uso de cabeceras y de algunos identificadores que ocupan una gran cantidad de espacio. Por ejemplo, de forma directa, para enviar System-Type-Spec-List es necesario enviar previamente el identificador del tipo (MDC_ATTR_SYS _TYPE_SPEC_LIST, de valor 0x0A 0x5A), la longitud del atributo (0x00 0x08), número de tipos y versiones (0x00 0x01) y longitud de tipo y versión (0x00 0x04); mientras que, usando XML, todos estos campos se expresan con código XML.

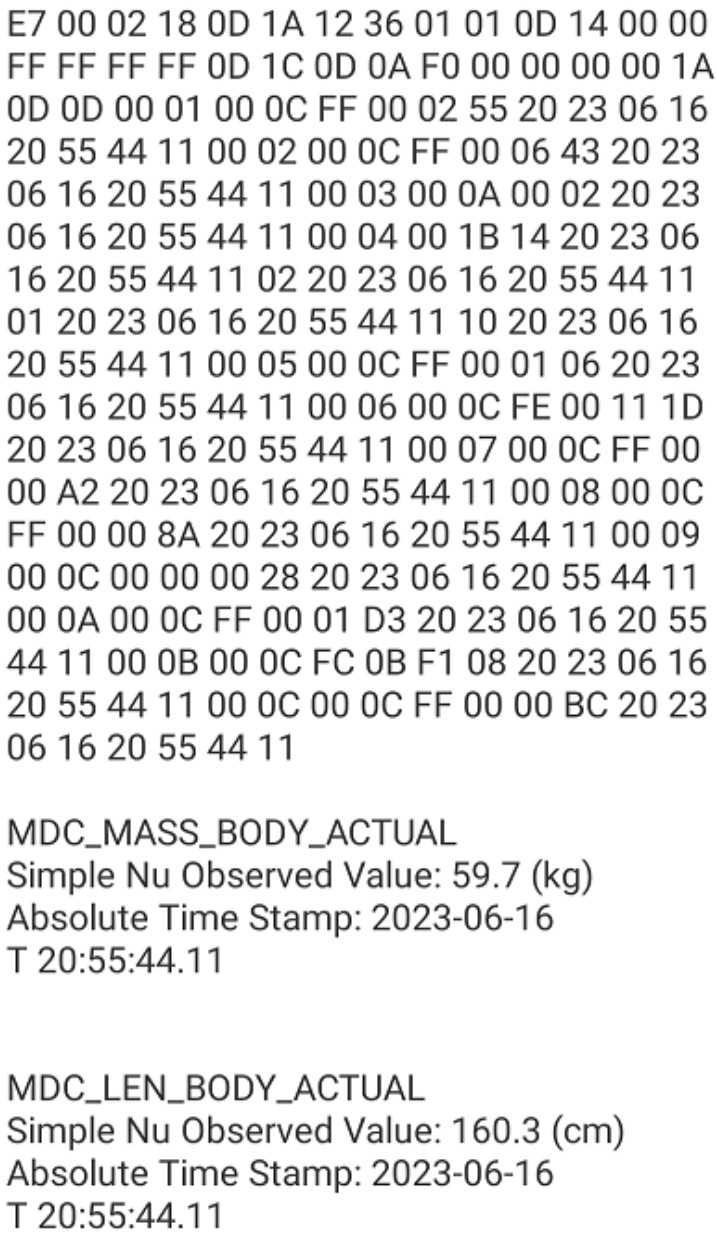

Figura 5-4. Primer paquete del envío de datos.

Debido a que el envío de datos ocupa más de 500 bytes, y al ser las notificaciones de tamaño 244 bytes, es necesaria la segmentación en 3 paquetes. En la Figura 5-4, se muestra la parte más importante de toda medida, los valores medidos. Aparece procesada solamente la información de los 2 primeros valores medidos porque es una captura realizada desde el móvil. No se muestran todos los paquetes del envío de datos ni todos los mensajes para no ocupar espacio innecesario, todos los mensajes intercambiados entre dispositivo BLE y teléfono móvil

se encuentran en el Anexo C.

En los siguientes apartados de este capítulo, se van a estudiar diferentes aspectos de la comunicación y, en algunos de ellos, se realiza una comparación de prestaciones de cada una de las dos implementaciones de este trabajo.

5.1 Potencia recibida

Como se comentó en 4.1, el dispositivo BLE puede ofrecer hasta 8 dBm de potencia transmitida y un mínimo de -26 dBm, la potencia transmitida por defecto es 0 dBm. Las siguientes figuras muestran la potencia recibida por el teléfono móvil en dBm frente a la distancia a la que se encuentran dispositivo BLE y móvil cambiando la potencia transmitida en cada figura. La sensibilidad en BLE a 1 Mbps de un smartphone es, aproximadamente, -95 dBm (0.1% BER), por lo que, se puede establecer la conexión e intercambiar información con esa potencia recibida, sin embargo, es recomendable que, al menos, esté en el rango de -85 a -90 dBm para evitar errores.

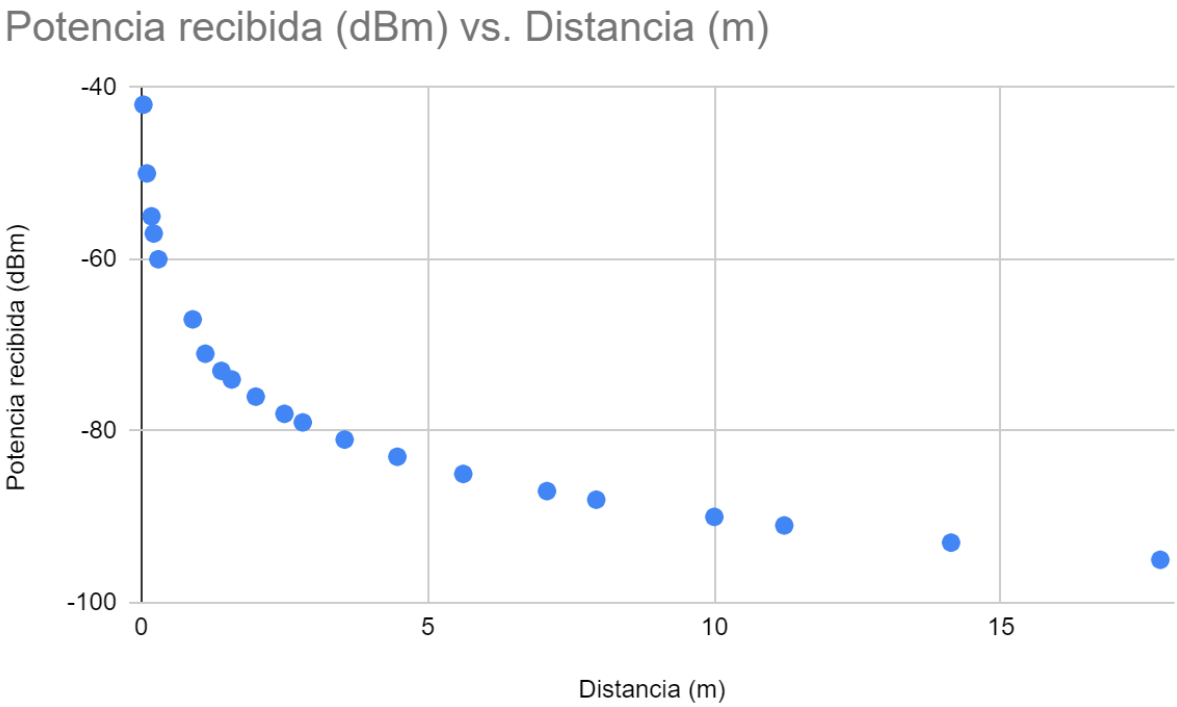

Figura 5-5. Potencia recibida frente a la distancia para 0 dBm TX.

La Figura 5-5 muestra el caso estándar, la transmisión de 0 dBm, donde la potencia recibida frente a la distancia es una función exponencial decreciente. Para este caso, se observa que hasta los 10 metros los niveles de potencia serían adecuados, incluso se podría aumentar la distancia hasta los 20 metros y seguiría funcionando. Esta potencia de transmisión es adecuada ya que el rango requerido en las comunicaciones con sensores biomédicos no suele ser alto.

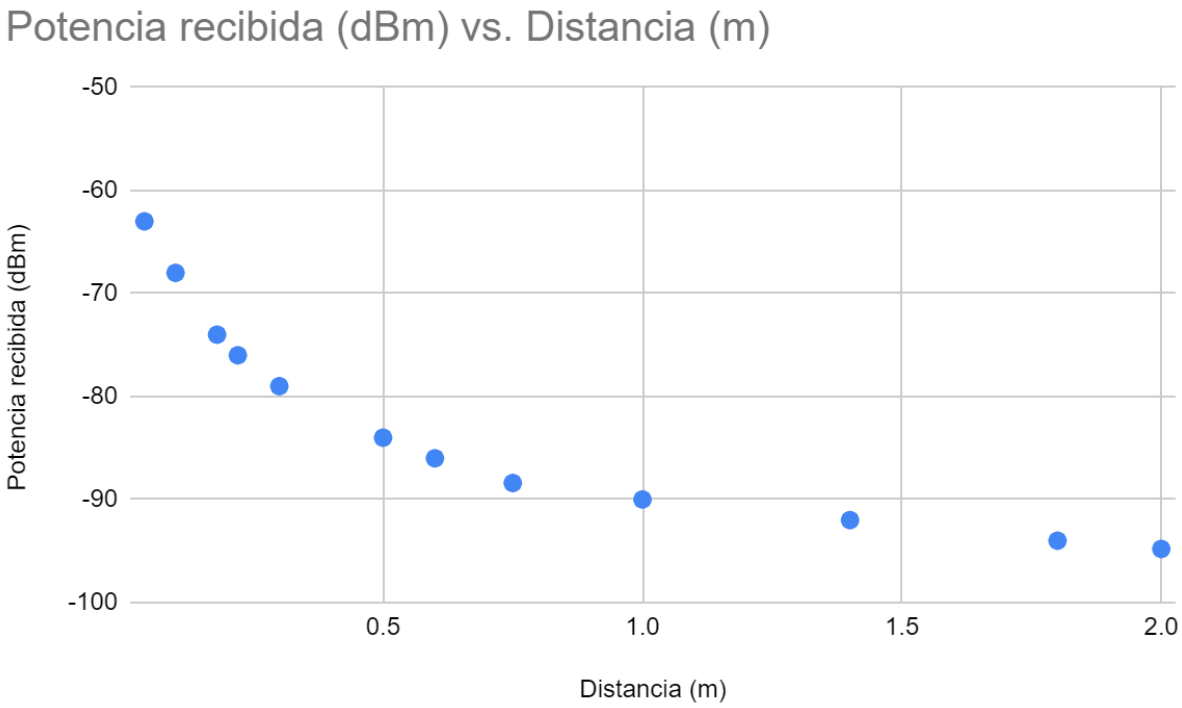

Figura 5-6. Potencia recibida frente a la distancia para -26 dBm TX.

Para el caso de la Figura 5-6, el desempeño es muy pobre. A una distancia de 2 metros, la comunicación es poco fiable, por lo tanto, esta potencia de transmisión está totalmente descartada.

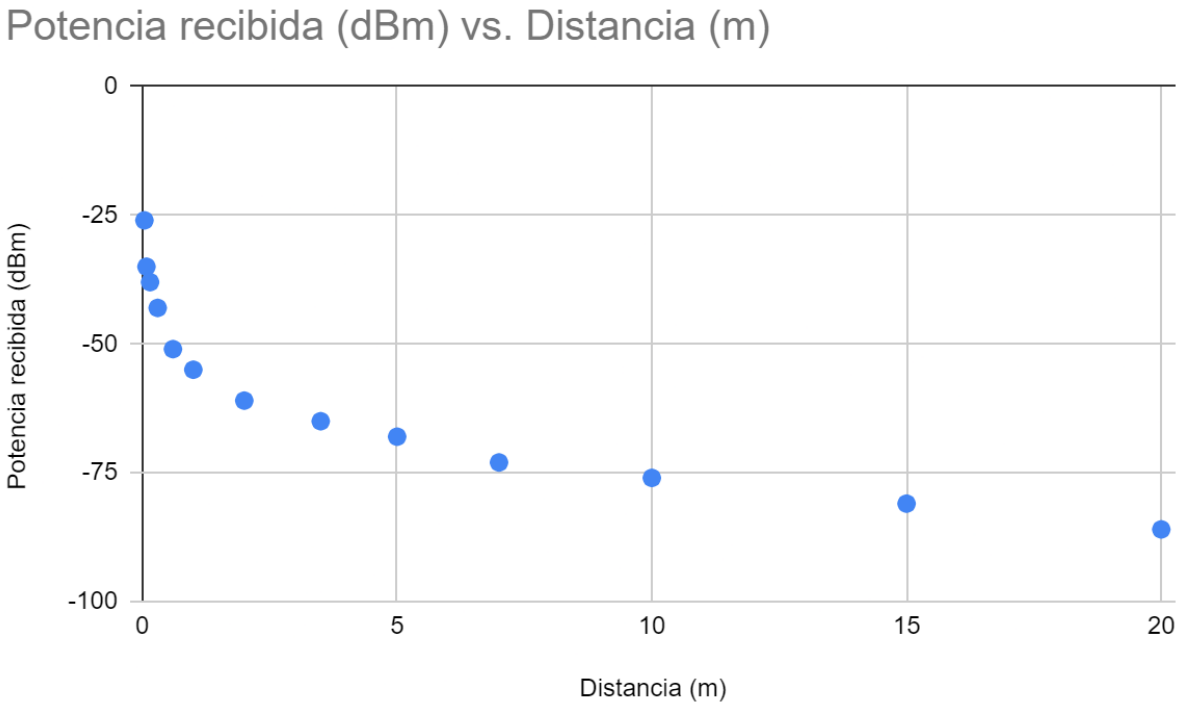

Figura 5-7. Potencia recibida frente a la distancia para 8 dBm TX.

En la Figura 5-7, se encuentra el caso para la mayor potencia de transmisión del dispositivo BLE de este trabajo y, como es lógico, los resultados de potencia recibida son los mejores. La transmisión es muy fiable a 20 metros de distancia e, incluso, se podría aumentar el rango a 30 o 40 metros.

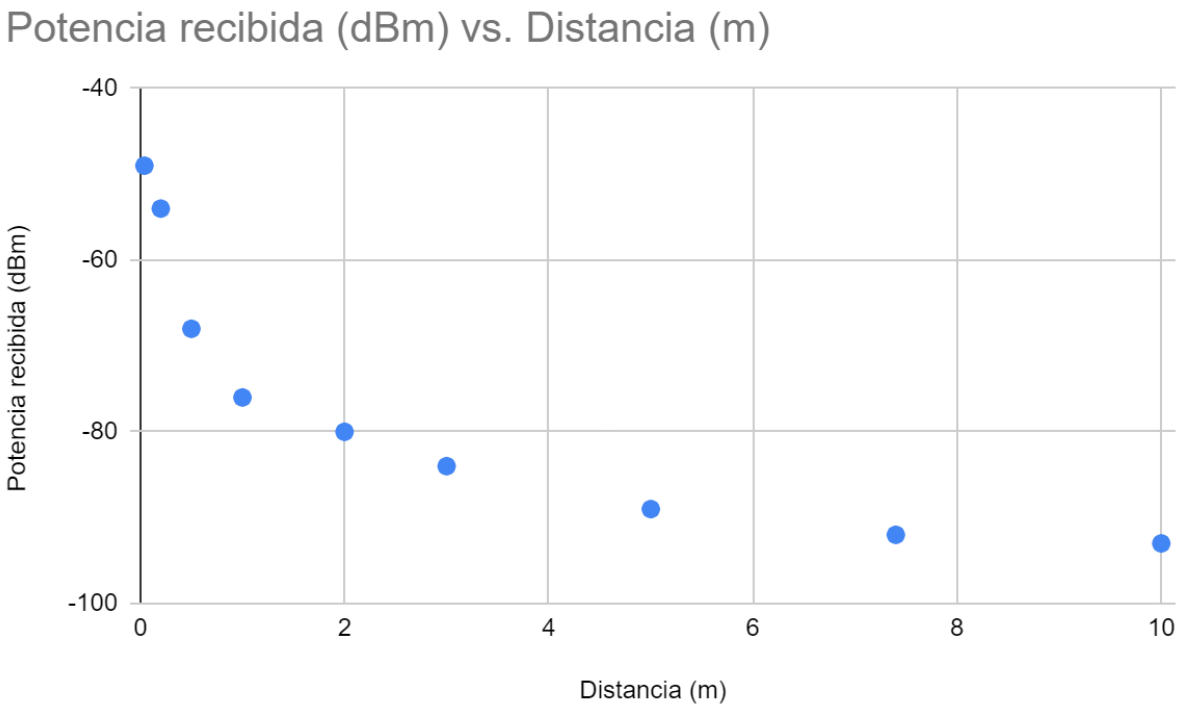

Figura 5-8. Potencia recibida frente a la distancia para -10 dBm TX.

El último caso explorado en este trabajo es el de la Figura 5-8, donde la potencia transmitida es -10 dBm. En este caso, la transmisión es fiable hasta los 5 metros y posible hasta los 10 o 15 metros. Este resultado es aceptable para las comunicaciones entre un sensor biomédico y un teléfono móvil, pero sería positivo obtener un resultado más favorable.

Como conclusión a este apartado, las potencias de transmisión superiores a 0 dBm son adecuadas para esta comunicación. También es necesario tener en cuenta que una mayor potencia de transmisión supone un mayor consumo de energía para el dispositivo BLE, por lo que, la potencia transmitida que se usa en el trabajo es 0 dBm.

5.2 Tamaño de los mensajes

El tamaño de los mensajes es la diferencia principal entre ambas implementaciones, enviar menos datos entraña una menor dificultad y supone menos errores. La diferencia de tamaño entre ambas implementaciones se muestra en la siguiente tabla.

Tabla 5-1. Tamaño de los mensajes.

Mensaje	Tamaño en bytes para forma directa	Tamaño en bytes para forma con XML
Envío de datos	540	134
Configuración	1146	326
Resto de mensajes	155	44

La Tabla 5-1 muestra el tamaño por mensaje de cada implementación. Se observa que la longitud de los mensajes usando XML es, aproximadamente, 4 veces menor que la longitud de forma directa. Los mensajes más grandes son el de configuración, por contener los atributos de los objetos y el de envío de datos, porque los valores observados y las fechas y horas ocupan muchos bytes. Se agrupan el resto de mensajes en el mismo campo porque no suponen un gran número de bytes.

Todos los mensajes suceden solo una vez por conexión BLE (y asociación X73) a excepción del envío de los datos, que puede suceder en varias ocasiones. Por lo general, un mismo usuario no va a realizar múltiples medidas de bioimpedancia en un corto espacio de tiempo, sin embargo, en sensores biomédicos utilizados por varios usuarios, sí tendría sentido varios mensajes de envíos de datos en una sola conexión.

5.3 Throughput

El régimen binario de BLE es 1 Mbps y, a partir de Bluetooth 5.0, 2 Mbps. En este trabajo, se ha usado la transmisión a 1 Mbps pese a que este dispositivo es capaz de enviar a 2 Mbps, ya que es el modo predeterminado y la velocidad no es un factor realmente importante en una medida de bioimpedancia.

A pesar de que el régimen binario sea 1 Mbps, la tasa de transferencia efectiva (throughput) es menor debido a la existencia de cabeceras y tiempos de guarda. En el caso de las notificaciones, método usado para el envío de datos de forma directa, el throughput máximo teórico es de 780 kbps [36] [37] [38]. Este cálculo se realiza con el número de bytes en la notificación, 251, menos 4 bytes de cabecera L2CAP, menos 3 bytes de cabecera GATT, lo que hace un total de 244 bytes. El número de bytes se divide entre 2500 us, provenientes de la suma de 2120 us de tiempo efectivo de transmisión, 150 us IFS (interframe space), 80 us de recepción y 150 us de otra IFS. Sin embargo, a pesar de que el throughput teórico sea 780 kbps, ninguna conexión BLE es capaz de alcanzarlo.

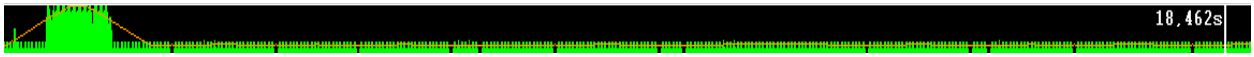

Figura 5-9. Paquetes enviados entre dispositivo y móvil.

En la Figura 5-9, se muestran los paquetes enviados entre el dispositivo BLE y el móvil con la herramienta Network Analyzer de Simplicity Studio. La captura de paquetes ocupa del segundo 2 al segundo 19, es decir, 17 segundos. Las marcas verdes corresponden con los paquetes enviados y la marca blanca es una notificación 244 bytes que se ha marcado. Donde se encuentra la mayor densidad de paquetes es alrededor de los 3 segundos, pues se estableció la conexión en ese instante. La mayoría de paquetes verdes son Empty PDU, paquetes sin información dedicados a expresar que la conexión sigue activa. Haciendo zoom en la notificación marcada de blanco, se aprecia la Figura 5-10.

Figura 5-10. Notificación de 244 bytes.

La transmisión de un paquete por medio de notificación de 244 bytes dura 8 milisegundos. Por lo que, el throughput usando notificaciones es de 244 kbps. Este es el método usado para la transmisión de forma directa debido a que se usan paquetes. Para el envío de XML, se usa el permiso de lectura en el perfil GATT.

Figura 5-11. Lectura de datos.

La lectura de datos, implica la existencia de ACK, es decir, reconocimiento de que ha llegado la información, así que supone un mayor tiempo de transmisión. En este caso, ocupa 50 ms, por lo que, si se escriben 134 bytes, el throughput para su lectura es de 21.44 kbps.

El throughput para el envío de datos usando XML es significativamente mayor, sin embargo, al enviar muchos menos bytes, eso hace que el tiempo de transmisión sea solamente ligeramente superior. De todas formas, el tiempo de transmisión no es algo fundamental para las comunicaciones en un sensor biomédico de bioimpedancia, aunque sí podría ser de mayor interés para otros sensores biomédicos.

5.4 Interoperabilidad

La interoperabilidad es el motivo central de la creación de la familia de estándares IEEE 11073, por lo que, resulta de gran importancia cuál de las dos implementaciones aporta una mayor facilidad para la interoperabilidad entre dispositivos. Cabe destacar que X73 y BLE son incompatibles, por lo tanto, la dificultad es mucho mayor. En este trabajo, se propone el envío de datos por el perfil GATT, usado por muchos fabricantes para el intercambio de información entre sensores biomédicos por BLE. Sin embargo, los fabricantes usan soluciones propietarias, este proyecto propone el uso de GATT mediante dos formas, de forma directa y usando XML.

La forma directa supone un entendimiento total y directo de los estándares IEEE 11073 en caso de recibir la información correctamente. Así que, lo necesario para fomentar la interoperabilidad entre todos los dispositivos es recibir los datos adecuadamente, para ello, deben reunirse los paquetes que han sufrido la segmentación. De esta manera, ya se pueden procesar los datos normalmente según X73.

Para el envío usando XML hay más complicaciones. El fichero XML usado debe ser entendido por ambas partes y, a día de hoy, no hay ninguna propuesta de este tipo de ficheros presente en los estándares. Hace más de una década se creó la librería Antidote de código abierto para el envío de datos X73 sobre Bluetooth clásico, pero aquello ha quedado desfasado. En este trabajo, se proponen varios ficheros XML, con el objetivo de que sirvan de referencia, para fomentar la interoperabilidad total entre dispositivos.

5.5 Dificultades

Las dificultades son mayúsculas en todo lo que respecta a X73. Son unos estándares complicados de entender y con muy poca información existente acerca de ellos. No existen implementaciones completas de IEEE 11073 en Internet y, mucho menos, si es sobre Bluetooth Low Energy, por lo que, este trabajo arroja luz sobre ese problema. Sin embargo, para cualquier implementación de X73, es necesario un conocimiento extenso de los estándares y solo usar de referencia otros trabajos, pues lo contrario puede acarrear multitud de equivocaciones y confusiones.

En cuanto a cada implementación de este trabajo, la que hace uso de XML requiere un esfuerzo menor, debido a que se transmite menos información y, en consecuencia, hay menos datos que adaptar y que conocer. Además, se evita mucha programación.

La implementación de forma directa es algo más compleja si se hace de forma sencilla. En este trabajo, se ha optado por la creación de una lista de configuración con los parámetros requeridos en X73 de forma que solo con la configuración de esos parámetros se pueda adaptar cualquier medida a X73. Esto ha complicado mucho

el trabajo debido a la necesidad de mucha programación para la generación de todos los datos de forma automática mediante el uso de bucles, sin embargo, ha resultado en un trabajo mucho más completo.

6 CONCLUSIONES

Vale más el fin de algo que su principio. Vale más la paciencia que la arrogancia.

- Eclesiastés 7:8-10 -

En este capítulo, se presentan las conclusiones que surgen de la implementación y el análisis de una medida de bioimpedancia siguiendo la familia de estándares IEEE 11073 y enviándose a través de Bluetooth Low Energy.

Debido a la incompatibilidad entre Bluetooth LE y X73, es necesaria la adaptación de sus protocolos para integrar estos estándares. Una debilidad de IEEE 11073 es la inexistencia de consideración del medio de transporte en sus estándares, por lo tanto, el envío de los datos X73 debe ser ideado y realizado por el usuario. Aparte de que la complejidad de ello es elevada, los usuarios pueden idear soluciones diferentes, frenando así el objetivo principal de IEEE 11073, la interoperabilidad. Como solución a este problema, sería adecuado la inclusión de, al menos, algunas consideraciones acerca de los medios de transporte más populares (Bluetooth Low Energy, Zigbee…) dentro de los estándares. En este trabajo, se han propuesto algunas posibles soluciones.

Otra conclusión principal del proyecto es que la complejidad de X73 limita su uso por los fabricantes, aunque en esa complejidad radica la cualidad diferencial de estos estándares. Los fabricantes pueden encontrarse abrumados ante las dificultades a la hora de comprender IEEE 11073 y, posteriormente, implementarlo con algún medio de transporte. Sin embargo, X73 aporta una enorme cantidad de información muy detallada acerca de la medida y de su contexto, lo que puede resultar verdaderamente útil para un análisis adecuado de la medida y de todos los factores asociados a la misma. Por lo tanto, al principio, la inclusión de IEEE 11073 por parte de los fabricantes en las comunicaciones en sensores biomédicos es complicado y costoso, pero el resultado será muy superior. En el presente trabajo, se ha demostrado experimentalmente que la integración de IEEE 11073 sobre BLE es posible siempre que disponga de un conocimiento extenso de los estándares.

En cuanto a los beneficios de cara al usuario, pueden ser mayúsculos de igual forma. La interoperabilidad entre dispositivos puede dar lugar a una comunicación efectiva entre sensores biomédicos y centros de salud a través del dispositivo móvil completamente estandarizada. A día de hoy, las comunicaciones entre sensores biomédicos y centros de salud son más limitadas, haciendo uso de soluciones propietarias del fabricante o usando otros estándares para el intercambio de información clínica que son mucho más limitados en cuanto a las posibilidades de los sensores biomédicos.

Como conclusión principal del presente trabajo, la implementación de la familia de estándares IEEE 11073, usando como transporte Bluetooth Low Energy, puede suponer una revolución en los sensores biomédicos, permitiendo un acceso a la información sencillo por parte de todos los dispositivos involucrados en la comunicación gracias a la interoperabilidad.

BIBLIOGRAFÍA

[1] Torab-Miandoab, A., Samad-Soltani, T., Jodati, A., Rezaei-Hachesu, P.:Interoperability of heterogeneous health information systems: a system-atic literature review. BMC Medical Informatics and Decision Making23(1), 18 (2023).

[2] Schwiebert, L., Gupta, S. K., & Weinmann, J. (2001, July). Research challenges in wireless networks of biomedical sensors. In Proceedings of the 7th annual international conference on Mobile computing and networking (pp. 151-165).

[3] "ISO/IEC/IEEE International Standard - Health informatics - Point-of-care medical device communication - Application profile - Base standard," in ISO/IEEE 11073-20101:2004(E) , vol., no., pp.1-92, 15 Dec. 2004.

[4] Caranguian LP, Pancho-Festin S, Sison LG. Device interoperability and authentication for telemedical appliance based on the ISO/IEEE 11073 Personal Health Device (PHD) Standards. Annu Int Conf IEEE Eng Med Biol Soc. 2012; 2012:1270-3.

[5] A. F. Harris III, V. Khanna, G. Tuncay, R. Want and R. Kravets, "Bluetooth Low Energy in Dense IoT Environments," in IEEE Communications Magazine, vol. 54, no. 12, pp. 30-36, December 2016.

[6] F. Laamarti, H. F. Badawi, Y. Ding, F. Arafsha, B. Hafidh and A. E. Saddik, "An ISO/IEEE 11073 Standardized Digital Twin Framework for Health and Well-Being in Smart Cities," in IEEE Access, vol. 8, pp. 105950-105961, 2020.

[7] H.-W. Kang, C.-M. Kim, S.-J. Koh. “ISO/IEEE 11073-based healthcare services over IoT platform using 6LoWPAN and BLE: Architecture and Experimentation”, IEEE Int. Conf. on Networking and App. 2016, pp. 313-318.

[8] M. Schmidt and R. Obermaisser, "Middleware for the integration of Bluetooth LE devices based on MQTT and ISO/IEEE 11073", 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), pp. 1-4, 2017.

[9] F. Battaglia, G. Gugliandolo, G. Campobello and N. Donato, "EEG-Over-BLE: A Low-Latency, Reliable, and Low-Power Architecture for Multichannel EEG Monitoring Systems," in IEEE Transactions on Instrumentation and Measurement, vol. 72, pp. 1-10, 2023.

[10] S.F. Khalil, M.S. Mohktar, F. Ibrahim, “The theory and fundamentals of bioimpedance analysis in clinical status monitoring and diagnosis of diseases,” Sensors, 14 (2014), pp. 10895-10928.

[11] Michel Y. Jaffrin, Hélène Morel. “Body fluid volumes measurements by impedance: A review of bioimpedance spectroscopy (BIS) and bioimpedance analysis (BIA) methods,” Medical Engineering & Physics, Volume 30, Issue 10, 2008, pp 1257-1269.

[12] Mendes Jr., J.J.A.; Vieira, M.E.M.; Pires, M.B.; Stevan Jr., S.L. Sensor Fusion and Smart Sensor in Sports and Biomedical Applications. Sensors 2016, 16, 1569.

[13] A. Molina Rivero, J. Reina Tosina, D. Naranjo Hernández. (2020). Diseño y Desarrollo Software para la Gestión de Sensores Biomédicos basado en la Norma X73. Universidad de Sevilla, Sevilla.

[14] "IEEE Standard for Health informatics--Point-of-care medical device communication - Part 10101: Nomenclature," in IEEE Std 11073-10101-2019 (Revision of ISO/IEEE 11073-10101:2004), vol., no., pp.1-1061, 9 Oct. 2019.

[15] "ISO/IEC/IEEE International Standard - Health informatics--Device interoperability--Part 10201: Point-of-care medical device communication--Domain information model," in ISO/IEEE 11073-10201:2020(E), vol., no., pp.1-182, 25 May 2020.

[16] "IEEE Health informatics--Personal health device communication - Part 20601: Application profile--Optimized Exchange Protocol," in IEEE Std 11073-20601-2019 (Revision of IEEE Std 11073-20601-2014), vol., no., pp.1-283, 20 Dec. 2019.

[17] "IEEE Standard - Health informatics--Personal health device communication Part 10404: Device specialization--Pulse oximeter," in IEEE Std 11073-10404-2020 (Revision of IEEE Std 11073-10404-2008) , vol., no., pp.1-83, 7 Jan. 2021.

[18] "IEEE Approved Draft Standard for Health Informatics--Personal Health Device Communication Part 10406: Device Specialization--Basic Electrocardiograph (ECG) (1- to 3-lead ECG)," in IEEE P11073-10406/D5, December 2022 , vol., no., pp.1-80, 22 Feb. 2023.

[19] "IEEE Health informatics--Personal health device communication Part 10407: Device specialization--Blood pressure monitor," in IEEE Std 11073-10407-2020 (Revision of IEEE Std 11073-10407-2008) , vol., no., pp.1-69, 30 April 2020.

[20] "Health informatics--Device interoperability - Part 10420: Personal health device communication--Device specialization--Body composition analyzer," in IEEE Std 11073-10420-2020 (Revision of IEEE Std 11073-10420-2010) , vol., no., pp.1-78, 17 Nov. 2020.

[21] "IEEE Health Informatics--Device Interoperability--Part 10471: Personal Health Device Communication--Device Specialization--Independent Living Activity Hub," in IEEE Std 11073-10471-2023 (Revision of IEEE Std 11073-10471-2008), vol., no., pp.1-124, 22 June 2023.

[22] Bluetooth Special Interest Group (SIG). "Bluetooth Core Specification." Version 5.3.

[23] C. Gómez, J. Oller, J. Paradells. Overview and evaluation of bluetooth low energy: An emerging low-power wireless technology. *sensors*, 2012, vol. 12, no 9, p. 11734-11753.

[24] Bulić, P.; Kojek, G.; Biasizzo, A. Data Transmission Efficiency in Bluetooth Low Energy Versions. Sensors 2019, 19, 3746.

[25] Mathworks, «Bluetooth Protocol Stack», 2021.

[26] Cristobal-Huerta, A.; Torrado-Carvajal, A.; Rodriguez-Sanchez, C.; Hernandez-Tamames, J.A.; Luaces, M.; Borromeo, S. Implementation of ISO/IEEE 11073 PHD SpO2 and ECG Device Specializations over Bluetooth HDP following Health Care Profile for Smart Living. Sensors 2022, 22, 5648.

[27] J. D. Trigo et al., "Interoperability in Digital Electrocardiography: Harmonization of ISO/IEEE x73-PHD and SCP-ECG," in IEEE Transactions on Information Technology in Biomedicine, vol. 14, no. 6, pp. 1303-1317, Nov. 2010.

[28] Chan-Yong Park, Joon-Ho Lim and Soojin Park, "ISO/IEEE 11073 PHD adapter board for standardization of legacy healthcare device," 2012 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, 2012, pp. 482-483.

[29] Hai-Long Li, Zhi-Bin Duan, Jin-Zhong Cui and Zhen-Wei Chen, "A design of general medical data adapter based on ISO/IEEE 11073 standards," 2015 12th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), Chengdu, China, 2015, pp. 404-407.

[30] S.M. Kuptsov, D.A. Raznometov, I.N. Korsakov , V.V. Feklistov , M.A. Sumskoy. On Demand Plugins for X73-PHD Manager , in: 2nd International Conference on Advanced Computing, Engineering and Technology, Malaysia, October 14-15, 2013. The World Academy Of Research In Science And Engineering, 2013. P. 35-40.

[31] Signove Tecnologia, Antidote, an open-source IEEE 11073-20601 stack, 2014.

[32] Bluetooth R SIG, GATT REST API Whitepaper, 2014.

[33] Bluetooth SIG, «Personal Health Devices Transcoding,» Bluetooth White Paper, 2015.

[34] Silicon Labs, Simplicity Studio 5.

[35] BGM220P Wireless Gecko Bluetooth Module Data Sheet

[36] Badawi, H.F.; Laamarti, F.; El Saddik, A. ISO/IEEE 11073 personal health device (X73-PHD) standards compliant systems: Asystematic literature review. IEEE Access 2018,7, 3062–3073.

[37] Huang ZY, Wang Y, Wang L. ISO/IEEE 11073 Treadmill Interoperability Framework and its Test Method: Design and Implementation. JMIR Med Inform. 2020 Dec 9;8(12):e22000.

[38] Gaoyang Shan, Sun-young Im and Byeong-hee Roh, "Optimal AdvInterval for BLE scanning in different number of BLE devices environment," 2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), San Francisco, CA, USA, 2016, pp. 1031-1032.

[39] Nordic Semiconductor, «Bluetooth Low Energy data throughput», 2021.

[40] Silicon Labs, «Throughput with Bluetooth Low Energy Technology», 2021.

[41] F. J. Dian, A. Yousefi and S. Lim, "A practical study on Bluetooth Low Energy (BLE) throughput," 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2018, pp. 768-771.

ANEXO A: LIBRERÍA IEEE 11073

Este anexo contiene el fichero 'biblio.h', contenedor de todos los identificadores X73 y sus valores. La versión en este anexo es una versión reducida debido a que la extensión de este fichero es gigantesca, por lo que, aquí solo se encuentran los valores que se usan en el trabajo, además de todas las especializaciones.

```
#ifndef BIBLIO_H
#define BIBLIO_H



#include <string.h>
#include <stdint.h>
#include <stddef.h>
#include <stdbool.h>
#include "sl_status.h"

/*    OBJETOS     */
#define MDC_PART_OBJ 1 /* Object Infrastr. */
#define MDC_MOC_VMO_METRIC 4 /* */
#define MDC_MOC_VMO_METRIC_ENUM 5 /* */
#define MDC_MOC_VMO_METRIC_NU 6 /* */
#define MDC_ATTR_ID_HANDLE 2337 /* */
#define MDC_ATTR_ID_TYPE 2351 /* */
#define MDC_ATTR_NU_CMPD_VAL_OBS 2379 /* */
#define MDC_ATTR_NU_VAL_OBS 2384 /* */
#define MDC_ATTR_SYS_ID 2436 /* */
#define MDC_ATTR_SYS_TYPE 2438 /* */
#define MDC_ATTR_TIME_ABS 2439 /* */
#define MDC_ATTR_TIME_STAMP_ABS 2448 /* */
#define MDC_ATTR_UNIT_CODE 2454 /* */
#define MDC_ATTR_VAL_ENUM_OBS 2462 /* */
#define MDC_ATTR_DEV_CONFIG_ID 2628 /* */
#define MDC_ATTR_MDS_TIME_INFO 2629 /* */
#define MDC_ATTR_METRIC_SPEC_SMALL 2630 /* */
#define MDC_ATTR_SOURCE_HANDLE_REF 2631 /* */
#define MDC_ATTR_SIMP_SA_OBS_VAL 2632 /* */
```

```
#define MDC_ATTR_ENUM_OBS_VAL_SIMP_OID 2633 /* */
```

```
#define MDC_ATTR_ENUM_OBS_VAL_SIMP_STR 2634 /* */
#define MDC_ATTR_SCAN_HANDLE_ATTR_VAL_MAP 2643 /* */
#define MDC_ATTR_ATTRIBUTE_VAL_MAP 2645 /* */
#define MDC_ATTR_NU_VAL_OBS_SIMP 2646 /* */
#define MDC_ATTR_SYS_TYPE_SPEC_LIST 2650 /* */
#define MDC_ATTR_METRIC_ID_PART 2655 /* */
#define MDC_ATTR_ENUM_OBS_VAL_PART 2656 /* */
#define MDC_ATTR_ENUM_OBS_VAL_SIMP_BIT_STR 2661 /* */
#define MDC_ATTR_ENUM_OBS_VAL_BASIC_BIT_STR 2662 /* */
#define MDC_ATTR_METRIC_STRUCT_SMALL 2675 /* */
#define MDC_ATTR_NU_CMPD_VAL_OBS_SIMP 2676 /* */
#define MDC_ATTR_NU_CMPD_VAL_OBS_BASIC 2677 /* */
#define MDC_ATTR_SOURCE_HANDLE_REF_LIST 2681 /* */

/*      NOTIFICACIONES      */
#define MDC_NOTI_CONFIG 3356 /* */
#define MDC_NOTI_SCAN_REPORT_FIXED 3357 /* */
#define MDC_NOTI_SCAN_REPORT_VAR 3358 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_VAR 3368 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_FIXED 3369 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_GROUPED 3370 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_MP_VAR 3371 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_MP_FIXED 3372 /* */
#define MDC_NOTI_BUF_SCAN_REPORT_MP_GROUPED 3373 /* */

/*      ESPECIALIDADES      */
#define MDC_DEV_SPEC_PROFILE_HYDRA 4096 /* Hydra device */
#define MDC_DEV_SPEC_PROFILE_PULS_OXIM 4100 /* Pulse oximeter */
#define MDC_DEV_SPEC_PROFILE_ECG 4102 /* Basic ECG */
#define MDC_DEV_SPEC_PROFILE_BP 4103 /* Blood pressure */
#define MDC_DEV_SPEC_PROFILE_TEMP 4104 /* Thermometer */
#define MDC_DEV_SPEC_PROFILE_SCALE 4111 /* Weighing scale */
#define MDC_DEV_SPEC_PROFILE_GLUCOSE 4113 /* Glucose meter */
#define MDC_DEV_SPEC_PROFILE_COAG 4114 /* International normalized ratio */
#define MDC_DEV_SPEC_PROFILE_INSULIN_PUMP 4115 /* Insulin pump */
```

```
#define MDC_DEV_SPEC_PROFILE_BCA 4116 /* Body composition nalyzer*/
```

```
#define MDC_DEV_SPEC_PROFILE_PEFM 4117 /* Peak expiratory flow monitor */
#define MDC_DEV_SPEC_PROFILE_URINE_ANALYZER 4118 /* Urine analyzer */
#define MDC_DEV_SPEC_PROFILE_SLEEP_QUALITY 4119 /* Sleep quality monitor */
#define MDC_DEV_SPEC_PROFILE_SABTE 4120 /* Sleep apnoea breathing therapy equipment */
#define MDC_DEV_SPEC_PROFILE_CGM 4121 /* Continuous glucose monitor */
#define MDC_DEV_SPEC_PROFILE_PSM 4124 /* Power status monitor */
#define MDC_DEV_SPEC_PROFILE_HF_CARDIO 4137 /* Cardiovascular fitness and activity monitor */
#define MDC_DEV_SPEC_PROFILE_HF_STRENGTH 4138 /* Strength fitness equipment */
#define MDC_DEV_SPEC_PROFILE_AI_ACTIVITY_HUB 4167 /* Independent living activity hub */
#define MDC_DEV_SPEC_PROFILE_AI_MED_MINDER 4168 /* Medication Monitor */
#define MDC_DEV_SPEC_PROFILE_GENERIC 4169 /* Generic device */

#define MDER "8000"
#define MDER_encoded "1235"

/*      APDUs       */
#define aarqAPDU  "E200"
#define rlrqAPDU  "E400"
#define prstAPDU  "E700"

/*      UNIDADES         */
#define MDC_DIM_DIMLESS          512
#define MDC_DIM_KILO_G           1731
#define MDC_DIM_CENTI_M          1297
#define MDC_DIM_TOD              2400 /*hh:mm:ss*/
#define MDC_DIM_DATE             2432 /*yyyy-mm-dd*/
#define MDC_X_KHZ               62100 /*Existe Hz pero no kHz*/
#define MDC_DIM_X_OHM            4288
#define MDC_DIM_ANG_DEG           736
#define MDC_DIM_NANO_F           62101 /*Existe Faradios pero no nF*/
#define MDC_DIM_NANO_S           62102 /*Existe segundos pero no ns*/
#define MDC_DIM_PERCENT           544
#define MDC_DIM_X_L              1600
#define MDC_DIM_KG_PER_M_SQ      1955
```

```
/*     ATRIBUTOS MEDIDOS     */
#define MDC_MASS_BODY_ACTUAL      57664
#define MDC_LEN_BODY_ACTUAL       57668
#define MDC_SEXO                  64011
#define MDC_FECHA_NACIMIENTO      64012
#define MDC_CONFIGURACION_BIOIMPEDANCIA          64029
#define MDC_LISTA_CONFIGURACIONES_BIOIMPEDANCIA   64030
#define MDC_MODELO_BIOIMPEDANCIA  64013
#define MDC_BCM                   57704
#define MDC_BCM_PORCENTAJE        64014
#define MDC_ECW                   64015
#define MDC_ICW                   64016
#define MDC_TBW                   64017
#define MDC_BODY_WATER            57692
#define MDC_MASS_BODY_FAT_FREE    57684
#define MDC_FFM_PORCENTAJE        64018
#define MDC_FFMI                  64019
#define MDC_BODY_FAT              57676
#define MDC_FM_PORCENTAJE         64020
#define MDC_FMI                   64021
#define MDC_RES_FREC_0            64022
#define MDC_RES_FREC_INF          64023
#define MDC_RES_EXTRA             64024
#define MDC_RES_INTRA             64025
#define MDC_BIOIMPED_CAPACIDAD    64026
#define MDC_BIOIMPED_ALPHA        64027
#define MDC_BIOIMPED_RETRASO      64028


#endif // BIBLIO_H
```

ANEXO B: FUNCIONES DE APOYO EN C

Debido a que algunos datos se definen como cadenas de caracteres, otros como enteros, otros como punto flotanto, otros como hexadecimal, entre otros, este anexo contiene las funciones para el intercambio entre tipos de datos en C.

B.1 Conversiones entre valores enteros y hexadecimales

Este primer apartado comienza con una función que recibe una cadena de caracteres que expresa un número hexadecimal y la convierte en un número entero.

```
unsigned int convertirHexAEntero(const char* hexString) {
    unsigned int numero = 0;
    int i;

    for (i = 0; i < strlen(hexString); i++) {
        char c = hexString[i];

        if (c >= '0' && c <= '9') {
            numero = (numero << 4) + (c - '0');
        } else if (c >= 'A' && c <= 'F') {
            numero = (numero << 4) + (c - 'A' + 10);
        } else if (c >= 'a' && c <= 'f') {
            numero = (numero << 4) + (c - 'a' + 10);
        }
    }

    return numero;
}
```

También contiene su función dual, el cambio de entero a una cadena de caracteres que representa un número hexadecimal.

```
void enteroAHexadecimal(int numero, char* hexString, int ancho) {
    const char* digitosHex = "0123456789ABCDEF";
    int indice = 0;

    if (numero == 0) {
```

```
    hexString[indice++] = '0';
  } else {
    if (numero < 0) {
      hexString[indice++] = '-';
      numero = -numero;
    }

    while (numero != 0) {
      int residuo = numero % 16;
      hexString[indice++] = digitosHex[residuo];
      numero = numero / 16;
    }
  }
```

Como apoyo a esta última función, se encuentra la siguiente función, que en caso de que el número hexadecimal sea de 2 bytes, lo separa en cadenas de caracteres de 2 dígitos cada una, es decir, 1 byte.

```
void dividirCadena(const char* cadena, char** parte1, char** parte2) {
  int longitud = strlen(cadena);

  // Asignar memoria para las partes
  *parte1 = (char*)malloc(3 * sizeof(char));
  *parte2 = (char*)malloc((longitud - 2 + 1) * sizeof(char));

  // Copiar los primeros dos caracteres a la parte1
  strncpy(*parte1, cadena, 2);
  (*parte1)[2] = '\0';

  // Copiar el resto de la cadena a la parte2
  strncpy(*parte2, cadena + 2, longitud - 2);
  (*parte2)[longitud - 2] = '\0';
}
```

B.2 Conversión a UTF-8

En este apartado, se presenta una función que convierte una cadena de caracteres en su valor UTF-8. Por ejemplo, “Hola” tiene una codificación UTF-8 {0x48, 0x6F, 0x6C, 0x61}.

```
void convertirUTF8(const char* cadena, uint8_t* vector) {
```

```
    size_t len = strlen(cadena);

    for (size_t i = 0; i < len; i++) {
        vector[i] = (uint8_t)cadena[i];
    }
}
```

B.3 Conversión de entero a hexadecimal especial

Esta conversión especial es la conversión que se usa para Time Stamp, en la que la hora 20:45:32 se codifica en hexadecimal como {0x20, 0x45, 0x32}.

```
uint8_t obtenerHexadecimal(int decimal) {
    int aux = decimal;
    for(int i = 10; i < 100; i+=10){
        if(aux>=i)
            decimal+=6;
    }
    return decimal;  // Valor predeterminado si no se encuentra una conversión
}
```

B.4 Conversión de hexadecimal a ByteArray

En este apartado, se convierte una cadena de caracteres que representa un número hexadecimal en un ByteArray, es decir, un vector de valores uint8_t (Byte).

```
void hexStringToByteArray(const char* hexString, uint8_t** byteArray, size_t* byteArrayLen) {
    size_t hexStringLength = strlen(hexString);
    size_t i = 0;

    if (hexStringLength % 2 != 0) {
        printf("Error: Longitud de cadena hexadecimal inválida.\n");
        *byteArray = NULL;
        *byteArrayLen = 0;
        return;
    }

    *byteArrayLen = hexStringLength / 2;
```

```
    *byteArray = (uint8_t*)malloc(*byteArrayLen * sizeof(uint8_t));

    if (*byteArray == NULL) {
        printf("Error: No se pudo asignar memoria para el arreglo.\n");
        *byteArrayLen = 0;
        return;
    }

    while (i < hexStringLength) {
        sscanf(hexString + i, "%2hhx", &(*byteArray)[i / 2]);
        i += 2;
    }
}
```

B.5 Conversión de punto flotante a FLOAT-Type

Este apartado es especialmente importante porque la mayoría de valores transmitidos se transmiten en formato FLOAT-Type y no es difícil de encontrar información acerca de FLOAT-Type, por lo que, las siguientes funciones son bastante útiles.

La primera función cuenta los decimales de un valor tipo punto flotante.

```
int cuentaDecimales(float number) {
    int cuenta = 0;
    while (number != (int)number) {
        number *= 10;
        cuenta++;
    }
    return cuenta;
}
```

La segunda función genera el exponente de FLOAT-Type, que son sus dos primeros Bytes.

```
uint32_t exponenteFloat(uint32_t x) {
  return 0xFF000000 - ((x - 1) << 24);
}
```

La tercera y principal función es la que, apoyándose de las dos funciones anteriores, convierte un valor de punto flotante a FLOAT-Type.

```
uint32_t convertToFloatType(float value) {
   // Obtenemos el valor entero de la parte decimal multiplicándolo por 10

   int decimales = countDecimals(value);
   if (decimales > 4)
     decimales = 4;
   int32_t decimalPart = (int32_t)(value * pow(10,decimales));

   uint32_t aux = (uint32_t) decimales;
   uint32_t exp = exponenteFloat(aux);

   // Construimos el valor uint32_t combinando los bytes
   uint32_t result = exp | (decimalPart);

   return result;
}
```

B.6 Estructura del usuario

Este apartado contiene el 'struct' que adapta la medida de bioimpedancia a C para el usuario, pero primera hay que comenzar con la estructura de la fecha de nacimiento, que está contenida en la estructura de los usuarios.

```
typedef struct FechaNacimiento {
  int day;
  int month;
  int year;
} FechaNacimiento;
```

A continuación, se encuentra la estructura del usuario:

```
typedef struct usuario {
  float peso;
  float estatura;
  float ffm;
  float fm;
  int sexo;
  struct FechaNacimiento fecha;
  float bcm;
```

```
  float bcmperc;
  float ecw;
  float icw;
  float tbw;
  float ffmperc;
  float ffmi;
  float fmperc;
  float fmi;
  float res_0;
  float res_inf;
  float res_extra;
  float res_intra;
  float capacidad;
  float alpha;
  float retraso;
  float frecuencia1;
  float modulo1;
  float fase1;
  float frecuencia2;
  float modulo2;
  float fase2;
  float frecuencia3;
  float modulo3;
  float fase3;
  int numfrecuencias;
} usuario;
```

ANEXO C: MENSAJES X73 PARA LA MEDIDA DE BIOIMPEDANCIA

Este anexo sirve para aliviar el contenido del capítulo 4 debido a su extensión. A continuación, se van a ofrecer los mensajes entre agente y gestor X73 para la medida de bioimpedancia de este trabajo siguiendo el formato de los estándares IEEE 11073.

C.1 Asociación de los dispositivos

La asociación entre los dispositivos está compuesta por dos mensajes, el agente X73 enviando la petición y el gestor X73 enviando la respuesta a la petición.

C.1.1 Petición de asociación

El formato es los Bytes transmitidos en orden están a la izquierda y la información de lo que significan esos Bytes a la derecha. Algunos de los atributos presentes se mencionan y explican en el trabajo y otros no, debido a que generaría una extensión enorme y tampoco aportan lo suficiente.

Este mensaje contiene la petición del agente informando de que tiene extended configuration.

0xE2 0x00	APDU de petición (AarqAPDU)
0x00 0x32	Longitud del mensaje = 50 Bytes
0x80 0x00 0x00 0x00	Versión 1
0x00 0x01 0x00 0x2A	data-proto-list.count = 1 | Longitud = 42
0x50 0x79	data-proto-id = 20601
0x00 0x26	data-proto-info longitud = 38
0x70 0x00 0x00 0x00	Versión del protocolo (soporta V2, V3 y V4)
0x80 0x00	Reglas de codificación = MDER
0xE0 0x00 0x00 0x00	Versión de la nomenclatura (soporta v1,v2 y v3)
0x00 0x00 0x00 0x00	functionalUnits – No puede realizar tests de asociación
0x00 0x80 0x00 0x00	systemType = sys-type-agent (agente)
0x00 0x08	system-id longitud = 8 y valor (fabricante y modelo)
0x12 0x34 0x56 0x78 0x90 0xAB 0xCD 0xEF	(valor inventado porque no hay sensor biomédico)
0x40 0x00	dev-config-id – extended configuration
0x00 0x01	data-req-mode-flags
0x01 0x00	data-req-init-agent-count, data-req-init-manager-count
0x00 0x00 0x00 0x00	optionList.count = 0 | optionList.length = 0

C.1.2 Respuesta a la petición de asociación

La respuesta del gestor X73 es que acepta la asociación, pero debe pasar por la etapa de configuración, ya que hay una configuración desconocida.

0xE3 0x00	APDU de respuesta (AareApdu)
0x00 0x2C	Longitud del mensaje = 44 Bytes
0x00 0x03	Resultado = Asociación aceptada, configuración desconocida
0x50 0x79	data-proto-id = 20601
0x00 0x26	data-proto-info longitud = 38
0x10 0x00 0x00 0x00	Versión del protocolo (elige V4)
0x80 0x00	Reglas de codificación = MDER
0x20 0x00 0x00 0x00	nomenclatureVersion(v3)
0x00 0x00 0x00 0x00	functionalUnits – Asociación normal
0x80 0x00 0x00 0x00	systemType = sys-type-manager (gestor)
0x00 0x08	system-id length = 8 y valor (fabricante y modelo)
0x88 0x77 0x66 0x55 0x44 0x33 0x22 0x11	(valor aleatorio debido a que no es importante en este caso)
0x00 0x00	La respuesta del gestor a config-id es siempre 0
0x00 0x00	La respuesta del gestor a data-req-mode-flags es siempre 0
0x00 0x00	Es siempre 0
0x00 0x00 0x00 0x00	optionList.count = 0 | optionList.length = 0

Si la configuración hubiera sido conocida, el único cambio sería que el campo resultado valdría solo asociación aceptada y equivaldría a {0x00, 0x00}.

C.2 Configuración

Este apartado solo aparece si la configuración es desconocida.

C.2.1 Mensaje de configuración del agente

0xE7 0x00	APDU de presentación (PrstApdu)
0x01 0x78	Longitud del mensaje = 376 Bytes
0x01 0x76	Longitud = 374 Bytes
0x12 0x35	invoke-id = 0x1235 (Comienzo de los datos)
0x01 0x01	(Configuración | Evento confirmado)
0x01 0x70	CHOICE.length =368
0x00 0x00	obj-handle = 0 (Objeto MDS)
0xFF 0xFF 0xFF 0xFF	event-time = 0xFFFFFFFF

0x0D 0x1C	event-type = MDC_NOTI_CONFIG (Evento de configuración)
0x01 0x66	event-info.length = 358 (Comienzo del evento)
0x40 0x00	config-report-id (Configuración)
0x00 0x1A	config-obj-list.count = 26 objetos medidos
0x01 0x60	config-obj-list.length = 352
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x01	obj-handle = 1 (El primer objeto es el peso)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xE1 0x40	MDC_PART_SCADA | MDC_MASS_BODY_ACTUAL
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0xC3	MDC_DIM_KILO_G
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x02	obj-handle = 2 (El segundo objeto es la estatura)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xE1 0x44	MDC_PART_SCADA | MDC_LEN_BODY_ACTUAL
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x48	intermittent, stored data, upd & msmt aperiodic, agent init, manual
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x05 0x11	MDC_DIM_CENTI_M
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP

0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_SIMP, 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS, 8
0x00 0x05	obj-class = MDC_MOC_VMO_METRIC_ENUM
0x00 0x03	obj-handle = 3 (El tercer objeto es el sexo)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x0B	MDC_PART_SCADA | MDC_SEXO
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x48	intermittent, stored data, upd & msmt aperiodic, agent init, manual
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x02 0x00	MDC_DIM_DIMLESS
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x02	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 2
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x04	obj-handle = 4 (El cuarto objeto es la fecha de nacimiento)
0x00 0x04	attributes.count = 5
0x00 0x24	attributes.length = 76
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x0C	MDC_PART_SCADA | MDC_FECHA_NACIMIENTO
0x0A 0x73	attribute-id = MDC_ATTR_METRIC_STRUCT_SMALL
0x00 0x02	attribute-value.length = 2
0x00 0x01	ms-struct-compound
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x48	intermittent, stored data, upd & msmt aperiodic, agent init, manual
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE

0x00 0x02	attribute-value.length = 2
0x05 0x11	MDC_DIM_DATE
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_CMPD_VAL_OBS_SIMP, Longitud = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS, 8
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_CMPD_VAL_OBS_SIMP, Longitud = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS, 8
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_CMPD_VAL_OBS_SIMP, Longitud = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS, 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x05	obj-handle = 5 (El quinto objeto es la masa celular corporal (BCM))
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0XE1 0x68	MDC_PART_SCADA | MDC_BCM
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0xC3	MDC_DIM_KILO_G
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4

0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x06	obj-handle = 6 (El sexto objeto es %BCM)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x0D	MDC_PART_SCADA | MDC_BCM_PORCENTAJE
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x02 0x20	MDC_DIM_PERCENT
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x07	obj-handle = 7 (El séptimo objeto es el agua extracelular (ECW))
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x0E	MDC_PART_SCADA | MDC_ECW
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0x40	MDC_DIM_X_L
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4

0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x08	obj-handle = 8 (El octavo objeto es el agua intracelular (ICW))
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x0F	MDC_PART_SCADA | MDC_ICW
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0x40	MDC_DIM_X_L
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x09	obj-handle = 9 (El noveno objeto es el agua total (TBW))
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x10	MDC_PART_SCADA | MDC_TBW
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0x40	MDC_DIM_X_L
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4

0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x0A	obj-handle = 10 (El décimo objeto es FFM)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xE1 0x54	MDC_PART_SCADA | MDC_MASS_BODY_FAT_FREE
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0xC3	MDC_DIM_KILO_G
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x0B	obj-handle = 11 (El undécimo objeto %FFM)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x12	MDC_PART_SCADA | MDC_FFM_PORCENTAJE
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x02 0x20	MDC_DIM_PERCENT
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4

0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x0C	obj-handle = 12 (El duodécimo objeto es FFMI)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xFA 0x13	MDC_PART_SCADA | MDC_FFMI
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x07 0xA3	MDC_DIM_KG_PER_M_SQ
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4
0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8
0x00 0x06	obj-class = MDC_MOC_VMO_METRIC_NU
0x00 0x0D	obj-handle = 13 (El decimotercer objeto es FM)
0x00 0x04	attributes.count = 4
0x00 0x24	attributes.length = 36
0x09 0x2F	attribute-id = MDC_ATTR_ID_TYPE
0x00 0x04	attribute-value.length = 4
0x00 0x02 0xE1 0x4C	MDC_PART_SCADA | MDC_BODY_FAT
0x0A 0x46	attribute-id = MDC_ATTR_METRIC_SPEC_SMALL
0x00 0x02	attribute-value.length = 2
0xF0 0x42	intermittent, stored data, upd & msmt aperiodic, agent init, calculated
0x09 0x96	attribute-id = MDC_ATTR_UNIT_CODE
0x00 0x02	attribute-value.length = 2
0x06 0xC3	MDC_DIM_KILO_G
0x0A 0x55	attribute-id = MDC_ATTR_ATTRIBUTE_VAL_MAP
0x00 0x0C	attribute-value.length = 12
0x00 0x02	AttrValMap.count = 2
0x00 0x08	AttrValMap.length = 8
0x0A 0x56 0x00 0x04	MDC_ATTR_NU_VAL_OBS_ SIMP | Longitud del valor = 4

0x09 0x90 0x00 0x08	MDC_ATTR_TIME_STAMP_ABS| Longitud del valor = 8

Esta es la configuración de los primeros trece objetos, los siguientes trece objetos poseen una configuración similar.

C.2.2 Respuesta a la configuración por parte del gestor

0xE7 0x00	APDU de presentación (PrstApdu)
0x00 0x16	Longitud del mensaje = 22 Bytes
0x00 0x14	Longitud = 20 Bytes
0x12 0x35	invoke-id = 0x1235 (El mismo que antes)
0x02 0x01	(Respuesta remota | Evento confirmado)
0x00 0x0E	CHOICE.length =14
0x00 0x00	obj-handle = 0 (Objeto MDS)
0x00 0x00 0x00 0x00	currentTime = 0
0x0D 0x1C	Tipo de evento = MDC_NOTI_CONFIG
0x00 0x04	event-reply-info.length = 4
0x40 0x00	ConfigReportRsp.config-report-id = 0x4000
0x00 0x00	ConfigReportRsp.config-result = Configuración aceptada

C.3 Atributos MDS

Este apartado comparte los atributos del objeto MDS.

C.3.1 Petición de los atributos MDS

0xE7 0x00	APDU de presentación (PrstApdu)
0x00 0x0E	Longitud del mensaje = 14 bytes
0x00 0x0C	Longitud = 12 bytes
0x00 0x03	invoke-id = 3
0x01 0x03	CHOICE (Invocación remota | Get)
0x00 0x06	CHOICE.length = 6
0x00 0x00	handle = 0 (Objeto MDS)
0x00 0x00	attribute-id-list.count = 0 (todos los atributos)
0x00 0x00	attribute-id-list.length = 0

C.3.2 Respuesta con los atributos MDS

0xE7 0x00 APDU CHOICE Type (PrstApdu)
0x00 0x5C CHOICE.length = 92
0x00 0x5A OCTET STRING.length = 90
0x12 0x34 invoke-id = 0x1234 (El mismo que la petición)
0x02 0x03 CHOICE (Respuesta a la operación remota | Get)
0x00 0x54 CHOICE.length = 84
0x00 0x00 handle = 0 (MDS object)
0x00 0x05 attribute-list.count = 5
0x00 0x4E attribute-list.length = 78
0x0A 0x5A attribute id = MDC_ATTR_SYS_TYPE_SPEC_LIST
0x00 0x08 attribute-value.length = 8
0x00 0x01 TypeVerList count = 1
0x00 0x04 TypeVerList length = 4
0x10 0x14 type = MDC_DEV_SPEC_PROFILE_BCA
0x00 0x02 version = version 2 of the specialization
0x09 0x28 attribute-id = MDC_ATTR_ID_MODEL
0x00 0x18 attribute-value.length = 24
0x00 0x0E 0x42 0x69 0x6F 0x69 0x6D 0x70 0x65 0x64 0x61 0x6E 0x63 0x69 0x61 0x73
string length = 14 | “Bioimpedancias”
0x00 0x0A 0x46 0x61 0x62 0x72 0x69 0x63 0x61 0x6E 0x74 0x65
string length = 10 | “Fabricante”
0x09 0x84 attribute-id = MDC_ATTR_SYS_ID
0x00 0x0A attribute-value.length = 10
0x00 0x08 0x12 0x34 0x56 0x78 0x90 0xAB 0xCD 0xEF octet string length = 8 | EUI-64
0x0A 0x44 attribute-id = MDC_ATTR_DEV_CONFIG_ID
0x00 0x02 attribute-value.length = 2
0x40 0x00 dev-config-id = 16384 (extended-config-start)
0x09 0x90 attribute-id =MDC_ATTR_TIME_STAMP_ABS
0x00 0x08 attribute-value.length = 8
0x20 0x23 0x06 0x16 0x20 0x55 0x44 0x11 Absolute-Time-Stamp = 2023-06-16 T20:55:44.11

C.4 Envío de los datos

Este apartado transmite los valores medidos por el sensor biomédico.

C.4.1 Transmisión de los datos por parte del agente

0xE7 0x00	APDU de presentación (PrstApdu)
0x02 0x18	CHOICE.length = 536
0x02 0x16	OCTET STRING.length = 534
0x12 0x36	invoke-id = 0x1236
0x01 0x01	CHOICE(Remote Operation Invoke | Confirmed Event Report)
0x02 0x10	CHOICE.length = 528
0x00 0x00	obj-handle = 0 (Objeto MDS)
0xFF 0xFF 0xFF 0xFF	event-time = 0xFFFFFFFF
0x0D 0x1D	event-type = MDC_NOTI_SCAN_REPORT_FIXED
0x00 0x36	event-info.length = 54
0xF0 0x00	ScanReportInfoFixed.data-req-id = 0xF000
0x00 0x00	ScanReportInfoFixed.scan-report-no = 0
0x00 0x02	ScanReportInfoFixed.obs-scan-fixed.count = 3
0x00 0x2E	ScanReportInfoFixed.obs-scan-fixed.length = 46
0x00 0x03	Handle del primer objeto = 1
0x00 0x0A	Tamaño = 12 bytes
0xFF 0x00 0x01 0x1B	Simple-Nu-Observed-Value = 59.7 (kg)
0x20 0x23 0x06 0x16 0x20 0x55 0x44 0x11	Absolute-Time-Stamp = 2023-06-16 T20:55:44.11

Se presenta solamente el primer objeto porque los demás ocuparían espacio innecesario, ya que el primer objeto es representativo de todos los demás.

C.4.2 Respuesta del gestor

0xE7 0x00	APDU de presentación (PrstApdu)
0x00 0x12	Longitud del mensaje = 18 bytes
0x00 0x10	Longitud = 16 bytes
0x00 0x04	invoke-id = 4 (El mismo que el anterior)
0x02 0x01	Respuesta de operación remota | Evento Confirmado
0x00 0x0A	Longitud = 10 bytes
0x00 0x00	obj-handle = 0 (Objeto MDS)
0x00 0x00 0x00 0x00	currentTime = 0
0x0D 0x1D	Tipo de evento = MDC_NOTI_SCAN_REPORT_FIXED
0x00 0x00	event-reply-info.length = 0

C.5 Desasociación

Este apartado contiene los mensajes para la desasociación entre los dispositivos.

C.5.1 Petición de desasociación

0xE4 0x00	APDU de petición de desasociación (RlrqApdu)
0x00 0x02	Longitud del mensaje = 2 bytes
0x00 0x00	reason = normal (razón normal)

C.5.2 Respuesta a la petición de desasociación

0xE5 0x00	APDU de respuesta a la desasociación (RlreApdu)
0x00 0x02	Longitud del mensaje = 2 bytes
0x00 0x00	reason = normal (razón normal)