**E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación**

DISEÑO Y ESTUDIO DE UN SISTEMA DE COMUNICACIÓN INALÁMBRICO BASADO EN TECNOLOGÍA BLUETOOTH LOW ENERGY CON DESARROLLO DE PROTOCOLO PROPIO DE ENRUTAMIENTO.

Grado en Ingeniería
en Tecnologías de Telecomunicación

Trabajo Fin de Grado

Luis Garijo Gutierrez

Francisco Falcone Lanas

Pamplona, 30/06/2016

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Resumen

El proyecto consiste en desarrollar un protocolo propio de encaminamiento de paquetes sobre la tecnología inalámbrica *Bluetooth Low Energy*. Este protocolo basará su comportamiento en el de una red inalámbrica de topología *mesh* (malla) en la que no hace falta que todos los nodos estén en el rango de cobertura del punto de acceso de la red o entre sí para comunicarse y transmitirse información.

Se estudiará el funcionamiento y la estructura de la tecnología inalámbrica *Bluetooth Low Energy* con el objetivo de comprenderla con la profundidad necesaria para abordar el proyecto con garantías. Así mismo se desarrollarán perfiles *GAP* y *GATT* de conexión propios, para gestionar los mensajes tanto de anuncio como de datos, las conexiones y el resto de parámetros necesarios.

PALABRAS CLAVE

- **Mesh**
- **Bluetooth Low Energy**
- **Enrutamiento**
- **CC2541**

Summary

The project consists of developing a proprietary protocol packet routing on the Bluetooth Low Energy wireless technology. This protocol will base it is behavior on a wireless mesh network topology which does not need all nodes thath are in the coverage range of the network access point or with each other to communicate and transmit information.

We will study the functioning and structure of the Bluetooth Low Energy wireless technology with the aim of understanding with the depth needed to tackle the project with guarantees. Likewise GATT profiles GAP and developed own connection to manage both messages and data, connections and other required parameters.

KEYWORDS

- **Mesh**
- **Bluetooth Low Energy**
- **Routing**
- **CC2541**

ÍNDICE

1 INTRODUCCIÓN

En los últimos 10 años las tecnologías inalámbricas han ido creciendo y ganándole el terreno a las tecnologías alámbricas. Un ejemplo sencillo, en los hogares el teléfono fijo se está viendo desplazado por el teléfono inalámbrico. Cada día que pasa las personas tienen más necesidad de estar conectados constantemente con el mundo que les rodea y, como las personas están en continuo movimiento, las tecnologías cableadas no pueden satisfacer estas necesidades. El tamaño y la autonomía de los dispositivos también son muy importantes y no hay que perder de vista el coste final del producto. Las diferentes tecnologías inalámbricas se han ido adaptando a estas demandas. El crecimiento ha sido exponencial, la investigación y el desarrollo ha conllevado una disminución del coste de producción facilitando el acceso final de los usuarios.

Actualmente los teléfonos inteligentes son verdaderos ordenadores portátiles y su capacidad de procesamiento y autonomía ha generado un crecimiento de las redes inalámbricas de área personal (WPAN). Alrededor del teléfono inteligente, actuando a modo de nodo central, se han desarrollado multitud de dispositivos: relojes inteligentes, cascos inalámbricos, podómetros inalámbricos, medidores de la actividad física, etc. Estos dispositivos deben ser pequeños, baratos y con una cierta autonomía. También el hogar se ha convertido en un entorno con el que interactuar de forma inteligente: neveras que te hacen la compra, puertas que te avisan que están abiertas, luces, ventanas y persianas controladas a remoto, etc. Pero para que esa nevera inteligente pueda hacer la compra debe tener acceso a internet, es aquí donde surge el concepto de IoT, el internet de las cosas. Los dispositivos ya no sólo se conectan con otros que estén en su rango de cobertura, si no que a través de internet se pueden conectar con cualquier otro que se encuentre en cualquier otra parte. Esto abre las puertas a dispositivos más inteligentes e independientes, con capacidad para tomar decisiones en función de parámetros consultados a servidores o a otros dispositivos.

El internet de las cosas no solo incumbe al nivel de usuario, su implantación va más allá. Ciudades inteligentes (*smart cities*), eficientes y capaces de auto gestionarse. Por ejemplo: muchas veces esperamos ante un semáforo en rojo de forma totalmente absurda, pues no circula ningún coche ni ninguna persona a nuestro alrededor. Gracias al internet de las cosas, estos semáforos pueden conectarse a un circuito de cámaras distribuidas por la ciudad que identifican el nivel de tráfico y de movimiento de masas, evitando así esas absurdas esperas en las zonas de escaso movimiento.

Si llevamos el internet de las cosas a terrenos más amplios como la seguridad nacional o las empresas, la trascendencia y las posibilidades son aún mayores. Por ejemplo: huertos automatizados, alumbrados inteligentes, supervisión de máquinas…

Es en el ámbito de las ciudades inteligentes y en el ámbito de las empresas donde se está llevando a cabo una implantación, lenta, pero real del internet de las cosas. No son solamente redes inalámbricas de área personal, sino que entran en juego también redes de área extensa. Tecnologías como LoRa, una red inalámbrica de baja potencia y de área extensa (LPWAN), especificación pensada para dispositivos que operen con batería en una red regional, nacional o global. Este estándar facilita la interoperabilidad sin fisuras entre los dispositivos inteligentes sin necesidad de instalaciones complejas dando libertad al usuario, al desarrollador y a las empresas en el despliegue del internet de las cosas [1]. Se pueden desplegar de forma rápida y sencilla,

por ejemplo, redes de sensores para controlar farolas, contenedores o semáforos sin tener que desplegar nodos centrales o puntos de acceso propios para recopilar los datos, sino que se enviarán a través de LoRa.

Existen empresas encargadas de diseñar soluciones completas: hardware, software y gestión de datos. Una de ellas la española Libelium [2] que posee una larga lista de productos enfocados a las ciudades inteligentes: aparcamientos inteligentes, salud de las estructuras, mapas de ruido urbanos, detección de teléfonos inteligentes, medición de niveles de los campos electromagnéticos, congestión del tráfico, alumbrado inteligente, gestión de residuos o carreteras inteligentes. También tienen productos para medioambiente inteligente: detección de fuegos en bosques, contaminación del aire, prevención de avalanchas o detección de terremotos. Otro ejemplo de empresa dedicada al sector del IoT es la navarra Embeblue [3] que diseña y desarrolla hardware a medida de una forma ágil y flexible, una forma de enfocar la creación de hardware para el internet de las cosas.

La tecnología *Bluetooth* está presente en la gran mayoría de dispositivos como *smartphones, tablets* y PCs, con cerca de tres billones de dispositivos vendidos anualmente. *BLE* quiere ser la tecnología sobre la que se vertebre el internet de las cosas, IoT. El *Bluetooth Special Interest Group* (SIG), organización encargada de regular y estandarizar la tecnología *Bluetooth*, es consciente de que las limitaciones en el rango de cobertura de las redes inalámbricas es una barrera para el IoT, barrera que tecnologías como ZigBee rompen con la capacidad de trabajar en redes en malla, en la que dos nodos fuera de rango de cobertura pueden comunicarse entre sí a través de otros nodos intermedios. Este tipo de redes permiten además que se puedan añadir y eliminar nodos sin que ello afecte al funcionamiento de la red, simplificando su despliegue.

"Bluetooth was initially complementary to platforms like WiFi and ZigBee, but Bluetooth is starting to pick up features found in these other platforms. For example, one of ZigBee's advantages has been its mesh network support." [4]

SIG ha creado un grupo de trabajo para investigar y crear un estándar propio de red en malla y se espera que para finales de 2016 esté concluido. Esto junto al nuevo perfil IPSP (*Internet Protocol Support Profile)*, introducido en la versión 4.2 de las especificaciones Bluetooth, que posibilita el acceso a internet directamente a través de *IPv6/6LoWPAN*, abrirá una vía de trabajo para los desarrolladores y proveedores de soluciones para el IoT [5].

2 OBJETIVO

El objetivo es diseñar y desarrollar un protocolo propio de enrutamiento similar al comportamiento de una red con topología en malla (*mesh*) y se basará en tecnología Bluetooth Low Energy. El protocolo contendrá un nodo actuando como raíz, encargado de centralizar la recepción de los datos. Este nodo podría estar conectado a un host al que pasarle la información recibida. Podemos asumir que es una red *ad hoc* en la que cada nodo participa en el encaminamiento mediante el envío de los datos para otros nodos, hacía que nodo se envían los datos se calcula dinámicamente en función de la conectividad de la red. Esto permitirá que los nodos puedan abandonar o unirse a la red sin que afecte a la transmisión de los datos hacía el nodo central. Por tanto, el protocolo admite nodos en movimiento.

Para comprobar el protocolo de enrutamiento se plantea el objetivo secundario de desplegar una pequeña red compuesta por una serie de *gadgets* enchufables a la corriente eléctrica o bien alimentados por batería, en función del requerimiento posicional. Nodos emisores de información hacia un nodo central conectado inalámbricamente a un *host* encargado de procesar la información.

En principio, los *gadgets* estarán formados por módulos *BLE* HM-10 basado en el chip de *Texas Instruments* CC2541. Como IDE se utilizará *IAR Embedded Workbench*. *Texas Instruments* proporciona la pila de protocolos con varios proyectos de ejemplo: *Glucose sensor, Heart rate, Running sensor, etc…,* y otros con los roles GAP (*Generic Acces Profile)*: *SimpleBLEBroadcaster, SimpleBLECentral, SimpleBLEOberserver* y *SimpleBLEPeripheral*. Estos proyectos proporcionan a los desarrolladores una base para comenzar su trabajo, pero no todas las capas de las que se compone la pila son editables ni están accesibles. Partiremos de estos proyectos de ejemplo para desarrollar nuestros perfiles.

Se estudiará el funcionamiento y la estructura de la tecnología inalámbrica *Bluetooth Low Energy* con el objetivo de comprenderla con la profundidad necesaria para abordar el proyecto con garantías. Este estudio pretende conocer el alcance y las posibilidades futuras que ofrece la tecnología BLE.

3 ESTADO DEL ARTE

3.1 ¿QUÉ ES BLUETOOTH LOW ENERGY?

Bluetooth Low Energy (de aquí en adelante BLE) es la característica principal de la versión 4.0 del núcleo de especificaciones Bluetooth. Esta versión fue introducida en junio de 2010 por el grupo especial de interés Bluetooth (SIG). Pese a vertebrarse sobre Bluetooth clásico, BLE, es en sí misma una tecnología inalámbrica diseñada para cubrir objetivos distintos a su antecesor. El principal objetivo es diseñar un estándar de radio con el menor consumo de energía posible, especialmente optimizado para tener un bajo coste, con bajo ancho de banda, baja potencia y baja complejidad.

BLE, en contraste con su antecesora, facilita las comunicaciones de corto alcance entre dispositivos que no requieren grandes transferencias de datos, con la principal idea de proveer de una tecnología eficiente para la monitorización y control de aplicaciones donde las cantidades de datos son típicamente muy bajas, como el envío de los valores de sensores o mensajes de control.

La configuración clásica Bluetooth (BR/EDR) y la nueva, BLE, no son compatibles directamente. La forma de entenderse entre ellas es a través de un módulo doble. Nos encontramos con tres tipos de configuraciones posibles:

- Bluetooth clásico (BR/EDR): implementa el estándar inalámbrico clásico Bluetooth.
- Monomodo (BLE, Bluetooth Smart): implementa el estándar BLE y puede comunicarse con otros módulos monomodo y con los de modo dual pero no con los que sólo soportan BR/EDR.
- Modo dual (BR/EDR/LE, Bluetooth Smart Ready): implementa ambos estándares, BR/EDR y BLE y es capaz de conectarse con cualquier dispositivo Bluetooth.

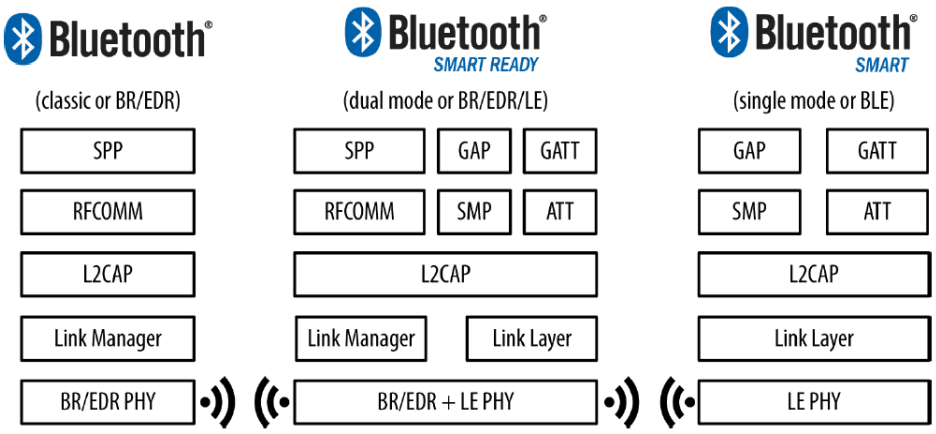

FIGURA 1-CONFIGURACIONES ENTRE VERSIONES Y DISTINTOS DISPOSITIVOS BLUETOOTH. **[6]**

En la figura 1 podemos ver las diferentes configuraciones entre las distintas versiones y dispositivos Bluetooth.

Cada vez más los nuevos dispositivos como smartphones, tablets y PCs que aparecen en el mercado incluyen, además de la versión clásica, de Bluetooth la nueva BLE. Algunos dispositivos como los sensores habitualmente son mono modo.

Desde la primera versión de la especificación en la que se incluyó BLE, la 4.0 (junio de 2010), han surgido dos más, la 4.1 (diciembre de 2013) y 4.2 (diciembre de 2014). Todas ellas publicadas por el SIG Bluetooth. Estas versiones introducen mejoras respecto a sus antecesoras, extendiendo las funcionalidades proporcionadas por la versión 4.0. Cabe destacar algunas diferencias importantes entre las dos últimas, 4.1 y 4.2.

- Los dispositivos que implementen Bluetooth 4.2 serán compatibles con dispositivos con versiones 4.1 y 4.0.
- Todas las características de la Bluetooth 4.1 son compatibles con la especificación de Bluetooth 4.2.
- En la versión 4.1 se introdujeron canales, en la capa L2CAP, dedicados para la conexión a internet que, junto a los nuevos servicios y perfiles introducidos en la versión 4.2, proporcionan a los desarrolladores y fabricantes la conexión a internet de forma flexible.
- La versión 4.2 extiende la longitud de los paquetes de datos de los 27 bytes a los 251 bytes. En la cabecera, el campo de longitud de paquete aumenta de 5 bits a 8 bits:

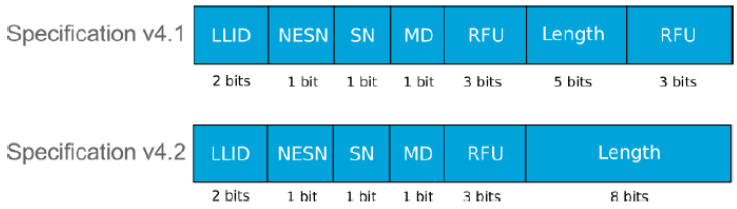

FIGURA 2- COMPARACIÓN DE DATOS DE LA CABECERA PDU. **[7]**

En la figura 2 podemos ver como el campo de RFU de la versión 4.1 es usado en la 4.2

para indicar el aumento de la longitud. Este aumento de la longitud del paquete es extremadamente importante en las aplicaciones emergentes del internet de las cosas (Iot).

- La versión 4.2 introduce el protocolo de seguridad ECDH (*Ellimptic Curve Diffie-Hellman*). Un protocolo criptográfico que permite que dos dispositivos que no hayan interactuado antes puedan intercambiar información de forma segura. Este nuevo método de seguridad se llama LE conexiones seguras.

Como vamos a ver en detalle más adelante en este documento, la pila de protocolos de BLE se compone de tres partes distintas:

- Aplicación: aplicación de usuario que cubre determinados casos de usos.
- Host: capas superiores de la pila de protocolos.
- Controlador: capas inferiores de la pila de protocolos, incluyendo la radio.

El controlador y el host interactúan a través del interfaz de control del host (HCI). Este estándar está definido por Bluetooth, lo que permite la interoperabilidad entre hosts y controladores de diferentes compañías.

Esto permite distintas configuraciones de hardware. Las diferentes capas pueden implementarse en un único chip o circuito integrado (IC), o pueden ser separado en varios ICs conectados a través de una capa de comunicación como UART, USB o SPI, tal y como podemos ver en la figura 2.

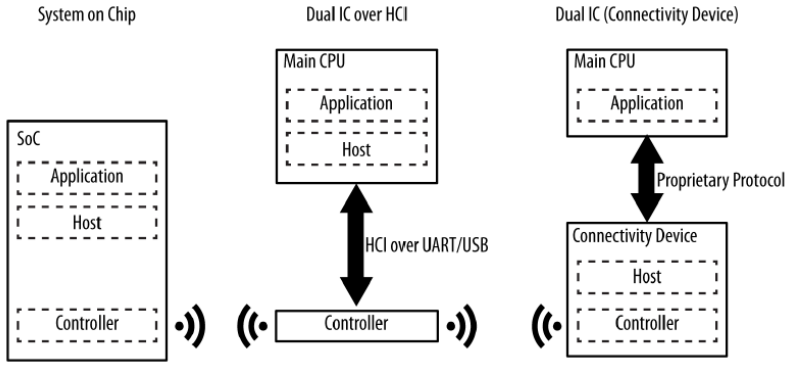

FIGURA 3- DISTINTAS CONFIGURACIONES HARDWARE. **[6]**

En el caso del IC dual aplicación- (host y controlador) la aplicación debe adaptarse al protocolo específico de cada vendedor del IC que incluye el host y el controlador.

Los sensores suelen utilizar la configuración SoC de forma que el coste y la complejidad del PCB sea reducido. Por el contrario, los dispositivos como tablets o smartphones optan por una configuración IC dual sobre HCI, ya que normalmente disponen de una CPU potente.

3.2 BLE, TECNOLOGÍA FUTURA

Quizás uno de los factores clave menos visibles que contribuyen al éxito de BLE es que fue diseñado para servir como un marco extensible para intercambiar datos. Esta es una diferencia fundamental con Bluetooth clásico, que se centró en un estricto conjunto de casos de uso. BLE, por el contrario, fue concebido para permitir que cualquier persona con una idea y un montón de datos procedentes de un accesorio, como un sensor, pueda desarrollar su propio proyecto sin necesidad de conocer en profundidad la tecnología. Los vendedores de teléfonos inteligentes entienden el valor de esta propuesta desde el principio, y nos proporcionaron API flexible y relativamente de bajo nivel para dar a los desarrolladores de aplicaciones móviles la libertad de usar el marco BLE en todo lo que consideran necesario.

A la hora de comparar BLE con otras tecnologías inalámbricas hay que tener en cuenta varios parámetros importantes como el rango de cobertura, la tasa de transmisión de datos, el consumo de energía y la topología de red.

BLE trabaja en el rango de las WPAN (*Wireless Personal Area Network*). Con una cobertura típica de 10 metros. Un PAN inalámbrica común es un smartphone conectado a través de Bluetooth a accesorios como auriculares inalámbricos, relojes inteligentes o dispositivo de control para el ejercicio físico. Estos tipos de dispositivos tienen tasas de transmisión bajas, pocas cantidades de datos. Por otra parte, los dispositivos inalámbricos para PAN suelen tener una potencia de transmisión de radio baja dada su naturaleza de corta distancia, lo que conlleva uso de fuentes de alimentación pequeñas, como pilas o pequeñas baterías.

El siguiente nivel en el rango de cobertura son las WLAN (*Wireless Local Area Network*), que normalmente cubren hasta los 100 metros. La tecnología inalámbrica predominante que trabaja en este rango es el Wi-Fi, otras son ZigBee o 6LowPAN como podemos ver en la Figura 4.

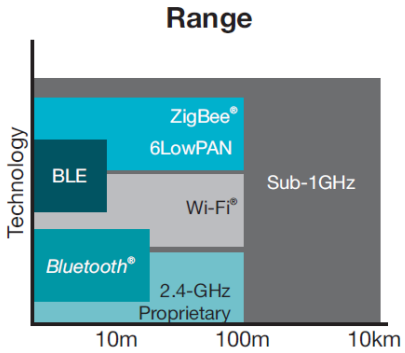

FIGURA 4- COMPARACIÓN ENTRE RANGO DE COBERTURA DE TECNOLOGÍAS INALÁMBRICAS. **[8]**

Las dos topologías de red fundamentales son en estrella y en malla, como se ve en la Figura 5. En una topología en estrella en la que todos los nodos están conectados a un nodo central, que típicamente también se utiliza como la puerta de enlace a Internet. Un ejemplo popular de una topología en estrella es una red Wi-Fi, donde el nodo central es llamado un punto de acceso (AP) y los otros nodos se suelen denominar estaciones.

En una red en malla cada nodo puede conectarse a varios nodos. Uno o más nodos de la red pueden servir como una puerta de enlace a Internet. En el ejemplo de la Figura 5 todos los nodos de la red están conectados entre ellos.

En una red en malla de ZigBee uno de los nodos se llama coordinador, y por lo general sirve también como una puerta de enlace a Internet.

Sin embargo, las redes de malla son más complejas de diseñar y pueden exhibir un mayor retraso al encaminar un mensaje desde un nodo remoto a través de la red, en comparación con redes en estrella. El gran beneficio de una topología de malla, como ya hemos explicado, es que se puede extender el rango de la red a través de múltiples saltos, mientras que el requerimiento de la potencia de transmisión se mantiene baja. También pueden conseguir mejor confiabilidad al permitir que más de una ruta se pueda utilizar para transmitir un mensaje a través de la red. BLE contará en poco tiempo con la posibilidad de trabajar en redes en malla, un paso importante para competir con el resto de tecnologías inalámbricas.

EL tamaño de la red, o el número máximo de dispositivos conectados al mismo tiempo, es también una consideración importante en el diseño del sistema.

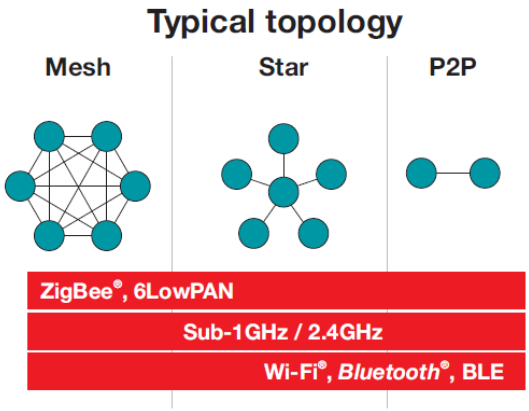

FIGURA 5- COMPARACIÓN ENTRE TOPOLOGÍAS DE TECNOLOGÍAS INALÁMBRICAS. **[8]**

La Figura 6 muestra una comparación entre las tasas máximas de transmisión alcanzables por las principales tecnologías inalámbricas.

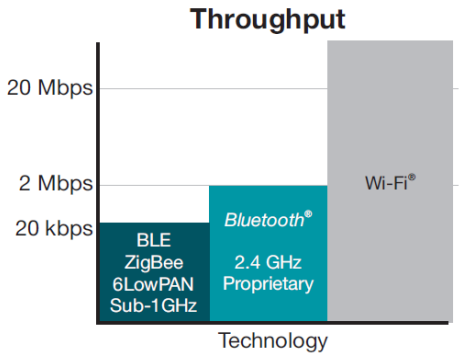

FIGURA 6- COMPARACIÓN ENTRE TASAS DE TRANSMISIÓN DE TECNOLOGÍAS INALÁMBRICAS. **[8]**

La máxima tasa de transmisión está ligada con el consumo energético y, por tanto, con el tipo

de alimentación necesaria, como podemos ver en la Figura 7.

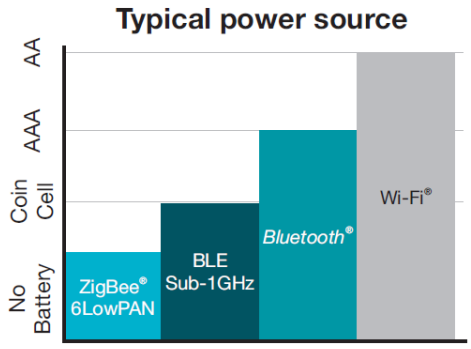

FIGURA 7- COMPARACIÓN ENTRE LAS FUENTES DE ALIMENTACIÓN DE TECNOLOGÍAS INALÁMBRICAS. **[8]**

3.3 MODELOS DE ENRUTAMIENTO

Hemos hablado de protocolo de enrutamiento, pero ¿qué es exactamente un protocolo de enrutamiento? ¿En qué consiste? ¿Qué tipos existen?

Empecemos definiendo la palabra enrutamiento o encaminamiento. Según la RAE (Real Academia Española) encaminamiento: acción y efecto de dirigir algo hacía un punto determinado.

Con lo que un protocolo de encaminamiento o protocolo de enrutamiento es el conjunto de reglas utilizadas por los routers o nodos de una red cuando se comunican con otros routers o nodos con el fin de compartir información de enrutamiento, información para poder dirigir los datos a transmitir por la ruta más conveniente. Esta información la almacenan los routers en tablas de enrutamiento.

Encontramos enrutamiento estático y dinámico. El enrutamiento estático tiene tablas de enrutamiento fijas, configuradas de forma manual, por lo que un router no puede adaptarse autónomamente a los cambios que se puedan producir en la topología de la red. El enrutamiento dinámico mantiene tablas de enrutamiento dinámicas por medio de mensajes que contienen información acerca de los cambios sufridos en la red y que generan la actualización de la información de la tabla de enrutamiento. Depende del origen de la información intercambiada se pueden clasificar en tres categorías:

-Adaptativo centralizado: existe un nodo central encargado de recoger toda esa información de control del resto de nodos y calcular la tabla de encaminamiento.

-Adaptativo distribuido: todos los nodos son iguales, calculan y recalculan continuamente la tabla de enrutamiento a partir de la información que les llega.

-Adaptativo aislado: se caracterizan por su sencillez, su respuesta a los cambios de tráfico o de topología se obtiene a partir de la información propia y local. El caso más habitual es el de encaminamiento por inundación: cada nodo reenvía cada paquete

recibido con destino a otros nodos, por todos los enlaces excepto por el que le llega.

La elección por parte del algoritmo de una ruta sobre otra y, por tanto la información que se mostrará en la tabla de rutas, puede depender de diversos parámetros. Los más habituales son:

-Número de saltos: número de nodos o routers por los que pasa un paquete.

-Retraso: según la cantidad de actividad existente en un recurso de red, como un router o nodo.

-Ancho de banda: capacidad de datos de un enlace.

-MTU: unidad máxima de transmisión, longitud máxima de trama en octetos que puede ser aceptada por todos los enlaces de la ruta.

En función de los parámetros y tipos de enrutados existen varios tipos de protocolos de enrutamiento, algunos de ellos son:

Protocolo de Difusión directa (modelo de un salto)

Este es el modelo más simple y representa la comunicación directa. Todos los nodos en la red transmiten a la estación base. Es un modelo caro en términos de consumo energético, así como inviable porque los nodos tienen un rango de transmisión limitado. Sus transmisiones no pueden siempre alcanzar la estación base, tienen una distancia máxima de radio, por ello la comunicación directa no es una buena solución para las redes inalámbricas.

Modelo Multi-salto (*multi-hops*)

En este modelo un nodo transmite a la estación base reenviando sus datos a uno de sus vecinos, el cual está más próximo a la estación base, a la vez que éste enviará a otro nodo más próximo hasta que llegue al nodo base. Por lo que la información viaja de la fuente al destino salto a salto desde un nodo a otro hasta que llega al destino. En vista de las limitaciones de los sensores, es una aproximación viable. Un gran número de protocolos utilizan este modelo.

Modelo esquemático basado en *clústeres*

Algunos protocolos usan técnicas de optimización para mejorar la eficacia del modelo anterior. Una de ellas es la agregación de datos usada en todos los protocolos de enrutamiento basados en clústeres. Una aproximación esquemática rompe la red en capas de clústeres. Los nodos se agruparán en clústeres con una cabeza, la responsable de enrutar desde ese clúster a las cabezas de otros clústeres o al nodo central. Los datos viajan desde un clúster de capa inferior a uno de capa superior. Aunque salta de uno a

otro, lo está haciendo de una capa a otra, por lo que cubre mayores distancias. Esto hace que, además, los datos se transfieran más rápido al nodo central.

Teóricamente, la latencia en este modelo es mucho menor que en la de *MultiHop*. Este modelo será mejor que los anteriores para redes con gran cantidad de nodos en un espacio amplio (del orden de miles de sensores y cientos de metros de distancia).

Protocolos centrados en el dato (*Data-centric*)

Si tenemos un número enorme de sensores, es difícil identificar de qué sensor queremos obtener un dato de una determinada zona. Una aproximación es que todos los sensores envíen los datos que tengan. Esto causa un gran despilfarro de energía. En este tipo de protocolo, se solicita el dato de una zona y espera a que se le remita. Los nodos de la zona negocian entre ellos la información más válida. Solo ésta es enviada, con el consiguiente ahorro de energía.

Para nuestro proyecto el protocolo de enrutamiento será dinámico adaptativo distribuido, cada nodo calculará y obtendrá la mejor ruta hasta el nodo central según la información que le llegue del resto de nodos y la propia almacenada. El parámetro que tomará nuestro algoritmo para elegir la mejor ruta será el número de saltos entre los distintos nodos y el nodo central.

4 ARQUITECTURA DE BLUETOOTH LOW ENERGY

BLE está formado por diferentes capas que interactúan entre sí. Cada una de ellas tiene una función con ciertas limitaciones y requerimientos. Estas diferentes capas forman la pila de protocolos.

La pila de protocolos de divide en tres partes; controlador, host y la aplicación. Vamos a ver de qué está compuesta para después profundizar en ellas.

4.1 ESTRUCTURA

-Aplicación

Es la capa superior y encargada de manipular los datos. Su arquitectura depende de la implementación particular de cada caso.

-Host

Está formado por diferentes capas:

- Perfil genérico de acceso (GAP).
- Perfil genérico de atributo (GATT).
- Protocolo de atributo (ATT).
- Protocolo de gestión de seguridad (SMP).
- Protocolo de control lógico y adaptación de enlace (L2CAP).
- Interfaz de control del host (HCI), en el lado del host.

-Controlador

Está formado por diferentes capas:

- Interfaz de control del host (HCI), en el lado del controlador.
- Capa de enlace (LL).
- Capa física (PHY).

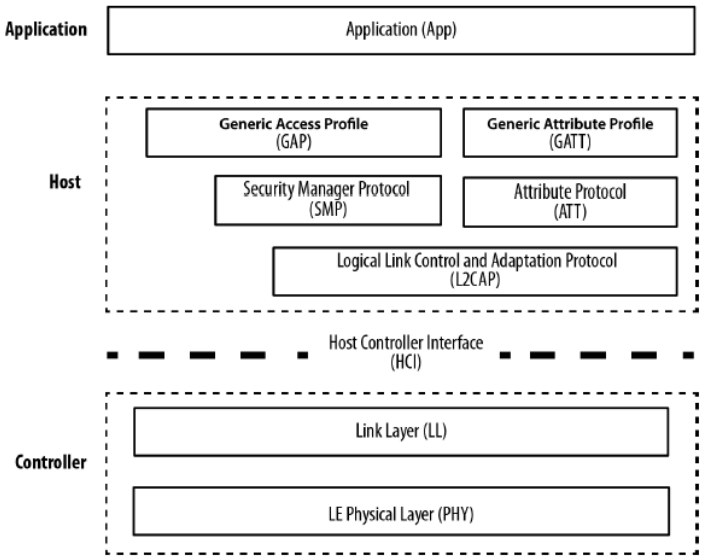

FIGURA 8-PILA DE PROTOCOLOS DE BLE. [1]

4.2 CONTROLADOR

4.2.1 La capa física (PHY)

Está formada por los circuitos que proporcionan la comunicación: modulación y demodulación de señales, transformación analógico-digital, etc…

Opera en la banda ISM de licencia libre de 2.4000GHz a 2.4835GHz, se definen 40 canales de 2MHz de ancho. Dos tipos de canales: de anuncio y de datos. Los canales de anuncio se utilizan para descubrimiento de dispositivos, transmisión de mensajes de *broadcast* y configuración de conexiones.

Los dispositivos BLE que operan en modo de anuncio transmiten periódicamente en tres canales (37, 38 y 39), que han sido asignados específicamente para minimizar la colisión con los canales de Wi-Fi que habitualmente más se usan (1, 6 y 11).

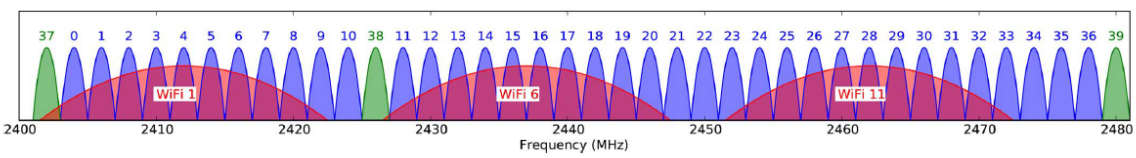

FIGURA 9- LOS 40 CANALES DE BLE Y LOS TRES MÁS HABITUALES DE WI-FI. CANALES 37, 38 Y 39 DEDICADOS A ANUNCIOS. [2]

BLE activa su radio solamente de 0.6 a 1.2 ms para escanear en busca de otros dispositivos utilizando los canales de anuncio (37, 38 y 39).

Los canales de datos se utilizan para las comunicaciones bidireccionales entre dispositivos conectados. Debido a que la banda ISM está abierta a cualquiera que desee utilizarla, el sistema de radio Bluetooth está preparado para evitar las múltiples interferencias que se puedan producir con una técnica llamada *spread-spectrum frequency hopping* o, lo que es lo mismo, saltos de frecuencia de espectro ensanchado. Los sistemas con dicha tecnología dividen la banda de frecuencias en varios canales y, durante el transcurso de la comunicación, se produce una transición brusca (salto o *hopping*) de un canal a otro siguiendo la fórmula:

$$canal = (canal\ actual + salto) \cdot mod\ 37$$

El valor del salto se comunica al establecer la conexión y es diferente por cada nueva conexión establecida. Esto hace que sea raro para más de un dispositivo transmitir en la misma frecuencia al mismo tiempo.

Todos los canales utilizan un esquema de modulación GFSK (*Gaussian Frecuency Shift Keying*) que permite reducir el pico de consumo.

El máximo rango de cobertura de BLE está alrededor de los 100 m. La tasa de modulación para BLE esta fija en 1 Mbps que, por consiguiente, es el límite de rendimiento físico superior para esta tecnología.

Añadir que, aunque el máximo ratio de modulación es de 1Mbps, éste nunca se alcanza, principalmente debido a los gastos introducidos en cada una de las capas.

4.2.1.1 Topología de red

La topología de red que predomina en BLE es la piconet. Un piconet es una red ad hoc que une un grupo de usuarios de dispositivos inalámbricos a través de protocolos sobre tecnología Bluetooth. Cada enlace está formado por un esclavo y un maestro. A diferencia de las conexiones en Bluetooth clásico, en BLE los esclavos no comparten el mismo canal físico con el maestro. Cada esclavo se comunica con el maestro en un canal físico separado. También se dan redes de dispersión (*scatternet*). En la figura 10 podemos ver ejemplos de posibles topologías que se pueden dar en BLE.

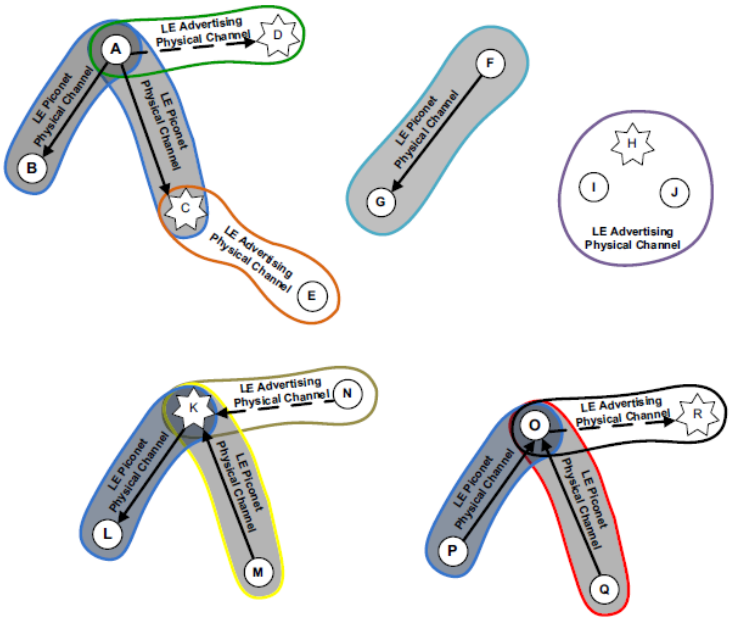

FIGURA 10- EJEMPLO DE TOPOLOGÍA EN BLE. **[9]**

En la figura 10 las flechas solidas señalan el sentido maestro esclavo. Las flechas de guiones indican un inicio de conexión y el comienzo de la flecha indica el iniciador de la conexión.El dispositivo anunciador se indica con una estrella.

- El dispositivo A es maestro en una piconet (representada por el área sombreada) con los dispositivos B y C como esclavos. Es también iniciador en la red de dispersión con el dispositivo D que es el anunciador.
- El dispositivo F es maestro en una piconet con el dispositivo G como esclavo.
- El dispositivo K se encuentra en una red de dispersión. También es maestro de dispositivo L y esclavo del M. También es anunciador en la red de dispersión con el dispositivo N que es iniciador.
- El dispositivo O se encuentra en una red de dispersión. Es esclavo de P y de Q.
- El dispositivo E es un escáner y el C es anunciador.
- El dispositivo H es anunciador y los dispositivos I y J son escáneres.
- El dispositivo R es anunciador y el O es iniciador.

4.2.2 La capa de enlace (LL)

Interactúa directamente con la capa física. Es la responsable de cumplir con todos los requisitos de tiempo definidos por la especificación, esto genera que normalmente permanezca aislada de las capas superiores de la pila de protocolos, es además computacionalmente costosa.

LL se encarga de gestionar como se conectan los dispositivos unos con otros. Un dispositivo BLE puede ser maestro, esclavo o ambos. Los dispositivos que inician una conexión serán maestros y los que las aceptan esclavos. Un maestro puede conectarse con múltiples esclavos y un esclavo

puede estar conectado con múltiples maestros.

BLE tiene una estructura asimétrica entre maestro y esclavo, se requieren más recursos para actuar como maestro. Por eso dispositivos como smartphones o tablets suelen actuar como maestros mientras que dispositivos como sensores, más pequeños, más simples y con limitaciones de memoria suelen actuar como esclavos.

Se definen 4 roles en la capa de enlace:

-*Anunciante*: el dispositivo envía mensajes de anuncio.

-*Escaneador*: el dispositivo escanea paquetes de anuncio.

-*Maestro*: el dispositivo que inicia la conexión y la gestiona después.

-*Esclavo*: el dispositivo que acepta peticiones de conexión.

Además de los roles que existen se describe el funcionamiento de la capa de enlace en términos de una máquina de estados con los siguientes cinco estados:

- Estado de espera- no transmite o recibe paquetes. A este estado se puede acceder desde cualquier otro estado.

- Estado de anuncio- transmite paquetes de anuncio y cuando sea necesario escucha y contesta a los posibles mensajes de petición y repuesta de escaneo. Correspondería al rol de anunciante. A este estado se puede acceder desde el estado de espera.

- Estado de escaneo- escucha a paquetes de anuncio en los canales de anuncio. Correspondería al rol de escaneador. A este estado se puede acceder desde el estado de espera.

- Estado de inicio- escucha a paquetes de anuncio y responder para iniciar una conexión con otro dispositivo. A este estado se puede acceder desde el estado de espera.

- Estado de conexión- a este estado se puede acceder desde el estado de escaneo o de inicio. Se definen dos roles:
    - Rol maestro – cuando se accede desde el estado de inicio.
    - Rol esclavo – cuando se accede desde el estado de anunciado.

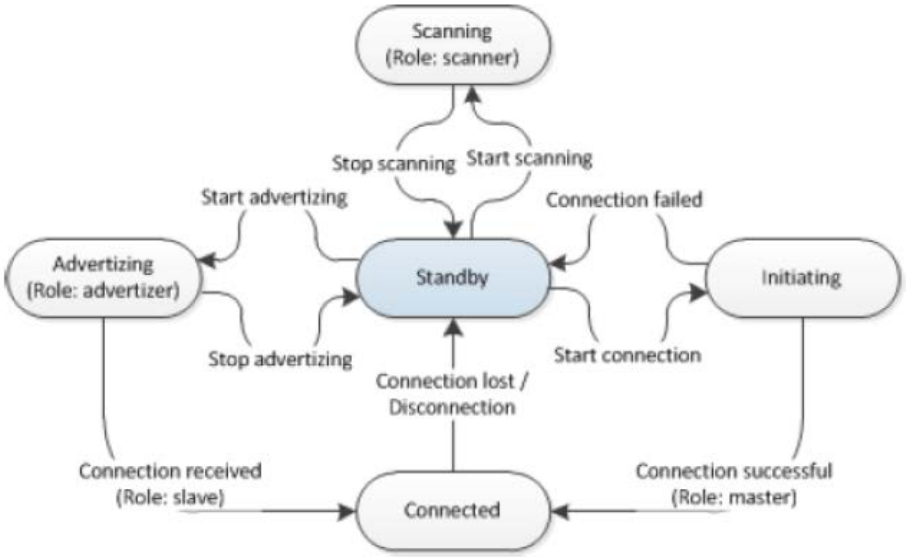

FIGURA 11- PROCESO ENTRE ESTADOS.

Existen direcciones a dos niveles distintos. La primera es la dirección de acceso perteneciente al canal físico. Para mitigar posibles efectos no deseados producidos por colisiones en el mismo canal, cada transmisión en un canal físico se inicia con una dirección de acceso que se utiliza como un código de correlación por parte de los dispositivos conectados al canal físico. Como podemos observar en la figura 12 la dirección de acceso está presente al inicio de cada paquete transmitido. Tiene un tamaño de 4 bytes y para todos los paquetes de los canales de anuncio tiene un valor fijo, 0x8E89BED6. Sin embargo, para los paquetes de los canales de datos la dirección de acceso debe ser diferente para cada conexión. Debe ser un valor aleatorio (con ciertas restricciones como no más de seis ceros o unos consecutivos) de 5 bytes de longitud generado por el dispositivo que inicia la conexión y enviado en el paquete de petición de conexión (CONNECT_REQ).

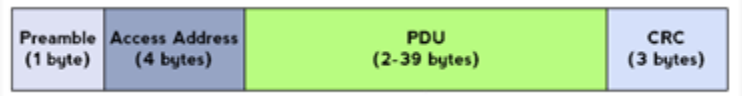

FIGURA 12- ESTRUCTURA DEL PAQUETE BLE. **[9]**

La otra dirección es la utilizada a la hora de identificar a un dispositivo Bluetooth. Existen dos tipos de direcciones, de 6 bytes de tamaño y que se encuentran en la PDU tras los 2 bytes del campo cabecera.

El primer tipo de dirección de dispositivo es la pública. Ésta dirección es fija, está programada en fábrica, nunca va a cambiar durante la vida útil del dispositivo y debe estar registrada en el IEEE. Se divide en dos campos: el primero, formado por los 3 bytes menos significativos, es el

campo asignado a la compañía y el segundo, formado por los 3 bytes más significativos, es el campo de identificador de compañía.

El segundo tipo de dirección de dispositivo es la aleatoria. Puede ser preprogramada en el dispositivo o generada dinámicamente durante el tiempo de ejecución.

La estructura general del paquete de la capa de enlace en el interfaz de aire refleja fielmente la arquitectura de capas del sistema BLE, como podemos ver en la Figura 13.

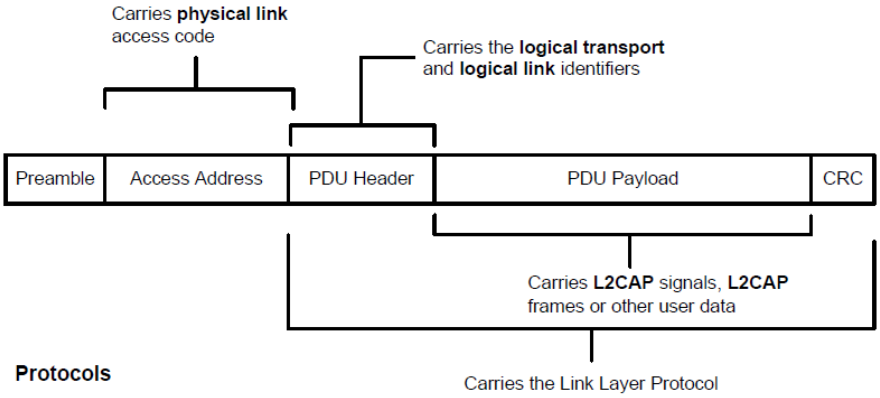

FIGURA 13- ESTRUCTURA DEL PAQUETE DE BLE. **[9]**

La capa de enlace compara todos los paquetes recibidos con sus CRC respectivos y pide la retransmisión cuando la comprobación de errores detecta un fallo en la transmisión. No hay límite superior para el número de retransmisiones, se reenviará hasta que la recepción sea correcta.

Una de los puntos que simplifica la implementación de la pila de protocolos es que solamente existe un formato de paquete y dos tipos de paquetes: de anuncio y de datos. La composición del paquete es la que podemos ver en la figura 5.

4.2.2.1 PAQUETES DE ANUNCIO

Los paquetes de anuncio tienen dos propósitos, publicitación de datos para aplicaciones que no necesitan establecer conexión y descubrir esclavos y conectarse con ellos. Cuando estos paquetes se envían previamente no se conoce si habrá algún dispositivo escaneando.

En el caso del tipo de paquete de anuncio tienen de carga útil hasta 31 bytes. La PDU de un paquete de anuncio tiene la siguiente composición:

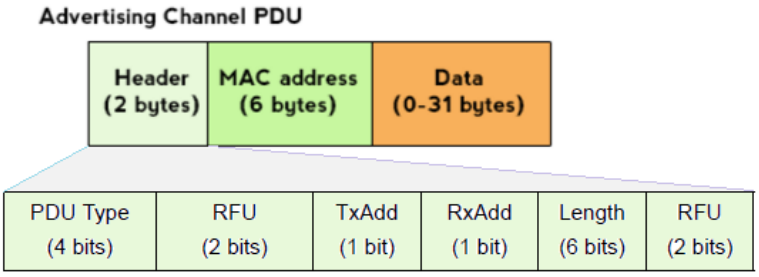

FIGURA 14- ESTRUCTURA DE LA CABECERA DE PAQUETE DE ANUNCIO.

El campo tipo de PDU puede tomar los siguientes valores:

PDU Type $b_3b_2b_1b_0$	Packet Name
0000	ADV_IND
0001	ADV_DIRECT_IND
0010	ADV_NONCONN_IND
0011	SCAN_REQ
0100	SCAN_RSP
0101	CONNECT_REQ
0110	ADV_SCAN_IND
0111-1111	Reserved

FIGURA 15- TIPOS DE CONTENIDO EN PAQUETE DE ANUNCIO SEGÚN CAMPO DE TIPO DE PDU. **[9]**

El campo TxAdd incida que la dirección contenida en el campo *MAC address* es pública (TxAdd=0) o aleatoria (TxAdd=1). El campo RxAdd indica si la dirección del iniciador del campo *data* es pública (RxAdd=0) o aleatoria (RxAdd=1).

Dentro del tipo de paquete de anuncio existen diferentes PDUs y cada una tiene un fin distinto. Antes de explicar cada una de ellas conviene definir tres propiedades de los paquetes de anuncio.

Conectabilidad:

- Conectable: un escáner puede iniciar una conexión al recibir este tipo de paquete.
- No conectable: un escáner no puede iniciar una conexión al recibir este tipo de paquete.

Escaneabilidad:

- Escaneable: un escáner puede emitir una petición de escáner al recibir este tipo de paquete.

- No escaneable: un escáner no puede emitir una petición de escáner al recibir este tipo de paquete.

Directividad:

- Directivo: un paquete de este tipo contiene en su carga útil solamente la dirección del dispositivo anunciador y del dispositivo escaneador objetivo. No se permite otro tipo de datos. Todos los paquetes directivos son por tanto conectables.
- No directivo: un paquete de este tipo no está dirigido a ningún dispositivo escaneador en particular.

4.2.2.1.1 PDUs de anuncio

Las PDUs de anuncio utiliza el rol anunciador de la capa de enlace. El campo AdvA contendrá la dirección pública o aleatoria del dispositivo anunciante según indique el campo TxAdd, es pública (TxAdd=0) o aleatoria (TxAdd=1). El campo RxAdd indica si la dirección del iniciador del campo InitA, para el caso de ADV_DIRECT_IND, es pública (RxAdd=0) o aleatoria (RxAdd=1).

La carga útil tiene la siguiente estructura:

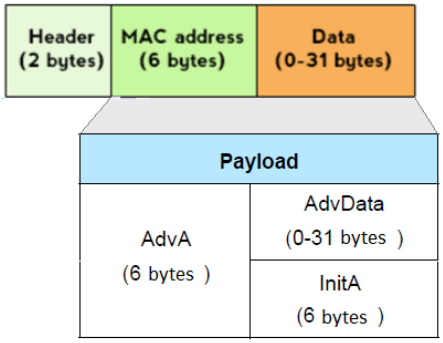

FIGURA 16- CARGA ÚTIL DE LAS PDU ANUNCIO.

- ADV_IND: tiene las propiedades de ser conectable, escaneable y no directivo.
- ADV_DIRECT_IND: conectable, no escaneable y directivo.
- ADV_NONCONN_IND: no conectable, no escaneable y no directivo.
- ADV_SCAN_IND: no conectable, escaneable y no directivo.

4.2.2.1.2 PDUs de escaneo

La carga útil tiene la siguiente estructura:

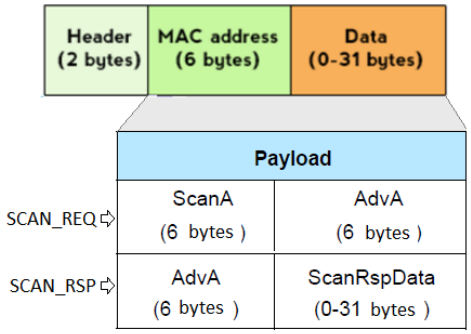

FIGURA 17- CARGA ÚTIL DE LAS PDU DE ESCANEO.

- SCAN_REQ: lo envía la capa de enlace de rol escaneador al recibir un paquete que en su PDU tiene la propiedad de escaneable. Lo recibe el anunciador que envió el paquete de anuncio escaneable. Este paquete no contiene datos del host.
- SCAN_RSP: lo envía el dispositivo anunciador como contestación a un paquete SCAN_REQ. Este paquete puede contener datos del host en el campo *ScanRspData*.

4.2.2.1.3 PDUs de inicio de conexión

Enviado por la capa de enlace del dispositivo en estado de inicio y recibo por el dispositivo en estado de anuncio. La carga útil tiene la siguiente estructura:

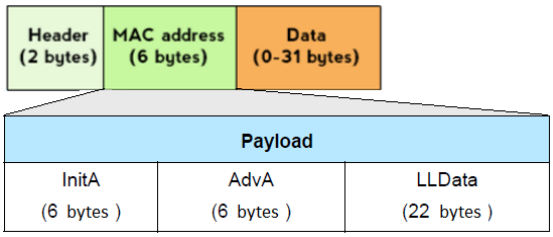

FIGURA 18- CARGA ÚTIL DE LA PDU DE INICIO DE CONEXIÓN.

- CONNECT_REQ: la carga útil consiste en el campo InitA que indica si la dirección del iniciador es pública (TxAdd=0) o aleatoria (TxAdd=1), el campo AdvA si la dirección del anunciador es pública (RxAdd=0) o aleatoria (RxAdd=1) y el campo LLData, que contiene 10 campos que indican las normas de la conexión, como el incremento de salto de frecuencia.

4.2.2.2 PAQUETES DE DATOS

Los paquetes de datos son usados para transportar los datos de usuario bidireccionalmente entre maestro y esclavo. Respecto a la carga útil existe una importante diferencia entre la última versión de la pila de protocolos, 4.2 es de 27 bytes, pero los protocolos superiores limitan la cantidad de la carga útil a 20 bytes por paquete, dependiendo de los protocolos usados.

La estructura actual de la cabecera de los paquetes de datos es la siguiente:

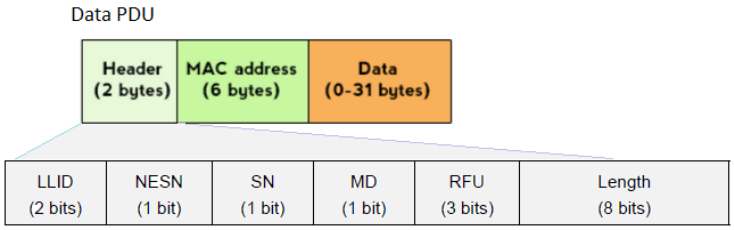

FIGURA 19-ESTRUCUTRA DE LA CABECERA DE LOS PAQUETES DE DATOS.

Está compuesta por una serie de campos, sobre los que no se va a profundizar. El significado de sus siglas es:

- LLID: identificador de enlace lógico. Define el tipo de carga útil (Logical Link Identifier).
- NESN: siguiente número de secuencia esperado (Next Expected Sequence Number).
- SN: número de secuencia (Sequence Number).
- MD: más datos (More Data).
- RFU: reservado para futuro uso (Reserved for Future Use).
- Length: indica el tamaño, en bytes, de la carga útil.
-

4.2.2.3 PROCESO DE ANUNCIO

El intervalo de publicidad es el que marca el tiempo de envío entre paquetes, es múltiplo entero de 0.625 ms y puede ir de un mínimo de 20 ms a un máximo de 10.24 s. En el evento de anuncio se puede enviar hasta un mensaje por cada uno de los tres canales de anuncio disponibles y los dispositivos que escanean lo hacen durante un tiempo en cada uno de esos tres canales, por lo que para que se reciba correctamente el paquete ambos deben coincidir en el mismo canal al mismo tiempo.

Si el tipo de paquete de anuncio es escaneable no dirigido (ADV_SCAN_IND) o no conectable no dirigido (ADV_NONCONN_IND) el intervalo de anuncio no será menor a 100 ms. Si el tipo de paquete de anuncio es conectable no dirigido (ADV_IND) o conectable dirigido (ADV_DIRECT_IND) el intervalo de anuncio puede ser de 20 ms o mayor.

El tiempo entre el inicio de dos eventos de anuncio consecutivos está formado por el intervalo

de publicidad y un retardo pseudoaleatorio, entre 0 ms y 10 ms, generado por la capa de enlace para cada evento de anuncio. En la figura 6 podemos ver lo descrito anteriormente: *T_advEvent = advInterval + advDelay.*

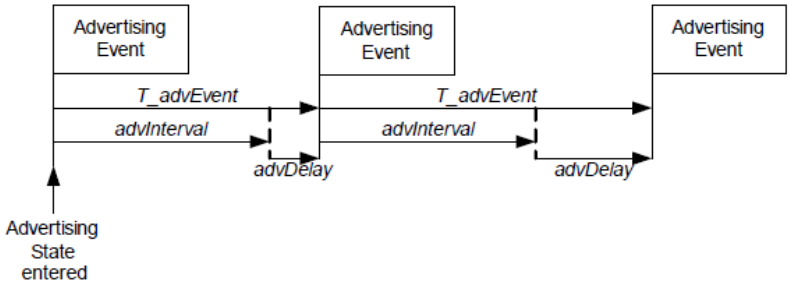

FIGURA 20- PROCESO DE ANUNCIADO. **[9]**

Independientemente del tipo de paquete de anuncio que se envíe, el tiempo de inicio entre dos paquetes de anuncio debe ser menor o igual a 10 ms (ver figura 7).

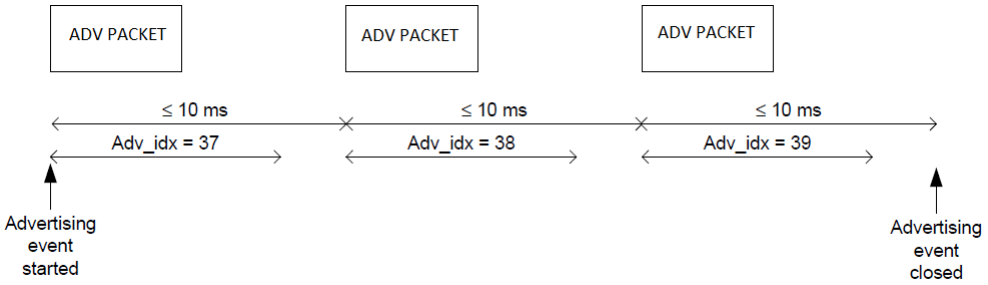

FIGURA 21- ESTRUCTURA DE UN EVENTO DE ANUNCIO. **[9]**

Si el tipo de paquete de anuncio es escaneable o conectable el escáner que los reciba podrá o bien mandar un SCAN_REQ o un INITIATION_REQ. Estos paquetes los mandará el escáner nada más recibir el paquete de anuncio y en el mismo canal que el paquete de anuncio. La figura 15 muestra este proceso, en este caso es el paquete de anuncio sobre el canal 38 el que recibe el escáner y al que contesta.

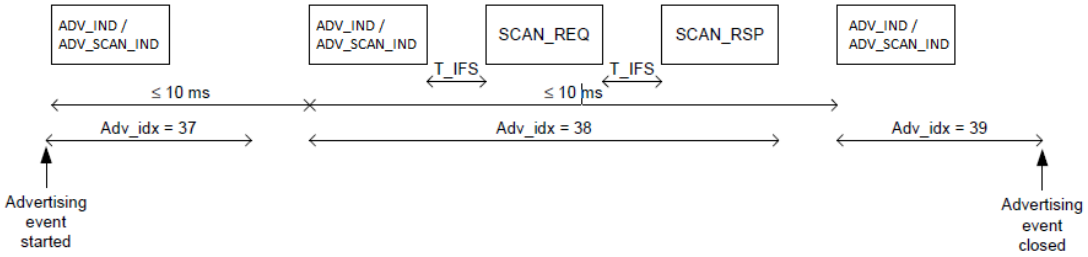

FIGURA 22- ESTRUCTURA DE UN EVENTO DE ANUNCIO ESCANEABLE CON ENVÍO DE SCAN_REQ. **[9]**

Como vemos en la figura 15 el envío, petición y respuesta debe suceder en los 10 ms.

Para el caso de una petición de conexión:

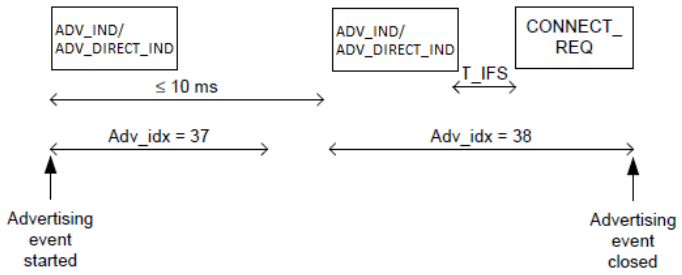

FIGURA 23- ESTRUCTURA DE UN EVENTO DE ANUNCIO CONECTABLE CON ENVIO DE CONNECT_REQ. **[9]**

Al recibir la petición de conexión el proceso de anunciado finaliza.

4.2.2.4 PROCESO DE ESCANEO

El proceso de escaneado viene marcado por dos parámetros:

- la ventana de escaneo: durante cuánto tiempo escucha.
- el intervalo de escaneo: con qué frecuencia ocurre la escucha.

Estos parámetros junto al intervalo de publicidad tienen un gran impacto en el consumo energético.

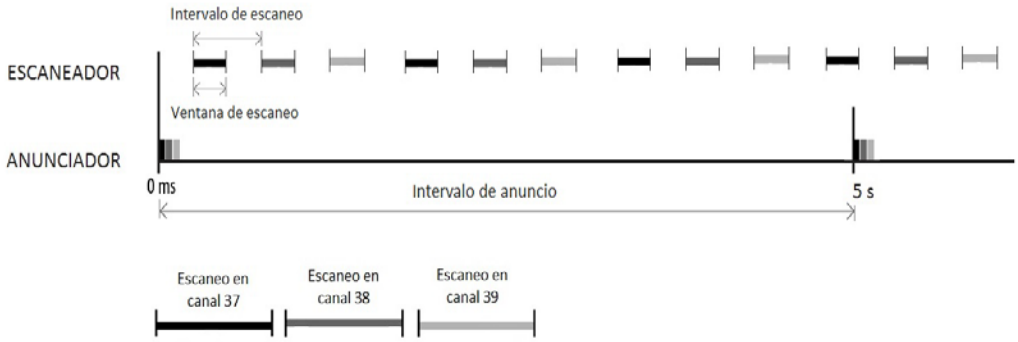

FIGURA 24-ANUNCIADO Y ESCANEADO.

Se especifican dos tipos de escaneado: el escaneo pasivo y el escaneo activo.

- *Escaneo pasivo:* el escáner solamente escucha paquetes de anuncio, por lo que la capa de enlace solamente puede recibir paquetes, no puede enviar, excepto para el caso en el que es objetivo de un ADV_DIRECT_IND. En ese caso sí que puede contestar con un CONNETC_REQ.
- *Escaneo activo:* el escáner escucha paquetes y puede emitir un paquete de solicitud de escaneo para que el anunciador le conteste con un paquete de respuesta de escaneo que puede contener cierta información sobre el anunciador.

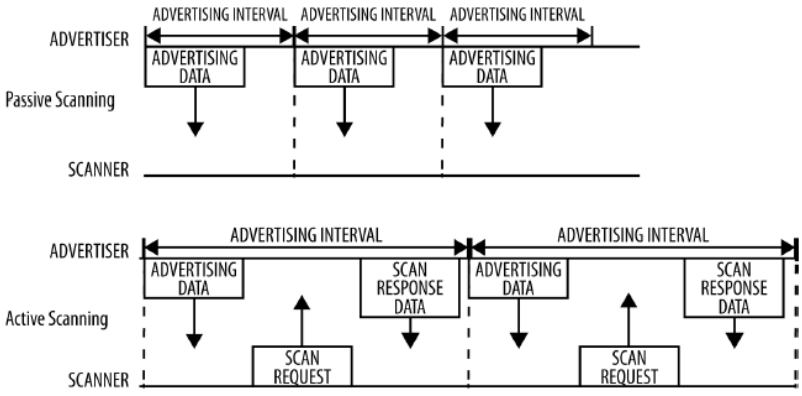

FIGURA 25- ESCANEO PASIVO Y ESCANEO ACTIVO. **[6]**

4.2.2.5 PROCESO DE CONEXIÓN

Cuando un maestro que escanea recibe un mensaje de anuncio conectable envía un paquete de petición de conexión (CONNECT_REQ) y, siempre y cuando el esclavo responda, se establece una conexión. Como ya hemos visto, el paquete de petición de conexión incluye el incremento de salto de frecuencia.

La conexión es un simple intercambio de datos entre esclavo y maestro en unos instantes predefinidos. Cada intercambio se llama evento de conexión, en la Figura 23 podemos ver un ejemplo de conexión.

Durante el proceso de establecimiento de conexión hay tres parámetros claves que el maestro comunica al esclavo y que determinan la conexión:

- Intervalo de conexión: tiempo que transcurre ente el inicio de dos eventos de conexión consecutivos (ver Figura 23). Puede estar entre los 7.5 ms hasta 4 s.

- Latencia del esclavo: el número de eventos de conexión que el esclavo puede saltarse sin riesgo de desconexión.

-Tiempo de supervisión de conexión: máximo tiempo entre dos paquetes de datos recibidos correctamente antes de que una conexión sea considerada perdida.

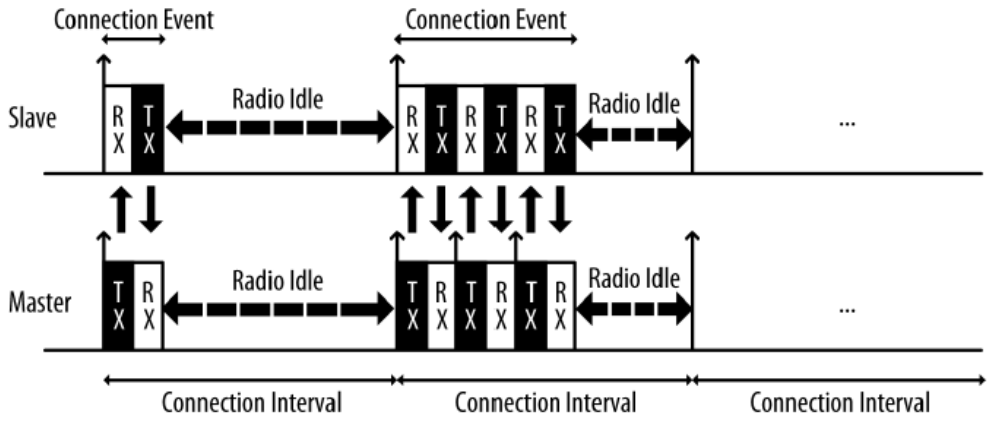

FIGURA 26- CONEXIÓN ESCLAVO-MAESTRO. **[6]**

Hay dos procesos de control relevantes que la capa de enlace lleva a cabo y que hacen referencia a las conexiones. El primero de ellos es el cambio de los parámetros de la conexión. Como ya hemos explicado, los parámetros de conexión (intervalo de conexión, latencia de eslavo, etc.) los establece inicialmente el maestro, pero las condiciones y requerimientos de la conexión pueden cambiar durante el tiempo de vida de la conexión. La capa de enlace permite al maestro y esclavo solicitar nuevos parámetros de conexión. En el caso del maestro, los puede fijar de manera unilateral en cualquier momento. Para el caso del esclavo necesita la aprobación del maestro. Por tanto, cada conexión puede ser ajustada para proporcionar el mejor balance entre tasa de transmisión y consumo de energía.

El otro proceso es el de encriptado del enlace para el intercambio de datos. Las claves las genera y gestiona el host, por lo que la capa de enlace realiza el cifrado y el descifrado de datos transparentemente para las capas superiores.

4.3 HOST

4.3.1 La capa de interfaz de control del host (HCI)

Es la encargada de permitir la comunicación entre el host y el controlador. En la mayoría de smartphones, tablets y ordenadores personales, el host y la aplicación se ejecutan en la CPU principal mientras que el controlador se encuentra en un chip independiente conectado vía UART o USB. Sin embargo, para aplicaciones con dispositivos embebidos, como redes de sensores, es preferible una gran integración que reduzca el tamaño y el precio del dispositivo: un sensor en un solo chip capaz de ejecutar las tres capas de forma simultánea con una CPU de bajo consumo. Esto es posible gracias a que la tecnología de semiconductores se ha vuelto lo suficientemente barata.

Solamente existe un tipo de formato de mensaje, aunque tiene dos tipos: paquetes de datos y paquetes de anuncio. El paquete de anuncio tiene una carga útil de datos máxima de 31 bytes y el paquete de datos de 27 bytes (pero se suele ver limitada por los protocolos superiores).

4.3.2 El protocolo de control lógico y adaptación de enlace (L2CAP)

Multiplexa los protocolos de las capas superiores y los encapsula en el formato estándar de paquete de BLE, y viceversa. Es similar a TCP, que permite que un gran rango de protocolos coexista sin problemas a través de un único enlace físico, cada uno con sus diferentes tamaños de paquete y requerimientos.

Son dos protocolos los que tiene por encima y de los que se encarga: protocolo de atributos (ATT) y protocolo de gestión de seguridad (SMP). El protocolo de atributos es la base del intercambio de datos en BLE y el protocolo de gestión de seguridad proporciona el marco para generar y distribuir las claves de seguridad entre pares.

La cabecea del paquete L2CAP ocupa 4 bytes, por lo que el tamaño de carga útil de datos de la capa de enlace se ve reducida en 4 bytes y, de forma análoga, la carga útil efectiva de usuario.

4.3.3 El protocolo de atributo (ATT)

Es un protocolo simple cliente/servidor sin estados. La comunicación se compone de pares independientes de solicitud y respuesta, basado en los atributos que contiene cada servidor. Estos atributos no son más que la forma en la que el servidor tiene organizados los datos. En BLE

cada dispositivo puede ser cliente, servidor o ambos, independientemente de si es maestro o esclavo.

Cada atributo tiene asignado un manejador de atributo de 16 bits (identificador usado para acceder al valor del atributo), un identificador único universal (que identifica el tipo de datos contenidos en el valor), un conjunto de permisos (escritura y lectura) y un valor.

Cuando un cliente quiere leer o escribir el valor de un atributo desde o hacia un servidor, emite una petición de lectura o escritura al servidor con el manejador de ese atributo. El servidor le puede contestar con el valor del atributo o con una aceptación. Durante una operación de escritura el servidor espera que el cliente proporcione datos en consonancia con el tipo de atributo y, en caso de que no lo sean, puede rechazar la operación.

4.3.4 El protocolo de gestión de seguridad (SMP)

Está compuesto por una serie de algoritmos de seguridad que aportan a la pila de protocolos de Bluetooth la capacidad de generar e intercambiar claves de seguridad, permitiendo, por ejemplo, ocultar la dirección pública Bluetooth para evitar seguimientos maliciosos de un dispositivo en particular.

Se definen dos roles:

-*Iniciador*: siempre se corresponde con el rol de maestro de la capa de enlace (LL).

-*Respondedor*: siempre se corresponde con el rol de esclavo de la capa de enlace (LL).

El protocolo de gestión de seguridad proporciona apoyo para los tres siguientes procedimientos:

-*Emparejamiento*: proceso en el cual se genera una clave de cifrado común y temporal para poder pasar a una conexión cifrada segura. Esta clave no se almacena y no podrá ser usada en conexiones posteriores.

-*Vinculación*: secuencia de emparejamiento seguida por la generación e intercambio de claves de seguridad permanentes, que se almacenan en la memoria no volátil y que crean un vínculo permanente entre dos dispositivos, que les permitirá establecer una conexión segura en conexiones posteriores sin tener que llevar a cabo el proceso de emparejamiento de nuevo.

-*Cifrado* de restablecimiento: después de un proceso de vinculación las claves de seguridad podrían haber sido almacenadas en ambos lados de la conexión. Si las claves de encriptado han sido almacenadas, este procedimiento define cómo usarlas en posteriores conexiones para restablecer una conexión cifrada segura sin tener que pasar el procedimiento de emparejamiento (o vinculación) de nuevo.

4.3.5 El perfil genérico de atributo (GATT)

Se basa en el protocolo de atributo (ATT) y define cómo se organizan e intercambian los datos entre aplicaciones. Se mantiene la arquitectura cliente/servidor, pero ahora los datos son encapsulados en servicios, que forman un perfil y que consisten en una o más características formadas a su vez por atributos que tienen diferentes funciones: valor, descripción, configuración, etc.

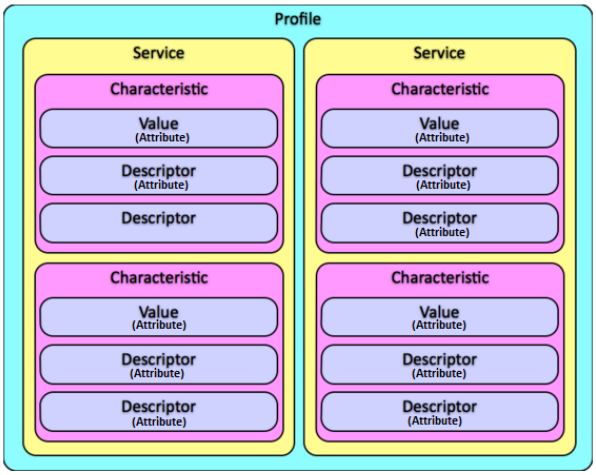

FIGURA 27-ESTRUCTURA DE PERFIL GATT. **[10]**

Existen perfiles que cubren todos los procedimientos y formatos de datos requeridos para cada caso de uso en particular. Es una de las grandes diferencias con el Bluetooth clásico que, aunque también define varios perfiles de uso como: perfil básico de imagen (BIP), perfil de distribución de audio avanzada (A2DP), perfil de transferencia de ficheros (FTP) o perfil para auriculares (HSP), son perfiles de uso muy generales y de dificultosa modificación. Sin embargo, la creación de perfiles de uso en BLE es muy sencilla, lo que permite que se puedan ajustar a casos muy específicos logrando una mejora de rendimiento y consumo energético.

El Bluetooth SIG define algunos perfiles GATT de uso aprobados: *Find Me Profile, Proximity Profile, Glucose Profile,* etc… (lista completa con descripción de cada perfil en https://developer.bluetooth.org/gatt/services/Pages/ServicesHome.aspx).

Los dispositivos pueden adoptar dos perfiles a la hora de interactuar entre ellos:

- **Cliente**: corresponde al cliente ATT. Envía peticiones al servidor y recibe respuestas desde el servidor. El cliente GATT no conoce a priori nada sobre los atributos del servidor, así que lo primero que debe hacer es preguntar a cerca de la presencia y naturaleza de los atributos mediante el descubrimiento de servicios.

- **Servidor**: al igual que el cliente, corresponde al servidor ATT. Recibe peticiones desde el cliente y envía respuestas. Es el responsable de almacenar los datos, organizados en atributos, y hacer posible su uso por el cliente.

Como ya hemos explicado, en ATT cada atributo tiene asignado un manejador de atributo de 16

bits (identificador usado para acceder al valor del atributo), un identificador único universal (que identifica el tipo de datos contenidos en el valor), un conjunto de permisos (escritura y lectura) y un valor. Los atributos, cada uno cumpliendo una función específica, forman las características de un servicio. Cada servicio viene descrito por un UUID (*Universally unique identifier*, identificador universal único).

Los UUIDs son usados por muchos protocolos y aplicaciones y tienen un tamaño de 16 bytes. Por eficiencia, la especificación BLE añade dos formatos nuevos de UUID: 2 bytes y 4 bytes.

Para reconstruir el UUID de 16 bytes completo, se insertan (indicados por XXXXXXXX) los UUIDs acortados en la base UUID Bluetooth:

XXXXXXXX -0000-1000-8000-00805F9B34FB

Estos formatos acortados sólo pueden ser usados con UUIDs definidos por la especificación Bluetooth y que corresponden a los servicios y perfiles registrados y especificados por el SIG. Para otros UUIDs hará falta usar el valor completo de 16 bytes.

Type (hex)	Type (text)	Hex value	GATT Server Permissions	Description/Value (text)
0x2800	GATT Primary Service Declaration	0x1800	R	Generic Access Service
0x2800	GATT Primary Service Declaration	F000AA50-0451-4000-B000-000000000000	R	Gyroscope Service

FIGURA 28- ATRIBUTOS QUE CONTIENEN LA DECLARACIÓN DE SERVICIOS.

En la figura superior podemos ver dos atributos que declaran dos servicios. El primero de ellos es un servicio de acceso genérico definido por el SIG y que, por tanto, tiene un UUID acortado:

Generic Access	org.bluetooth.service.generic_access	0x1800	Adopted

FIGURA 29- RECORTE DE LA WEB DEL SIG BLUETOOTH DONDE SE MUESTRAN LOS SERVICIOS DEFINIDOS.

En la figura 29 podemos ver la definición por parte del SIG Bluetooth del servicio de acceso genérico y el valor de su UUID de 2 bytes, 0x1800.

Volviendo a la figura 28, el segundo servicio, el de giroscopio, tiene un valor de UUID de 16 bytes. Este servicio es propio de Texas Instruments y no está recogido por el SIG Bluetooth. El valor del servicio está contenido en el valor del atributo.

Cada atributo contiene información sobre sí mismo y sobre los datos que contiene. Vamos a explicar cómo se estructuran los atributos:

Manejador	Tipo	Permisos	Valor

El primer campo es el manejador. Un identificador de 2 bytes para cada atributo que lo hace direccionable en un servidor GATT y está garantizado que no cambia. El valor 0x0000 significa manejador inválido, por lo que el rango de manejadores disponibles por todos los servidores de

un GATT va desde 0x0001 a 0xFFFF (65534), aunque típicamente suelen ser de pocas docenas.

El siguiente campo es el tipo de atributo. Es un UUID que puede ser de 2, 4 o 16. El tipo de atributo determina el tipo de datos presente en el valor del atributo. Por ejemplo, el 0x2800 indica que el atributo es una declaración de servicio primario, el 0x2801 indica declaración de característica.

El campo permisos especifica qué tipo de operaciones pueden ser ejecutadas sobre un atributo particular y con qué requerimientos de seguridad. Están los permisos de acceso:

-Ninguno: el atributo no puede ser escrito o leído por el cliente.

-Legible: el atributo puede ser leído por el cliente.

-Escribible: el atributo puede ser escrito por el cliente.

-Legible y escribible: el atributo puede ser leído y escrito por el cliente.

Tenemos también el requerimiento de nivel de encriptado necesario para ser accedido por el cliente:

-No se requiere encriptado: el cliente puede acceder al atributo en una conexión no encriptada.

-Autentificación de encriptado no requerida: la conexión debe ser encriptada para acceder al atributo, pero las claves de encriptado no deben ser autentificadas.

- Autentificación de encriptado requerida: la conexión debe ser encriptada con claves autentificadas para poder acceder al atributo.

Autorización, que determina si hace falta autorización por parte del usuario para acceder al atributo.

-Autorización no requerida.

-Autorización requerida.

Todos los permisos son independientes unos de otros y el servidor los puede combinar, almacenándolos en una base.

El último de los campos, el campo valor. No existen restricciones en cuanto al tipo de datos que puede contener el campo valor. Sin embargo, sí que existe una limitación en cuanto al tamaño, el máximo indicado por la especificación es de 512 bytes. Es la parte de un atributo a la que el cliente puede acceder libremente (con los permisos adecuados) para leer o escribir. El resto de campos del atributo no pueden ser modificados por el cliente ni éste puede acceder a ellos directamente.

Handle	Type	Permissions	Value
0x0201	$UUID_1$ (16-bit)	Read only, no security	0x180A
0x0202	$UUID_2$ (16-bit)	Read only, no security	0x2A29
0x0215	$UUID_3$ (16-bit)	Read/write, authorization required	"a readable UTF-8 string"
0x030C	$UUID_4$ (128-bit)	Write only, no security	{0xFF, 0xFF, 0x00, 0x00}
0x030D	$UUID_5$ (128-bit)	Read/write, authenticated encryption required	36.43
0x031A	$UUID_1$ (16-bit)	Read only, no security	0x1801

FIGURA 30- REPRESENACTIÓN DE ATRIBUTOS DE SERVIDOR GATT FICTICIO. **[6]**

En la figura 30 podemos ver los atributos de un servidor GATT ficticio. Los manejadores no tienen por qué ser consecutivos, pero sí que deben estar en orden, de menor valor a mayor.

4.3.6 El perfil genérico de acceso (GAP)

Especifica cómo los dispositivos llevan a cabo procedimientos de control como la detección de dispositivos, la conexión, difusión o sistemas de seguridad para asegurar la interoperabilidad y permitir que se lleve a cabo el intercambio de datos entre dispositivos de diferentes vendedores. Para ello establece diferentes conjuntos de normas y conceptos para regular y estandarizar el nivel bajo de funcionamiento de los dispositivos.

Los perfiles Bluetooth definen las funciones y características requeridas en cada capa física hasta la L2CAP, definen la iteración vertical entre las capas como también las interacciones entre pares de capas específicas entre dispositivos. Además, los comportamientos de la aplicación y los formatos de datos también están definidos por el perfil.
Todos los perfiles describen los requisitos de descubrimiento de servicios necesarios para que los dispositivos pueden conectarse, encontrar servicios de aplicaciones disponibles e información de conexión necesarios para realizar las conexiones a nivel de aplicación.

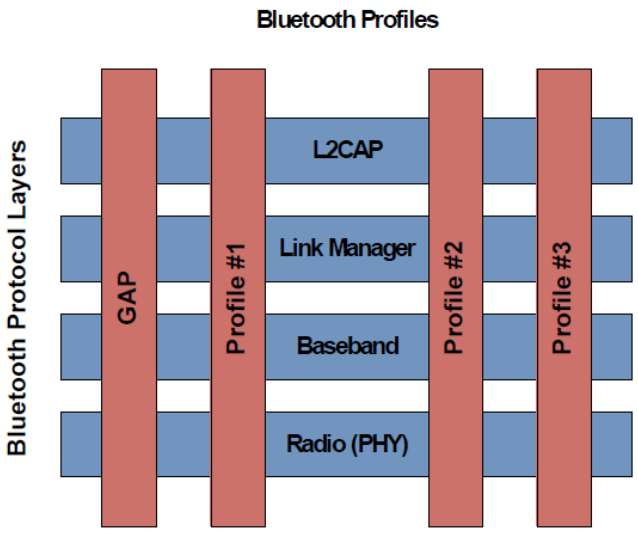

FIGURA 31- PERFILES BLUETOOTH. **[9]**

ROLES

Cada dispositivo puede operar en uno o más roles al mismo tiempo. Cada rol impone restricciones y hace cumplir ciertos requerimientos de comportamiento. Son cuatro posibles roles los que un dispositivo puede adoptar.

- ***Emisor***: este rol envía periódicamente mensajes de anuncio con datos. Esta optimizado para aplicaciones que distribuyan datos regularmente. Usa el rol de anunciante de la capa de enlace (LL).
- ***Observador***: esta optimizado para aplicaciones que solamente quieren recopilar datos de los dispositivos emisores. Escucha los datos incluidos en los mensajes de anuncio. Usa el rol de escaneador de la capa de enlace (LL).
- ***Central***: un dispositivo es capaz de establecer múltiples conexiones con otros dispositivos y es el iniciador de éstas. El rol central corresponde con el rol maestro de la capa de enlace. Comienza escuchando paquetes de anuncio de otros dispositivos y entonces inicia una conexión con el dispositivo que seleccione. El protocolo BLE es asimétrico, los requerimientos computacionales del rol maestro de la capa de enlace son mayores que los del rol esclavo. Por eso el rol central se encuentra en smartphones o tablets, mayores recursos de memoria y CPU.
- ***Periférico***: este rol corresponde con el rol esclavo de la capa de enlace. Utiliza paquetes de anuncio para permitir al rol central encontrarle y establecer después una conexión. El protocolo BLE está optimizado para necesitar pocos recursos, potencia de procesamiento y memoria, para la implementación de periférico. Esto ayuda a que desarrolle un gran mercado de dispositivos BLE periféricos baratos.

Cada dispositivo pude operar en uno o más roles a la misma vez. Muchas veces se asocian erróneamente los roles GATT de servidor y cliente con los roles GAP. No hay conexión entre ellos, cualquier dispositivo puede ser cliente GATT, servidor GATT o ambos, depende de la aplicación y la situación.

4.3.6.1 Descubrimiento: modos y procedimientos

La detectabilidad de un dispositivo se refiere a cómo un periférico anuncia de su presencia a otros dispositivos y qué deben hacer esos dispositivos con esa información. Las diferencias entre los diferentes modos de detección y procedimientos de descubrimiento se refieren a si el anunciado y el escaneo finalmente se llevan a cabo, pero también tienen en cuenta la la naturaleza de los datos incluidos en los paquetes de publicidad. Existe un campo opcional que se incluye en los datos del paquete de anuncio llamado *Flags AD* que controlan el modo de detectabiliad del dispositivo.

Data Type	Octet	Bit	Description
<<Flags>>	0	0	LE Limited Discoverable Mode
	0	1	LE General Discoverable Mode
	0	2	BR/EDR Not Supported. Bit 37 of LMP Feature Mask Definitions (Page 0)
	0	3	Simultaneous LE and BR/EDR to Same Device Capable (Controller). Bit 49 of LMP Feature Mask Definitions (Page 0)
	0	4	Simultaneous LE and BR/EDR to Same Device Capable (Host). Bit 66 of LMP Feature Mask Definitions (Page 1)
	0	5..7	Reserved

FIGURA 32- TIPOS DE BANDERAS (FLAGS) DE DATOS. **[11]**

Modos de detectabilidad

Los siguientes modos de detectabilidad permiten una cierta flexibilidad a los diseñadores de periféricos en función de las prioridades de diseño.

- *Modo no detectable*: otros dispositivos no pueden aprender sobre la presencia del periférico o realizar una consulta sobre su naturaleza. Un dispositivo en este modo puede seguir enviando paquetes de anuncio, pero bit 0 y 1 de la figura 32 deben estar a 0. Si se decide enviar mensajes de anuncio estando en ese modo, éstos deben ser del tipo ADV_NONCONN_IND o ADV_SCAN_IND.

- *Modo de descubrimiento limitado*: permite al dispositivo ser descubierto por un tiempo limitado, para ello el bit 0 de la figura 32 estará a 1 y el bit 1 estará a 0. Los dispositivos centrales que estén en procedimiento de descubrimiento limitado sólo podrán detectar dispositivos en este modo.

- *Modo de descubrimiento general*: este modo hace al dispositivo detectable durante el tiempo que se requiera. El bit 1 de la figura 32 estará a 1 y el bit 0 estará a 0. Sólo un dispositivo central que esté en procedimiento de descubrimiento general podrá encontrar dispositivos periféricos en este modo.

Procedimientos de descubrimiento

La especificación proporciona dos procedimientos de descubrimiento.

- *Procedimiento de descubrimiento limitado*: el dispositivo central comienza con un escaneo activo, sin filtrar y analizando cada paquete que recibe. Si la bandera Limited Discoverable del paquete recibido está a 1 se le pasa a la aplicación para más acciones.

- *Procedimiento de descubrimiento general*: el dispositivo central comienza con un escaneo activo, sin filtrar y analizando cada paquete que recibe. Si la bandera Limited Discoverable del paquete recibido está a 1 o la bandera General Discoverable está a 1 se le pasa a la aplicación para más acciones.

4.3.6.2 Establecimiento de conexión: modos y procedimientos.

Para que un central pueda iniciar el establecimiento de una conexión con un periférico, este último debe estar en modo conectable. Existen varios modos y procedimientos que controlan cómo interaccionan entre dispositivos.

Modos de establecimiento de conexión

Las diferencias entre los modos de establecimiento de conexión reflejan el uso por parte del periférico de los tipos de paquetes de anuncio.

- *Modo no conectable*: un dispositivo en este modo o no envía mensajes de anuncio o si los envía son del tipo ADV_NONCONN_IND o ADV_SCAN_IND. En cualquiera de los casos un central no podrá establecer conexión con él.
- *Modo de conexión directo*: un dispositivo en este modo envía paquetes del tipo ADV_DIRECT_IND. Estos paquetes no contendrán carga útil si no que contienen la dirección Bluetooth destino del central. Esto proporciona una rápida reconexión, se envían a la máxima frecuencia y durante un corto periodo. Estos paquetes solo los recibirá el central objetivo.
- *Modo de conexión indirecto*: un dispositivo en este modo envía paquetes de anuncio del tipo ADV_IND. Este es el estándar de modo conectable, a través del cual un periférico se hace a si mismo conectable durante un largo periodo de tiempo y trata de conectarse con un central que o ya conoce o es nuevo.

Procedimientos de establecimiento de conexión

Debido a que un dispositivo central no tiene medios para seleccionar los tipos de paquetes de anuncio que recibirá al escanear, con la intención de conectar (que siempre serán de tipo ADV_IND o ADV_DIRECT_IND), las diferencias entre los procedimientos de establecimiento de conexión no dependen de los tipos de paquetes de publicidad. En su lugar, el tipo de procedimiento de establecimiento de conexión utilizado depende del tipo de filtrado que el central impone sobre los paquetes entrantes.

- *Procedimiento de establecimiento de conexión automática*: el host introduce en una lista blanca una serie de dispositivos periféricos conocidos y luego pide al controlador que se conecte a la primera que se detecte. Este procedimiento es útil cuando el central ya conoce un conjunto limitado de dispositivos y no tiene una preferencia sobre a cuál conectarse.
- *Procedimiento de establecimiento de conexión general*: este es un proceso de dos pasos y es el usado comúnmente para conectarse a un periférico nuevo y desconocido. El central comienza escaneando sin una lista blanca, aceptando todos los paquetes de anuncio entrantes. Por cada periférico detectado la aplicación decide si conectarse o pasar al siguiente. Para ello la aplicación puede solicitar al usuario o analizar los datos contenidos en la carga del paquete. Una vez que el periférico es elegido, el central se conecta con él utilizando el procedimiento de establecimiento de conexión directa.

- *Procedimiento de establecimiento de conexión selectiva*: este procedimiento es idéntico al procedimiento de establecimiento de conexión general con la excepción de que el host usa una lista blanca con dispositivos previamente conocidos para filtrar los paquetes que entran. Este caso es útil cuando el usuario sólo quiera conectarse a ciertos periféricos conocidos.
- *Procedimiento de establecimiento de conexión directo*: en ese procedimiento se intenta establecer una conexión con un periférico particular. El host usa la capa de enlace para iniciar una conexión con un único dispositivo identificado por su dirección Bluetooth sin tener constancia de su presencia, por eso puede fallar si ese dispositivo objetivo no está disponible o está en un modo no conectable.

Vale la pena reiterar que un host central tiene dos formas diferentes para iniciar una conexión. El primer método requiere dos pasos: en primer lugar, el escaneo y luego conexión directa a un dispositivo (especificando su dirección Bluetooth) detectado durante la fase de escaneo. El segundo método se salta el paso de escaneo explícito y en su lugar utiliza al controlador para seleccionar uno o más dispositivos a los que conectarse, sin saber si están en ese momento cerca.

5 HERRAMIENTAS PARA DESARROLLO DEL PROYECTO

5.1 SOFTWARE

5.1.1 IAR Embedded Workbench

IAR Systems es la principal compañía a nivel mundial en la provisión de software para crear aplicaciones en sistemas embebidos. Como IDE usaremos la versión 9.30 para la arquitectura 8051. Existe la posibilidad de descargar una versión de prueba de 30 días de duración o una versión de prueba con límite de 4Kbytes de tamaño de código. Finalmente hemos optado por la de 30 días. Esta versión se puede descargar en la siguiente URL:

https://www.iar.com/iar-embedded-workbench/#!?currentTab=free-trials

En la figura 33 podemos ver el aspecto de la interfaz gráfica del programa.

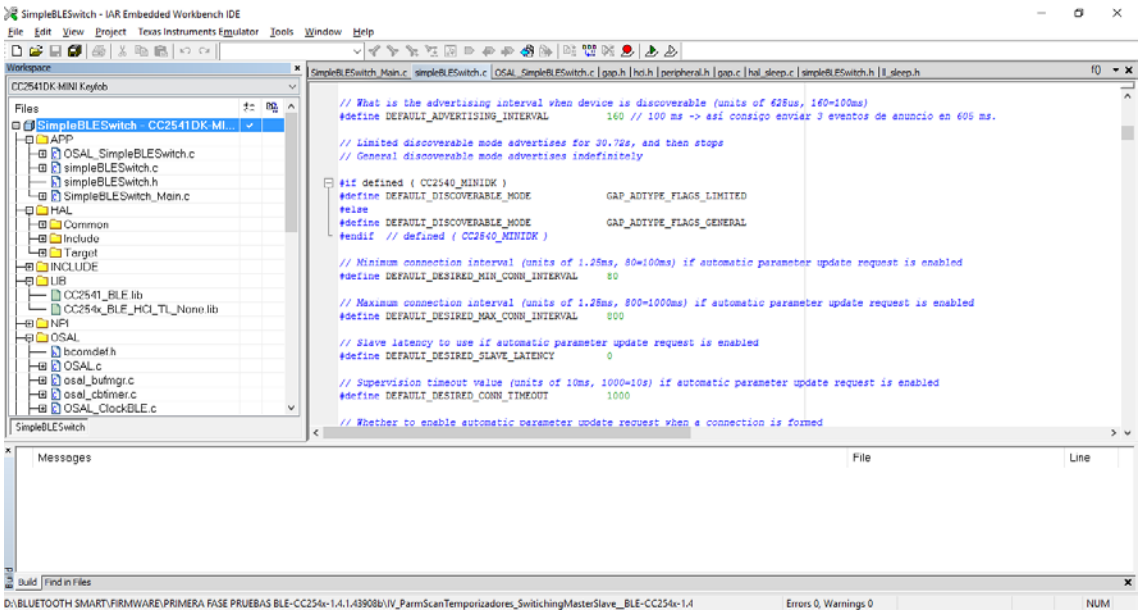

FIGURA 33- CAPTURA DE LA INTERFAZ GRÁFICA DE IAR EMBEDDED WORKBENCH.

En la parte izquierda de la pantalla tenemos el área de trabajo donde se muestran todos los archivos que contiene el proyecto. Estos archivos los podemos mostrar en la parte central de la pantalla. En la inferior se encuentra el área donde se muestran los diferentes mensajes que van apareciendo durante la elaboración del proyecto.

5.2 HARDWARE

5.2.1 Posibles módulos BLE (SoC)

En el mercado podemos encontrar fabricantes tanto de chips BLE propios como de módulos con chips de otros fabricantes o con chips propios. El fabricante principal es Texas Instruments y su principal chip es el CC2541 (ver Figura 34). Su familia más novedosa es la CC26xx, que monta un moderno ARM-M3, con una considerable reducción del consumo energético. Es destacable el fabricante Bluegiga, una compañía de Silicon Labs que usa el chip CC2541 de TI en su módulo BLE113 (https://www.bluegiga.com/en-US/products/), aunque actualmente posee un SoC BLE propio, el Blue Gecko, que monta en módulos propios. BLE113 incluye soporte para BGScript (lenguaje propio de Bluegiga), lo que le permite programar ciertos tipos de aplicaciones que utilizan simples archivos XML. Bluegiga también proporciona una API de lenguaje C para trabajar con estos módulos utilizando una MCU externa, que se comunica con el módulo a través de UART.

	Mode	Integrated Processor	Flash	RAM	Current Consumption BLE (RX/TX)	Average Current 1 sec / 4sec connection interval
TI CC2540/CC2541	Single Mode v4.0	8051	128kB/256kB	8kB	17.9mA / 18.2mA to 14.7mA / 14.3mA	24uA / 6.8uA
Texas Instruments CC256x	Dual Mode Classic + BLE/ANT	No - External	None	None	-	-
Texas Instruments CC26xx	Single Mode BLE v4.1	Cortex-M3	128kB	20kB	5.9mA	-
Nordic Semiconductor nRF51822	Single Mode v4.1 / ANT	Cortex-M0	128kB / 256kB	16kB / 32kB	9.7mA / 8mA	15.5uA / 5.6uA
Nordic Semiconductor nRF8001	Single Mode v4.0	None	None	None	14.6mA/12.7mA	-
Dialog Semiconductor DA14580	Single Mode BLE v4.1	Cortex-M0	32kB OTP	42kB + 8kB	4.9mA / 4.9mA	-
Cypress Semiconductor PSoC 4 BLE / PRoC BLE	Single Mode BLE v4.1	Cortex-M0	128kB / 256kB	16kB / 32kB	15.6mA / 16.4mA	18.9uA / 6.2uA
CSR CSR101x	Single Mode BLE v4.1	16-bit RISC	64kB	64kB	16mA	28uA / 10.8uA

FIGURA 34-COMPARACIÓN ENTRE DIFERENTES CHIPS BLUETOOTH Y VENDEDORES. **[12]**

El otro gran fabricante de chips BLE es Nordic Semiconductor [13] que lleva muchos años involucrado en soluciones inalámbricas de baja potencia y, como miembro de la junta de Bluetooth SIG, ha ayudado a definir y dar forma al núcleo del estándar BLE desde su creación. Fue una de las primeras empresas en conseguir un periférico BLE económico, el nRF8001 (ver

Figura 34). Su familia más reciente es la nRF51XX que supone una importante mejora respecto a su primer SoC, el nRF8001, que fue de los primeros existentes en el mercado.

La empresa Laird tiene un módulo BLE, el BL600 [14], que monta el chip nRF51822 de Nordic Semiconductor. Además de toda la funcionalidad incluida en el nR51822, estos módulos añaden una programación orientada a eventos, lenguaje smartBASIC, que le permite crear fácilmente aplicaciones básicas sin tener que aprender o invertir en IDEs comerciales caros y compiladores, o tener que programar en lenguajes de niveles bajos como C o C ++.
Proporciona libertad para programar los módulos directamente utilizando el código estándar de C y el SDK de Nordic y kit de herramientas para la nRF51822, pero la opción de lenguaje smartBASIC podría ser útil para facilitar el uso en los casos en los que sólo se tiene que añadir un enlace inalámbrico con un mínimo esfuerzo de desarrollo y sin tener que aprender una nueva pila ni la tecnología en profundidad.

5.2.2 CC2541 de Texas Instruments

5.2.2.1 Texas Instruments

Texas Instruments es una empresa norteamericana con sede en Dallas (Texas, EEUU) que desarrolla y comercializa semiconductores y tecnología para ordenadores. Pionera en el campo de la electrónica y los circuitos integrados (patentado por ellos en 1958). Por poner un ejemplo, en 1973 TI compartió con Intel el mérito por la invención casi simultánea del microprocesador. En la actualidad, TI es el tercer mayor fabricante de semiconductores del mundo tras Intel y Samsung y es el mayor suministrador de circuitos integrados para teléfonos móviles. Igualmente, es el mayor productor de procesadores digitales de señal y semiconductores analógicos. Otras áreas de actividad incluyen circuitos integrados para módems de banda ancha, periféricos para ordenadores, dispositivos digitales de consumo y RFID (identificación por radiofrecuencia).

Texas Instruments tiene varios modelos de SoC para aplicaciones BLE, producto de una continua investigación. El CC2540 fue el primer chip a nivel mundial, presentado en septiembre de 2010 por TI. En febrero de 2012 se presentó el CC2541 que introdujo ciertas mejoras respecto a su antecesor: optimización del consumo de energía y remplazamiento del interfaz USB por un I2C. Basada en la séptima generación de núcleos Bluetooth y compatible con la versión 4.1, la familia CC256X implementa el modo dual Bluetooth (Bluetooth Smart Ready, BR/EDR/LE): CC2560 y su mejora, el CC2564. Su coste es superior al de la familia CC254X.La última familia en el mercado es la CC26XX, familia de dispositivos de ultra baja potencia, de coste rentable y bajo consumo

energético. De esta familia, el enfocado a la tecnología BLE es el CC2640 que contiene un procesador *ARM Cortex-M3* de 32 bits y que posee mejores prestaciones que el CC2541.

5.2.2.2 Kit de desarrollo

TI comercializa un kit de desarrollo para su chip CC2541. Este kit es muy completo, incluye:

- Un **adaptador USB** con el chip CC2540. Se puede conectar a un puerto USB de un ordenador que esté funcionando con Windows y TI proporciona varias aplicaciones que permiten que actúe como si se tratara de un puerto serie, enviándole comandos byte a byte o como un *sniffer*.

FIGURA 35-ADAPTADOR USB CC2540. **[18]**.

- Un **mando/llavero** con el chip CC2541. Esta placa tiene dos botones, un led de dos colores, un timbre y un acelerómetro de 3 ejes. Todo ello va alimentado con una pila de botón tipo CR2032 de 3 voltios. TI ofrece programas para mostrar el funcionamiento de los elementos de este dispositivo y que pueden servir como punto de partida para futuras implementaciones.

FIGURA 36-MANDO CC2541. **[18]**

- Un **depurador/programador** para los dos dispositivos anteriores (*CC Debugger*). Permite la programación ilimitada de cualquiera de los dos chips. Para ello TI incluye también un cable USB – Mini USB para conectarlo al ordenador y un cable adaptador para pasar de los 10 pines de salida del programador a los 10 pines (de tamaño más reducido) de las placas de los dispositivos.

FIGURA 37-DEPURADOR/PROGRAMADOR. **[18]**

Su precio es muy asequible, 99$. Permite hacer pruebas y modificar los programas que da TI para conseguir diferentes objetivos. Ha sido una de las herramientas usadas durante el proceso de aprendizaje. Esta primera toma de contacto con el CC2541 es una de las razones por las que para el desarrollo del proyecto se le elija a él y no al CC2640.

5.2.2.3 Características del chip CC2541

El CC2541 es un SoC, desarrollado para optimizar el consumo de energía. Este SoC trabaja con aplicaciones propietarias de Bluetooth Low Energy a 2.4 GHz. Combina un destacado transceptor RF con un MCU 8051 mejorado. El bajo consumo se consigue con diferentes modos de funcionamiento y cortas transiciones entre ellos. El CC2541 viene en dos versiones diferentes: CC2541F128 con 128 Kb de memoria flash y el CC2541F256 con 256 Kb de memoria flash. Algunas de las características más destacables son:

- Soporta velocidades de 250 Kbps, 500 Kbps, 1 Mbps y 2 Mbps.
- Potencia programable de salida hasta 0 dBm.
- Excelente sensibilidad de recepción, hasta -94 dBm a 1 Mbps, selectividad y capacidad de bloqueo.
- 8 Kb de RAM.
- Microcontrolador 8051 de gran rendimiento y bajo consumo con captación previa de código.
- Interface I2C.
- 23 patas de entrada/salida de propósito general.

Texas Instruments proporciona la pila de protocolos para la configuración hardware con todas las capas implementadas en un único chip, la típica disposición hardware de los sensores.

La pila de protocolos, que tiene un consumo optimizado, incluye la capa de controlador y host:

- GAP: rol central, periférico, observador y anunciador (incluye también combinación de roles como periférico-anunciador o central-observador).
- ATT/GAT- rol cliente y servidor.
- SMP- AES-128 encriptación y desencriptación.
- L2CAP.

Se incluyen también aplicaciones y perfiles de ejemplo:

- Aplicaciones genéricas como: *SimpleBLEBroadcaster, SimpleBLECentral, SimpleBLEOberserver* y *SimpleBLEPeripheral.*
- Aplicaciones específicas: *Glucose sensor, Heart rate, Running sensor, etc…*

También permite varias configuraciones, como en un único chip, permitiendo ejecutarse aplicaciones sobre un CC2540 o CC2541 o un interfaz de procesador de red para ejecutar aplicaciones desde un microcontrolador externo.

Incluye además guías, en formato PDF, de documentación sobre las diferentes capas:

- *TI BLE Sample Applications Guide.*
- *TI BLE Software Developer's Guide.*
- *TI BLE Vendor Specific HCI Guide.*
- *TI CC2541 ARC Quick Start Guide.*
- *TI CC2541 ARC User Guide.*
- *HAL Driver API.*
- *OSAL API*.

Estos documentos son una herramienta imprescindible para poder comprender el funcionamiento qué, junto a una guía API en forma de interfaz de navegación sencillo, con el siguiente aspecto:

FIGURA 38-INTERFAZ DE NAVEGACIÓN DE LA GUÍA API DE TI.

Describe procesos de la propia API de la pila de protocolos BLE: GAP, GATT o SM, como también sobre los perfiles GAP: rol periférico, multi-rol periférico/broadcaster, rol central o gestor de vínculos.

TI proporciona una serie de programas:

-**SmartRF Flash Programmer**: este programa permite programar la memoria flash, leer y escribir la dirección IEE/MAC y actualizar el firmware del CC2540/41.

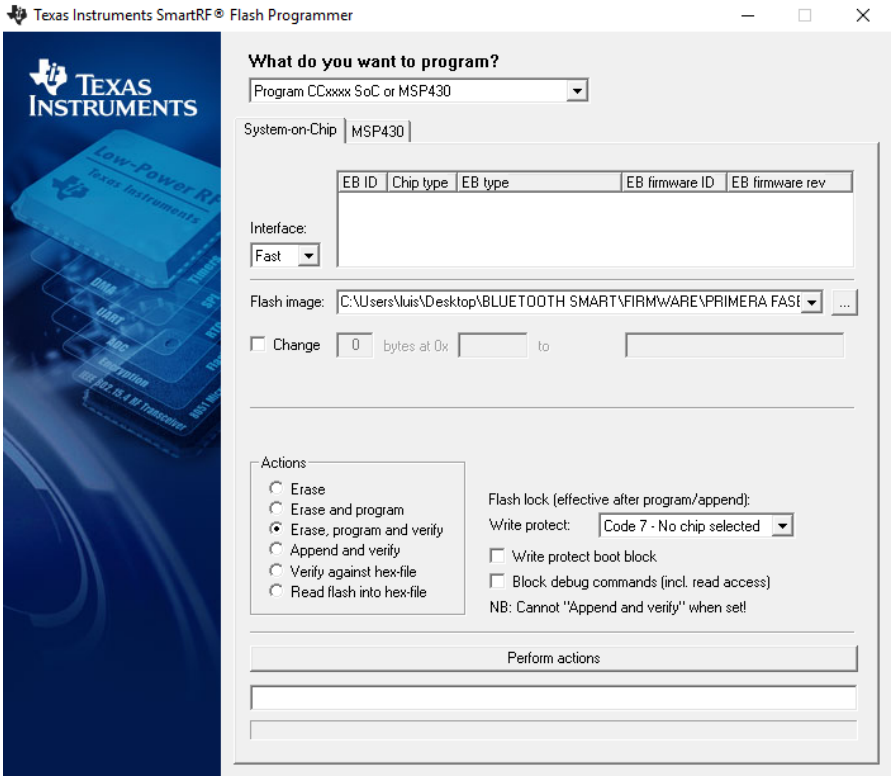

FIGURA 39- SMARTRF FLASH PROGRAMMER DE TI.

-**Packet Sniffer**: este programa permite ver y almacenar paquetes de RF capturados, en este caso con el CC2540 USB Dongle programado con *sniffer_fw_cc2540_usb.hex* (este archivo lo proporciona TI). Filtra y decodifica los paquetes mostrándolos de forma cómoda.

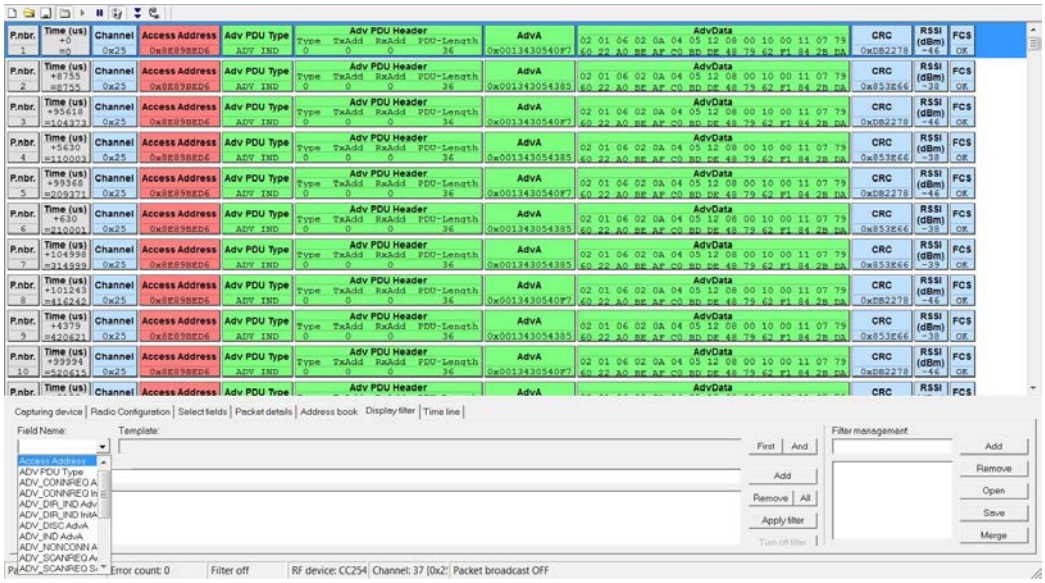

FIGURA 40- PACKET SNIFFER DE TI.

-**BLE Device Monitor**: este programa muestra los servicios, características y atributos de cualquier dispositivo BLE. También se puede utilizar para descargar el firmware a través del aire (OAD) para los dispositivos de TI que soportan esta característica. El Device Monitor BLE requiere de un CC2540 USB Dongle programado con *CC2540_USBdongle_HostTestRelease_All.hex*(este archivo lo proporciona TI). Este es el mismo firmware en que se basa BTooL.

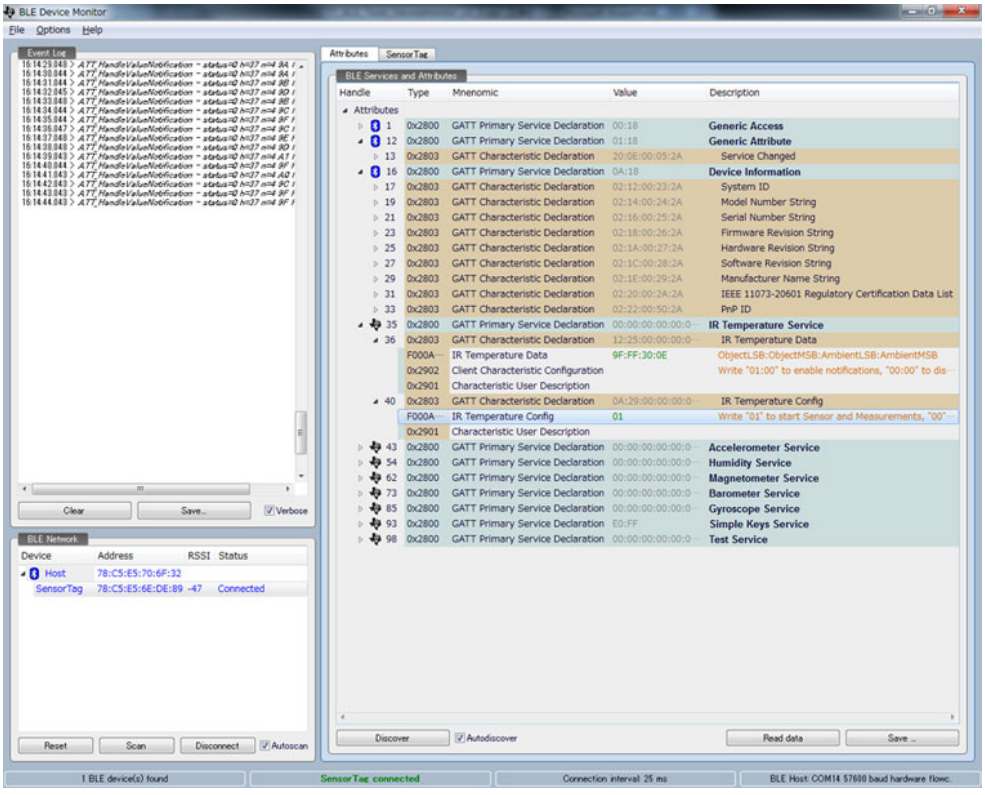

FIGURA 41- BLE DEVICE MONITOR DE TI.

-**BTool**: este programa permite formar una conexión entre dos dispositivos BLE mediante la comunicación con el CC2540 USB Dongle, actuando como un procesador de red, por medio de comandos HCI. El software Dongle USB y el controlador crean un puerto serie virtual a través del interfaz USB. Btool se comunica con el USB Dongle a través de este puerto serie virtual. El archivo *CC254041 Mini Development Kit* contiene ejemplos de cómo usar BTool.

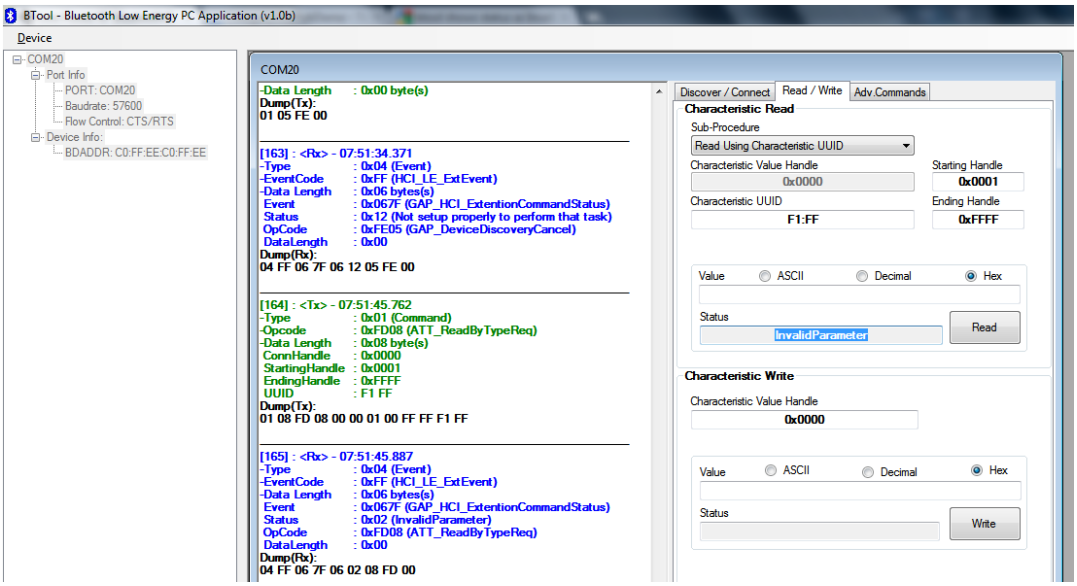

FIGURA 42- BTOOL DE TI.

Toda la documentación junto a los ejemplos de aplicaciones, perfiles y los programas *BLE Device Monitor, BTool, Packet Sniffer y Smart Flash programmer* ofrecidos por TI componen una herramienta fundamental para entender el funcionamiento y estructura de la pila de protocolos.

5.2.2.4 Arquitectura de memoria del MCU 8051

Es importante tener una breve idea de cómo está compuesta la memoria de un microcontrolador 8051, que es un circuito integrado que contiene una CPU, una memoria e interfaces de periféricos. También se conocen como SoC.

La arquitectura de la CPU del 8051 tiene cuatro espacios de memoria diferentes. Tiene espacios de memoria separados para memoria de programa y memoria de datos. Son los siguientes:

CODE: un espacio de memoria de sólo lectura para memoria de programa. Es un espacio de direcciones de memoria de 64 Kb.

DATA: un espacio de memoria de datos de lectura o escritura al que se puede acceder

directamente o indirectamente con un solo ciclo instrucciones de la CPU. Este espacio de direcciones de memoria es 256 bytes. A los 128 bytes menores del espacio de la memoria DATA se puede acceder directa o indirectamente, sin embargo, a los 128 bytes mayores sólo indirectamente.

XDATA: un espacio de memoria de datos de lectura y escritura, el acceso a él por lo general requiere 4-5 CPU ciclos de instrucciones. Este espacio de direcciones de memoria es de 64 KB. El acceso a la memoria XDATA es más lento que a la memoria DATOS.

SFR: un espacio de memoria de datos de lectura o escritura al que se puede acceder directamente por una sola instrucción de la CPU. Este espacio de memoria consta de 128 bytes. Para SFR cada dirección es divisible por ocho y cada bit también es direccionable individualmente.

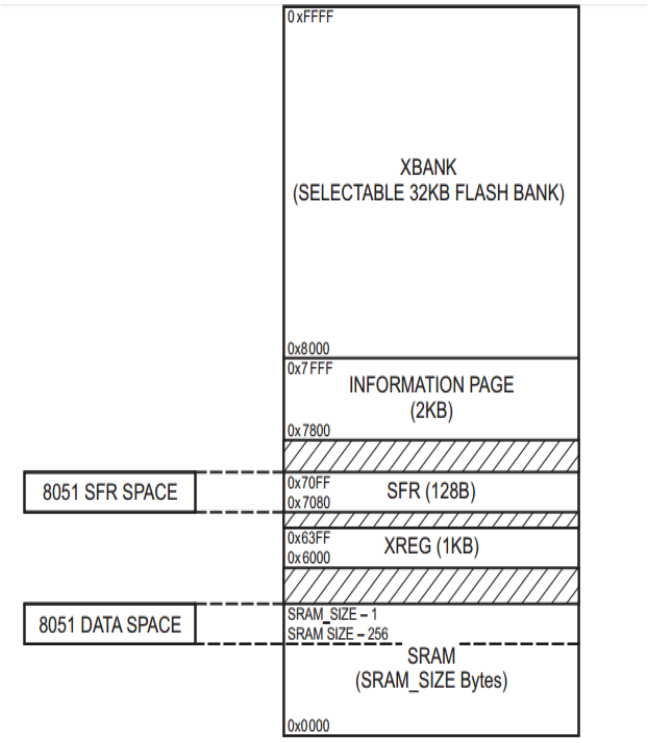

FIGURA 43-ESPACIO DE MEMORIA XDATA (MOSTRANDO SFR Y LA ASIGNACIÓN DE DATOS). [10]

5.2.2.5 Estructura de la pila BLE de TI

Además de los proyectos, merecen mencionarse dos carpetas de código incluido en todos ellos: la capa del sistema operativo (OSAL) y la capa de abstracción de hardware (HAL).

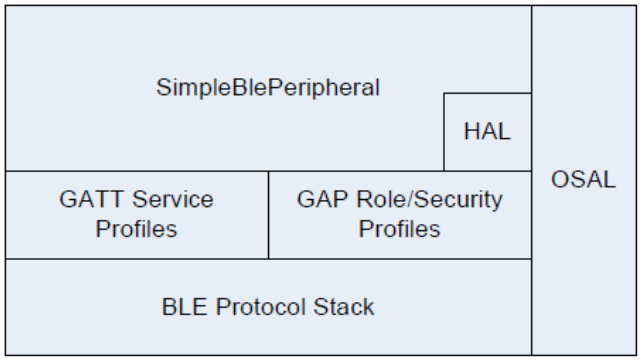

FIGURA 44- ESTRUCTURA DE LAS CAPAS DE LA PILA BLE DE TI. **[15]**

En la Figura 44 podemos ver cómo se encuentra estructuradas las capas de la pila de protocolos BLE de TI.

La capa OSAL

La capa OSAL, cuyas siglas significan *Operating System Abstraction Layer*, proporciona a las aplicaciones, a los perfiles y a la pila BLE un marco sobre el que desarrollarse y funcionar. Es una capa "vertical" como podemos ver en Figura 44. OSAL no es un sistema operativo en el sentido tradicional, más bien sería un bucle de control que permite al programa controlar la ejecución de eventos.

Toda función recibe un identificador, que debe ser llamado en una rutina de inicialización por parte de esta capa.

```
/*********************************************************************
 * @fn      osalInitTasks
 *
 * @brief   This function invokes the initialization function for each task.
 *
 * @param   void
 *
 * @return  none
 */
void osalInitTasks( void )
{
  uint8 taskID = 0;

  tasksEvents = (uint16 *)osal_mem_alloc( sizeof( uint16 ) * tasksCnt);
  osal_memset( tasksEvents, 0, (sizeof( uint16 ) * tasksCnt));

  /* LL Task */
  LL_Init( taskID++ );

  /* Hal Task */
  Hal_Init( taskID++ );

  /* HCI Task */
  HCI_Init( taskID++ );

#if defined ( OSAL_CBTIMER_NUM_TASKS )
  /* Callback Timer Tasks */
  osal_CbTimerInit( taskID );
  taskID += OSAL_CBTIMER_NUM_TASKS;
#endif

  /* L2CAP Task */
  L2CAP_Init( taskID++ );

  /* GAP Task */
  GAP_Init( taskID++ );

  /* SM Task */
  SM_Init( taskID++ );

  /* GATT Task */
  GATT_Init( taskID++ );

  /* Profiles */
  GAPRole_Init( taskID++ );
  GAPBondMgr_Init( taskID++ );

  GATTServApp_Init( taskID++ );

  /* Application */
  SimpleBLEPeripheral_Init( taskID );
}
```

FIGURA 45- LA FUNCIÓN OSALINITTASK INICIALIZA CADA FUNCIÓN LLAMÁNDOLAS POR SU IDENTIFICADOR DE TAREA. CAPTURA DE IAR EMBEDDED WORKBENCH.

Por ejemplo, las funciones relacionadas con el control de la conexión BLE tienen su identificador y son ejecutadas desde esta capa, aunque reciben una alta prioridad debido a las exigencias de tiempo concreto de las comunicaciones BLE. Por tanto, cualquier aplicación que se quiera desarrollar en modo *Single-Device* deberá tener en cuenta la priorización de eventos y administrarlos correctamente mediante esta capa. La Información detallada de su uso se puede encontrar en los manuales de TI (véase la referencia [5]).

La capa HAL

La capa HAL, cuyas siglas significan *Hardware Abstraction Layer*, proporcionan una interfaz de comunicación entre software y hardware. Esto permite el desarrollo de nuevo hardware (como por ejemplo una nueva PCB) sin necesidad de realizar cambios en la pila de protocolos ni en la aplicación. El código que proporciona TI está desarrollado para su propia plataforma por lo que, a no ser que se trate de una empresa a la que le interese diseñar su propia placa donde conectar el CC2541, no se necesitaría modificar nada de esta capa.

5.2.3 Módulo HM-10

El HM-10 es un módulo Bluetooth Low Energy basado en los chips de Texas Instruments CC2540 o CC2541. El fabricante chino Jinan Huamao Technology es el creador de la placa y del firmware original.

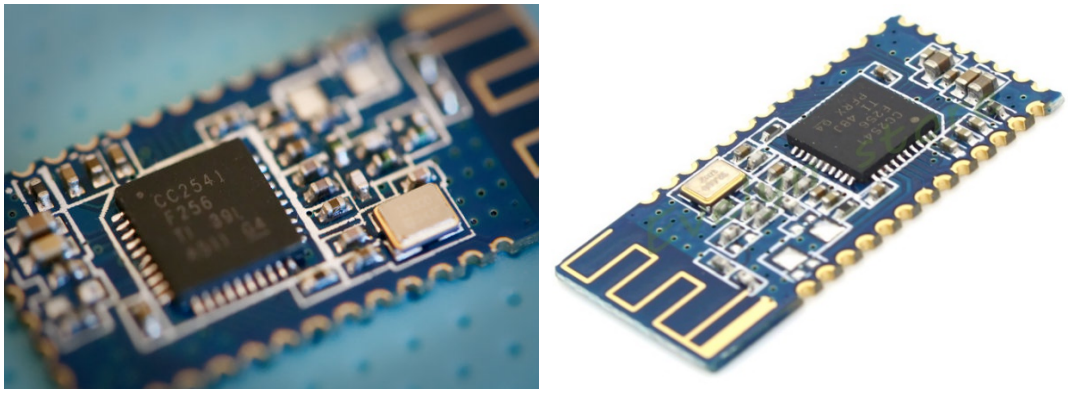

FIGURA 46-MÓDULO HM-10 COMPLETO (DERECHA) Y DETALLE DEL SOC CC2541 (IZQUIERDA).

Un aspecto importante de este módulo es que, una vez configurado, puede funcionar de forma autónoma sin necesidad de estar conectado a un microcontrolador que lo gestione, simplemente alimentando el módulo. Los chips de Texas Instruments y el módulo básico funcionan a 3.3V.

El módulo se puede adquirir también montado sobre una placa con seis pines macho (ver Figura 43). Este es el formato más sencillo de integrar en un proyecto si no se quiere soldar (los pines pueden conectarse a un Arduino u a otro microcontrolador por medio de una placa de prueba o de cables tipo jumper).

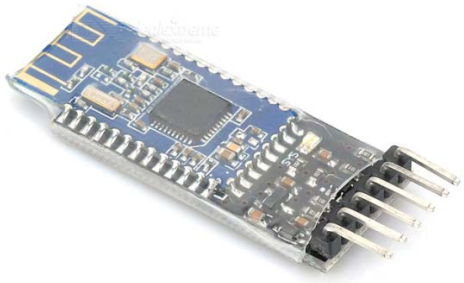

FIGURA 47-MÓDULO HM-10 SOBRE PLACA CON PINES.

Para nuestro proyecto usaremos el módulo HM-10 simple, para poder integrarlo en una placa junto a algún tipo de soporte para una pila AAA y a un conjunto de 8 pines, para poder conectar el dispositivo programador/depurador del kit de Texas Instruments y poder así programar el CC2541 del HM-10.

Este módulo ofrece la opción de ser configurado por medio de comandos AT. En la documentación sobre el HM-10 se explican todos los comandos que permiten controlar y configurar muchos de los parámetros. Se puede configurar como rol central, como baliza IBeacon o como rol periférico que simula una conexión serie.

Otro factor determinante para su elección es su precio. Podemos encontrarlo en páginas web de venta online como *AliExpress* desde 2.5 € la unidad para la versión básica y desde 3 € la versión sobre placa con pines. Si quisiéramos comprar chips CC2541 su precio es de unos 1.60 € la unidad, pero hay que añadirle la dificultad técnica que tiene soldar componentes tan pequeños. El precio de estos elementos es muy bajo y pone de manifiesto el peso y la importancia que tiene la tecnología Bluetooth para los fabricantes de electrónica de consumo que han apostado por ella, invirtiendo en investigación para aumentar la capacidad y reducir los costes de producción.

6 DESARROLLO DEL PROYECTO

6.1 INTRODUCCIÓN

La idea principal del proyecto es desarrollar un protocolo de enrutamiento similar al de las redes en malla, capaz de transmitir la información de cualquier nodo al nodo central incluso si estos dos no están en rango de cobertura.

Los nodos de la red, los dispositivos, pueden ser estáticos, con lo que su posición en la red no cambia, o no estáticos, móviles, comportamiento más cercano a lo que puede ocurrir en un entorno real.

El nodo central (raíz) es fijo y será el nodo que esté conectado con el host, siendo el punto de acceso de la red. No existirá entonces, dado el carácter fijo del nodo central, un método de balanceo que en caso de existir una descompensación en los saltos entre nodos intermedios y central otorgue las funciones de nodo central a otro que esté mejor posicionado.

Los nodos intermedios se van pasando la información hasta que ésta llega al nodo central siguiendo la ruta óptima: la más corta y menos saturada. Para poder llevar esto a cabo cada nodo debe saber cuál es esa ruta óptima hasta el nodo central. Este proceso de cálculo de ruta debe hacerse periódicamente para que, en caso de que un nodo se caiga de la red, los que dependían de él para alcanzar el nodo central recalculen su ruta. De la misma forma, si un nodo se une a la red podrá calcular su ruta óptima anunciando su presencia al resto de nodos y pudiendo ser salto intermedio para otros nodos.

6.2 PLANTEAMIENTO DEL MÉTODO DE FUNCIONAMIENTO

Se hará uso de los mensajes de anuncio para difundir el parámetro que utilizarán los nodos para configurar la ruta hacia el nodo central. Estos mensajes de anuncio son simples, no requieren confirmación de haber sido recibidos y son de fácil configuración. Además el consumo energético para enviarlos y recibirlos es bajo ya que el sistema radio pasa la mayor parte del tiempo apagado.

Cada dispositivo difundirá su distancia en saltos hasta el nodo central en mensajes de anuncio no conectables (ADV_NONCONN_IND). Se introducirá un byte en la carga útil del paquete, el campo distancia en saltos. El campo distancia en saltos lo leerán el resto de nodos y cuando sea menor que el que tienen almacenado lo sustituirán guardando también la dirección del nodo más cercano, de forma que siempre tendrán almacenada la dirección del nodo más cercano. A su vez modificarán el valor de saltos que difunden en sus mensajes de anuncio. El nodo central emitirá un valor de salto específico, 204 (en decimal, 0xCC en hexadecimal o 11001100 en binario) de forma que los nodos puedan distinguir los paquetes del nodo central.

Cuando haga falta transmitir información, que tendrá como destino el nodo central, el dispositivo anunciador se trabajará en rol periférico y se utilizarán mensajes de anuncio conectables y dirigidos (ADV_DIRECT_IND) a la dirección del nodo almacenada como salto más cercano. Este tipo de paquetes contiene en su carga útil solamente la dirección Bluetooth del anunciador y la del escáner objetivo, la del siguiente salto debe seguir nuestra información para llegar al nodo central. El dispositivo escaneador objetivo, al recibir este paquete, trabajará en rol central e iniciará una conexión mandando de vuelta una petición de conexión (CONNECT_REQ). Una vez establecida la conexión, el central (maestro) leerá los servicios del periférico (esclavo) en busca de uno en concreto que será el que contendrá la información que se quiere hacer llegar al nodo central. El nodo (maestro) que ha leído la información a transmitir la almacenará.

Cada dispositivo deberá ser capaz de recibir mensajes de anuncio y escanear en busca de posibles mensajes de anuncio de otros nodos. Esto implica que cada dispositivo debe funcionar simultáneamente en dos roles diferentes de la capa de enlace: anunciador y escaneador. En este punto nos encontramos con el primer problema, ya que los dispositivos no pueden trabajar simultáneamente enviando mensajes y escaneando en los canales de anuncio. Por lo tanto, la radio o se encuentra transmitiendo o escuchando. Se debe recordar que para que un mensaje de anuncio se reciba correctamente debe ocurrir que la ventana de tiempo de escaneo se produzca en el mismo canal en el que se envía el mensaje.

También se ha detectado un posible problema de interferencia en los canales de anuncio si varios dispositivos en rango de cobertura emiten los mensajes en intervalos de anuncio cortos. Para evitarlo se ha pensado que el intervalo de anuncio, periodo que incluye el evento de anuncio: envío de los mensajes en los tres canales (no es obligatorio enviar por los tres canales, también se puede sobre uno sólo) y la espera hasta el siguiente evento de anuncio, ocurra cada 5 segundos. El estándar nos dice que el intervalo de anuncio puede estar entre 20 ms y 10.24 s y que entre intervalos de anuncio se introduce un retardo aleatorio que va entre 0 ms y 10 ms. De esta manera se reduce mucho la probabilidad de coincidencia de los eventos de anuncio y con ello la interferencia cocanal.

La posible solución a la interferencia cocanal, separando tanto los eventos de anuncio (con aumento del intervalo de anuncio), dificulta mucho la rápida detección de los mensajes por parte del escáner, como se puede ver en la Figura 48.

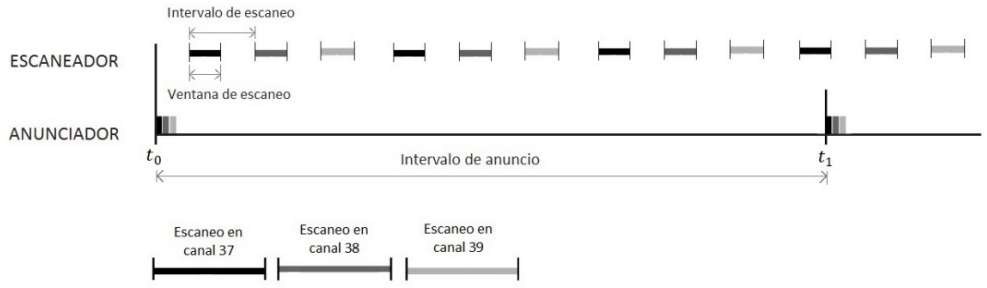

FIGURA 48- PROCESO ANUNCIO-ESCANEO CON INTERVALO DE ESCANEO GRANDE.

Para solucionar esto una de las opciones planteadas consiste en que los intervalos de escaneo sean lo más cortos posible, frecuencia de escaneo alta, y maximizar el tiempo de la ventana de escaneo (tiempo en el que se escanea el canal). En la Figura 49 podemos ver como la reducción del intervalo de escaneo provoca que la detección del paquete sea más rápida. De esta forma aseguraríamos el menor tiempo de respuesta en cuanto a detección se trata, pero por el contrario tendríamos un mayor consumo ya que la radio se encuentra más tiempo encendida. En nuestro proyecto no existen requerimientos de bajo consumo, pero de cara a una futura implementación es conveniente tenerlo en cuenta.

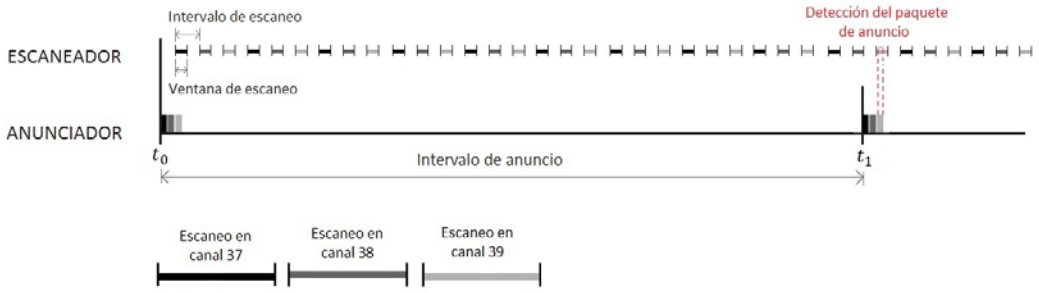

FIGURA 49- PROCESO ANUNCIO-ESCANEO CON INTERVALO DE ESCANEO PEQUEÑO.

Otra opción sería que la frecuencia de escaneo no fuese tan alta, pero esto ralentizaría el proceso de descubrimiento de las rutas. La disminución de la probabilidad de coincidencia escaneo-mensaje incrementaría el tiempo de detección de los mensajes.

Por tanto, la mayor parte del tiempo del proceso de cálculo de ruta óptima lo dedicarán los nodos intermedios a escanear en busca de mensajes de anuncio. El hecho de que se emitan tan pocos mensajes de anuncio y tan separados aumenta la probabilidad de que no se detecten, teniendo que pasar varios intervalos de anuncio para que ocurra la detección. Para aumentar esta probabilidad de detección podrían enviarse dos o más eventos de anuncio consecutivamente para después entrar en un periodo largo de escaneo. El intervalo de escaneo debe ser corto para aumentar las probabilidades de coincidencia mensaje de anuncio-canal escaneado.

Existen limitaciones en cuanto al mínimo tiempo de intervalo de anuncio, al ser paquetes de anuncio ADV_NONCONN_IND el intervalo de anuncio no será menor a 100 ms.

Para asegurar el descubrimiento de las emisiones hay una regla general, el intervalo de anuncio más diez debe ser menor que la ventana de escaneo del dispositivo observador. Un intervalo de emisión más bajo permitiría que los datos difundidos fueran escaneados con mayor rapidez, aunque requiere más energía [16].

En cuanto al consumo energético, aunque en este proyecto no sea una restricción a continuación, se establece una comparación entre diferentes modelos de valores de ventana de escaneo, intervalo de escaneo y periodo de escaneo. Como en el proyecto usamos un escaneo activo los casos con los que se compara también utilizan este tipo de escaneado. Para el análisis, el consumo que genera el despertado del dispositivo (unos 0.4 ms), el pre proceso a escanear (unos 0.35 ms) y el post proceso (1.3 ms), los vamos a despreciar ya que en comparación a los

200 ms de la ventana de escaneo no tiene relevancia. Nos quedamos solamente con el consumo que se produce durante la ventana de escaneo (unos 17.5 mA), el resto del tiempo entre ventanas de escaneo el dispositivo se duerme, así que el consumo no es destacable frente a los 17.5 mA. Estos valores son estimados en función de los cálculos de Texas Instruments [17]. Como recordatorio, la estructura del proceso de escaneado es la siguiente:

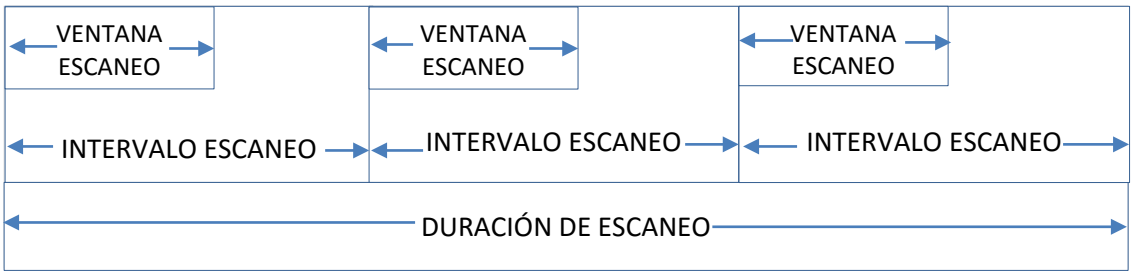

FIGURA 50- COMPOSICIÓN DEL PROCESO DE ESCANEADO.

En función del intervalo de escaneo, y por consiguiente la duración de escaneo, y la de la ventana de escaneo, la variación del consumo cambia. Se proponen 3 casos de análisis.

CASO 1 (como en el proyecto)

- Ventana de escaneo 200 ms
- Intervalo escaneo 220 ms
- Duración de escaneo 660 ms

CASO 2

- Ventana de escaneo 120 ms
- Intervalo escaneo 220 ms
- Duración de escaneo 660 ms

CASO 3

- Ventana de escaneo 200 ms
- Intervalo escaneo 440 ms
- Duración de escaneo 1230 ms

El consumo medio de duración de escaneo (escaneo en los tres canales) para el caso 1 es de 1.590 mA.

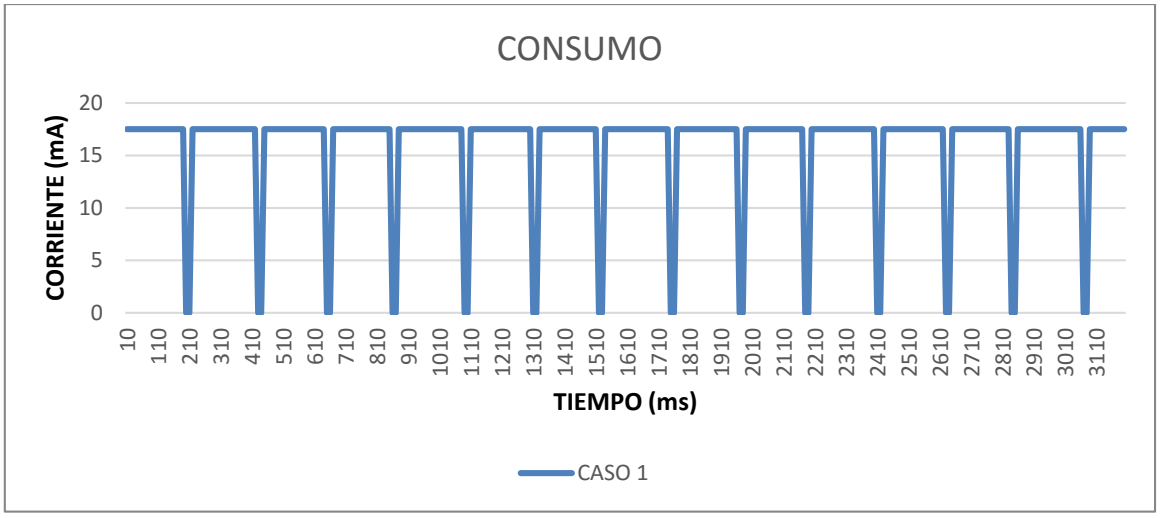

FIGURA 51- CORRIENTE VS. TIEMPO, CONSUMO CASO 1.

El consumo medio de duración de escaneo (escaneo en los tres canales) para el caso 2 es de 0.954 mA.

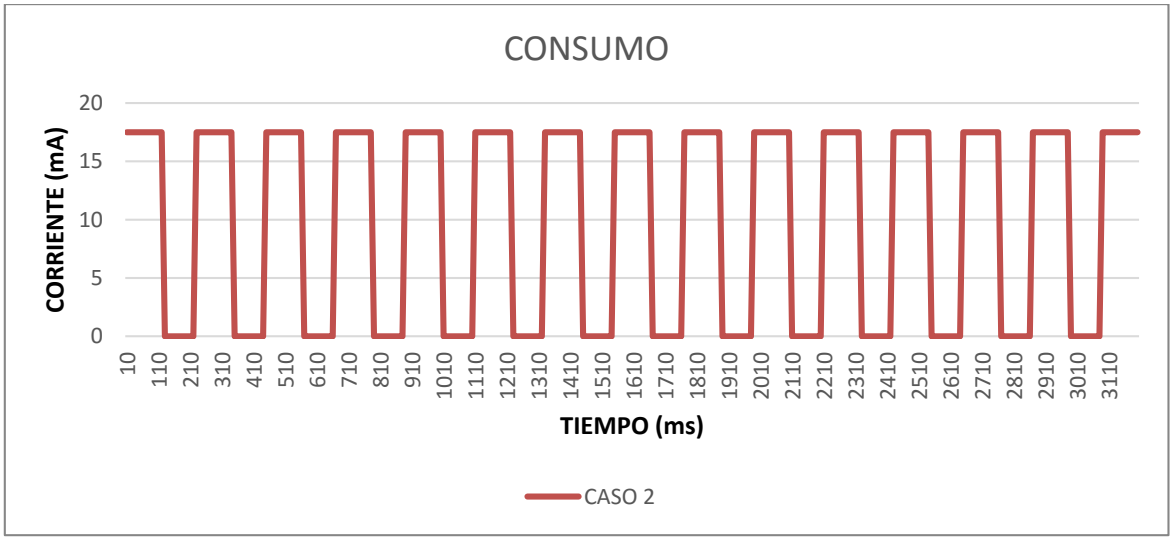

FIGURA 52-CORRIENTE VS. TIEMPO, CONSUMO CASO 2.

El consumo medio de duración de escaneo (escaneo en los tres canales) para el caso 3 es de 0.795 mA.

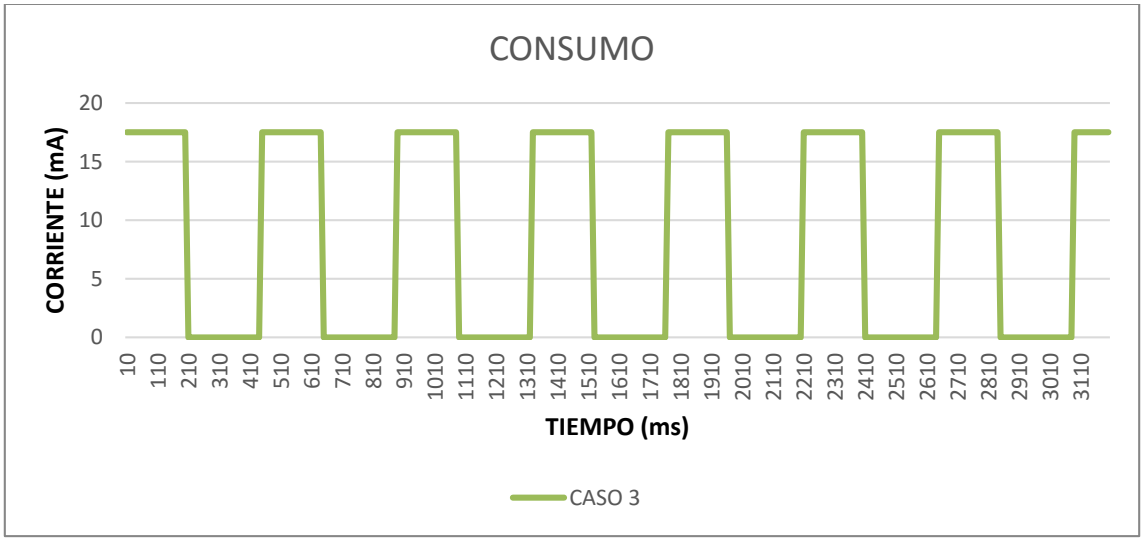

FIGURA 53-CORRIENTE VS. TIEMPO, CONSUMO CASO 3.

El caso 1 corresponde al comportamiento propuesto para el proyecto, donde el consumo no es una limitación, pero sí lo es la rápida detección de los mensajes. En la gran mayoría de proyectos existe un claro compromiso entre consumo y detección de los mensajes. Como podemos ver entre el caso 1 y el caso 3, en el caso 3 las ventanas de escaneo ocurren cada el doble de tiempo que en el caso 1, el consumo también se reduce a la mitad, el mayor consumo se produce durante la ventana de escaneo, ya que, durante ventanas de escaneo, obviando los post y pre procesos, el consumo ronda los 5 µs.

El proceso por el cual cada nodo calcula su distancia en saltos hasta el nodo central ha de producirse de forma continua. Será el proceso de fondo y, cuando se deba mandar información de alguno de los nodos al nodo central, el proceso de cálculo se detendrá. Dada la importancia del proceso de cálculo, tendrá prioridad durante el desarrollo del proyecto.

Cuando alguno de los nodos generé información que deba llegar al nodo central detendrá su proceso de cálculo de ruta y entrará en un proceso de transmisión. En el proceso de transmisión el dispositivo actuará como un rol periférico, tendrá la capacidad de establecer conexiones y de que accedan a su perfil GATT que contiene diferentes servicios, entre ellos el que contiene la información que queremos que llegue al nodo central. En este estado, el nodo enviará mensajes de anuncio conectables y dirigidos al nodo con el que quiere establecer una conexión para que éste lea el servicio que contiene la información a transmitir.

Texas Instruments proporciona un proyecto de trabajo llamado *MasterSlaveSwitch* [18], un

perfil de intercambio de estado ente maestro y esclavo. Es necesario recordar que los roles maestro y esclavo corresponden a la capa de enlace y que indican cuál de los dispositivos propone el inicio de la conexión y cuál lo acepta y está basado en la pila de protocolos para BLE 1.4.1, la más actual de Texas Instruments para los CC254X. Este perfil se basa en el proyecto *SimpleBLEPeripheral* y añade componentes de *SimpleBLECentral* otorgándole al dispositivo la capacidad de cambiar entre los roles periférico y central. Se puede pasar de emitir mensajes de anuncio conectables a escanear para establecer una conexión y viceversa sin límite. Nos vamos a basar en este proyecto que nos proporciona un excelente punto de partida. Lo modificaremos para que el rol periférico sea capaz de enviar mensajes de anuncio del tipo *ADV_NONCONN_IND* y del tipo *ADV_DIRECT_IND.* Se implementará una función que maneje el cambio entre roles, contenido de los mensajes, parámetros de escaneo y anunciado, etc… La Figura 54 ilustra cómo se va utilizar el funcionamiento de este perfil de trabajo para nuestro propósito. Las flechas indican el sentido de los datos.

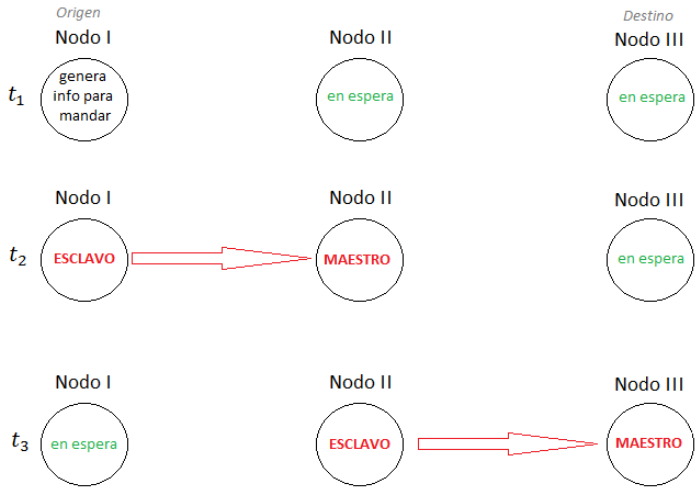

FIGURA 54-MODO DE FUNCIONAMIENTO DEL PERFIL INTERCAMBIO MAESTRO-ESCLAVO.

En el instante de tiempo t_1 el nodo I genera información que desea enviar al nodo central. En su proceso de cálculo de ruta óptima averiguó que su siguiente salto debía ser hacía el nodo II. Para el instante de tiempo t_2 el nodo I envía un mensaje de anuncio conectable dirigido al nodo II, éste al recibir el mensaje de anuncio dirigido a él le envía al nodo I un mensaje de petición de conexión, el nodo I la acepta y se establece la conexión. El nodo II lee la información del servicio del nodo I. Para el instante de tiempo t_3 se repite el proceso del instante de tiempo t_2.

Otra opción planteada ha sido utilizar dos módulos HM-10 unidos a través de una placa intermedia. Cada uno de los módulos actuando en un rol diferente: uno de ellos de periférico (anunciador) y el otro de central (observador).

Adoptar la primera opción reduce el número de módulos HM-10 necesarios y con ello el coste y la complejidad de la red.

En cuanto a la transmisión pura de la información, se usará el perfil GATT creándose un servicio propio, con una única característica. Existe una limitación en cuanto al tamaño máximo del atributo que contiene el valor de la característica. Según la especificación es de 512 bytes, pero de forma práctica el tamaño de carga útil de datos de un paquete en la capa de enlace es de 31 bytes. Además, se ha comprobado que con un tamaño de atributo superior a los 20 bytes

aparecen problemas, por lo que 20 bytes será el tamaño máximo de información que el protocolo podrá transmitir. Se podrían usar más características y que cada una contuviera 20 bytes de información, pero por simplicidad se usará una única característica.

6.3 ESTRUCTURA

Vamos a distinguir dos procesos:

- Cálculo de la ruta óptima: difusión de mensajes de anuncio, que contienen el valor del número de saltos necesario para llegar al nodo central y escucha de esos mensajes con la intención de establecer la ruta óptima hacia el nodo principal, punto de acceso a la red. Deberán alternar entre escanear (rol central) y anunciar (rol periférico) (ver Figura 55).

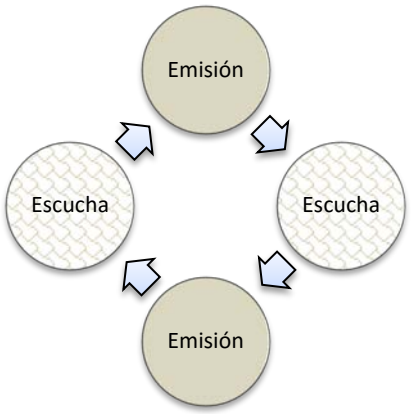

FIGURA 55- ESQUEMA SIMPLE DEL FUNCIONAMIENTO DEL PROCESO DE CÁLCULO DE RUTA ÓPTIMA.

- Proceso de intercambio de información: la información generada por los nodos y que tiene como destino el nodo central se transmite de nodo en nodo en función de la ruta óptima calculada en el proceso anterior. Este proceso se compondrá de cuatro pasos:
    - *Establecimiento de la conexión, rol maestro*: el dispositivo recibe un mensaje de anuncio dirigido a su dirección e inicia una conexión.
    - *Transferencia de la información, rol cliente*: una vez establecida la conexión, el dispositivo actúa en rol GATT cliente leyendo en el servicio del dispositivo con el que inició la conexión.
    - *Establecimiento de la conexión, rol esclavo*: una vez que el dispositivo ya tiene la información a transmitir, leída en el paso anterior, pasa a enviar un mensaje de anuncio dirigido a la dirección del dispositivo que tiene almacenada con la intención de establecer una conexión con él.
    - *Transferencia de la información, rol servidor*: una vez establecida la conexión, el dispositivo actúa en rol GATT servidor y contiene en su servicio la información que debe llegar al nodo central y que el dispositivo con el que inició la conexión debe leer.
    - *Fin de proceso* de transmisión de información y retorno al proceso de cálculo de ruta óptima.

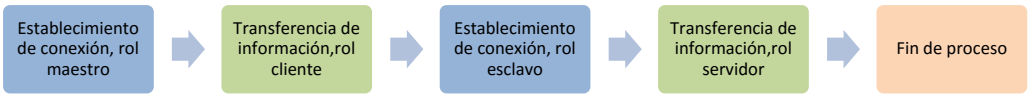

FIGURA 56- ESQUEMA SIMPLE DEL PROCESO DE INTERCAMBIO.

El proceso de cálculo de ruta óptima será un proceso de fondo (ver Figura 56), que se ejecutará continuadamente hasta que el dispositivo genere información para enviar o durante el escaneo se reciban paquetes de anuncio dirigidos. Ambos provocarán que el proceso de transmisión, que se mantenía en espera, se active y sea el proceso de cálculo de ruta óptima el que pase a un modo de espera. No existirá ejecución simultánea de procesos.

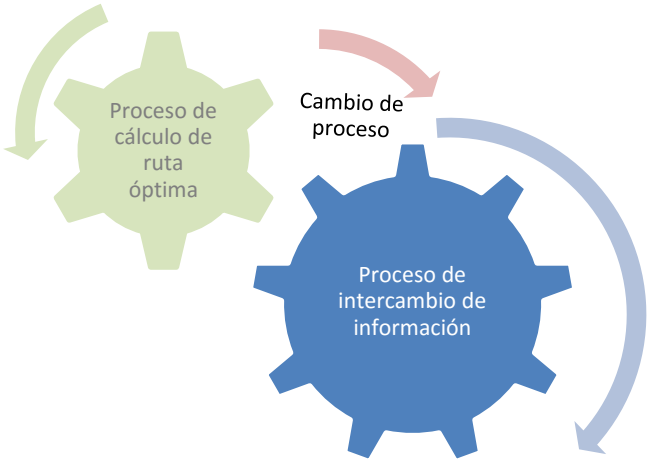

FIGURA 57- ESQUEMAS DE LOS PROCESOS.

A modo de recordatorio, los mensajes de anuncio para difundir el parámetro del número de saltos hasta el nodo central serán del tipo no conectables, no escaneables y no dirigidos (*ADV_NONCONN_IND*). Deben existir otro tipo de paquetes de anuncio para establecer la conexión entre nodos cuando sea necesario transmitir la información hasta el nodo central. Para este caso los paquetes de anuncio serán conectables, no escaneables y dirigidos (*ADV_DIRECT_IND*), dirigidos a la dirección que el nodo tendrá almacenada en su memoria flash y que corresponderá al nodo más cercano en la ruta óptima hacia el nodo central.

Cada uno de los dos procesos generales agrupa una serie de subprocesos que vamos a explicar a continuación. En el Anexo (Desarrollo en IAR Embedded Workbench) se detallarán los pasos seguidos en su implementación, los problemas encontrados durante el mismo y las soluciones propuestas.

Configuración inicial del dispositivo recién arrancado:

- Rol periférico, anunciado.
- Mensajes ADV_NONCONN_IND: no conectables, no escaneables y no dirigidos (queremos difusión de información)
- Duración del periodo de anuncio, 2.5 s.
- Duración del periodo de escaneo, 9.9 s.
- Duración del intervalo de anuncio, 100 ms.
- Duración del intervalo de escaneo, 220 ms.
- Duración de la ventana de escaneo, 200 ms.
- Duración de escaneo, 660 ms.

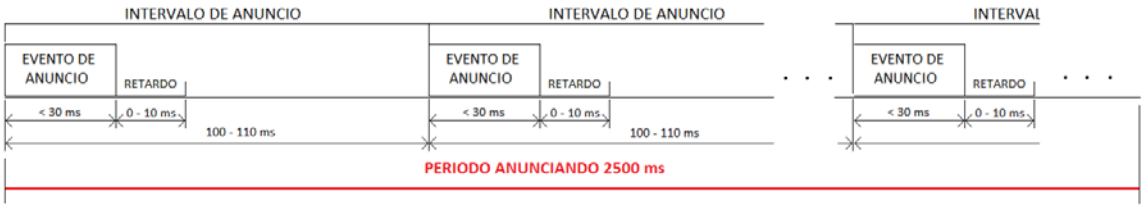

FIGURA 58- INTERVALOS DE ANUNCIO FRENTE A TEMPORIZADOR.

El documento con las especificaciones de BLE [9] que proporciona el SIG en su versión 4.2, muestra en el apéndice A una serie de valores recomendados y requeridos para constantes y temporizadores. Algunos de los que nos interesan son:

TABLA 1- PARÁMETROS GAP.

Nombre del temporizador	Valor	Descripción	Requerimiento o recomendación
$\mathbf{T_{GAP}}$(scan_slow_window1)	11.25 ms	Ventana de escaneo en cualquier proceso de establecimiento de descubrimiento o conexión	Valor recomendado
$\mathbf{T_{GAP}}$(gen_disc_scan_min)	10.24 s	El tiempo mínimo para llevar a cabo el escaneo cuando se está en el procedimiento de detección general	Valor recomendado

$\mathbf{T_{GAP}}$(lim_disc_scan_min)	10.24 s	El tiempo mínimo para llevar a cabo el escaneo cuando se está en el procedimiento de detección limitado	Valor recomendado
$\mathbf{T_{GAP}}$(lim_disc_scan_int)	11.25 ms	Escaneo de intervalo usado durante el procedimiento descubrimiento limitado	Valor recomendado

6.3.1 Proceso de cálculo de ruta óptima

Se van a detallar los diferentes pasos requeridos para llevar a cabo el proceso de cálculo de ruta óptima. La potencia de transmisión preestablecida es de 0 dBm y no se modificará.

6.3.1.1 Almacenado en memoria

El primer paso es definir los valores que se van a almacenar en la memoria flash de cada nodo:

TABLA 2- DATOS ALMACENADOS EN MEMORIA.

Dirección del dispositivo	**Campo Salto**	**Campo RSSI**
6 bytes	1 byte	1 byte

Como podemos ver en la Tabla 1 son tres valores: la dirección del dispositivo que ocupa 6 bytes, el campo salto que ocupa 1 bytes y el campo RSSI que ocupa otro byte. En total son 8 bytes los que tenemos que almacenar.

El SoC CC2541 proporciona a las aplicaciones una forma de almacenar información persistentemente en su memoria flash. El usuario puede leer y escribir un elemento entero, pero no puede leer o escribir parcialmente el elemento. Cada elemento tiene un único ID. Hay un rango de ID específicos para la aplicación, algunos otros ID reservados y otros usados por la pila de protocolos (ver Figura 59).

Para almacenar nuestros datos deberemos usar IDs del rango reservado para las aplicaciones.

VALUE	USER
0x00	Reserved
0x01 – 0x6F	Reserved for ZigBee RF4CE network layer
0x70 – 0x7F	Reserved for ZigBee RF4CE application framework (RTI)
0x80 – 0xFE	Application
0xFF	Reserved

FIGURA 59-VALOR DE IDS Y USOS RESERVADOS. **[19]**

Hay que tener en cuenta que las funciones que se usan para leer y escribir en la memoria no volátil es mejor llamarlas en momentos en los que no entren en conflicto con otras operaciones de sincronización. Un buen momento para escribir sería cuando el receptor está apagado. Es mejor realizar con poca frecuencia las funciones de escritura ya que se necesita tiempo y energía. La mayoría de los dispositivos flash tienen un número limitado de ciclos de borrado, es necesario tener esto en cuenta. Las funciones OSAL necesarias para escribir y leer en la memoria son las siguientes:

```
uint8 osal_snv_write( osalSnvId_t id, osalSnvLen_t len, void *pBuf );
```

- id – identeficador de la posición en la memoria, definida por el usuario (para nuestro caso dentro del rango 0x80-0xFE).
- len – tamaño de los datos a escribir.
- *pBuf – datos a escribir.

```
uint8 osal_snv_read( osalSnvId_t id, osalSnvLen_t len, void *pBuf );
```

- id – identeficador de la posición en la memoria, definida por el usuario (para nuestro caso dentro del rango 0x80-0xFE).
- len – tamaño de los datos a leer.
- *pBuf – datos a leer.

6.3.1.2 Recepción de paquetes de anuncio: extracción de información

Una vez explicado cómo se almacenarán los datos nos queda aclarar cómo se obtendrán. Los obtendremos de los mensajes de anuncio envidaos por los nodos. La dirección está incluida en la carga útil del paquete, el campo salto en los datos de la carga útil del paquete (ver Figura 13) y el campo RSSI lo calcula la capa física al recibir el paquete.

Existen una serie de eventos relacionados con la capa de GAP que se puede devolver a la aplicación desde la pila BLE. Algunos de estos eventos se transmitirán directamente a la aplicación y otros los manejarán las capas de GAPRole o GAPBondMgr.

Para leer el contenido de los paquetes de anuncio recibidos haremos uso del evento:

(GAP_DEVICE_INFO_EVENT)

El dispositivo escaneando recibe un paquete de anuncio. Su capa física se lo pasa a la capa de enlace y esta genera un evento (*GAP_DEVICE_INFO_EVENT)* y un elemento tipo *struct*

(*gapDeviceInfoEvent_t*) para la capa GAP. Este elemento contiene, entre otros valores: el tipo de mensaje de anuncio, el tipo de dirección, la dirección del anunciador, el valor del RSSI, los datos que contenidos en el paquete

La Figura 60 muestra una captura de la descripción que da Texas Instruments del contenido y tipo de cada variable:

- GAP_DEVICE_INFO_EVENT: Sent during the Device Discovery Process when a device is discovered.

```
typedef struct
{
  osal_event_hdr_t  hdr;     //!< GAP_MSG_EVENT and status
  uint8 opcode;              //!< GAP_DEVICE_INFO_EVENT
  uint8 eventType;           //!< Advertisement Type: @ref GAP_ADVERTISEMENT_REPORT_TYPE_DEFINES
  uint8 addrType;            //!< address type: @ref GAP_ADDR_TYPE_DEFINES
  uint8 addr[B_ADDR_LEN];    //!< Address of the advertisement or SCAN_RSP
  int8 rssi;                 //!< Advertisement or SCAN_RSP RSSI
  uint8 dataLen;             //!< Length (in bytes) of the data field (evtData)
  uint8 *pEvtData;           //!< Data field of advertisement or SCAN_RSP
} gapDeviceInfoEvent_t;
```

FIGURA 60- DESCRIPCIÓN DEL CONTENIDO DE GAPDEVICEINFOEVENT_T. **[20]**

Cuando ocurra el evento se llamará a una función que se encargará de procesar los datos que nos hacen falta. Esta función tendrá como parámetros de entrada *addr[B_ADDR_LEN]*, *rssi*, *dataLen* (necesario para saber la longitud del campo datos y poder extraer el byte que contiene el campo salto) y **pEvtData.*

6.3.1.3 Función de procesado de datos

Ésta es la función presentada en el punto 6.3.1.2 y que ahora explicamos en profundidad. Será la encargada de procesar los datos extraídos de los paquetes de anuncio y compararlos con los almacenados, que como configuración inicial valdrán:

TABLA 3-VALORES INICIALES.

Dirección del dispositivo	Campo Salto	Campo RSSI
0x000000000000	0xFF	0x00

Leerá y escribirá en la memoria no volátil haciendo uso de las funciones presentadas en el punto 6.3.1.1. El nodo central no incluirá esta función, el contenido de su mensaje de anuncio será invariable.

Al llamar a la función se le pasarán las variables: *addr[B_ADDR_LEN]*, *rssi*, *dataLen* y **pEvtData.* A modo de filtrado de paquetes de anuncio de otras direcciones que no nos interesen se colocará un byte con valor 0xBB, en hexadecimal (187 en decimal) antes byte del campo salto en el contenido del paquete. El byte de control y el byte de campo salto estarán colocados en la primera y segunda posición, respectivamente, de la carga útil del paquete para simplificar su localización. Queremos minimizar las funciones de escritura en la memoria. El diagrama de flujo de la función es el siguiente:

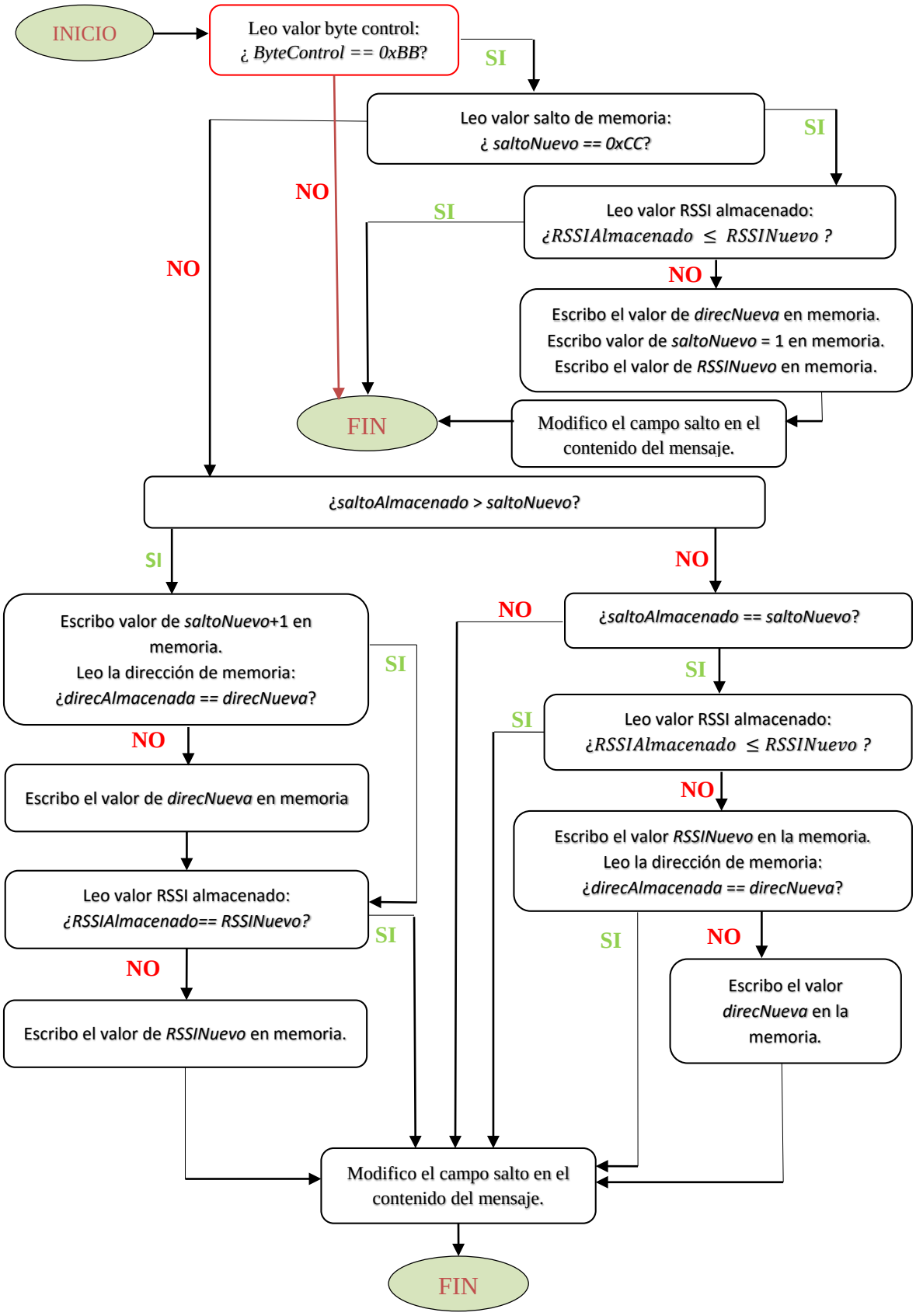

FIGURA 61- DIAGRAMA DE FLUJO DE LA FUNCIÓN.

6.3.1.4 Función de conmutación entre roles periférico y central

Queremos que un nodo durante un periodo de tiempo emita mensajes de anuncio y durante otro periodo de tiempo escanee en busca de mensajes de anuncio de otros nodos y que de nuevo pase a emitir y así continuamente. Éste es el requerimiento principal que nos hace elegir el proyecto *MasterSlaveSwitch* como punto de partida. Recordar que este proyecto permite pasar de emitir mensajes de anuncio conectables a escanear para establecer una conexión y viceversa sin límite. Este proyecto está pensando para el dispositivo mando/llavero (ver Figura 36) y que el cambio entre los roles se produzca al pulsar el botón izquierdo. Para nuestro caso el cambio entre los roles ha de ser automático, así que haremos uso de temporizadores.

Se van a utilizar dos temporizadores. El primero de ellos controlará el tiempo de emisión de paquetes, por tanto, el tiempo que el dispositivo se encuentra en rol periférico y el segundo temporizador controlará el tiempo que estamos escaneando, rol central.

Cuando inicializas un temporizador ajustas el tiempo que tarda en expirar y el identificador de evento, valor definido por el usuario y que se devolverá cuando el tiempo se agote pudiendo así reconocer la tarea. Existen dos tipos de temporizadores.

El primero. Se inicializa el temporizador y cuando este expira el temporizador termina se genera el bit del evento definido por el usuario. Cuando expira el temporizado este no se recarga:

```
uint8 osal_start_timerEx(uint8 taskID, uint16 event_id, uint32 timeout_value );
```

- taskID es el identificador de tarea de la tarea que se va a obtener cuando expire el temporizador
- event_id es el bit del evento definido por el usuario. Cuando el tiempo se agota, al usuario se le notificará la tarea de llamada.
- timeout_value es la cantidad de tiempo (en milisegundos) antes de que se establezca el evento de temporizador, el tiempo de duración del temporizador.

El otro, inicializa el temporizador que cuando expira genera el evento con el ID de la tarea definida. Cuando expira el temporizador este se actualiza automáticamente:

```
uint8 osal_start_reload_timer(uint8 taskID, uint16 event_id, uint32
timeout_value );
```

- taskID es el identificador de tarea de la tarea que se va a obtener cuando expire el temporizador
- event_id es el bit del evento definido por el usuario. Cuando el tiempo se agota, al usuario se le notificará la tarea de llamada.
- timeout_value es la cantidad de tiempo (en milisegundos) antes de que se establezca el evento de temporizador, el tiempo de duración del temporizador.

PROBLEMÁTICA ENCONTRADA

A la hora de implementar los dos temporizadores surgieron imprevistos. El cambio entre roles no era periódico como debería haber sido. Además, pasado un tiempo, que era dependiente del valor de los temporizadores, el dispositivo dejaba de emitir mensajes. Como solución se eliminó uno de los temporizadores pasando a trabajar con un único temporizador. El valor del periodo

con el que se reiniciaba el temporizador al expirar cambiaba en función del rol en el que nos encontráramos.

6.3.2 Proceso de intercambio de información

El segundo de los procesos fundamentales del protocolo define la forma en la que los dispositivos interactúan para intercambiar la información que ha de llegar al nodo central.

El proceso de cálculo de ruta óptima estará ejecutándose de forma continua hasta que el dispositivo tenga información que quiere hacer llegar al nodo central, esto está pensado para poder implementar el protocolo en una red de sensores que cada cierto tiempo o bajo ciertas circunstancias recojan datos y quieran transmitirlos. Para ello el dispositivo que quiere enviar la información enviará un paquete de anuncio conectable dirigido (ADV_DIRECT_IND), dirigido al a la dirección Bluetooth que tiene almacenada en memoria y que corresponde con el siguiente salto que debe dar la información para llegar de la manera más rápida al nodo central. La recepción de este paquete por parte del escáner objetivo, que detiene su proceso de cálculo e inicia el de intercambio, provoca que envié un CONNECT_REQ para iniciar la conexión.

En el modo en que se inicia el proceso de transmisión se deben distinguir dos casos, uno particular y otro general. El particular, se da en el dispositivo iniciador de la transmisión, es decir, el que genera la información a transmitir. Para este caso particular los dispositivos inician el proceso de intercambio de información directamente en el sub proceso de establecimiento de la conexión, rol esclavo (ver Figura 58). El caso general, se da para el resto de nodos que actúan como puentes reenviando la información recibida y en este caso el proceso de intercambio comienza desde el sub proceso de establecimiento de conexión, rol maestro (ver Figura 58). Esta diferencia solamente afecta al comienzo del establecimiento de conexión.

FIGURA 62- PROCESO DE INTERCAMBIO CON DETALLE DE INICIO DE CASO GENERAL Y PARTICULAR.

Como podemos ver en la Figura 62 son cinco subprocesos los que forman el proceso de intercambio de información. Los dos procesos de establecimiento de conexión funcionan de la misma manera, pero un nodo tiene papeles diferentes en cada uno de ellos, por eso se explican de manera separada.

6.3.2.1 Establecimiento de conexión, rol maestro

Ciertos valores de algunos de los parámetros configurados para el proceso de cálculo de ruta óptima han de ser modificados, como el tipo de mensaje de anuncio que ha de pasar de ADV_NONCONN_IND (paquetes no conectables, no dirigidos y no escaneables) a ADV_DIRECT_IND (paquetes conectables, dirigidos y no escaneables).
Una función será la encargada de modificar estos parámetros.
En la Figura 63 podemos observar al nodoB, que es el nodo que se encuentra en el proceso de establecimiento de conexión, rol maestro. El nodoB se encuentra escaneando en el proceso de cálculo de ruta óptima justo antes de recibir el paquete de ADV_DIRECT_IND del nodoA dirigido a él. En cuanto lo reciba, el nodoB pasará al proceso de intercambio de información y, dentro de este proceso, al subproceso de establecimiento de conexión, rol maestro. En este subproceso enviará una petición de conexión al nodoA y se establecerá la conexión. El nodoB es rol maestro en la capa de enlace (LL), ya que es el que inicia la conexión.

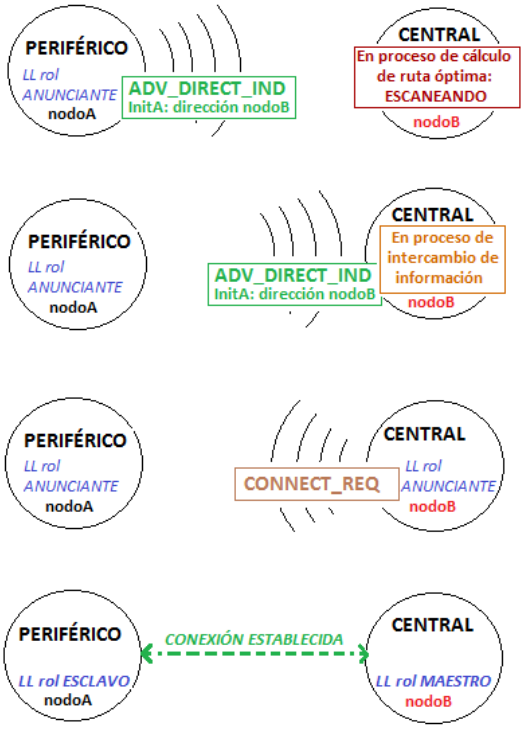

FIGURA 63- FASES DEL PROCESO DE ESTABLECIMIENTO DE CONEXIÓN, ROL MAESTRO.

Una función procesará el tipo de paquete de anuncio recibidos. Cuando sea un paquete de anuncio dirigido a su dirección se pasará al proceso de intercambio de información, paralizando el proceso de cálculo de ruta óptima.

6.3.2.2 Transferencia de información, rol cliente

Una vez establecida la conexión debemos conseguir que el nodoB obtenga del nodoA la información que se desea que llegue al nodo central. Para este proceso haremos uso de los perfiles GATT, que es la forma en la que las aplicaciones de los dispositivos intercambian los

datos en BLE. Cada nodo contendrá entre sus servicios uno en el que almacenar la información a transmitir. Se llamará servicio de Protocolo. Su estructura será la siguiente:

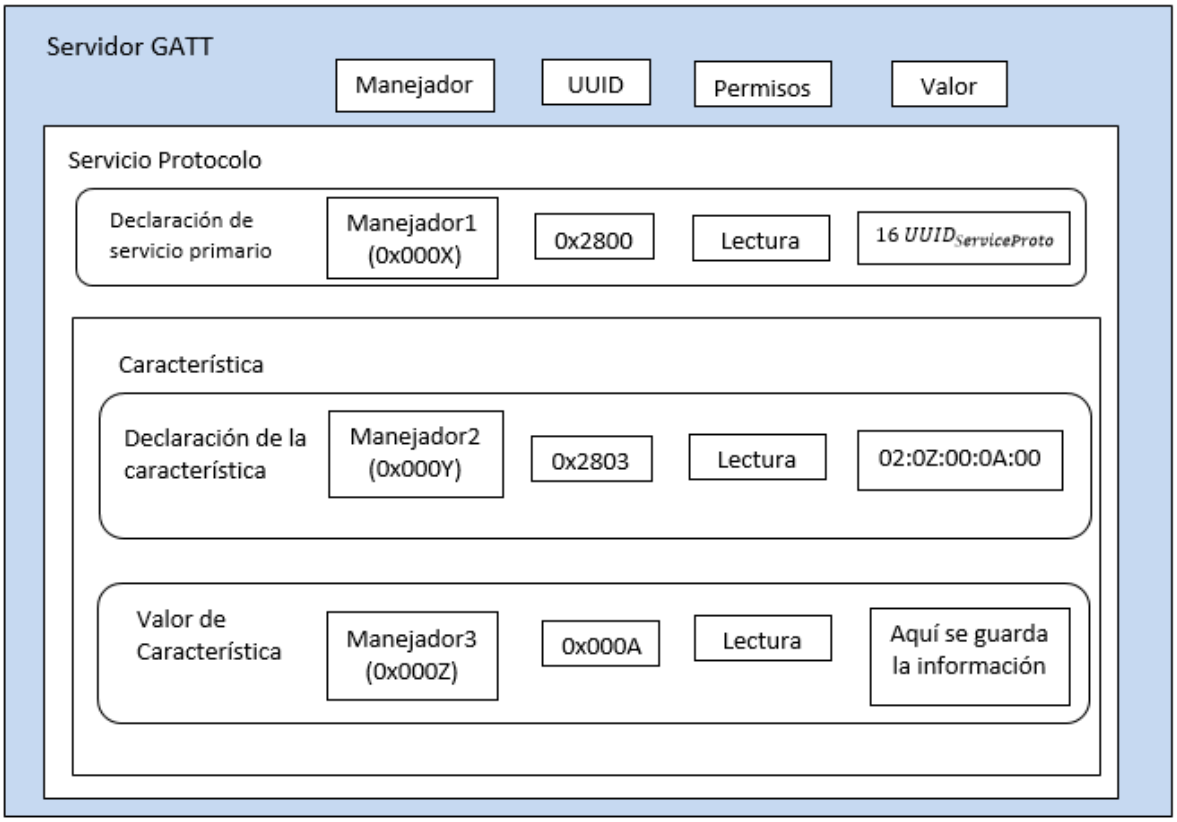

FIGURA 64- ESTRUCTURA DEL SERVICIO PROTOCOLO DEL SERVIDOR GATT.

El valor del manejador lo configura el dispositivo al arrancar, no es un valor que se preconfigure. En la Figura 64 se hacen referencia a ellos enumerándolos, siendo X>Y>Z.
El valor del atributo *declaración de la característica* contiene en su valor información sobre el atributo *valor de característica*:

- 02: el atributo valor solamente tiene permisos de lectura.
- 0Z:00: es el manejador del atributo que contiene el valor (0x000Z).
- 0A:00: es el UUID del atributo que contiene el valor (0x000A).

El UUID de 16 bytes para el servicio Protocolo ha sido generado en la página web de la ITU http://www.itu.int/en/ITU-T/asn1/Pages/UUID/uuids.aspx.

$$16\ UUID_{SERVICE\ PROTOCOL} = \text{14D93400-2BB6-11E6-9DCA-0002A5D5C51B}$$

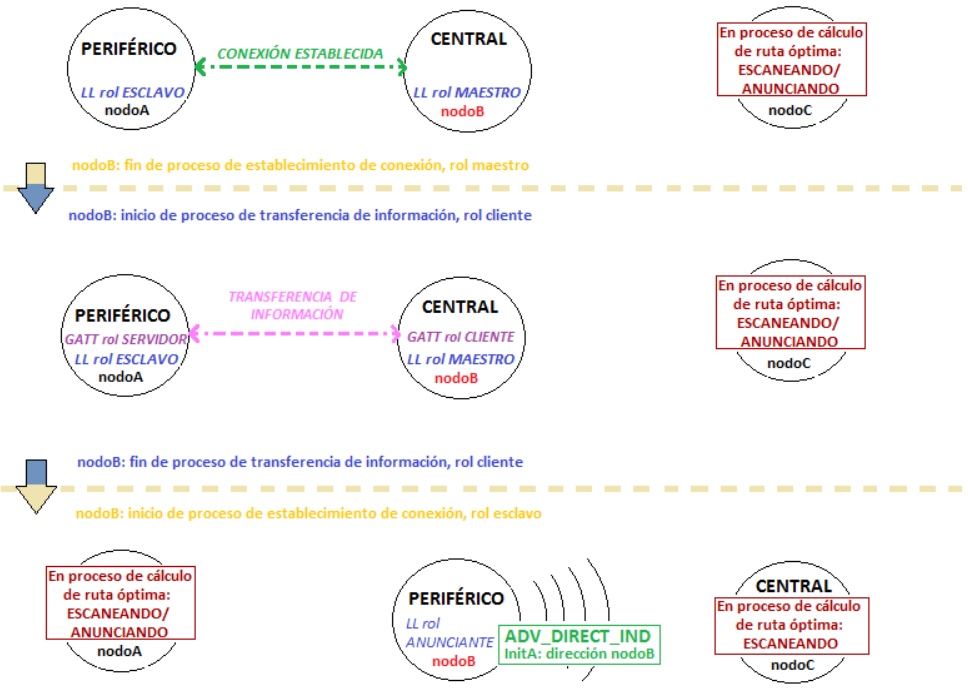

FIGURA 65- PROCESO DE TRANSFERENCIA DE INFORMACIÓN, ROL CLIENTE.

En la Figura 65 podemos ver como se sitúa en el tiempo el proceso de transferencia de información rol cliente respecto al resto de procesos.

Una función leerá los servicios GATT del nodoA filtrando por UUID, 16 $UUID_{ServiceProto}$ y valor de la única característica del servicio, que es la información que debe llegar al nodo central. La función modificará el servicio Protocolo del nodoB escribiendo en el atributo que contiene en su valor el valor de la característica. Con la confirmación de que la lectura del servicio GATT del nodoA ha sido correcta se terminará con la conexión entre ambos nodos.

6.3.2.3 *Establecimiento de conexión, rol esclavo*

En este subproceso el nodo es el generador del paquete de anuncio dirigido a la dirección del siguiente salto que tiene almacenada en su memoria flash y que calculó durante el proceso de cálculo de ruta óptima.

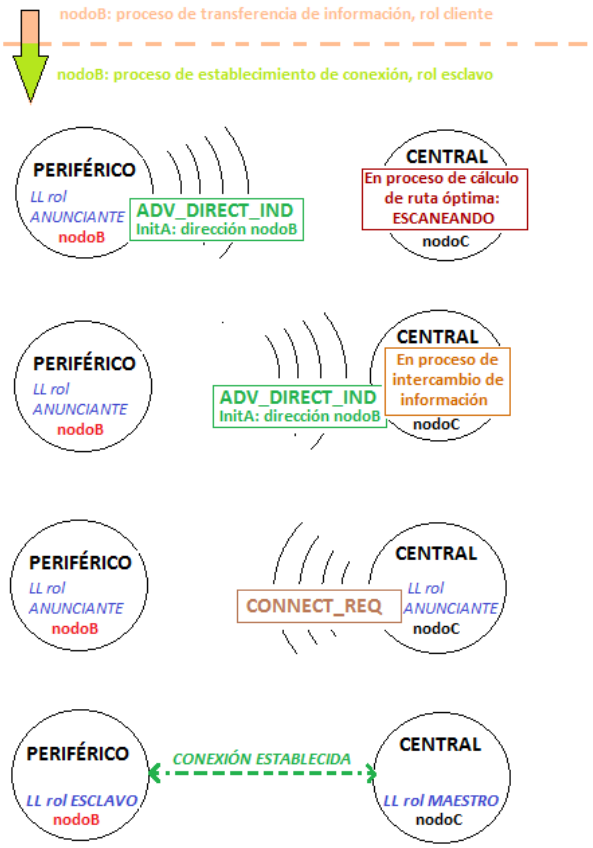

FIGURA 66- FASES DEL PROCESO DE ESTABLECIMIENTO DE CONEXIÓN, ROL ESCLAVO.

En la Figura 66 podemos observar al nodoB, que se encontraba en el proceso de transferencia rol cliente, en el que obtuvo la información a transferir, y que ahora ha pasado al proceso de establecimiento de conexión rol esclavo, para pasar esa información que ha obtenido al siguiente nodo. A diferencia de lo que ocurre en el proceso de establecimiento de conexión rol maestro, ahora el nodoB es el que envía el mensaje de anuncio dirigido para que el nodo objetivo establezca la conexión con él.

Una función se encargará de llevar a cabo el cambio entre roles para enviar el paquete de anuncio dirigido. Leerá de la memoria la dirección almacenada y modificará el contenido de los paquetes de anuncio. Esta función será llamada al terminarse la conexión entre los nodos que intercambian la información en el proceso de transferencia de información rol cliente.

6.3.2.4 *Trasferencia de información, rol servidor*

Este proceso es parecido al proceso anterior de transferencia, pero ahora el nodoB hará de rol servidor conteniendo la información, información que será leída por el nodoC. En la figura 63 podemos situar el proceso de transferencia de información respecto a los procesos que le siguen y preceden.

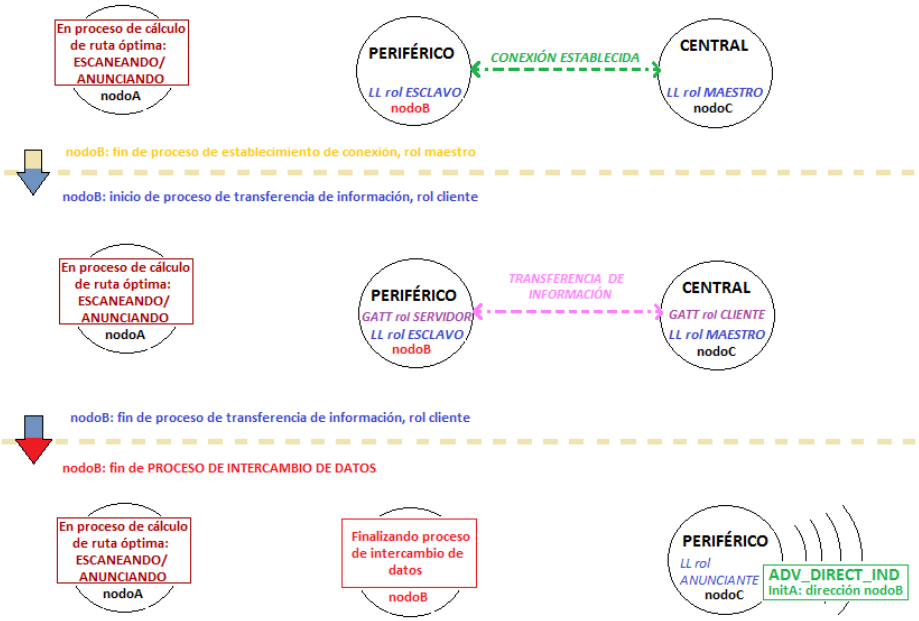

FIGURA 67- PROCESO DE TRANSFERENCIA DE INFORMACIÓN, ROL SERVIDOR.

Este proceso conlleva una actitud pasiva, es decir el nodoB deja que lean su contenido.

6.3.2.5 *Fin de transmisión, retorno al proceso cálculo de ruta*

Una vez la lectura por parte del nodoC del servicio de protocolo del nodoB haya terminado, el nodoC finalizará la conexión, una función captará esto y retomará el proceso de cálculo de ruta óptima, dando por finalizado el proceso de intercambio de información.

6.4 ESTADO DE LA IMPLEMENTACIÓN

En este apartado se explican tanto las tareas definidas en la propuesta de proyecto como los hitos técnicos planteados durante el proyecto y cuáles se han completado.

En la propuesta del proyecto se plantearon varias tareas a desarrollar, eran las siguientes:

- *Estudio del Arte y del Mercado*: sondeo previo de la situación tecnológica actual y de las distintas plataformas y productos disponibles en el mercado antes de definir en detalle el alcance y los objetivos del proyecto.

- *Redefinición de Objetivos y alcance de los desarrollos a realizar*: consiste en la elaboración de los requisitos y especificaciones del sistema a implementar y la descripción de las funcionalidades a cubrir por el mismo.

- *Diseño de Alto y Bajo Nivel*: en este apartado se definirán los esquemas de bloques con sus funciones y la interacción entre ellos, tanto en lo relativo al hardware a desarrollar (si fuere necesario) como a la programación de los procedimientos correspondientes (firmware y/o software) asociados.

- *Desarrollo*: consiste en el desarrollo de la plataforma electrónica necesaria, así como la implementación de código descrito para el funcionamiento de la misma. También puede incluir el desarrollo de herramientas software de alto nivel para PC o dispositivos móviles que ayuden a la mejor comprensión del funcionamiento del sistema y a la demostración de funcionalidades.

- *Plan de pruebas*: Consiste en la definición y ejecución de un plan en el que se detallen las diferentes pruebas a realizar junto con una valoración de las mismas y de sus resultados.

- *Conclusiones*: es una de las tareas más importantes, y consiste en la valoración del trabajo realizado, así como la redefinición del alcance de los desarrollos realizados y la posible definición de nuevos objetivos, tanto en el campo del I+D como de la creación, industrialización y comercialización de nuevos productos.

Se observa que estas tareas cumplen con todos los pasos requeridos para el desarrollo de cualquier proyecto técnico. Con el diseño completo, se definieron diferentes hitos técnicos a cumplir en el proceso de desarrollo de la programación. Estos son los hitos principales, los tres primeros corresponderían al proceso de cálculo de ruta óptima y los dos siguientes al proceso de intercambio de información:

- *Cambio de rol dinámico*: el cambio entre rol maestro y rol esclavo debe ser dinámico, manejado por un temporizador. Ajustar los parámetros de escaneo y anunciado.
- *Procesado de los paquetes recibidos*: capacidad para procesar los paquetes recibidos así como su contenido.
- *Implementación de la función de almacenamiento y modificación de datos*: la encargada de procesar el contenido de los paquetes, escritura y lectura en memoria.
- *Configuración de la conexión*: gestión de la conexión con modificación de parámetros para poder establecerla.
- *Procesado de la información a transmitir*: proceso de tratamiento de la información a transmitir al nodo central.

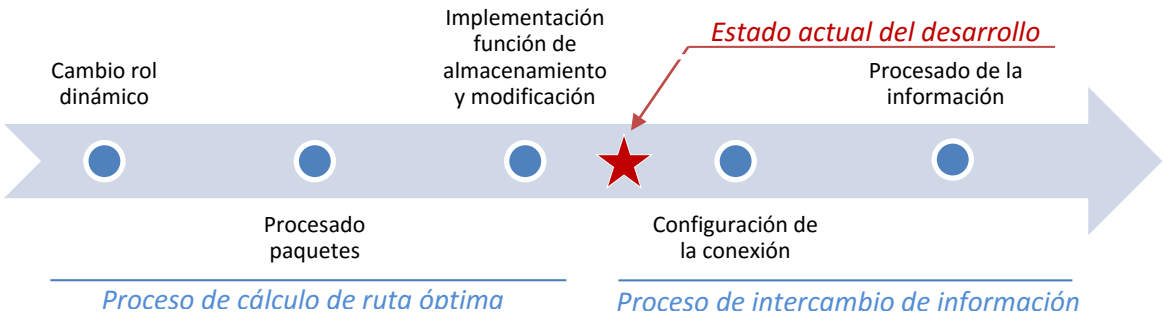

FIGURA 68- ESTADO DEL DESARROLLO TÉCNICO, HITOS.

En la Figura 68 la flecha representa el eje temporal y la estrella roja el punto donde se encuentra el desarrollo técnico al finalizar el tiempo propuesto para realizar el TFG. Tal y como se observa, el proyecto se encuentra en el punto anterior al proceso de intercambio de información, no habiendo completado todos los hitos establecidos inicialmente. El proceso de cálculo de ruta óptima se ha implementado, comprobando el correcto funcionamiento mediante pruebas. Para poder llevar a cabo las pruebas de funcionamiento se han usado otros dispositivos además de los incluidos en el kit de desarrollo de Texas Instruments (ver apartado 5.2.2.2). Estos dispositivos se han creado utilizando los módulos HM-10, soldados a una placa especialmente diseñada junto a unos pines para descargar el firmware y unos cables para alimentarlo (ver Figura 69).

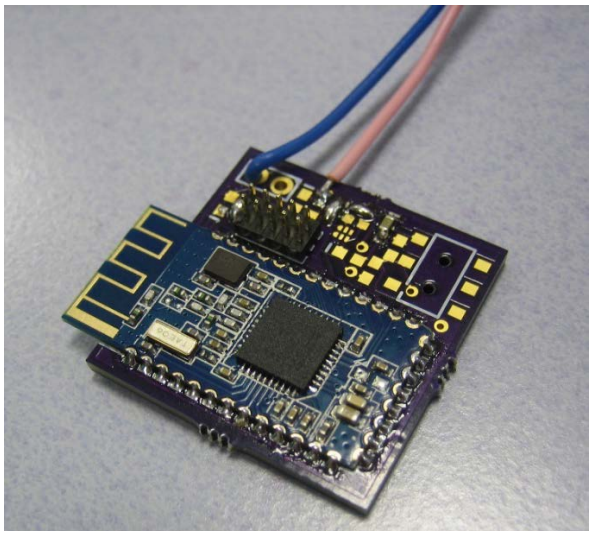

FIGURA 69- MÓDULO HM-10 MONTADO EN PLACA JUNTO A PINES Y CABLES DE ALIMENTACIÓN. DISPOSITIVO PARA PRUEBAS.

La placa con circuito impreso que vemos en la Figura 69 está diseñada para soportar un HM-10, unos pines, un interruptor y un puerto micro USB, aunque solamente se han montado los pines y el módulo HM-10. En la siguiente figura se puede ver el *layout* del diseño:

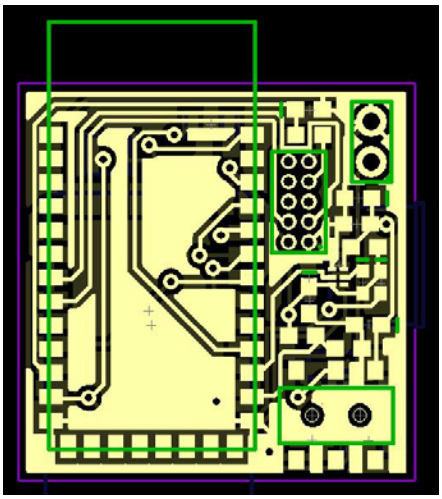

FIGURA 70- LAYOUT DE LA PLACA CON CIRCUITO IMPRESO.

El rectángulo verde más grande situado a la izquierda en la Figura 70 corresponde al HM-10, el rectángulo verde más pequeño con ocho círculos corresponde a los pines, el de la esquina superior derecha son las conexiones para la alimentación y el de la parte inferior es donde iría conectado el interruptor. El puerto micro USB estaría conectado por la cara del *layaout.*

La fabricación se llevó a cabo a través de la web https://oshpark.com. Esta web está pensada para que la gente mande sus pequeños diseños de PCB y, cuando han completado un panel lo mandan fabricar, de forma que se divide el coste entre todos los participantes consiguiéndose hacer placas de circuito económicas. En este caso 3 PCB de nuestro diseño costaron alrededor de 5 dólares y tardaron en llegar casi tres semanas desde que se realizó el pedido.

7 CONCLUSIONES Y LÍNEAS FUTURAS

El crecimiento e implantación del internet de las cosas trae consigo la llegada de miles de dispositivos que se comunican de forma inalámbrica. Dispositivos en su gran mayoría de poco tamaño y alimentados con pilas o baterías, colocados en lugares de todo tipo: tiendas, hogares, fábricas, entornos rurales, etc. Un ejemplo sería una red de sensores distribuida por una fábrica en la que fuese necesario actualizar el firmware (microprograma) porque el tratamiento de los datos se ha modificado, por ejemplo. Una actualización vía cable sería muy completa teniendo que ir dispositivo por dispositivo. De ahí la importancia de la actualización vía aire, en inglés mayormente conocida como *over the air programming (OAT)* o *over the air download (OAD).* La OAT se ha utilizado en los últimos años para la reprogramación de dispositivos móviles, tales como teléfonos celulares. Sin embargo, ahora está cambiando de dirección y su importancia ha aumentado.

De cara a mejorar el protocolo desarrollado en este proyecto, una de las líneas de trabajo sería la creación de un perfil de OAD, para poder actualizar el *firmware* de los nodos de forma inalámbrica. Una gran herramienta que facilitaría introducir mejoras y hacer pruebas.

Sería interesante que el protocolo no almacenará una única dirección de siguiente salto, si no que tuviese una segunda (o incluso tercera) que asegurase una vía secundaría en caso de no poder establecer conexión de manera inmediata con el dispositivo más óptimo.

En este proyecto el consumo no era una de las limitaciones a la hora del desarrollo, pero en un entorno real en el que los nodos estuvieran alimentados por baterías sería necesario tener algún tipo de herramienta de control del estado de esas baterías o pilas. Se podría introducir un servicio GATT de batería (ver Anexo 10.3.1) que al sobrepasar un límite de carga concreto enviase un aviso al nodo central utilizando el protocolo de enrutamiento. Este aviso contendría la carga de la batería y el dispositivo que generó el aviso. Por tanto, haría falta crear un modo de diferenciar a los dispositivos unívocamente. Algún tipo de nomenclatura que relacionase las distancias entre los nodos de manera que facilitara el trabajo de localizarla físicamente para que el técnico encargado de sustituir las baterías lo tuviera fácil. Habría que tener en cuenta que la capacidad del protocolo de soportar la entrada y salida automática de nodos a la red no puede verse afectada. Se podría colocar un led al módulo que sirviera como elemento visible en caso de querer localizarlo y que parpadeara de una manera determinada, por ejemplo.

El protocolo podría ser bidireccional, es decir, que desde un equipo conectado al nodo central pudiéramos interactuar con cierto nodo. Esto abriría la puerta a nuevas implementaciones.

En este proyecto, por su simplicidad y requerimientos, no se ha implementado ningún tipo de seguridad o codificación, pero BLE tiene formas y niveles de codificación de canal o de encriptado entre extremos. Otra vía de mejora sería alcanzar ciertos niveles de seguridad de cara a una implantación real del protocolo.

Bluetooth Low Energy nos permite, por su estructura, adaptar su funcionamiento a infinidad de situaciones. Dada su gran implantación, podemos encontrarlo en prácticamente todos los teléfonos inteligentes, abriendo la posibilidad a muchas aplicaciones. A priori se puede pensar que el uso del teléfono va ligado a aplicaciones de ocio, a nivel de usuario, pero también es una herramienta útil en el entorno laboral. Sólo la falta de ideas novedosas puede frenar este

crecimiento.

8 ABREVIATURAS

API	Application Programming Interface
ARM	Advanced RISC Machine
BR	Basic Rate
CPU	Central Processing Unit
EDR	Enhanced Data Rate
I2C	Inter-Integrated Cirtcuit
IC	Integrated Circuit
IDE	Integrated Development Environment
ITU	International Telecommunication Union
LE	Low Energy
MCU	Microcontroller
OSAL	Operating System Abstraction Layer
PCB	Printed Circuit Board
PDU	Protocol Data Unit
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
SDK	Software Development Kit
SoC	System on chip
SPI	Serial Peripheral Interface
TI	Texas Instruments
UART	Universal Asynchronous Receiver-Transmitter

9 BIBLIOGRAFÍA

9.1 REFERENCIAS

[1] LoRa Alliance, «lora-alliance,» [En línea]. Available: https://www.lora-alliance.org/What-Is-LoRa/Technology.

[2] Libelium, «libelium,» [En línea]. Available: http://www.libelium.com/.

[3] Embeblue, «Embeblue,» [En línea]. Available: http://www.embeblue.com/.

[4] W. Wong, «Bluetooth Mesh, Far-field Voice Processing, and More at CES 2016,» electronic design, 1 Febrero 2016. [En línea]. Available: http://electronicdesign.com/embedded/bluetooth-mesh-far-field-voice-processing-and-more-ces-2016.

[5] Bluetooth SIG, «new Bluetooth® specifications enable ip connectivity and deliver industry-leading privacy and increased speed,» [En línea]. Available: https://www.bluetooth.com/news/pressreleases/2014/12/03/new-bluetoothspecifications-enable-ip-connectivity-deliver-industry-leading-privacy-increased-speed.

[6] K. Townsend, C. Cufí, R. Davidson y Akiba, Getting Started with Bluetooth, Sebastopol, United States of America: O'Reilly Media, Inc, 2014.

[7] G. Litovsky, «edn,» 2015. [En línea]. Available: http://www.edn.com/design/wireless-networking/4439356/A-look-into-Bluetooth-v4-2-for-Low-Energy-Products.

[8] G. Reiter, «Wireless connectivity for,» Texas Instruments , Dallas, Texas, 2014.

[9] Bluetooth SIG, Specification of the Bluetooth System, Covered Core Package version: 4.2, Bluetooth SIG, 2014.

[10] D. Smith, «possiblemobile,» [En línea]. Available: https://possiblemobile.com/2013/12/bluetooth-smart-for-android/.

[11] T. Instruments, Supplement to the Bluetooth Core Specification, CSS version: 6, Texas Instruments , 2015.

[12] «ARGENOX,» 26 Agosto 2015. [En línea]. Available: http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-guide-to-selecting-a-bluetooth-chipset/.

[13] Nordic Semiconductor , [En línea]. Available: https://www.nordicsemi.com/eng.

[14] Laird , «lairdtech,» [En línea]. Available: http://www.lairdtech.com/products/BL600-series.

[15] Texas Instruments, «LPRF San Diego Bluetooth Low Energy Deep Dive,» [En línea]. Available: https://www.google.es/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=0ahUKEwjA4K2p9PnMAhVH1BoKHcxNDzYQFggpMAA&url=https%3A%2F%2Fe2e.ti.com%2Fsupport%2Fwireless_connectivity%2Fm%2Fvideos__files%2F653593%2Fdownload&usg=AFQjCNHiiS_6QLoK3Tt0eDetQ.

[16] Texas Instruments , Bluetooth Low Energy Beacons - document number: SWRA475, San Diego, California, USA: Texas Instruments , 2015.

[17] S. K. &. J. Lindh, *Application Note AN092,* Dallas: Texas Instruments, 2012.

[18] Instruments, Texas, «ti,» CC2541 Mini Development Kit, [En línea]. Available: http://www.ti.com/tool/cc2541dk-mini.

[19] I. Texas Instruments, «OS Abstraction Layer Application Programming Interface,» de *Document Number: SWRA194*, San Diego, California USA, Texas Instruments, Inc., 2005-2015, p. 27.

[20] Texas Instruments, «TI CC254x Bluetooth Low Energy Software Developer's Guide - SWRU271G Version 1.4.1,» Texas Instruments, 2015.

[21] P. D. M. Güneş, *Embedded Internet and the Internet of Things,* Institute of Computer Science Freie Universität Berlin.

[22] J. Schönwälder, *Internet of Things:802.15.4, 6LoWPAN, RPL, COAP,* Bremen, Alemania, 2010.

[23] J. H. David E. Culler, *6LoWPAN Tutorial, IP on IEEE 802.15.4 Low-Power Wireless Networks,* Arch Rock Corporation.

[24] M. Havlena, «havlena,» 27 03 2014. [En línea]. Available: http://www.havlena.net/en/location-technologies/ibeacons-how-do-they-technically-work/.

[25] T. Instruments, CC2540/41 System-on-Chip Solution for 2.4- GHz Bluetooth® low energy Applications, User's Guide, 2009- Revisado 2014.

[26] R. Faragher y R. Harle, «Location Fingerprinting With Bluetooth,» *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS,* vol. 33, nº 11, p. 2419, 2015.

[27] T. Instruments, Bluetooth® Low Energy Beacons, http://www.ti.com/lit/an/swra475/swra475.pdf, Junio 2015.

[28] Texas Instruments , «Processors Wiki,» MasterSlaveSwitch, [En línea]. Available: http://processors.wiki.ti.com/index.php/MasterSlaveSwitch.

9.2 PÁGINAS WEB CONSULTADAS

[w1] iee802, «iee802.15,» [En línea]. Available: http://www.ieee802.org/15/pub/TG4.html.

[w2] NordicSemiconductor, «nRF5 IoT SDK,» [En línea]. Available: https://developer.nordicsemi.com/nRF5_IoT_SDK/doc/0.9.0/html/a00011.html.

[w3] Bluetooth SIG, «Bluetooth,» [En línea]. Available: https://www.bluetooth.com/specifications/bluetooth-core-specification/technical-considerations.

[w4] J. Sens, «Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI,» 2013. [En línea]. Available: http://www.mdpi.com/2224-2708/2/3/589/htm.

[w5] tutorialspoint, «tutorials point, simply easy learning,» [En línea]. Available: http://www.tutorialspoint.com/cprogramming.

[w6] Texas Instruments , «TI 2E2, Support » [En línea]. Available: https://e2e.ti.com/.

9.3 LIBROS CONSULTADOS

[L1] C. C. A. R. D. Kevin Townsend, Getting Started with Bluetooth Low Energy, O'Reilly Media, 2014-04-29.

[L2] D. C. S. M. Alasdair Allan, Bluetooth LE Projects for Arduino, Raspberry Pi, and Smartphones, Maker Media, 2015-11-24.

[L3] Y. P. Yang Xiao, EMERGING WIRELESS LANs, WIRELESS PANS, AND WIRELESS MANs, WILEY, 2009.

10 ANEXO

10.1 DESARROLLO EN IAR EMBEDDED WORKBENCH

Todos los cambios llevados a cabo en este apartado han sido realizados sobre el proyecto *SimpleBLESwitch*.

10.1.1 Temporizador (evento periódico)

Para definir un temporizador (evento periódico)

En *CONSTANTS* dentro de *simpleBLESwitch.c*:

Se define el valor de la frecuencia del temporizador.

En *CONSTANTS* dentro de *simpleBLESwitch.h*:

Hay que definir un evento de tarea (*TASK_EVENTO_CAMBIO_ROL)* y asignarle un valor múltiplo de 2:

```
/*********************************************************************
 * CONSTANTS
 */

// Simple BLE Switch Task Events
#define SBP_START_DEVICE_EVT                              0x0001
#define SBP_PERIODIC_EVT                                  0x0002
#define TASK_EVENTO_FUNCION                               0x0004 //<<--
#define TASK_EVENTO_CAMBIO_ROL                            0x0008 //<<--
/*********************************************************************
```

Ahora hay que inicializar el temporizador, para ello disponemos de dos funciones *osal*:

Una de ellas inicializa el temporizador. Cuando éste expira, el temporizador termina y se genera el bit del evento definido por el usuario. Cuando expira el temporizado éste no se recarga:

```
uint8 osal_start_timerEx( uint8 taskID, uint16 event_id, uint32
timeout_value );
```

taskID es el identificador de tarea de la tarea que se va a obtener cuando expire el temporizador

event_id es el bit del evento definido por el usuario. Cuando el tiempo se agota, al usuario se le notificará la tarea de llamada.

timeout_value es la cantidad de tiempo (en milisegundos) antes de que se establezca el evento de temporizador, el tiempo de duración del temporizador.

La otra de ellas inicializa el temporizador que cuando expira genera el evento con el ID de la tarea definida. Cuando expira el temporizador éste se actualiza automáticamente.

```
uint8 osal_start_reload_timer( uint8 taskID, uint16 event_id, uint32
timeout_value );
```

taskID es el identificador de tarea de la tarea que se va a obtener cuando expire el temporizador

event_id es el bit del evento definido por el usuario. Cuando el tiempo se agota, al usuario se le notificará la tarea de llamada.

timeout_value es la cantidad de tiempo (en milisegundos) antes de que se establezca el evento de temporizador, el tiempo de duración del temporizador.

En este caso haremos uso de la segunda, que se actualiza automáticamente. La incluimos en la función de inicialización para que la llame y se inicialice nuestro temporizador.

```
/*********************************************************************
 * PUBLIC FUNCTIONS
 */


/*********************************************************************
 * @fn      SimpleBLESwitch_Init
 *
 * @brief   Initialization function for the Simple BLE Switch App Task.
 *          This is called during initialization and should contain
 *          any application specific initialization (ie. hardware
 *          initialization/setup, table initialization, power up
 *          notificaiton ... ).
 *
 * @param   task_id - the ID assigned by OSAL.  This ID should be
 *                    used to send messages and set timers.
 *
 * @return  none
 */
void SimpleBLESwitch_Init( uint8 task_id )
{
  simpleBLESwitch_TaskID = task_id;
.
.
.
osal_start_timerEx(simpleBLESwitch_TaskID,              TASK_EVENTO_FUNCION,
PERIODO_EVENTO_FUNCION);


  osal_start_reload_timer(simpleBLESwitch_TaskID,    TASK_EVENTO_CAMBIO_ROL,
PERIODO_EVENTO_CAMBIO_ROL);
}
```

Cuando el temporizador expira, el tiempo establecido se ha consumido y se genera un evento que nos devuelve como valor el identificador de la tarea que nosotros establecimos. Aprovechamos el evento para llamar a nuestra función.

```
/*********************************************************************
 * @fn      SimpleBLESwitch_ProcessEvent
 *
 * @brief   Simple BLE Switch Application Task event processor.  This function
 *          is called to process all events for the task.  Events
 *          include timers, messages and any other user defined events.
 *
 * @param   task_id  - The OSAL assigned task ID.
 * @param   events - events to process.  This is a bit map and can
 *                   contain more than one event.
 *
 * @return  events not processed
 */
uint16 SimpleBLESwitch_ProcessEvent( uint8 task_id, uint16 events )
{

  VOID task_id; // OSAL required parameter that isn't used in this function
.
.
.

  //gestiono cuando se cumple el tiempo del periodo y me devuelve el bit de tarea
  if( events & TASK_EVENTO_CAMBIO_ROL)
  {
    cambio_rol(); //llamo ami función en la que tengo los procesos
    return (events ^ TASK_EVENTO_CAMBIO_ROL);
  }


  // Discard unknown events
  return 0;
}
```

Nuestra función es *cambio_rol*. Esta función debemos declararla en *simpleBLESwitch.c:*

```
/*********************************************************************
 * LOCAL FUNCTIONS
 */
static void simpleBLESwitch_ProcessOSALMsg( osal_event_hdr_t *pMsg );
static void simpleBLESwitch_ProcessGATTMsg( gattMsgEvent_t *pMsg );
static void peripheralStateNotificationCB( gaprole_States_t newState );
static void performPeriodicTask( void );
static void calculaHop (void); //<<--función nueva
static void cambio_rol (void); //<<--función nueva
```

La función está definida en *simpleBLESwitch.c:*

```
/*********************************************************************
 * PUBLIC FUNCTIONS
 */
.
.
////////////////////////////////////////////////////////////////////////////
//              Función para gestionar el cambio de roles                 //
//                  central-periférico                                    //
////////////////////////////////////////////////////////////////////////////
/
/* @fn      cambio_rol
*
*/
static void cambio_rol (void)
{
.
.
.
}
```

10.1.2 Captura de evento de paquete recibido

El evento se llama:

GAP_DEVICE_INFO_EVENT

```
/*********************************************************************
 * @fn      simpleBLECentralEventCB
 *
 * @brief   Central event callback function.
 *
 * @param   pEvent - pointer to event structure
 *
 * @return  TRUE if safe to deallocate event message, FALSE otherwise.
 */
static uint8 simpleBLECentralEventCB( gapCentralRoleEvent_t *pEvent )
{
  switch ( pEvent->gap.opcode )
  {
case GAP_DEVICE_INFO_EVENT:
      {

    case GAP_DEVICE_INFO_EVENT:
       {
        //parpadeo de led
         HalLedBlink (2, 4, 10, 100);

          //le paso a mi función los datos
            capturaDatos( pEvent->deviceInfo.addr, //dirección
                          pEvent->deviceInfo.rssi,  // valor de RSSI
                          pEvent->deviceInfo.pEvtData,  //puntero al contenido del campo
datos
                          pEvent->deviceInfo.dataLen );  //longitud de datos
        }
      }
      break;
```

Enviado durante el proceso de descubrimiento de dispositivos cuando se descubre un dispositivo. Este evento es enviado como un mensaje OSAL definido como `gapDeviceInfoEvent_t` que tiene la siguiente estructura, definida en (`gap.h`):

```
typedef struct
{
  osal_event_hdr_t  hdr;    //!< GAP_MSG_EVENT and status
  uint8 opcode;             //!< GAP_DEVICE_INFO_EVENT
  uint8 eventType;          //!< Advertisement Type: @ref
GAP_ADVERTISEMENT_REPORT_TYPE_DEFINES
  uint8 addrType;           //!< address type: @ref GAP_ADDR_TYPE_DEFINES
  uint8 addr[B_ADDR_LEN];   //!< Address of the advertisement or SCAN_RSP
  int8 rssi;                //!< Advertisement or SCAN_RSP RSSI
  uint8 dataLen;            //!< Length (in bytes) of the data field (evtData)
  uint8 *pEvtData;          //!< Data field of advertisement or SCAN_RSP
} gapDeviceInfoEvent_t;
```

Está incluido en `gaCentralRoleEvent_t` donde lo define (en `central.h`) como `deviceInfo`:

```
/**
 * Central Event Structure
 */
typedef union
{
  gapEventHdr_t             gap;               //!< GAP_MSG_EVENT and status.
  gapDeviceInitDoneEvent_t  initDone;          //!< GAP initialization done.
  gapDeviceInfoEvent_t      deviceInfo;        //!< Discovery device information event structure.
  gapDevDiscEvent_t         discCmpl;          //!< Discovery complete event structure.
  gapEstLinkReqEvent_t      linkCmpl;          //!< Link complete event structure.
  gapLinkUpdateEvent_t      linkUpdate;        //!< Link update event structure.
  gapTerminateLinkEvent_t   linkTerminate;     //!< Link terminated event structure.
} gapCentralRoleEvent_t;

/**
```

10.1.3 Añadir el Servicio de Batería

Vamos a tratar de añadir un nuevo servicio a la capa GATT, Generic Attribute Profile, a modo de recordatorio. La capa GATT opera sobre la capa ATT (Attribute Protocol). Es la encargada de manejar el intercambio de datos entre los dispositivos, Servidor → Cliente. El dispositivo que actúa como Servidor es el que ofrece la información. Éste tiene un perfil que incluye todos los servicios de los que dispone. Cada servicio contiene diferentes características que están formadas por atributos. Existen una serie de servicios fijos, existentes en todos los dispositivos, por ejemplo, el que contiene la información del dispositivo incluida en diferentes características: *System ID, Model number string, Manufacturer name string,* etc…

Cada servicio tiene un UUID propio. Existen algunos servicios registrados por el SIG y que, por tanto, todos los dispositivos que tienen implementado BLE reconocen esos UUIDs y muestran el nombre del servicio asociado. Uno de los servicios registrados es el *Battery Service*.

SpecificationName	SpecificationType	AssignedNumber	SpecificationLevel
Alert Notification Service	org.bluetooth.service.alert_notification	0x1811	Adopted
Automation IO	org.bluetooth.service.automation_io	0x1815	Adopted
Battery Service	org.bluetooth.service.battery_service	0x180F	Adopted
Blood Pressure	org.bluetooth.service.blood_pressure	0x1810	Adopted
Body Composition	org.bluetooth.service.body_composition	0x181B	Adopted

FIGURA 71- PARTE LOS SERVICIOS DE LAS ESPECIFICACIONES DE GATT. 2016 BLUETOOTH SIG.

Este servicio está incluido en las librerías ofrecidas por la pila de protocolos de *Texas Instruments*. Con el nombre de *battservice.c* y *battservice.h.* Estos dos archivos han de ser incluidos en la carpeta *PROFILES* en el *Workspace* del proyecto en *IAR Embedded Workbench IDE*. Una vez incluidos hemos de conectarlos con el resto de capas, es en este proceso en el que reside la dificultad.

En *SimpleBLEPeripheral.c* incluimos la línea: #include “battservice.h”.

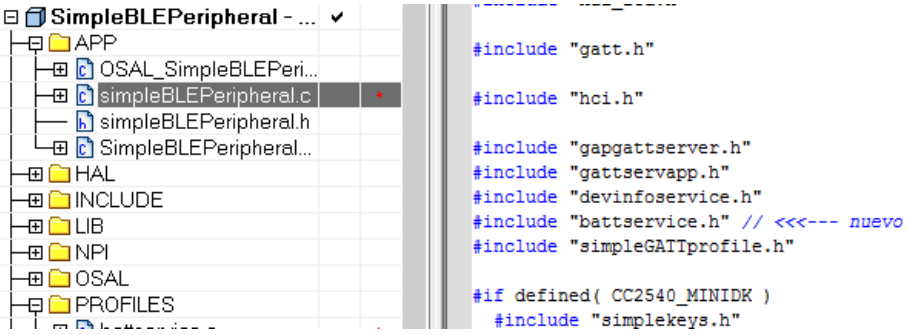

FIGURA 72-NUEVA LÍNEA.

En *simpleBLEPeripheral.c* añadimos el evento de tarea para el nuevo servicio en las constantes, incluimos la línea:

```
#include SBP_BATTERY_CHECK_EVT          0x0008
```

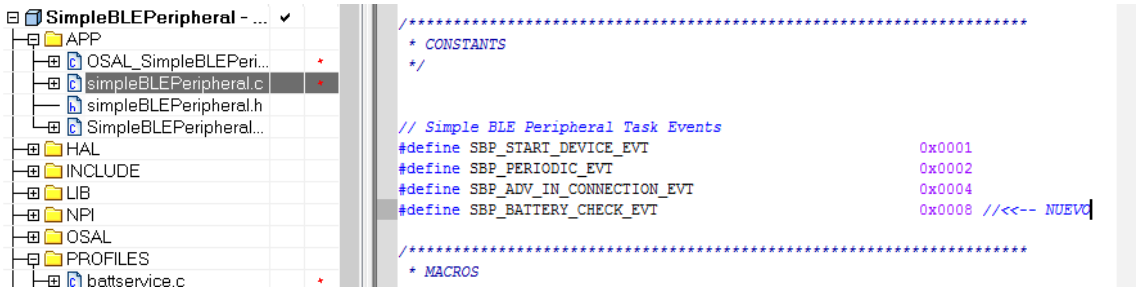

FIGURA 73-NUEVA LÍNEA.

Registramos el servicio de batería añadiéndolo a *SimpleBLEPeripheral_Init(…)* función de inicialización de tareas para la *App* de *Simple BLE Peripheral*, a la que se llama durante la inicialización y debe contener cualquier inicialización de aplicación específica (inicialización/configuración de hardware, tablas de inicialización, encendido de notificaciones…) añadiendo la línea: Batt_AddService(); .

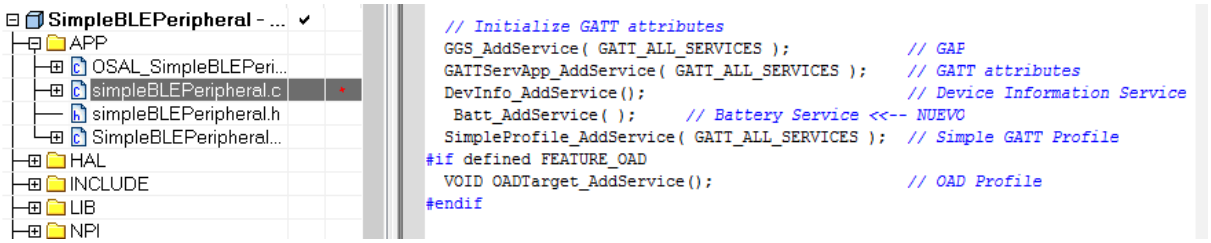

FIGURA 74-NUEVA LÍNEA.

Registramos el servicio de batería añadiéndolo en *SimpleBLEPeripheral_ProcessEvent(…)* procesador de eventos de tareas de la aplicación de *Simple BLE Peripheral*, a la que se llama para procesar todos los eventos para cada tarea. Los eventos incluyen timers, mensajes y otros eventos definidos por el usuario añadiendo la línea:

```
// Set timer for first battery read event
    osal_start_timerEx( simpleBLEPeripheral_TaskID, SBP_BATTERY_CHECK_EVT,
BATTERY_CHECK_PERIOD );
```

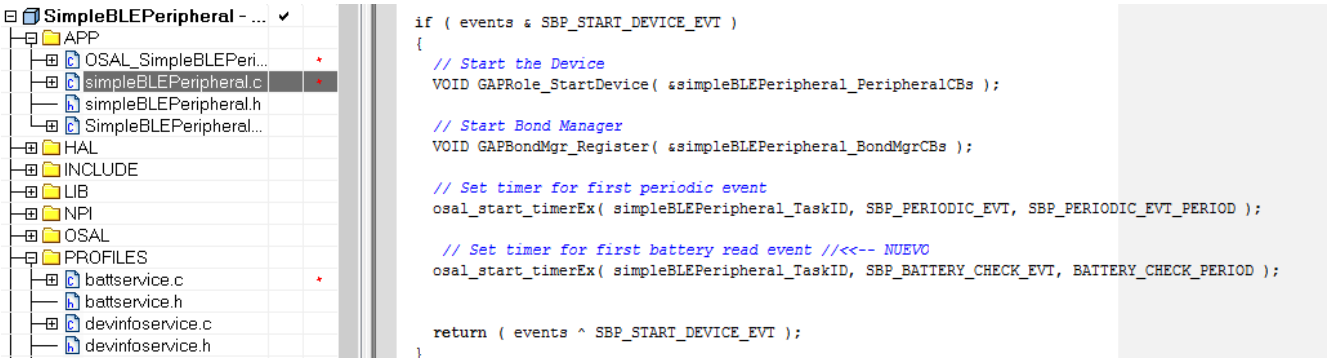

FIGURA 75-NUEVA LÍNEA.

Dentro de la misma función, un poco más abajo, incluiremos las siguientes líneas:

```
if ( events & SBP_BATTERY_CHECK_EVT )
{
// Restart timer
if ( BATTERY_CHECK_PERIOD )
{
  osal_start_timerEx( simpleBLEPeripheral_TaskID, SBP_BATTERY_CHECK_EVT,
BATTERY_CHECK_PERIOD );
}
// perform battery level check
      Batt_MeasLevel( );
      return (events ^ SBP_BATTERY_CHECK_EVT);
}
```

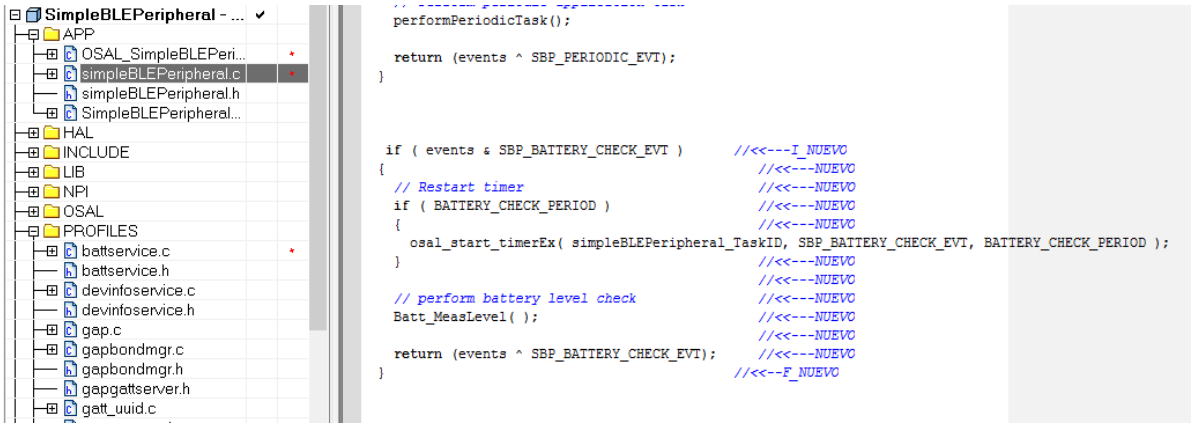

FIGURA 76-NUEVA LÍNEA.

Hay que definir el periodo de medición de la batería. En las constantes de *simpleBLEPeripheral.c* añadimos las líneas:

```
//How often to check battery voltage (in ms)
#define BATTERY_CHECK_PERIOD                  10000
```

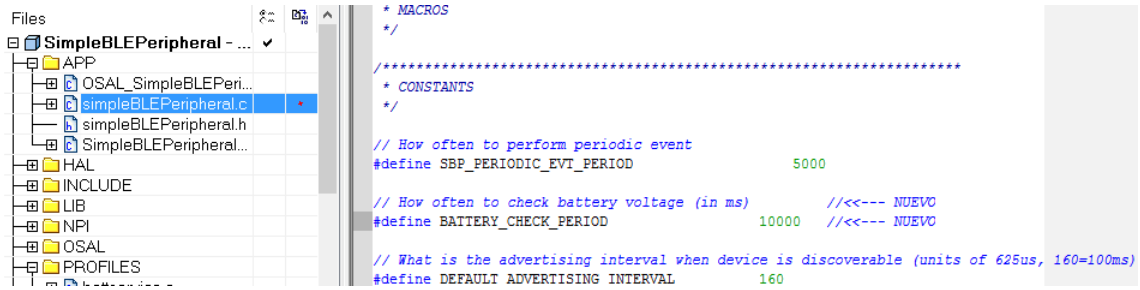

FIGURA 77-NUEVA LÍNEA.

Por último:

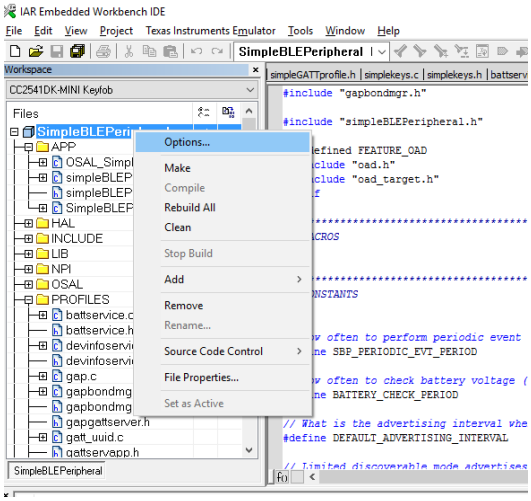

FIGURA 78- PASO A SEGUIR.

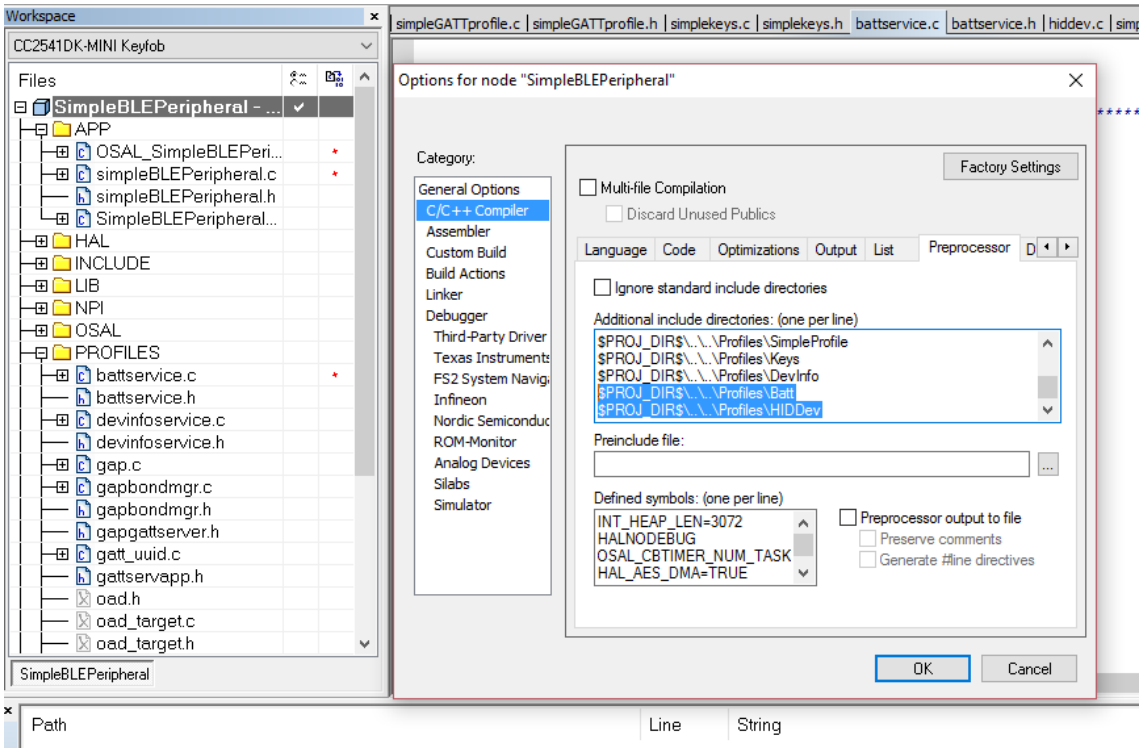

FIGURA 79- ELECCIÓN DE PREPROCESADOR.

Y añadimos esas dos líneas.
$PROJ_DIR$\..\..\Profiles\Batt
$PROJ_DIR$\..\..\Profiles\HIDDev

Pulsamos OK y ya tendremos nuestro servicio de batería listo y funcionando correctamente.

10.2 BREVE INTRODUCCIÓN A 6LOWPAN

IPV6 over Low-power Wireless Personal Area Netwoks *(IPV6 sobre redes de área personal de baja potencia).* Nombre propuesto por el **IETF** (The Internet Engineering Task Force, *Grupo de Trabajo de Ingeniería de Internet*).

El concepto de 6LoWPAN viene de la idea de que "el protocolo de Internet podría y debería aplicarse incluso a los dispositivos más pequeños" y que los dispositivos de baja potencia con la capacidad de procesamiento limitada deben ser capaces de participar del Internet de las Cosas (**IoT**).
6LoWPAN define los mecanismos de encapsulación y compresión de cabecera que permiten a los paquetes IPV6 ser enviados y recibidos sobre las redes basadas en IEEE 802.15.4.
IPV4 e IPV6 son el caballo de trabajo para la entrega de datos en redes de área local (LAN), redes de área metropolitana (MAN) y redes de área extensa (WAN) como Internet. Del mismo modo, el estándar del IEEE 802.15.4 proporciona a los dispositivos la capacidad de comunicación en dominio inalámbrico. Las naturalezas inherentes de las dos redes, sin embargo, son diferentes.

La especificación de base desarrollada por el grupo 6LoWPAN IETF es *RFC 4944* (actualizado por el *RFC 6282* con la compresión de cabecera y por el *RFC 6775* con optimizaciones de descubrimiento de vecino). El documento de declaración del problema es el *RFC 4919*.

- RFC4944 Features:
  - Basic LoWPAN header format
  - HC1 (IPv6 header) and HC2 (UDP header) compression formats
  - Fragmentation & reassembly
  - Mesh header feature (depreciation planned)
  - Multicast mapping to 16-bit address space

- RFC6282 Features:
  - New HC (IPv6 header) and NHC (Next-header) compression
  - Support for global address compression (with contexts)
  - Support for IPv6 extension header compression
  - Support for UDP
  - Support for compact multicast address compression

FIGURA 80- COMPARACIÓN ENTRE RFC4944 Y RFC6282 **[21]**.

ÁREAS DE APLICACIÓN

El objetivo de la creación de redes IP para la comunicación de radio de baja potencia son aplicaciones que necesitan conectividad inalámbrica a Internet a velocidades de datos de más bajos para los dispositivos con factor de forma muy limitada. Un ejemplo es la automatización de aplicaciones en entorno de casas, oficinas y fábricas. Los mecanismos de compresión de encabezado estandarizados en el *RFC 6282* pueden ser utilizados para promocionar la compresión de cabecera de paquetes de IPV6 en tales redes.
IPV6 también se está utilizando en la red inteligente que permite a los contadores inteligentes y otros dispositivos para construir una red de malla micro antes de enviar los datos al sistema de facturación con el backbone IPV6. Algunas de estas redes van sobre la radio IEEE 802.15.4, y por lo tanto utilizan la compresión de la cabecera y la fragmentación como se especifica en el *RFC 6282*.

FUNCIONES

Al igual que con todas las asignaciones de capa de enlace de IP, *RFC 4944* proporciona una serie de funciones. Más allá de las diferencias habituales entre redes L2 y L3, la cartografía de la red IPV6 a la red IEEE 802.15.4 plantea desafíos adicionales de diseño.

Adaptación del tamaño de los paquetes de las dos redes

IPV6 requiere que la unidad máxima de transmisión (MTU) no sea mayor que 1280 bytes. En contraste, IEEE 802.15.4 tiene un tamaño de paquete estándar de 127 bytes. Un máximo de sobrecarga de trama de 25 bytes ahorra, libera, a la capa de control de acceso al medio 102 bytes. Una característica de seguridad opcional pero altamente recomendada en la capa de enlace supone una sobrecarga adicional. Por ejemplo, 21 bytes se consumen para AES-CCM-128 dejando sólo 81 bytes para las capas superiores (81+21=102 bytes).

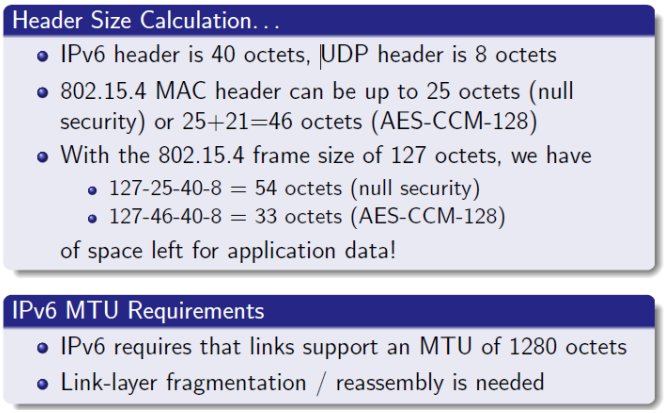

FIGURA 81- TAMAÑO CABECERA 802.15.4 Y REQUERIMIENTOS IPV6 **[22].**

Resolución de direcciones

A los nodos de IPV6 se les asigna una dirección IP de 128 bit (7 bytes) de manera jerárquica, a través de una longitud de prefijo de red arbitraria. Los dispositivos IEEE 802.15.4 pueden usar o la dirección extendida IEEE 64 bit (6 bytes), asociada a un evento posterior, o la dirección de 16 bit (4 bytes) que es única dentro de una PAN. Hay también un PAN-ID para un grupo de dispositivos con una ubicación física compartida en IEEE 802.15.4.

Diferentes diseños de los dispositivos

Los dispositivos IEEE 802.15.4 están limitados intencionadamente para reducir costes (lo que permite una red a gran escala de muchos dispositivos), reducir el consumo de energía (permitiendo que los dispositivos funcionen con baterías) y permitiendo la flexibilidad de la instalación (pequeños dispositivos para redes de uso corporal). Por otra parte, los nodos cableados (alámbricos) no están limitados de esta manera; pueden ser más grandes y hacer uso de fuentes de alimentación de red.

Diferentes objetivos en la optimización de los parámetros

Los nodos IPV6 están orientados en la consecución de altas velocidades. Algoritmos y protocolos implementados en las capas altas como el kernel TCP de TCP/IP están optimizados para manejar problemas típicos como la congestión. En los dispositivos IEEE 802.15.4 la conservación de

energía y la optimización de tamaño de la codificación están en la agenda del día.

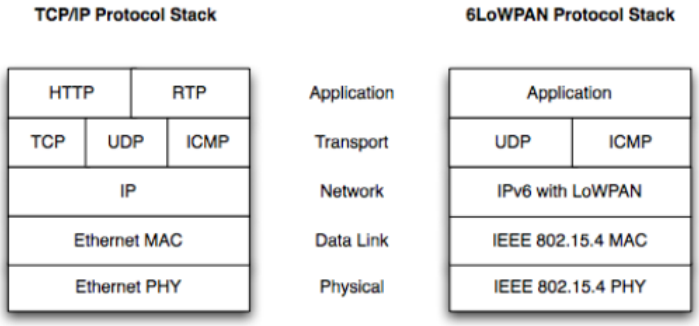

FIGURA 82- COMPARACIÓN ENTRE PILAS DE PROTOCOLOS **[21]**.

Capa de adaptación para la interoperabilidad y el formato de los paquetes

Un mecanismo de adaptación que permita la interoperabilidad entre el dominio IPV6 y el IEEE 802.15.4 puede ser visto como un problema de capa. Identificar la funcionalidad de esta capa y definir los nuevos formatos de los nuevos paquetes, si es necesario, es un área de investigación actual. *RFC 4944* propone una capa de adaptación que permite trasmitir datagramas de IPV6 sobre redes IEE 802.15.4.

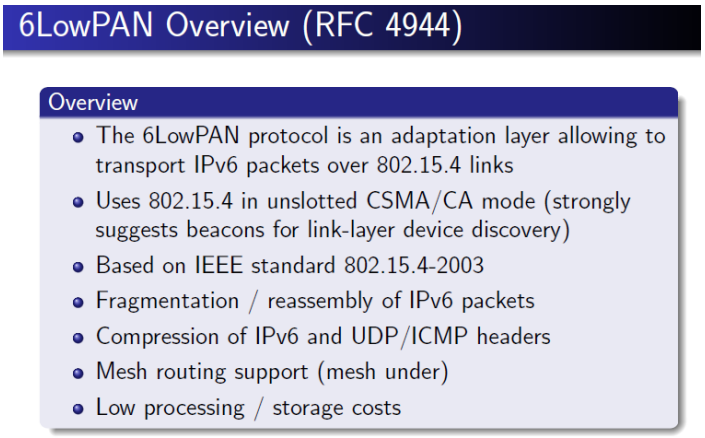

FIGURA 83- RESUMEN DE LA RFC 4944 **[22]**.

Mecanismos de gestión de direcciones

La gestión de las direcciones para los dispositivos que se comunican a través de los dos diferentes dominios, IPV6 e IEEE 802.15.4, es engorroso y exhaustivamente complejo.

Consideraciones de enrutamiento y protocolos para la topología de malla en 6LoWPAN

El enrutamiento, de por sí, es un problema de dos fases que se está considerando para redes IP de bajo consumo.

-Enrutamiento de red en el espacio de las Redes de Área Personal (PAN).

-La enrutabilidad de paquetes entre el dominio IPV6 y el dominio del PAN.

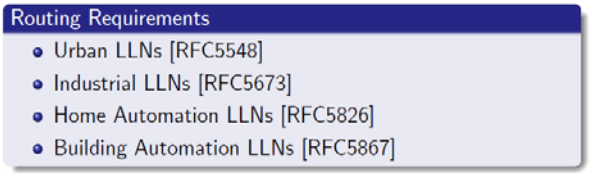

FIGURA 84- REQUERIMIENTOS DE ENRUTADO, LLN **[22]**.

Varios protocolos de enrutamiento han sido propuestos para 6LoWPAN: LOAD, DYMO-LOW, HI-LOW. Sin embargo, solamente dos están actualmente legitimados en el desarrollo a gran escala: **LOADng**, estandarizado por la ITU en la recomendación ITU-T G.9939 y **RPL** (*Routing Protocol for Low-Power and Lossy Networks*, RFC 6550), estandarizado por el grupo de trabajo IETF ROLL.

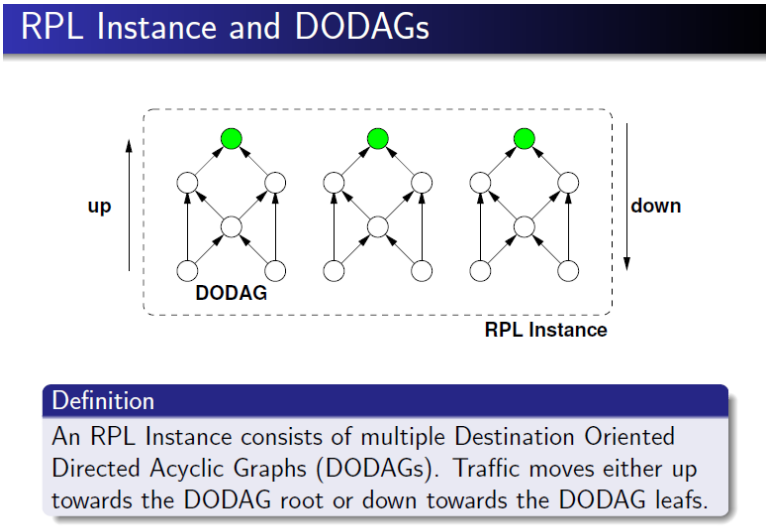

FIGURA 85- PROTOCOLO DE ENRUTAMIENTO PARA REDES DE BAJO CONSUMO **[22]**.

Dispositivos y servicio de descubrimiento

Dado que los dispositivos habilitados para IP pueden requerir la formación de redes ad hoc, tendrá que ser conocido el estado actual de los dispositivos vecinos y los servicios que alojan. La extensión de descubrimiento vecino IPV6 es un proyecto de Internet propuesto como una contribución en esta área.

Seguridad

Los nodos IEEE 802.15.4 pueden operar en modo seguro o en modo no seguro. Estos dos modos de seguridad se definen en la especificación con el fin de lograr diferentes objetivos de seguridad: Lista de Control de Acceso (ACL) y el modo Seguro.

PREGUNTAS FRECUENTES [23]

¿Cómo se compara 6LoWPAN con Zigbee, SP100.11a, …?

- Zigbee
  - Sólo define la comunicación entre nodos de 802.15.4 (“capa 2” en términos IP), no el resto de la red (otros enlaces, otros nodos)
  - Define nuevas capas superiores, todas en el camino a la aplicación, similar a IRDA, USB y Bluetooth, utilizando los estándares existentes.
  - Especificaciones todavía en progreso. (Carece de una capa de transporte).
- SP100.11a
  - Trata de abordar una variedad de enlaces, incluyendo 802.15.4, 802.11, WiMax y el futuro “saltos de frecuencia de banda estrecha”.
  - Las especificaciones están todavía en las primeras etapas, pero deberá redefinir, con IP, muchas de las que ya están definidas.
  - Mucha de la atención está en el arbitraje de los niveles bajos usando técnicas TDMA (cómo *token ring*) en lugar de CSMA (cómo Ethernet y WiFi). El problema es ortogonal para el formato de trama.
- 6LoWPAN define como establecer las capas de red IP utilizando 802.15.4
  - Permite la comunicación entre 802.15.4↔802.15.4 y 802.15.4↔no 802.15.4
  - Permite el uso de un amplio conjunto de normas existentes, así como del nivel más alto de protocolos, software y herramientas.
  - Es una extensión enfocada a las tecnologías IP que permite el uso de nuevas clases de dispositivos de una manera familiar.

¿Necesito IP para mi red independiente?

- Hoy en día, prácticamente todos los dispositivos utilizan la *stack* de red IP para comunicarse con otros dispositivos, si pertenece a una red independiente aislada, a una parte accesible de una empresa privada perteneciente a una empresa más grande o si es un punto de acceso público.
  - Cuando todos los dispositivos forman una subred no se requiere de enrutamiento, pero todo funciona de la misma manera.

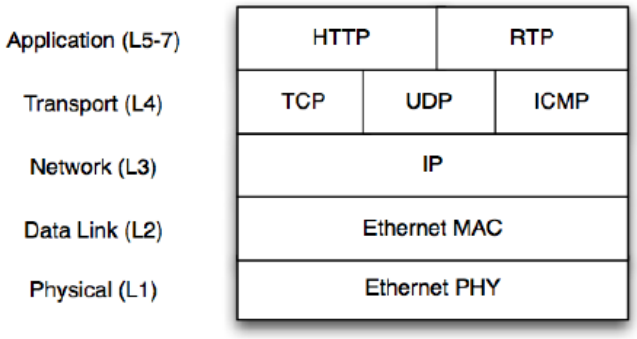

FIGURA 86- PILA DE PROTOCOLO IP **[21]**.

- El software, las herramientas y los estándares utilizan IP y las capas por encima ella, no el vínculo físico por debajo.

- El valor de hacerlo "todo lo mismo" supera con creces los moderados gastos generales.
- 6LoWPAN elimina la sobrecara donde más importa.

¿La facilidad de acceso con IP significa menos seguridad?

- No
- Las redes más altamente sensibles usan IP internamente, pero están completamente desconectadas de todos los demás equipos.
- Las redes IP en todo tipo de ajustes altamente valorados son protegidos por el establecimiento de los puntos de conexión de manera cuidadosa (cortafuegos, DMZ, listas de control de acceso…).
- Nodos no IP detrás de una puerta de enlace en una red no son más seguros que el dispositivo que actúa como la puerta de enlace. Esos dispositivos suelen ser numerosos y no suelen utilizar la tecnología más avanzada en seguridad.
- 802.15.4 proporciona una función de cifrado AES128 que se habilita por debajo de IP, como WPA en 802.11.

¿Usar 6LoWPAN significa renunciar al comportamiento determinista?

- No.
- El uso del formato 6LoWPAN para transportar tráfico sobre los enlaces 802.15.4 es ortogonal a sí esos enlaces fuesen programados deterministamente.
    - Medios deterministas de control de acceso (MAC) se pueden implementar con la misma facilidad con 6LoWPAN como con cualquier otro formato.
- Hay muchos mecanismos TDMA con IP, incluyendo Token Ring y FDDI.
    - Protocolos MAC como FPS o TSMP extienden esto a una red.
    - En última instancia, se quieren límites de carga y suficiente espacio libre para cubrir las pérdidas potenciales.
    - Los dispositivos que usan diferentes MACs sobre el mismo enlace (TDMA vs CSMA) pueden no ser capaces de comunicarse, aunque el formato de paquete sea el mismo.

¿Es 6LoWPAN menos eficiente energéticamente que otros protocolos propietarios?

- No.
- Otros protocolos transportan información de cabecera similar para el direccionamiento y el enrutamiento, pero de una manera más a medida.
- Mientras IP requiere mucho trabajo para el caso general, permite optimizaciones para casos específicos.
- 6LoWPAN optimiza la subred de baja potencia dentro de 802.15.4.
    - Más costoso sólo cuando vas más allá de ese enlace.
    - Otros protocolos deben proporcionar información análoga (a nivel de aplicación) para instruir a las pasarelas, vías (gateways).
- Básicamente, el rendimiento está determinado por la calidad de la implementación.
    - Con estándares IP abiertos las compañías deben competir en rendimiento y eficiencia, en lugar de los estándares propietarios.

¿Necesito ejecutar IPV6 en lugar de IPV4 en el resto de mi red para utilizar 6LoWPAN?

- No.
- IPV6 e IPV4 trabajan juntos en todo el mundo a través de la traducción 4-6.
- IPV6 está diseñada para soportar a billones de dispositivos de red no tradicionales y está diseñado de forma clara.
    - En realidad, es más sencillo de soportar en dispositivos pequeños a pesar de direcciones más largas.
- Los dispositivos 802.15.4 embebidos pueden hablar IPV6 con los rúters al resto de la red con la traducción 4-6.
    - Esa traducción ya está normalizada y disponible de manera amplia.

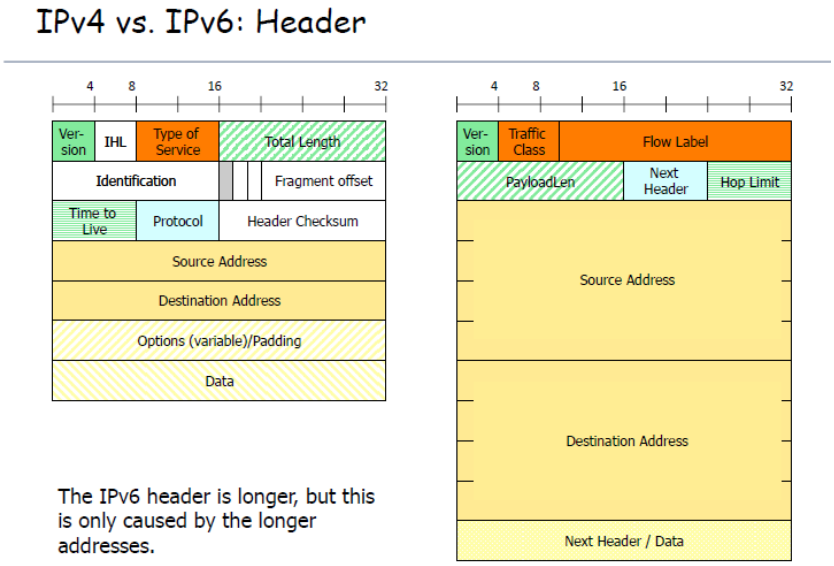

FIGURA 87- CABECERA IPV4 VS IPV6 **[23]**.