



Escola d'Enginyeria de Telecomunicació i
Aeroespacial de Castelldefels

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO DEL TFG: Bluetooth 4.0 Low Energy: Análisis de las prestaciones y aplicaciones para la automoción

TITULACIÓN: Grado en Ingeniería de Sistemas de Telecomunicación

AUTOR: Yassir Akhayad

DIRECTOR: José Luis Valenzuela González

FECHA: 08/02/2016

Título: Bluetooth 4.0 Low Energy: Análisis de las prestaciones y aplicaciones para la automoción

Autor: Yassir Akhayad

Director: José Luis Valenzuela González

FECHA: 08/02/2016

Resumen

En este trabajo se ha analizado la viabilidad de la nueva tecnología Bluetooth Low Energy (BLE) con el fin de proponer un nuevo uso en el sector automovilístico. Concretamente, una alternativa al sistema de detección de señales de tráfico basado en cámaras que presenta algunos problemas en su funcionamiento.

Las características que se han considerado relevantes para dicha propuesta son: el tiempo de descubrimiento para una detección eficiente que garantiza una reacción segura del conductor ante cualquier señal de tráfico y el consumo energético de los dispositivos BLE para responder a las exigencias económicas y obtener un impacto medioambiental responsable.

En primer lugar se presenta BLE describiendo su funcionamiento junto con las capas de su pila de protocolos. La descripción de la pila de protocolos se ha enfocado solamente en las capas que se han considerado relevantes para este trabajo. También se ha hecho un estudio comparativo con otros estándares donde se han destacado las ventajas que ofrece BLE para el objetivo de este trabajo frente a los demás estándares

A continuación se estudian los procedimientos de *Scanning/Advertising* con sus respectivos parámetros. A partir de los resultados obtenidos se ha hecho la elección del tipo de *Advertising* que debe implementarse para cumplir con las condiciones del sistema propuesto (*Broadcaster-Scanner*). Las medidas realizadas de tiempo de descubrimiento se han enfocado en el tipo *Nonconnectable Undirected Advertising* (Beacons), variando los parámetros de *Scanning/Advertising* y los chipsets del Scanner.

Para realizar las medidas del tiempo de descubrimiento se han programado algoritmos en el transmisor y el receptor (*Broadcaster/Scanner*). Para la realización del mismo ha sido necesario estudiar los dispositivos utilizados, sus parámetros, cómo modificarlos y configurarlos vía comandos HCI.

Por otra parte, se han realizado medidas de consumo de energía de diferentes configuraciones y chipsets donde se demuestra que dependiendo de la configuración y el chipset puede variar el consumo de energía, pero siempre cumpliendo con su característica de bajo consumo.

Por último se presenta un escenario de aplicaciones de detección y configuración adaptativa de señales de tráfico. También se han introducido nuevas ideas de señales de aviso que mejorarían la seguridad vial.

Title: Bluetooth 4.0 Low Energy: Análisis de las prestaciones y aplicaciones para la automoción

Author: Yassir Akhayad

Director: José Luis Valenzuela González

Date: 08/02/2016

Overview

The objective of this project is to analyze the viability of Bluetooth Low Energy and give it a new use in the automobile industry. Specifically, it will be used as an alternative to the traffic-sign detection system based on cameras that do not work properly.

The features chosen for such a purpose are: the discovery time for an efficient detection that guarantees a secure reaction of the driver to any traffic sign and the energy consumption of BLE device to respond to the economic demand and get a responsible environmental impact.

On first place we introduce BLE and its functioning along with the layers of its protocol stack. The protocol stack description is exclusively focused on the layers that have been considered relevant for this project. A comparative investigation has also been done with other standards where we highlighted the advantages that BLE offers for the objective of this project compared to other standards.

Hereafter, we study the scanning/advertising procedures and its respective parameters. The type of advertising that must be implemented to comply with the conditions of proposed system (broadcaster-scanner) has been selected from the results obtained. Time Discovery taken measures has been focused on the type of nonconnectable Undirected Advertising (Beacons), varying parameters of scanning/advertising and scanner chipsets.

To take the discovery time measures, algorithms in the transmitter and the receiver (broadcaster/scaner) have been programmed. To do so, it was necessary to study the used devices, their parameters and how to modify and configure them through HCI commands.

On the other side, we have measured the power consumption of different configurations and chipsets. We can see that power consumption can change depending on the configuration and chipset, but always meeting its low consumption feature.

At last, a setting of adaptative traffic sign detection and configuration application is introduced. New ideas of warning signs have been introduced, as well, to improve road safety.

INDICE

INTRODUCCIÓN	1
CAPÍTULO 1. DESCRIPCIÓN DE BLUETOOTH LOW ENERGY	4
1.1. Arquitectura	4
1.1.1. Capa Física	5
1.1.2. Capa de Enlace	6
1.1.3. Host/Controller Interface.....	8
1.2. Comparación con otros estándares	8
CAPÍTULO 2. DESCRIPCIÓN DEL PROCEDIMIENTO DE ADVERTISING... ..	10
2.1. El intervalo de Advertising	10
2.2. Advertising Channel PDU	11
2.3. Connectable Undirected	12
2.4. Connectable Directed.....	12
2.5. Nonconnectable Advertising	13
2.6. Discoverable Advertising	14
CAPÍTULO 3. IMPLEMENTACIÓN DE PROCEDIMIENTOS DE CONFIGURACIÓN.....	15
CAPÍTULO 4. IMPLEMENTACIÓN DE PROCEDIMIENTO DE SCANNING... ..	19
4.1. Passive Scanning	19
4.2. Active Scanning.....	22
CAPÍTULO 5. IMPLEMENTACIÓN DE PROCEDIMIENTOS DE BROADCASTING	23
5.1. Nonconnectable Advertising	23
5.2. Discoverable Advertising	28
CAPÍTULO 6. MEDIDAS SOBRE LOS PROCEDIMIENTOS DE ADVERTISING	29
6.1. Influencia de Scan Interval con Advertising Interval de 100 ms	29
6.2. Influencia de Scan Interval con Advertising Interval de 200 ms	35
6.3. Influencia del ciclo de trabajo	36
6.4. Influencia del chipset.....	37
CAPÍTULO 7. CARACTERIZACIÓN DE ENERGÍA	41
7.1. Consumo de Scanning al 50% de diferentes fabricantes	41
7.2. Consumo de pulsos de Advertising de diferentes chipsets	43
7.3. Consumo de Scan continuo de diferentes chipsets.....	44
CAPÍTULO 8. PROPUESTAS PARA EL RECONOCIMIENTO DE SEÑALES DE TRÁFICO	46
CAPÍTULO 9. CONCLUSIÓN E IMPACTO SOCIAL	50

BIBLIOGRAFÍA 51

ANEXOS 52

Anexo 1: List Of Error Codes 52

Anexo 2: Scripts 53

 A2.1: Habilitar Scan Pasivo 53

 A2.2: Habilitar Scan Activo..... 53

 A2.3: Habilitar Advertising tipo Nonconnectable 53

 A2.4: Habilitar Advertising tipo Discoverable 54

 A2.5: Script capturar medidas de configuraciones diferentes 55

Anexo 3: Dispositivos utilizados 57

 A3.1: Trust 57

 A3.2: Belkin 61

 A3.3: Sena 65

 A3.4: BLE Nano 66

 A3.5: Sensor de corriente 67

Anexo 4: Figuras de posibles casos de tiempo de descubrimiento 68

ÍNDICE DE FIGURAS

Fig. 1.1 Pila del protocolo BLE.....	5
Fig. 1.2 Distribución de canales en BLE	6
Fig. 1.3 Estados y roles de BLE.....	7
Fig. 2.1 <i>Advertising Interval</i>	10
Fig. 2.2 <i>Advertising Channel PDU</i>	11
Fig. 2.3 <i>Connectable Undirected Advertising</i>	12
Fig. 2.4 Flujo de mensajes del modo <i>Connectable Undirected Advertising</i>	12
Fig. 2.5 <i>Connectable Directed Advertising</i>	13
Fig. 2.6 Flujo de mensajes del modo <i>Connectable Directed Advertising</i>	13
Fig. 2.7 <i>Nonconnectable Advertising</i>	14
Fig. 2.8 <i>Discoverable Advertising</i>	14
Fig. 3.1 Comando hcitool dev	15
Fig. 3.2 Comando hcitool scan.....	16
Fig. 3.3 Comando hcitool inq	16
Fig. 3.4 Captura de Wireshark	18
Fig. 4.1 <i>Scan Interval/Window</i>	19
Fig. 4.2 Valores de los campos de parámetros de <i>Scan</i> pasivo.....	21
Fig. 4.3 Respuesta 0x0C del <i>Controller</i> vía terminal.....	21
Fig. 4.4 Respuesta 0x0C del <i>Controller</i> vista desde Wireshark	21
Fig. 5.1 <i>Nonconnectable Undirected Advertising PDU</i>	24
Fig. 5.2 Valores de campos de <i>Nonconnectable Advertising Channel PDU</i> y tiempo transcurrido entre dos paquetes consecutivos	27
Fig. 6.1 Algoritmo para obtener las medidas.....	30
Fig. 6.2 Distribución del tiempo de descubrimiento con <i>Advertising Interval</i> de 100 ms: <i>Scan</i> continuo de 10 ms (Izquierda), <i>Scan Interval</i> 50 ms y <i>Scan Window</i> 10 ms (Derecha).....	31
Fig. 6.3 Comportamiento <i>Scan</i> continuo de Trust.....	32
Fig. 6.4 Duración de pausas <i>Scan</i> continuo de Trust.....	32
Fig. 6.5 Duración pulsos de <i>Advertising</i> de Belkin	32
Fig. 6.6 Efecto de las pausas de <i>Scan</i> continuo en la detección de los <i>Advertising</i>	33
Fig. 6.7 Posible caso de <i>Scanning</i> al 20% con <i>Advertising</i> de intervalo 100 ms	34
Fig. 6.8 Evolución del tiempo de descubrimiento en función de <i>Scan Interval</i> con <i>Advertising Interval</i> de 100 ms	34
Fig. 6.9 Distribuciones de tiempo de descubrimiento con <i>Advertising Interval</i> de 200 ms: <i>Scan</i> continuo de 10 ms (Izquierda), <i>Scan Interval</i> 100 ms y <i>Scan Window</i> 10 ms (Derecha).....	35
Fig. 6.10 Evolución del tiempo de descubrimiento en función de <i>Scan Interval</i> con <i>Advertising Interval</i> de 200 ms	36

Fig. 6.11 Evolución del tiempo de descubrimiento en función de la combinación <i>Scan Interval-Window</i> de 33%	37
Fig. 6.12 Evolución del tiempo de descubrimiento en función de la combinación <i>Scan Interval-Window</i> de 50%	37
Fig. 6.13 Comportamiento de <i>Scan</i> continuo del dispositivo Sena	38
Fig. 6.14 Comportamiento de <i>Scan</i> continuo del dispositivo Trust	38
Fig. 6.15 Comportamiento de <i>Scan</i> continuo del dispositivo Belkin	39
Fig. 6.16 Evolución del tiempo de descubrimiento con configuraciones <i>Scan</i> continuo diferentes del dispositivo Belkin	39
Fig. 6.17 Evolución del tiempo de descubrimiento con configuraciones <i>Scan</i> continuo diferentes del dispositivo Trust	40
Fig. 6.18 Evolución del tiempo de descubrimiento con configuraciones <i>Scan</i> continuo diferentes del dispositivo Sena	40
Fig. 6.19 Evolución delta vs <i>Scan Interval/Window</i> en <i>Scan</i> continuo de diferentes chipsets	40
 Fig. 7.1 Función transformada V_{OUT} vs I_{IN} del sensor de corriente utilizado	41
Fig. 7.2 Consumo de <i>Scanning</i> al 50% del dispositivo Belkin (Izquierda <i>Scan Window</i> de 50 ms y <i>Scan Interval</i> 100 ms, derecha 500 ms y 1000 ms consecutivamente)	41
Fig. 7.3 Consumo <i>Scanning</i> al 50% del dispositivo Sena (Izquierda <i>Scan Window</i> de 50 ms y <i>Scan Interval</i> 100 ms, derecha 500 ms y 1000 ms consecutivamente)	42
Fig. 7.4 Consumo <i>Scanning</i> al 50% del dispositivo Trust (Izquierda <i>Scan Window</i> de 50 ms y <i>Scan Interval</i> 100 ms, derecha 500 ms y 1000 ms consecutivamente)	42
Fig. 7.5 Consumo de los pulsos de <i>Advertising</i> del dispositivo: Derecha Belkin, Izquierda Sena	43
Fig. 7.6 Consumo de los pulsos de <i>Advertising</i> del dispositivo Trust	43
Fig. 7.7 Consumo de los pulsos de <i>Advertising</i> del dispositivo BLE Nano	44
Fig. 7.8 Consumo modo <i>Scan</i> continuo del dispositivo: Izquierda Belkin, derecha Trust	45
Fig. 7.9 Consumo modo <i>Scan</i> continuo del dispositivo Sena	45
 Fig. 8.1 Croquis de posibles escenarios de señales de tráfico	46
Fig. 8.2 Ejemplo estructuras de tipos de datos propuesto.	49

ÍNDICE DE TABLAS

Tabla I.1. Prestaciones para el reconocimiento de señales de tráfico [1].....	2
Tabla 1.1. Comparación entre BLE y Bluetooth clásico	5
Tabla 1.2. Comparación de BLE con otros estándares	9
Tabla 2.1. Tipos de paquetes de <i>Advertising</i> y sus características	11
Tabla 4.1. <i>LE Set Scan Parameters Command</i>	20
Tabla 5.1. <i>Flags</i>	24
Tabla 5.2. <i>Local Name</i>	24
Tabla 5.3. <i>LE Set Advertising Data Command</i>	25
Tabla 5.4. <i>LE Set Advertising Parameters Command</i>	26
Tabla 8.1. Ejemplo de asignación de números para los tipos de datos propuestos.....	48
Tabla 8.2. Ejemplo de distribución de bits para el tipo de dato “Estado”	48
Tabla 8.3. Ejemplo de distribución de bytes y descripción de los tipos de datos propuestos.....	49

Índice de acrónimos

AD	Advertising Data
ADV_IND	Advertising Indicator Packet
ADV_NONCONN	Advertising Non Connectable Packet
API	Application Programming Interface
ATT	Attribute protocol
BD ADDR	Bluetooth Device Address
BLE	Bluetooth Low Energy
BR	Basic Rate
CRC	Cyclic Redundancy Check
DSSS	direct sequence spread spectrum
EDR	Enhanced Data Rate
FHSS	Frequency Hopping Spread Spectrum
GAP	Generic Access Profile
GATT	Generic Attribute Profile
GFSK	Gaussian Frequency Shift Keying
ISI	Intersymbol Interference
ISM	Industrial Scientific Medical
LE	Low Energy
LED	Light-Emitting diode
MAC	Media Access Control
NMEA	National Marine Electronics Association
OCF	OpCode Command Field
OFDM	Orthogonal Frequency Division Multiplexing
OGF	OpCode Group Field
PC	Personal Computer
PDU	Protocol Data Unit
SIG	Special Interest Group
TFG	Trabajo Final de Grado

Quiero agradecer por su ayuda, soporte y sobretodo su confianza en mí por asignarme este proyecto, a los profesores José Luis Valenzuela y David Pérez Díaz de Cerio.

Agradecer a mis padres, Saida y Mohammed que siempre me apoyaron incondicionalmente para poder llegar a ser lo que soy.

A mis hermanas y demás familiares en general.

A todos mis profesores por la ayuda y los conocimientos proporcionados.

A mis compañeros de la universidad con los que he compartido grandes momentos.

A mis amigos, por estar siempre a mi lado.

A todos aquellos que siguen estando cerca de mí y que le regalan a mi vida algo de ellos.

INTRODUCCIÓN

Este trabajo propone un uso de la tecnología Bluetooth al sector automovilístico dando una alternativa a los sistemas de detección de señales de tráfico mediante cámaras que llevan algunos vehículos de hoy en día. Los sistemas basados en cámara muestran una tasa de error alta en ciertas condiciones como por ejemplo las meteorológicas, lluvia, nieve, niebla y las derivadas del posicionamiento de la señal de tráfico respecto al vehículo. Sin embargo, estas desventajas que afectan directamente al funcionamiento de los sistemas visuales no existen en un sistema de detección de señales basado en sistemas de radiofrecuencia.

Bluetooth Low Energy (BLE) o Bluetooth Smart es la nueva especificación de Bluetooth. Se trata de una nueva tecnología destinada a sectores como la domótica, el deporte, la salud y por supuesto se ha integrado totalmente en los nuevos *smartphones*. Sus características, tales como el bajo coste, el bajo consumo de energía y su sencilla configuración abrieron muchas puertas para nuevas aplicaciones que pueden marcar una notable evolución en nuestra vida cotidiana.

El sistema de detección de señales basado en cámaras, que a menudo se ven instaladas en el parabrisas de algunos vehículos, es la solución adoptada por los gigantes automovilísticos. Estas cámaras detectan las señales de tráfico con visibilidad directa. Por lo cual, se ven afectadas por condiciones meteorológicas adversas y por la interposición de objetos.

Después de la detección, viene la parte del procesamiento de la imagen: como la distancia entre la imagen y la cámara cambia (vehículo en movimiento), obviamente su tamaño también cambia. Si la imagen es muy pequeña, el reconocimiento de la misma se hace difícil.

Para el correcto reconocimiento de la señal de tráfico, no solo influye el tamaño de la imagen (la distancia), sino también el ángulo que forma la cámara respecto la imagen. Esto exige algoritmos de procesamiento de imágenes robustos cuya complejidad puede aumentar el coste computacional, que de no utilizarse provocaría que muchos casos reales de señales de tráfico fueran irreconocibles por este sistema.

A continuación, se presentan los resultados [1] de pruebas que se han hecho sobre cinco coches de fabricantes diferentes y que muestran varias desventajas que afectan al funcionamiento de este sistema.

Se observa que en el caso de Volkswagen Phaeton, Opel Insignia Sport Tourer y Audi A8 se tienen problemas en el reconocimiento de señales de tráfico de LED's (variables). Respecto al reconocimiento nocturno, los que tienen un porcentaje bajo son todos menos Mercedes. Y en el caso del Opel Insignia, se tiene otro inconveniente que es el de la posición de la cámara, ya que invade el campo de visión, por lo tanto pasa desapercibido un porcentaje importante de señales de tráfico.

Tabla I.1. Prestaciones para el reconocimiento de señales de tráfico [1].

	Puntuación	Reconocimiento general	Reconocimiento diurno	Reconocimiento nocturno	Reconocimiento de señales fijas	Reconocimiento de señales variables	Reconocimiento en zonas de obras en carretera
Ponderación			3	1	1	1	1
Coche							
BMW 740d	1,9	92 %	95 %	82 %	95 %	95 %	90 %
Mercedes S 500 CGI	2,0	91 %	92 %	90 %	95 %	85 %	90 %
Audi A8	2,3	88 %	89 %	82 %	95 %	85 %	90 %
Opel Insignia	2,9	83 %	81 %	81 %	95 %	75 %	90 %
VW Phaeton	2,8	84 %	87 %	74 %	90 %	80 %	85 %

Este experimento con estos cinco coches no ha estudiado las situaciones meteorológicas tales como la lluvia, niebla y la nieve que seguramente afectarían a la visibilidad de las cámaras tal como lo hace el simple hecho de anochecer.

Por lo cual, dejando aparte todas las ventajas de BLE que se describirán en este trabajo, la propuesta de BLE para sustituir las cámaras de reconocimiento de señales de tráfico es completamente correcta, ya que con el simple hecho de que sea un protocolo inalámbrico solucionaría todos estos problemas (básicamente de visibilidad).

En lo que se refiere al problema de detección de las señales LED variables; en este trabajo se demuestra que la información que puede llevar algunos paquetes de *Advertising* es completamente modificable. Además de una manera muy sencilla, en tres líneas de comandos (deshabilitar, poner los datos y habilitar) se puede configurar la información que se desea transmitir.

Para esta nueva propuesta se ha realizado un estudio general de BLE, analizando el funcionamiento de los procedimientos de *Advertising/Scanning* y su configuración. Se han analizado medidas de tiempo de descubrimiento. El estudio se ha enfocado en los tipos de *Advertising* que encajan con el objetivo de la propuesta. También se han realizado medidas de consumo de energía y de cobertura.

Para llegar al objetivo, este trabajo se ha dividido en 9 capítulos:

1- Primero familiarizarse con la tecnología BLE, describiendo su arquitectura de protocolos, haciendo un estudio comparativo con otras tecnologías y explicando las soluciones que proporciona BLE a los problemas que tienen los sensores de detección de señales de tráfico.

- 2- El segundo capítulo se centra en la descripción de los procedimientos de *Advertising*, concretamente los tipos y su funcionamiento.
- 3- En el tercer capítulo se presenta la manera de configurar los dispositivos vía comandos y las herramientas que se han utilizado para obtener las medidas.
- 4- En el cuarto capítulo se explica la implementación del procedimiento de *Scanning* (Pasivo/Activo).
- 5- En el quinto capítulo se explica la implementación del procedimiento de *Advertising*, enfocando el estudio en los tipos de *Advertising* que podrían ser candidatos en aplicaciones de envío (difusión) de pequeñas cantidades de datos sin necesidad de establecer conexión.
- 6- Con el fin de llegar a conclusiones del rendimiento y del funcionamiento de esta nueva tecnología, hemos analizado el comportamiento de los *beacons* haciendo medidas de tiempo de descubrimiento variando parámetros de *Scanning* y de *Advertising*. En este capítulo se presenta los resultados de estas medidas.
- 7- Se presentan medidas de corriente en modo *Scanning* y *Advertising* de dispositivos de varios fabricantes, también se destaca el efecto del cambio de parámetros en el consumo energético.
- 8- Se describe cómo sería un sistema basado en BLE que sirve para la detección de señales de tráfico y sustituir los sensores que llevan los vehículos de hoy en día, solucionando problemas de funcionamiento de este último sistema.
- 9- Por último, una reflexión sobre el impacto social y las conclusiones.

CAPÍTULO 1. DESCRIPCIÓN DE BLUETOOTH LOW ENERGY

Bluetooth Low Energy (BLE) es la nueva especificación 4.0 [2], 4.1 y 4.2 de la tecnología Bluetooth desarrollado por Bluetooth Special Interest Group (SIG). Se ha diseñado como una tecnología complementaria a Bluetooth clásico para garantizar un consumo de energía bajo, y menor tiempo de establecimiento de conexión. A pesar del uso de la misma banda de frecuencia y las similitudes compartidas, BLE debe considerarse un nuevo estándar con objetivos y aplicaciones diferentes.

BLE está diseñado para la transmisión de pequeñas cantidades de datos (tiempos de transmisión muy pequeños) y por lo tanto de ultra-bajo consumo de energía. No está pensado para mantener una conexión entre dispositivos por un largo tiempo transmitiendo grandes cantidades de datos a alta velocidad. Esto permite que los dispositivos estén activos solo cuando se les pide la transmisión de datos. Con la idea de “Obtén lo que quieras cuando quieras” surgen aplicaciones muy útiles y en el caso de este trabajo, surge el posible uso de esta tecnología en el sector automóvil.

1.1. Arquitectura

La topología de red de BLE es de tipo estrella. Los dispositivos Master pueden tener varias conexiones de capa de enlace con periféricos (*Slaves*) y simultáneamente realizar búsquedas de otros dispositivos. Por otro lado un dispositivo en rol de esclavo solo puede tener una conexión de capa de enlace con un único Master. Además, un dispositivo puede enviar datos en modo *Broadcast*, eventos de *Advertising*, sin esperar ninguna conexión; esto permite enviar datos a los dispositivos en estado *Scanning* sin necesidad de establecer la conexión *Master-Slave*.

Para la gestión de los dispositivos, la conexión y la interfaz de las aplicaciones, el SIG de Bluetooth define una pila de protocolos. La pila del protocolo BLE se divide en tres partes básicas: *Controller*, *Host* y *Applications*. El *Controller* es el dispositivo físico que permite transmitir y recibir señales radio e interpretarlas como paquetes con información. Contiene la capa física “*Physical Layer*”, “*Direct Test Mode*”, la capa de enlace “*Link Layer*” y la interfaz de control de host “*Host Controller Interface*”.

El Host es la pila de software que administra cómo dos o más dispositivos se comunican entre ellos. No está definida ninguna interfaz superior para el Host, cada sistema operativo o entorno tiene su propia manera de exponer API's (*Application Programming Interface*) de Host para los desarrolladores. Esta parte de la pila contiene una capa de control de enlace lógico y de protocolo de adaptación “*Logical Link Control and Adaptation Protocol*”, administrador de seguridad “*Security Manager*”, protocolo de atributo “*Attribute protocol (ATT)*”, perfil de atributo genérico “*Generic Attribute Profile (GATT)*” y perfil de acceso genérico “*Generic Access Profile (GAP)*”.

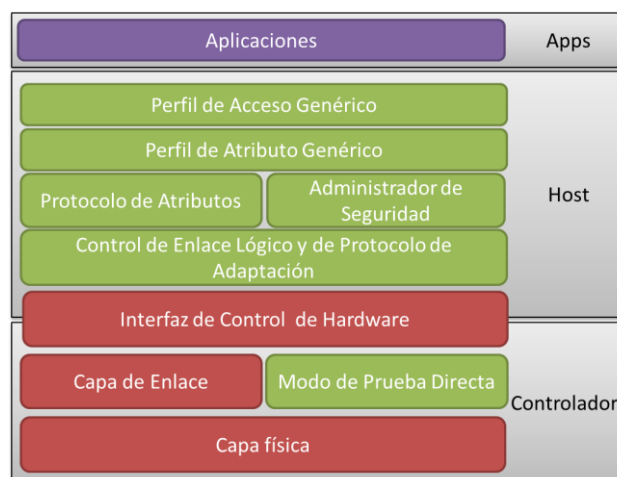


Fig. 1.1 Pila del protocolo BLE

Para cada caso de uso, las aplicaciones utilizan la pila de software, y a su vez ésta utiliza el controlador.

A continuación se describen las capas de la pila de protocolos (los marcados en color rojo en la figura 1.1) entrando más o menos en detalles, dependiendo de la importancia que tienen esos detalles en los experimentos que se han realizado y que se analizarán más adelante.

1.1.1. Capa Física

Bluetooth Low Energy comparte algunas similitudes con el Bluetooth clásico. Los dos usan la banda de 2.4 GHz. Bluetooth clásico y BLE usan la modulación GFSK a 1Mbps, pero con índices de modulación diferentes. *Enhanced Data Rate* (EDR) usa una modulación completamente diferente de la GFSK. El estándar Bluetooth clásico tiene 79 canales mientras BLE tiene 40 tal como indica la tabla 1.1. La separación entre canales también es diferente. Debido a estas dos diferencias entre BLE y Bluetooth clásico, éstos son incompatibles entre sí, por lo tanto no se pueden comunicar. Sin embargo, existen dispositivos *Dual Mode* que soportan las dos tecnologías conmutando los parámetros de modulación y los canales donde se está radiando.

Tabla 1.1. Comparación entre BLE y Bluetooth clásico

	BLE	BR	EDR
Modulación	GFSK 0.45 to 0.55	GFSK 0.28 to 0.35	DQPSK / 8DPSK
Tasa Mbit/s	1Mbit/s	1 Mbit/s	2 and 3 Mbit/s
Nº Canales	40	79	79
Separación	2MHz	1MHz	1MHz

La capa física es la que se encarga de enviar las señales al aire, transmitiendo y recibiendo bits usando ondas radio en la banda de frecuencia Industrial Scientific Medical (ISM) 2.4 GHz que se extiende desde 2402 MHz hasta 2480 MHz. La separación entre los 40 canales utilizados es de 2 MHz (numerados de 0 a 39 y de 1 MHz de anchura cada uno). Existen 3 canales dedicados para el *Advertising* y 37 para la transmisión de datos. Los canales 37, 38, y 39 son usados solo para el envío de paquetes de *Advertising*. El resto son usados para el intercambio de datos durante la conexión.

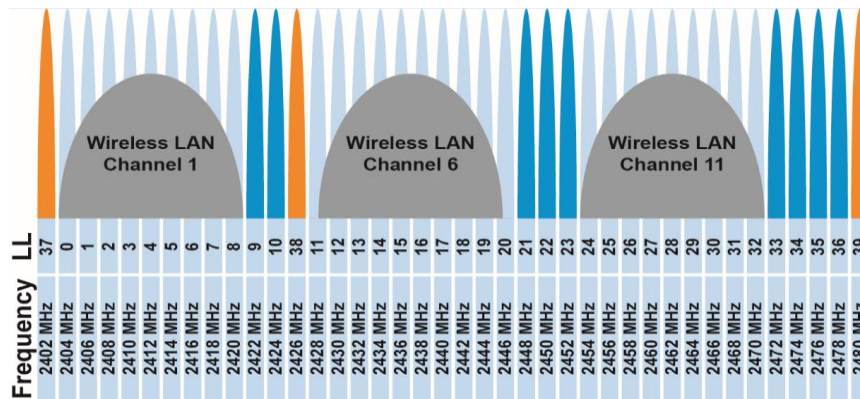


Fig. 1.2 Distribución de canales en BLE

Estos tres canales de *Advertising* están situados estratégicamente (**Fig. 1.2**) para evitar interferencias causadas por otras tecnologías que coexisten en el mismo espectro (IEEE 802, y ZigBee). Además, en estado de conexión, BLE utiliza la técnica de Frequency Hopping Spread Spectrum (FHSS) para reducir interferencias.

BLE utiliza una modulación gaussiana con desplazamiento de frecuencia. Ésta utiliza dos frecuencias para identificar el bit '1' o '0'. El filtro gaussiano se usa para suavizar las transiciones entre frecuencias y reducir el ensanchado de espectro causado por la ISI. La especificación de BLE [3] limita la potencia transmitida máxima a +10 dBm y la mínima en -20 dBm. La sensibilidad recibida mínima requerida para BLE es de -70 dBm, aunque la mayoría de dispositivos BLE tienen una sensibilidad menor a -85 dBm.

1.1.2. Capa de Enlace

Esta es la capa responsable de los estados de *Advertising*, *Scanning*, creación y mantenimiento de las conexiones. También es la responsable de la estructura de los paquetes.

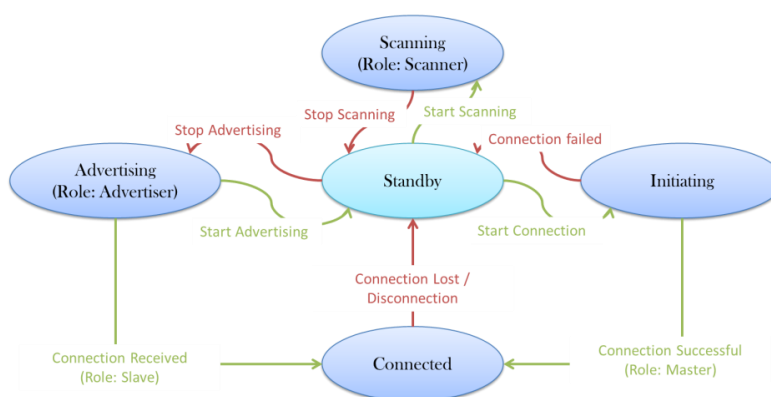


Fig. 1.3 Estados y roles de BLE

- **Standby:** básicamente el dispositivo ni transmite ni recibe. Por lo general, este estado está asociado con un sistema durmiente para conservar energía.
- **Advertising:** el dispositivo que tiene el rol de periférico entra en estado de *Advertising* en el que envía paquetes de *Advertising* en los canales de *Advertising*. En este estado también escucha cualquier respuesta (solicitud) de los paquetes desde el dispositivo central. Este modo es de los más críticos para analizar desde el punto de vista de la potencia ya que el dispositivo periférico tardará más o menos un tiempo de *Advertising* (anunciándose) dependiendo de la aplicación. Hay que tener en cuenta que el tiempo de transmisión afecta al consumo de energía, por tanto el intervalo de *Advertising* afecta directamente al consumo de potencia y la vida de las baterías.
- **Scanning:** se refiere a escuchar a paquetes de *Advertising* enviados a través de sus canales. Este modo se usa para explorar dispositivos.
- **Initiating:** en general, este estado es el estado al que entra el dispositivo central antes de pasar a estado de conexión. El dispositivo central escucha los *Advertising* de periféricos, pero una vez recibe el *Advertising* del periférico deseado, el central debe conectar enviando los datos correctos.

Para el *Slave*, el estado de *Advertising* también se considera como estado inicial antes del estado de conexión. El estado de conexión es el último estado en el que el *Slave* (periférico) y el *Master* (Central) pueden intercambiar datos. Éstos se intercambian datos periódicamente a través de eventos de conexión.

Se denomina *Advertiser*, al dispositivo BLE que utiliza los canales de *Advertising* para emitir datos, anunciar que es conectable y detectable o “descubrible”. A un dispositivo en estado de escaneo se le denomina *Scanner* y cuando se inicia una conexión, *Initiator*. Los canales de datos se utilizan una vez establecida la conexión entre dispositivos.

El dispositivo *Scanner* busca en los canales de *Advertising* paquetes de *Advertising* de otros dispositivos. Existen cuatro tipos de *Advertising*: *General*, *Directed*, *Nonconnectable* y *Discoverable*. Éstos se desarrollan con más detalle

en el capítulo siguiente, una descripción breve sería:

- **Connectable Undirected Advertising:** es el más común. El dispositivo escáner puede recibir los anuncios o ir a hacia una conexión.
- **Connectable Directed Advertising¹:** se usa cuando un dispositivo necesita conectarse de manera rápida. Este *Advertising* debe repetirse cada 3.75 milisegundos. Se permite estar en este estado un tiempo máximo de 1.28s.
- **Nonconnectable Advertising:** es usado por dispositivos que quieren emitir (difundir) datos. El dispositivo no puede estar detectable (no puede recibir *Scan Request*) ni en conexión.
- **Discoverable Advertising:** no se puede usar para iniciar una conexión, pero sí para ser escaneado por otros dispositivos y detectar *Scan Request*. El dispositivo escáner puede obtener datos del *Advertiser* ya que éste responde a cada *Scan Request* detectado con un *Scan Response*. También se puede utilizar para emitir datos.

1.1.3. Host/Controller Interface

Esta capa define la conexión física entre el *Host* y el *Controller* vía comandos HCI. La especificación define varias interfaces físicas: UART, 3Wire UART, USB y SDIO.

Algunos comandos están relacionados con la configuración del dispositivo y por lo tanto con su funcionamiento. Estos comandos son muy importantes en este trabajo ya que sirven para hacer pruebas de configuración y obtener medidas.

1.2. Comparación con otros estándares

La tabla 1.2 permite comparar las características de BLE con otras tecnologías de comunicación inalámbricas, que operan en la misma banda, y que además, se consideran en muchos sectores competencia directa. Zigbee, Wifi y BLE utilizan la banda de 2.4GHz, pero con capacidades completamente diferentes. El alcance máximo depende tanto de la sensibilidad como del chipset del fabricante, con un PIRE máximo en la banda de 2.4 GHz es de 100 mW. Experimentalmente se han obtenido distancias de conexión de más de 1 km.

¹ En la especificación v4.2 se introduce dos tipos de *Advertising* derivados del **Directed Advertising**: *High Duty Cycle Directed Advertising* y *Low Duty Cycle Directed Advertising*.

Tabla 1.2. Comparación de BLE con otros estándares

	BLE	Wi-Fi	Zigbee
Banda de Frecuencia	2.4 GHz	2.4 GHz / 5 GHz	2.4 GHz
Modulación	GFSK	OFDM, DSSS	DSSS
Rango	100 m	100 m	200 m
Tipología de Red	Scatternet	Star	Mesh
Tasa	1 Mbps	1 Mbps -7 Gbps	250 kbps
Consumo de Pico de Corriente	<15 mA	60 mA RX, 200 mA TX	19 mA RX, 35 mA TX
Corriente en Standby	< 2 μ A	< 100 μ A	5 μ A

A la hora de elegir la batería o la fuente de alimentación, es fundamental fijarse en el consumo de corriente. BLE tiene el valor más bajo, de unos 15 mA, sin tener en cuenta el consumo en el estado *Standby*, que es prácticamente nulo (2 μ A).

Por lo tanto, resumimos la elección de BLE para este estudio en 4 puntos:

- **Bajo consumo de potencia:** hoy en día, es importante para todos los proyectos el consumo de energía ya que se refleja directamente en el coste de las infraestructuras.
- **Fácil de implementar y de instalar:** dispositivos pequeños (ocupan poco espacio), configurables y que se les puede integrar antenas directivas, incluso se les puede integrar paneles fotovoltaicos para su alimentación.
- **Soportado por smartphones:** los *smartphones* están presentes en nuestra vida cotidiana, por lo tanto la propuesta de este trabajo debe ser completamente compatible con los teléfonos móviles de hoy en día.
- **Larga vida de batería:** se requiere una vida larga de batería para la tecnología elegida con el fin de reducir el mantenimiento de los equipos. Incluso se está estudiando la posibilidad de incorporar a su diseño baterías recargables mediante paneles fotovoltaicos.

Todos estos requisitos se cumplen perfectamente para el caso de BLE como se muestra en la tabla. Zigbee tiene desventajas que no le permiten competir para dar soluciones comerciales como por ejemplo el inconveniente de no estar soportado por los *smartphones*. Por otro lado, Wifi está desplegado en muchos espacios interiores (centros comerciales, aeropuertos, parkings...) e exteriores también. Sin embargo, esta infraestructura basada en routers, conmutadores y puentes tiene un alto consumo de energía, por lo tanto, hace difícil su mantenimiento y su despliegue en todas las carreteras.

CAPÍTULO 2. DESCRIPCIÓN DEL PROCEDIMIENTO DE *ADVERTISING*

En los capítulos anteriores se han descrito las dos maneras para obtener datos (información) de un dispositivo BLE:

- **Advertising y Scan Responses:** Los paquetes de *Advertising* incluyen campos donde se puede proporcionar información, y el dispositivo central puede solicitar más información con un *Scan Request*. No hay seguridad esencial para este método ya que en modo *Broadcasting* la emisión de datos es pública, sin embargo todos los dispositivos que detectan el *Advertising* pueden recibir la información al mismo tiempo.
- **Conexiones:** Durante una conexión, dos dispositivos pueden intercambiar información, lo cual requiere procedimientos más complejos comparados con otros métodos.

En este capítulo, se van a presentar los tipos de *Advertising* y su funcionamiento. A continuación se describe el parámetro del intervalo de *Advertising* que tienen en común los cuatro tipos.

2.1. El intervalo de *Advertising*

Durante el estado de *Advertising*, se envían paquetes de *Advertising* periódicamente en cada uno de los tres canales de *Advertising*. El intervalo de tiempo que separa el envío de estos paquetes es la suma de un intervalo fijo y un retardo aleatorio.

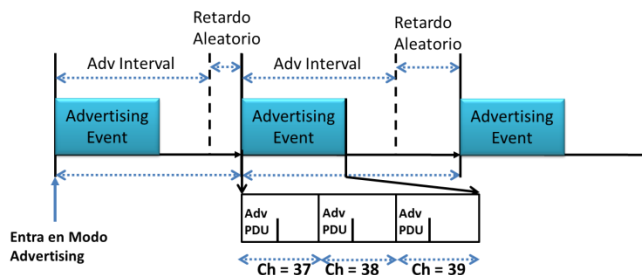


Fig. 2.1 Advertising Interval

El intervalo fijo es el *Advertising Interval* y se puede configurar entre 20 ms y 10.24 s, en pasos de 0.625 ms, excepto para los tipos *Non-connectable* y *Scannable* que como mínimo tiene que ser de unos 100 ms. El retardo es un valor aleatorio entre 0 ms y 10 ms que se añade automáticamente. Este último valor ayuda a reducir colisiones entre *Advertising* de dispositivos diferentes. De esta manera, BLE mejora la robustez del protocolo haciendo más fácil la búsqueda de los *Advertising* por parte del *Scanner*.

2.2. Advertising Channel PDU

La estructura del paquete *Advertising* es la misma que la de datos. Los bits se transmiten de derecha a izquierda. Por ejemplo, **0x80** se transmite como 00000001 enviando el bit menos significativo primero.

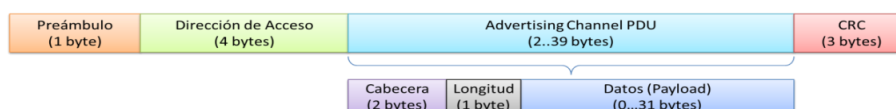


Fig. 2.2 Advertising Channel PDU

Preámbulo: los ocho primeros bits son 10101010 o 01010101.

Dirección de Acceso: existen dos tipos: direcciones de acceso de *Advertising* y direcciones de acceso de datos. El primero se usa para emitir (difundir) datos, escanear o inicializar conexiones y siempre es el mismo **0x8E89BED6**. El segundo se usa en una conexión y cambia en cada conexión a nivel de capa de enlace.

Cabecera: el contenido de este campo depende de si es un paquete de *Advertising* o de datos. Para un paquete de *Advertising* el campo de cabecera indica el tipo de paquete de *Advertising*.

Longitud: este campo indica la longitud de los datos.

Datos (Payload): es la información real que se transmite. Pueden ser datos de *Advertising* de un dispositivo *Broadcaster*, datos de *Scan Response*, o simplemente datos de comunicación entre dispositivos conectados.

CRC: son 3 bytes de Cyclic Redundancy Check.

A continuación, se presentan los 4 tipos de eventos de Advertising y su funcionamiento.

Tabla 2.1. Tipos de paquetes de *Advertising* y sus características

<i>Advertising Packet Type</i>	<i>Connectable</i>	<i>Scannable</i>	<i>Directed</i>	<i>GAP Name</i>
ADV_IND	Yes	Yes	No	Connectable Undirected Advertising
ADV_DIRECT_IND	Yes	No	Yes	Connectable Directed Advertising
ADV_NONCONN_IND	No	No	No	Non-connectable Undirected Advertising
ADV_SCAN_IND	No	Yes	No	Scannable Undirected Advertising

2.3. *Connectable Undirected*

Este tipo de eventos es el más común y el más genérico de todos, es genérico en el sentido de que no es dirigido y es conectable. O sea, un dispositivo central puede conectarse a otro periférico que está en estado de *Advertising* y no está dirigido hacia ningún dispositivo central concreto.

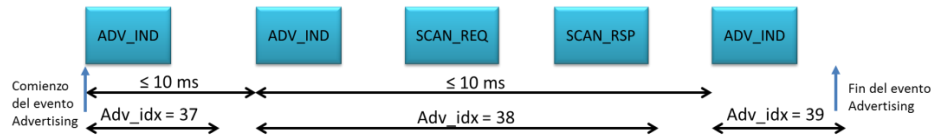


Fig. 2.3 *Connectable Undirected Advertising*

Cuando un periférico envía paquetes de *Advertising* (ADV_IND), está ayudando a dispositivos centrales a encontrarle, una vez encontrado, el dispositivo central puede empezar el proceso de conexión.

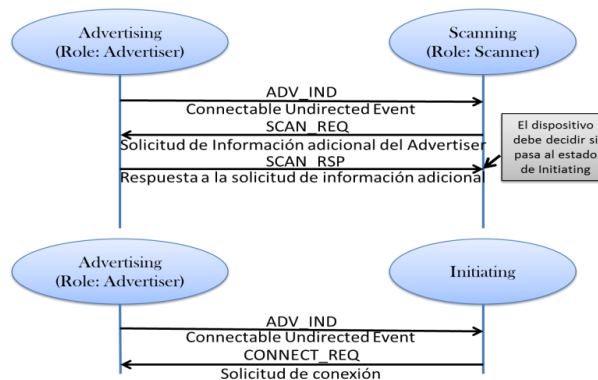


Fig. 2.4 Flujo de mensajes del modo *Connectable Undirected Advertising*

Para informar que ha sido encontrado por el *Scanner*, el *Advertiser* recibe un *Scan Request* al cual contesta con un *Scan Response* en el que completa la información necesaria para que el *Scanner* cambie de estado a *Initiating*. Estando en el estado *Initiating*, el siguiente *Advertising* que recibe se le contesta con un *Connect Request*, momento en el que empieza el proceso de conexión.

2.4. *Connectable Directed*

Este evento se usa cuando un *Advertiser* quiere que se conecte con un dispositivo BLE concreto. Después de recibir este tipo de PDU por el dispositivo

solicitado, este último puede contestar con un *Connect Request* para solicitar una conexión con el *Advertiser*.

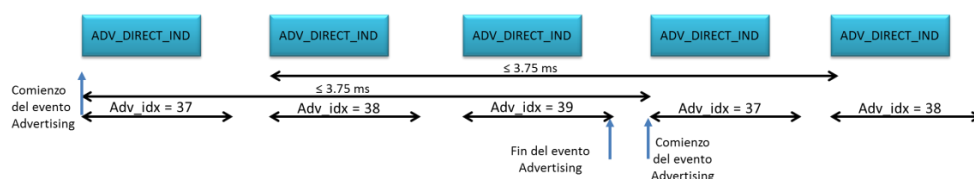


Fig. 2.5 *Connectable Directed Advertising*

Estos paquetes contienen dos direcciones: la dirección del *Advertiser* y la del *Initiator*. Además tienen que cumplir unos requisitos de tiempo especiales, dos paquetes de este tipo de *Advertising* enviados por el mismo canal (dos eventos consecutivos) tienen que estar separados como máximo unos 3.75 ms. Este requisito permite un escaneo de solamente 3.75 ms para detectar los *Advertising*.

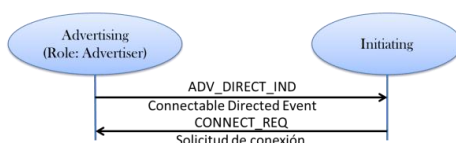


Fig. 2.6 Flujo de mensajes del modo *Connectable Directed Advertising*

Por lo tanto, de esta forma puede conectarse de manera muy rápida, pero esta estrategia hace que se congestionen los canales rápidamente. Por esta razón, no se permiten utilizar los *Advertising Directed* por un tiempo superior a 1.28 s. Si no se consigue establecer una conexión antes de los 1.28 s el *Controller* para los *Advertising* de forma automática. Hay que destacar unas características importantes: cuando se usa este tipo de *Advertising*, el dispositivo ignora cualquier *Scan Request* recibido. Además, sus paquetes no permiten tener ningún tipo de datos adicionales, solamente las dos direcciones afectadas y nada más.

2.5. *Nonconnectable Advertising*

Este tipo de eventos de *Advertising* es el utilizado por dispositivos que no tienen intención de conectarse a ningún otro dispositivo, incluso no quieren ni saber si están siendo escaneados por otros dispositivos centrales. Básicamente, se utilizan para emitir datos en modo *Broadcast*. Los dispositivos *Scanner* solo pueden escuchar la información emitida, no pueden ni conectarse ni solicitar más información.

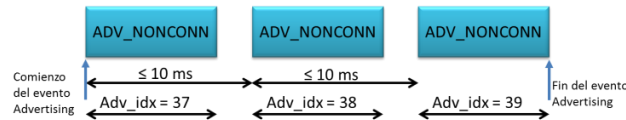


Fig. 2.7 *Nonconnectable Advertising*

El tiempo entre dos paquetes ADV_NONCONN enviados en canales consecutivos tiene que ser como máximo 10 ms.

Este tipo de *Advertising*, desde un principio interesa para llegar al objetivo de este TFG, ya que puede ser útil para aplicaciones que simplemente quieren detectar la existencia de un objeto o un estado, por ejemplo. Por lo tanto, más adelante se va a desarrollar más y se va a implementar para recoger medidas y conclusiones.

2.6. *Discoverable Advertising*

El último tipo es parecido al anterior con la diferencia de que es escaneable. O sea, no puede entrar en conexión pero puede además de enviar *Advertising*, escuchar *Scan Request* de un *Scanner* activo (Sección 4.2) y responder con un *Scan Response*. En este caso no solo se pueden enviar datos en los paquetes de *Advertising*, sino también se pueden enviar en paquetes *Scan Response*. Y por tanto puede ser utilizable para enviar información adicional.

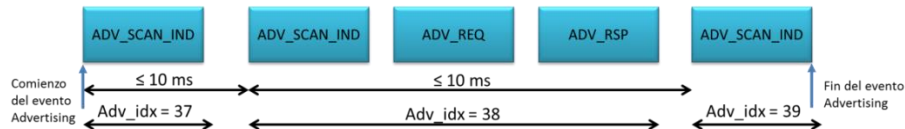


Fig. 2.8 *Discoverable Advertising*

CAPÍTULO 3. IMPLEMENTACIÓN DE PROCEDIMIENTOS DE CONFIGURACIÓN

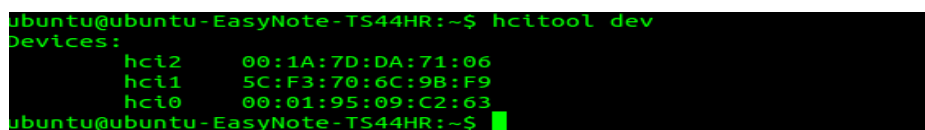
En este capítulo se explica las herramientas utilizadas, cómo instalarlas y comprobar su buen funcionamiento. Primero nos situamos en la pila del protocolo para saber desde donde se van a lanzar las órdenes: el terminal del sistema operativo Ubuntu 15.04 con un BLE conectado mediante un USB, esto equivale en la pila de BLE que desde el *Host* se enviarán las órdenes al *Controller* (concretamente a la capa de enlace, nivel más alto del *Controller*) mediante la interfaz física USB.

Para controlar los dispositivos Bluetooth desde Linux, BlueZ ofrece una pila del protocolo Bluetooth. Su objetivo es ofrecer una implementación de las especificaciones de los estándares inalámbricos Bluetooth para Linux. A partir de 2006, la pila BlueZ soporta todos los protocolos Bluetooth y sus capas. Por lo tanto, BlueZ es una herramienta fundamental que permite utilizar el terminal para enviar comandos HCI definidos en la especificación v4.0.

A continuación se muestra la serie de comandos para su instalación:

```
wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.18.tar.xz
dpkg --get-selections | grep -v deinstall | grep bluez
tar xvf bluez-5.18.tar.xz
sudo apt-get install libglib2.0-dev libdbus-1-dev libusb-dev libudev-dev libical-dev systemd libreadline-dev
./configure --enable-library
make -j8 && sudo make install
sudo cp attrib/gatttool /usr/local/bin/
```

Después de instalar correctamente BlueZ, se pueden utilizar comandos para obtener información, configurar un dispositivo, o para dar una orden. Para ello se conectan los dispositivos BLE que se quieren analizar y se ejecuta el comando **hcitool dev**. El resultado sería parecido a la captura siguiente:



```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hcitool dev
Devices:
    hci2    00:1A:7D:DA:71:06
    hci1    5C:F3:70:6C:9B:F9
    hci0    00:01:95:09:C2:63
ubuntu@ubuntu-EasyNote-TS44HR:~$
```

Fig. 3.1 Comando hcitool dev

El resultado muestra todos los dispositivos BLE que están conectados al PC vía USB con sus respectivas direcciones BD ADDR (Bluetooth Device Address), únicas como las conocidas MAC. También presenta el nombre con el que hay que referirse a la hora de realizar cualquier prueba u operación: por ejemplo, para escanear con el dispositivo de la BD ADDR 5C:F3:70:6C:9B:F9 hcitool ofrece el comando **scan**, pero antes debemos especificar con la opción -i el dispositivo elegido para escanear, en este caso sería hci1.

```

ubuntu@ubuntu-EasyNote-TS44HR:~$ hcitool -i hci1 scan
Scanning ...
    00:1A:7D:DA:71:06      ubuntu-EasyNote-TS44HR-0
    00:01:95:09:C2:63      ubuntu-EasyNote-TS44HR-2
ubuntu@ubuntu-EasyNote-TS44HR:~$

```

Fig. 3.2 Comando hcitool scan

Este tipo de escaneo es el que viene configurado en los dispositivos Bluetooth por defecto, y es el utilizado por el Bluetooth clásico. Su intervalo de escaneo es de 1.28 s, con este tipo de escaneo se detectan los dos dispositivos visibles que previamente se han conectado al mismo PC.

Para obtener más información sobre los dispositivos se usa el comando **inq**:

```

ubuntu@ubuntu-EasyNote-TS44HR:~$ hcitool -i hci1 inq
Inquiring ...
    00:1A:7D:DA:71:06      clock offset: 0x0000      class: 0x600100
    00:01:95:09:C2:63      clock offset: 0x38bd      class: 0x600100
ubuntu@ubuntu-EasyNote-TS44HR:~$

```

Fig. 3.3 Comando hcitool inq

En este caso se muestra además de la BD ADDR, el clock offset que indica la diferencia de relojes entre dispositivos y el parámetro class es (*Class of Device*) que indica de qué clase de dispositivo se trata.

Con **hcitool** se puede solicitar mucha información de los dispositivos, o especificar qué información se quiere como, por ejemplo, el nombre con el comando **name**, o **info** que devuelve además del nombre, la versión y otras características del dispositivo.

El comando de **hcitool** más utilizado en este trabajo va a ser **cmd** que permite configurar y ejecutar todos los parámetros HCI de los dispositivos Bluetooth.

Según la sección **HCI Command Packet** [4], el primer campo del paquete es de dos bytes e indican el código del comando, estos dos bytes se dividen en dos partes: la primera de 6 bits es para el campo de código de grupo o **OGF** (*OpCode Group Field*) la segunda de 10 bits indica el campo de código de comando o **OCF** (*OpCode Command Field*).

Primero se busca en la especificación el OGF que permite modificar parámetros de dispositivos *Low Energy*, en la sección **LE CONTROLLER COMMANDS** de [4] se encuentra que para comandos de *LE Controller* está definido el **0x08**. Después, cada comando tiene asignado un código. También para cada comando, se indican los campos que habrá que rellenar describiendo las opciones, longitud y más detalles.

Por ejemplo, para el caso del comando **HCI_LE_Set_Scan_Enable** que lo que hace es simplemente habilitar el escaneo en modo LE, tiene asignado como **OCF** el **0x000C**. Además de dos campos de un byte cada uno: el primero para

habilitar o deshabilitar el escaneo (**0x01** o **0x00** respectivamente) y el segundo para habilitar o deshabilitar el filtro de duplicados (**0x01** o **0x00**).

Entonces si lo que se quiere es habilitar un escaneo LE sin filtrar los paquetes de *Advertising* duplicados se ejecuta el comando siguiente:

```
Sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

Por lo tanto, así tendremos un dispositivo BLE en estado de *Scanning* (búsqueda de otros dispositivos BLE en estado de *Advertising*), además generando un *Advertising report* por cada paquete de *Advertising* recibido (filtro duplicados deshabilitado).

Otra familia de comandos útil es la de **hciconfig**, comando que sirve para la configuración de los dispositivos, sobre todo porque permite configurar con comandos predefinidos como por ejemplo, encender, apagar o hacer un reset. Incluso habilitar o deshabilitar un modo, como el siguiente comando:

```
Sudo hciconfig -a hci0 noscan
```

Este último comando deshabilita el escaneo del Bluetooth clásico (*Inquiry* y *Page Scan*) que viene habilitado por defecto. O sea, si se desea deshabilitar este modo, esta otra herramienta ahorraría el trabajo de búsqueda en la especificación sobre cómo sería el comando HCI en hexadecimal.

Para ver cómo se reciben los paquetes de *Advertising* se puede realizar de tres formas, las tres usan el programa de captura de tráfico de paquetes Wireshark. La primera forma de capturar el tráfico generado por la comunicación entre dispositivos BLE es utilizando el comando siguiente: `sudo hcidump -w inq.cap`

```
sudo hcidump -w inq.cap
```

Al matar el proceso (ctrl + z) se crea un fichero .cap (en este caso inq.cap) que permite analizar lo que ha ocurrido durante el tiempo de la captura, por lo tanto debemos ejecutar al mismo tiempo las órdenes que se quieren analizar (usando dos terminales). El fichero .cap se puede leer mediante Wireshark. Para instalar Wireshark:

```
$ sudo add-apt-repository ppa:pi-rho/security  
$ sudo apt-get update  
$ sudo apt-get install wireshark
```

Esta manera de capturar los paquetes no permite ver como éstos se reciben en tiempo real. Para visualizar el tráfico de paquetes en tiempo real, se selecciona la interfaz Bluetooth en el programa Wireshark. Pero antes, hay que configurar correctamente los permisos de acceso mediante Wireshark a estas interfaces, por lo contrario no va ser posible acceder. La serie de comandos que hay que ejecutar es la siguiente:

```

sudo apt-get install wireshark libcap2-bin
sudo groupadd wireshark
sudo usermod -a -G wireshark $USER
sudo chgrp wireshark /usr/bin/dumpcap
sudo chmod 755 /usr/bin/dumpcap
sudo setcap cap_net_raw,cap_net_admin=eip /usr/bin/dumpcap

```

Se puede parar la captura de paquetes y exportar los datos en varios formatos, además esta forma permite verificar si los campos introducidos (en hexadecimal) en el comando HCI son correctos. Por lo tanto, es como si fuera una interfaz gráfica para los comandos que ejecutamos a través del terminal. A continuación se muestra una imagen de una captura Wireshark seleccionando el paquete de comando HCI explicado anteriormente (HCI_LE_Set_Scan_Enable).

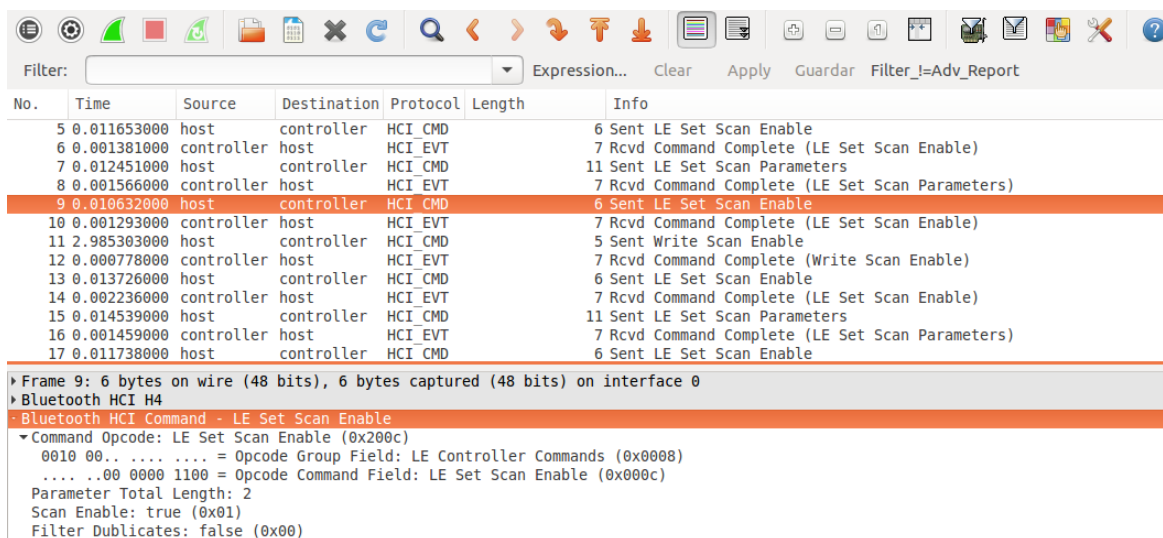


Fig. 3.4 Captura de Wireshark

La tercera manera es la que ofrece el comando **tshark** que permite capturar durante un determinado tiempo elegido por el usuario y guardar el resultado en un fichero .cap. Esta estrategia es muy útil a la hora de querer automatizar este proceso debido al gran número de pruebas que se desean capturar.

Conociendo estas herramientas, en los siguientes capítulos 4 y 5 se va a explicar la implementación de los estados *Scanning* y *Advertising*.

CAPÍTULO 4. IMPLEMENTACIÓN DE PROCEDIMIENTO DE SCANNING

Como ya se ha presentado anteriormente, el estado de *Scanning* es el estado que escucha paquetes de *Advertising*. Existen dos tipos de *Scanning*: *Active Scanning* y *Passive Scanning*. El pasivo solo recibe paquetes de *Advertising*. El activo escucha los paquetes de *Advertising* y cuando detecta uno envía paquetes *Scan Requests* para solicitar información adicional vía *Scan Response*.

Solo se puede pasar desde el estado de *Scanning* al estado *Standby*, esto ocurre cuando el dispositivo deja de escanear. La duración del estado *Standby* depende de los parámetros *Scan Window* y *Scan Interval*. Sabiendo que solamente se escanea durante *Scan Window*, el dispositivo pasa al estado *Standby* durante un tiempo *Scan Interval* – *Scan Window* cada *Scan Interval*.

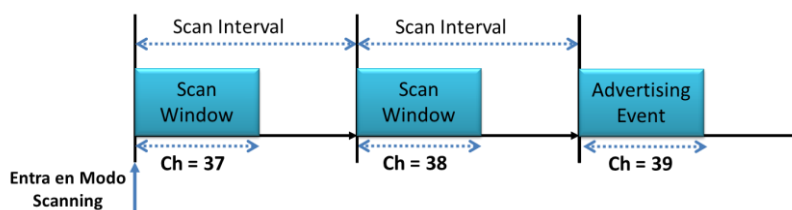


Fig. 4.1 *Scan Interval/Window*

Por lo cual, se puede pensar que si se igualan los parámetros *Scan Window* y *Scan Interval*, no entraría nunca en el estado de *Standby* excepto si se deshabilita el *Scanning*, y así debería escanear de manera continua.

Eso es cierto hasta un cierto punto. En el capítulo 6 se muestran capturas del consumo del escaneo continuo, donde se observa que los dispositivos hacen unas pequeñas pausas (*Standby*) y que la duración de estas pausas depende del fabricante y en algunos casos del tamaño de *Scan Window*.

4.1. *Passive Scanning*

Para configurar los parámetros, según la especificación, se debe utilizar el comando **LE Set Scan Parameters Command**.

Lo que muestra la tabla 4.1 son los campos de los parámetros que constituyen este comando que a su vez caracterizará el *Scanning* del dispositivo BLE.

El primer campo es el *LE_Scan_Type* de un byte, y solo hay dos opciones (**0x00** o **0x01**) que indica si va a ser pasivo o activo consecutivamente.

Tabla 4.1. *LE Set Scan Parameters Command*

Command	OCF	Command parameters	Return Parameters
HCI_LE_Set_Scan_Parameters	0x000B	LE_Scan_Type, LE_Scan_Interval, LE_Scan_Window, Own_Address_Type, Scanning_Filter_Policy	Status

El segundo y el tercero, LE_Scan_Interval y LE_Scan_Window, son campos de dos bytes, son tiempos comprendidos entre 2.5 ms y 10.24 s en pasos de 0.625 ms que después se tienen que convertir a hexadecimal. Por ejemplo, si se quiere poner 10 ms, se convertirá a hexadecimal la división de $10/0.625 = 16$ que sería **0x0010**. A la hora de escribir el comando, este campo se escribirá en formato Little Endian² tal como indica la especificación v4.0 en la sección **DATA AND PARAMETER FORMATS** de [4]. Por lo tanto, los bytes que hay que poner en el campo de *Scan Window* o *Scan Interval* equivalente a 10 ms sería **0x1000**. Es importante saber que el valor *Scan Window* siempre tiene que ser menor o igual que *Scan Interval*.

El campo de Own_Address_Type es de un byte y sirve para indicar si el dispositivo va a tener una dirección pública (por defecto) o aleatoria que sería en hexadecimal **0x00** o **0x01** respectivamente.

El último campo también es de un byte, Scanning_Filter_Policy es para poner alguna política de filtro de paquetes mediante una lista llamada *White List Only*. Si es **0x00** se aceptan todos los paquetes de *Advertising*, si es **0x01** se ignoran todos los paquetes que no pertenecen a la lista *White List Only*. Si *White List Only* no está configurada se ignorarán todos los paquetes que no estén dirigidos al *Scanner*.

Por lo tanto, si se quiere poner el *Scanning* en modo pasivo, con un escaneo continuo de *Scan Window* 10 ms, con una dirección pública y sin ninguna restricción de filtros:

```
sudo hcitool -i hci0 cmd 0x08 0x000B 00 10 00 10 00 00 00
```

Al ejecutar este comando, el terminal devuelve un estado que indica si ha ocurrido algún error. Si es un **0x00** es que todo ha ido bien. Si no es un **0x00** habrá que mirar qué significa en la especificación, concretamente buscar su significado en la lista de errores de código (Anexo 1).

² El formato Little Endian almacena los datos de más de un byte en un ordenador escribiendolos de derecha a izquierda, como el árabe, o el hebreo, frente a los que se escriben de izquierda a derecha, Big Endian.

También, como ya se explicó en el capítulo anterior, los campos de los paquetes se pueden decodificar mediante Wireshark. De esta manera se puede verificar que los parámetros de cada comando están bien puestos y ver las respuestas del *Controller* al *Host*.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	host	controller	HCI CMD	11	Sent LE Set Scan Parameters
2	0.001656000	controller	host	HCI EVT	7	Rcvd Command Complete (LE Set Scan Parameters)

▶ Frame 1: 11 bytes on wire (88 bits), 11 bytes captured (88 bits) on interface 0
 ▶ Bluetooth HCI H4
 ▶ Bluetooth HCI Command - LE Set Scan Parameters
 ▶ Command Opcode: LE Set Scan Parameters (0x200b)
 Parameter Total Length: 7
 Scan Type: Passive (0x00)
 Scan Interval: 16 (10 msec)
 Scan Window: 16 (10 msec)
 Own Address Type: Public Device Address (0x00)
 Scan Filter Policy: Accept all advertisements. Ignore directed advertisements not addressed to this device (0x00)

Fig. 4.2 Valores de los campos de parámetros de *Scan* pasivo

Por ejemplo, en este caso, vemos en el terminal que el *Controller* responde con el estado **0x0C**.

```

ubuntu@ubuntu-EasyNote-TS44HR:~$ sudo hcitool -i hci0 cmd 0x08 0x000B 00 10 00 10 00 00 00
< HCI Command: ogf 0x08, ocf 0x000b, plen 7
00 10 00 10 00 00 00
> HCI Event: 0x0e plen 4
01 0B 20 0C
ubuntu@ubuntu-EasyNote-TS44HR:~$
  
```

Fig. 4.3 Respuesta 0x0C del *Controller* vía terminal

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	host	controller	HCI CMD	11	Sent LE Set Scan Parameters
2	0.001656000	controller	host	HCI EVT	7	Rcvd Command Complete (LE Set Scan Parameters)

▶ Frame 2: 7 bytes on wire (56 bits), 7 bytes captured (56 bits) on interface 0
 ▶ Bluetooth HCI H4
 ▶ Bluetooth HCI Event - Command Complete
 Event Code: Command Complete (0x0e)
 Parameter Total Length: 4
 Number of Allowed Command Packets: 1
 ▶ Command Opcode: LE Set Scan Parameters (0x200b)
 Status: Command Disallowed (0x0c)

Fig. 4.4 Respuesta 0x0C del *Controller* vista desde Wireshark

Este estado de error es muy común, significa *Command Disallowed* y una de sus razones es el estado del dispositivo. Este comando no se acepta por el *Controller* si este ya está en el estado de *Scanning*. Por lo tanto, antes de

configurar los parámetros, hay que deshabilitar *LE Scan* mediante el comando **HCI_LE_Set_Scan_Enable** explicado en el capítulo anterior.

```
Sudo hcitool -i hci0 cmd 0x08 0x000C 00 00
```

Por último, después de poner los parámetros que interesan. Hay que volver a habilitar *LE Scan* y con eso habilitamos también los parámetros puestos.

```
Sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

Entonces, la serie de comandos que hay que ejecutar para un escaneo pasivo y continuo de *Scan Window* de 10 ms sin ninguna política de filtro de paquetes es:

```
sudo hcitool -i hci0 cmd 0x08 0x000C 00 00
sudo hcitool -i hci0 cmd 0x08 0x000B 00 10 00 10 00 00 00
sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

Muchas veces, escribir líneas de comandos en el terminal se hace tedioso, sobre todo si hay que repetir los mismos comandos muchas veces o si hay que ejecutar muchas líneas. Una solución a este problema es preparar scripts para cada configuración que interesa probar. En el Anexo 2 se muestran los scripts que se utilizan en este trabajo, el primero es el de esta sección (Habilitar un *Scan* pasivo).

4.2. *Active Scanning*

Para la implementación de *Scanning* activo, hay que seguir los mismos pasos explicados en *Scanning* pasivo cambiando solamente el parámetro **LE_Scan_Type** del comando **HCI_LE_Set_Scan_Parameters** a **0x01**.

Por lo tanto la secuencia de comandos que hay que ejecutar para una configuración de *Scanning* activo es:

```
sudo hcitool -i hci0 cmd 0x08 0x000C 00 00
sudo hcitool -i hci0 cmd 0x08 0x000B 01 10 00 10 00 00 00
sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

Script en Anexo 2 (Habilitar un *Scan* activo).

CAPÍTULO 5. IMPLEMENTACIÓN DE PROCEDIMIENTOS DE *BROADCASTING*

Tal como se ha explicado anteriormente, un dispositivo BLE puede anunciarse mediante paquetes de *Advertising*. Ahora bien, para que se considere como un dispositivo de difusión (*Broadcasting*) debe incluir datos útiles en los paquetes de *Advertising*. Para ello, se pueden utilizar tres de los cuatro tipos de eventos de *Advertising*: *Connectable Undirected Advertising*, *Nonconnectable Advertising* y *Discoverable Advertising*.

Para el caso del evento *Connectable Undirected Advertising*, cualquier dispositivo en modo *Scanning* puede iniciar una conexión una vez detectado el *Advertising*. Por lo tanto, este tipo de eventos va a ser descartado en este trabajo y se centrará solamente en *Nonconnectable Advertising* y *Discoverable Advertising*.

Los datos de difusión pueden ser recibidos por cualquier dispositivo en modo *Scanning* indiferentemente si es de tipo activo o pasivo. Frente a esta ventaja se encuentra la inexistencia del reconocimiento (*Acknowledgment*) de los paquetes enviados. Por lo tanto, el *Advertiser* no puede saber si los datos enviados han sido recibidos por los *Scanners* que están a su alrededor.

5.1. *Nonconnectable Advertising*

En modo difusión, los datos son etiquetados en los paquetes de *Advertising*. Esto es así porque no todos los dispositivos pueden entender todos los posibles datos emitidos.

Los tipos de datos dependen del perfil de la aplicación, por ejemplo, en el de la salud con tipos de datos como “*Glucose Meter*” o “*Personal Emergency Response Sensor*”... etc. Dado que cada vez se incorporan nuevas aplicaciones, se deben definir nuevos tipos de datos. Bluetooth asigna una numeración de tipos de datos en su página web³ para mantener a sus clientes actualizados. Sin embargo, existen tipos genéricos que están definidos en la propia especificación [4] concretamente en **APPENDIX C (NORMATIVE): EIR AND AD FORMATS**.

Tal como se observa en la figura, el campo de datos es de 32 bytes, que se puede dividir en varios bloques con estructura “longitud: tipo: datos”. La longitud es para identificar cuando empieza el siguiente dato, el tipo es la etiqueta que identifica la clase de datos y finalmente los datos.

³ <https://www.bluetooth.com/specifications/assigned-numbers/Health-Device-Profile>

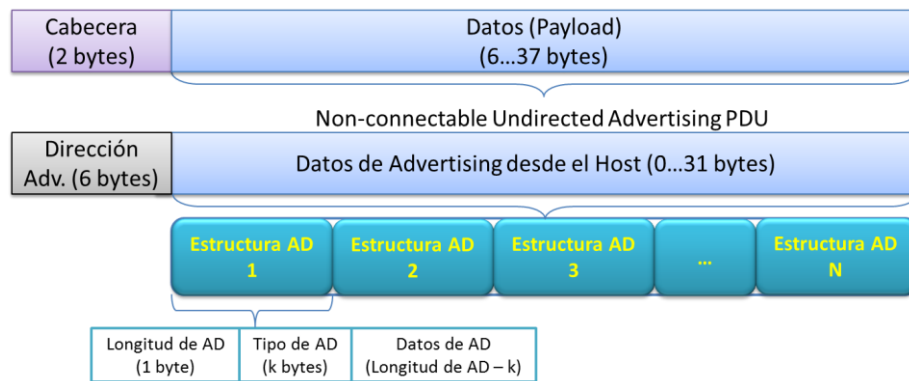


Fig. 5.1 *Nonconnectable Undirected Advertising PDU*

Una de las condiciones que exige el modo *Broadcasting* es desactivar dos opciones del tipo de datos Flags: “**LE General Discoverable Mode**” y “**LE Limited Discoverable Mode**”. La distribución de bits de este tipo de datos es:

Tabla 5.1. *Flags*

Value	Description	Bit	Information
0x01	Flags	0	LE Limited Discoverable Mode
		1	LE General Discoverable Mode
		2	BR/EDR Not Supported (i.e. bit 37 of LMP ExtendedFeature bits Page 0)
		3	Simultaneous LE and BR/EDR to Same Device Capable (Controller) (i.e. bit 49 of LMP Extended Feature bits Page 0)
		4	Simultaneous LE and BR/EDR to Same Device Capable (Host) (i.e. bit 66 of LMP Extended Feature bits Page 1)
		5..7	Reserved

Por lo tanto, siguiendo la estructura “longitud: tipo: datos”, si se quiere activar solamente la última opción sería “**0x02 0x01 0x08**”.

Otro ejemplo de tipo datos es *LOCAL NAME*:

Tabla 5.2. *Local Name*

Value	Description	Information
0x08	Local Name	Shortened local name
0x09	Local Name	Complete local name

Por ejemplo, si se quiere poner la palabra “STOP” que son 4 caracteres, convertidos en Hexadecimal serían “**0x73 0x74 0x6F 0x70**”. Se puede elegir uno de los dos valores del tipo **0x08** o **0x09**, en definitiva la longitud sería 5 bytes. Siguiendo la estructura “longitud: tipo: datos” el resultado sería “**0x05 0x09 0x73 0x74 0x6F 0x70**”.

Para poner estos datos, se utiliza el comando **LE Set Advertising Data Command**:

Tabla 5.3. LE Set Advertising Data Command

Command	OCF	Command parameters	Return Parameters
HCI_LE_Set_Advertising_Data	0x0008	Advertising_Data_Length, Advertising_Data	Status

El parámetro Advertising_Data_Length indica el total de los bytes significativos que se quieren insertar. El parámetro Advertising_Data es la serie de datos “longitud: tipo: datos” explicada anteriormente.

Por último, hay que saber que los bytes sobrantes de los 32 dedicados para los datos, se tienen que rellenar como ceros, aunque no se enviarán por el medio radio (solo los bytes significativos son enviados en paquetes de *Advertising*).

A continuación se muestra cómo quedaría el comando para insertar las dos estructuras de datos previamente explicadas (“**0x05 0x09 0x73 0x74 0x6F 0x70**” y “**0x02 0x01 0x08**”):

```
sudo hcitool -i hci0 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02 01 08 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Para configurar parámetros de *Advertising*, se usa el comando **LE Set Advertising Parameters Command**.

Tabla 5.4. *LE Set Advertising Parameters Command*

Command	OCF	Command parameters	Return Parameters
HCI_LE_Set_Advertising_Parameters	0x0006	Advertising_Interval_Min, Advertising_Interval_Max, Advertising_Type, Own_Address_Type, Direct_Address_Type, Direct_Address, Advertising_Channel_Map, Advertising_Filter_Policy	Status

- **Advertising_Interval_Min, Advertising_Interval_Max:** son campos de dos bytes que están para poner el intervalo de *Advertising* fijo (Capítulo 2.1). Según la especificación, el intervalo de *Advertising* para los casos *Nonconnectable Advertising* y *Discoverable Advertising* tiene que ser superior o igual a 100 ms. El valor mínimo y el máximo no tienen por qué coincidir, para dejar un margen de elección al *Controller* dependiendo de sus otras actividades.
- **Advertising_Type:** es un campo de un byte, para los tipos *Nonconnectable Advertising* y *Discoverable Advertising* es **0x03** y **0x02** respectivamente.
- **Own_Address_Type, Direct_Address_Type:** son campos de un byte, para poner estas direcciones como públicas (que es lo que viene por defecto), en estos campos se pone **0x00**.
- **Direct_Address:** es un campo de 6 bytes, se usa para especificar la dirección del destino en los tipos de *Advertising* dirigidos, en los casos de *Nonconnectable Advertising* y *Discoverable Advertising* se ignora y se pone todo a ceros.
- **Advertising_Channel_Map:** es un campo de un byte, se usa para la elección de los canales de *Advertising*. En el caso de querer enviar los paquetes de *Advertising* en los tres canales de *Advertising* se pone **0x07**.
- **Advertising_Filter_Policy:** es un campo de un byte, indica la política de filtro de paquetes. Para permitir *Scan Request* y *Connect Request* de cualquier dispositivo, este campo se pone a **0x00**.

Por lo tanto, el comando para forzar un intervalo de *Advertising* de 100 ms (**0xA0 0x00** en hexadecimal y en formato Little Endian) y de tipo *Nonconnectable Advertising* (**0x03**):

```
sudo hcitool -i hci0 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00 00 00 00 00 00 07 00
```

Por último, para habilitar el *Advertising* con los parámetros puestos previamente, se utiliza el comando **LE Set Advertise Enable Command** con OCF **0x000A** y que contiene un único parámetro para habilitar o deshabilitar el *Advertising* (**0x01** o **0x00** consecutivamente):

```
sudo hcitool -i hci0 cmd 0x08 0x000A 0x01
```

La secuencia de comandos para la implementación de un *Advertising* de tipo *Nonconnectable* con intervalo de *Advertising* de 100 ms e incluyendo datos en los paquetes es:

```
sudo hcitool -i hci0 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00 00 00 00 00 00 00 00 07 00
sudo hcitool -i hci0 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02 01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
sudo hcitool -i hci0 cmd 0x08 0x000A 0x01
```

Script en Anexo 2 (Habilitar *Advertising* tipo *Nonconnectable*).

Poniendo otro dispositivo en modo *Scanning*, independientemente si es activo o pasivo, se puede verificar mediante una captura de Wireshark los paquetes de *Advertising* que recibe y observar los campos que se han introducido.

57	0.104898000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)
58	0.106130000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)
59	0.100934000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)
60	0.099965000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)
61	0.108033000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)
62	0.104989000	controller host	HCI EVT	23 Rcvd LE Meta (LE Advertising Report)

Bluetooth HCI Event - LE Meta	
Event Code:	LE Meta (0x3e)
Parameter Total Length:	21
Sub Event:	LE Advertising Report (0x02)
Num Reports:	1
Event Type:	Non-Connectable Undirected Advertising (0x03)
Peer Address Type:	Public Device Address (0x00)
BD_ADDR:	Cc&CTech_6c:9b:f9 (5c:f3:70:6c:9b:f9)
Data Length:	9
▼ Advertising Data	
▼ Device Name:	stop
Length:	5
Type:	Device Name (0x09)
Device Name:	stop
▼ Flags	
Length:	2
Type:	Flags (0x01)
000.	= Reserved: 0x00
...0 ...	= Simultaneous LE and BR/EDR to Same Device Capable (Host): false (0x00)
.... 1...	= Simultaneous LE and BR/EDR to Same Device Capable (Controller): true (0x01)
.... .0..	= BR/EDR Not Supported: false (0x00)
.... ..0.	= LE General Discoverable Mode: false (0x00)
.... ...0	= LE Limited Discoverable Mode: false (0x00)
RSSI (dB):	-45

Fig. 5.2 Valores de campos de *Nonconnectable Advertising Channel* PDU y tiempo transcurrido entre dos paquetes consecutivos

En esta captura, además de ver los campos como el tipo de *Advertising* y los datos, se puede observar que los paquetes llegan con un intervalo de *Advertising* de 100 ms aproximadamente.

5.2. Discoverable Advertising

Para implementar este tipo de *Advertising*, se sigue la misma secuencia de comandos explicada en la sección anterior, cambiando solamente el campo **Advertising_Type** del comando **LE Set Advertising Parameters Command** a **0x02**.

Además, se tiene que saber que en este tipo de eventos, el *Advertiser* responde a los *Scan Requests* recibidos de los dispositivos *Scanner* en modo activo con paquetes *Scan Response*. Estos paquetes también contienen campos de información y tienen la misma estructura explicada anteriormente para los paquetes de *Advertising*.

Para poner los datos de *Scan Response* se utiliza el comando **LE Set Scan Response Data Command** cuyo OCF es **0x0009** y que está formado por los mismos campos de **LE Set Advertising Data Command**.

De manera que, para insertar los mismos datos del ejemplo que se han introducidos en la sección anterior (“**0x05 0x09 0x73 0x74 0x6F 0x70**” y “**0x02 0x01 0x08**”) sin olvidar el relleno, se debe ejecutar el siguiente comando:

```
sudo hcitool -i hci0 cmd 0x08 0x0009 09 05 09 73 74 6F 70 02 01 08 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

La secuencia de comandos⁴ para la implementación de este tipo de *Advertising* quedaría como la siguiente:

```
sudo hcitool -i hci0 cmd 0x08 0x0009 09 05 09 73 74 6F 70 02 01 08 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
sudo hcitool -i hci0 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02 01 08 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
sudo hcitool -i hci0 cmd 0x08 0x0006 A0 00 A0 00 02 00 00 00 00 00 00
00 00 07 00
sudo hcitool -i hci0 cmd 0x08 0x000A 0x01
```

Script en Anexo 2 (Habilitar *Advertising* tipo *Discoverable*).

⁴ El orden de los comandos no importa siempre que se deje el comando de habilitación como último paso.

CAPÍTULO 6. MEDIDAS SOBRE LOS PROCEDIMIENTOS DE *ADVERTISING*

En este capítulo se presentan medidas del tiempo de descubrimiento para diferentes configuraciones.

Los dispositivos BLE utilizados son de diferentes fabricantes cuya información se presenta en el Anexo 3.

El ciclo de trabajo representa el tiempo de escaneo de un dispositivo BLE durante un tiempo dado de *Scan Interval*. Se define como la proporción de tiempo donde el dispositivo BLE está escaneando:

$$\rho = \frac{\tau_{SW}}{\tau_{SI}} \quad (6.1)$$

6.1. Influencia de *Scan Interval* con *Advertising Interval* de 100 ms

La primera prueba que se ha realizado consiste en cambiar valores de *Scan Interval* entre 10 ms y 1 s respecto a *Scan Window* fijo de 10 ms y por parte del *Advertiser*, un *Advertising* de tipo *Nonconnectable* de intervalo de 100 ms. En esta prueba el dispositivo BLE Trust se puso como *Scanner* y el dispositivo Belkin como *Advertiser*.

Por lo tanto, el ciclo de trabajo de la primera configuración medida en esta prueba es 1 (100% del tiempo escaneando) ya que es un escaneo continuo (*Scan Interval* = *Scan Window* = 10 ms), y a partir de ahí se va ir reduciendo.

$$0 < \rho < 1 \quad (6.2)$$

Para una implementación correcta del *Scanning* en modo LE, primero se tiene que deshabilitar el *Inquiry Scan* y *Page Scan*, procedimientos utilizados para el descubrimiento e inicio de conexión con dispositivos Bluetooth clásicos, que generalmente vienen habilitados por defecto. Para ello, la familia de comandos **hciconfig** ofrece el comando **noscan**.

```
Sudo hciconfig -a hci0 noscan
```

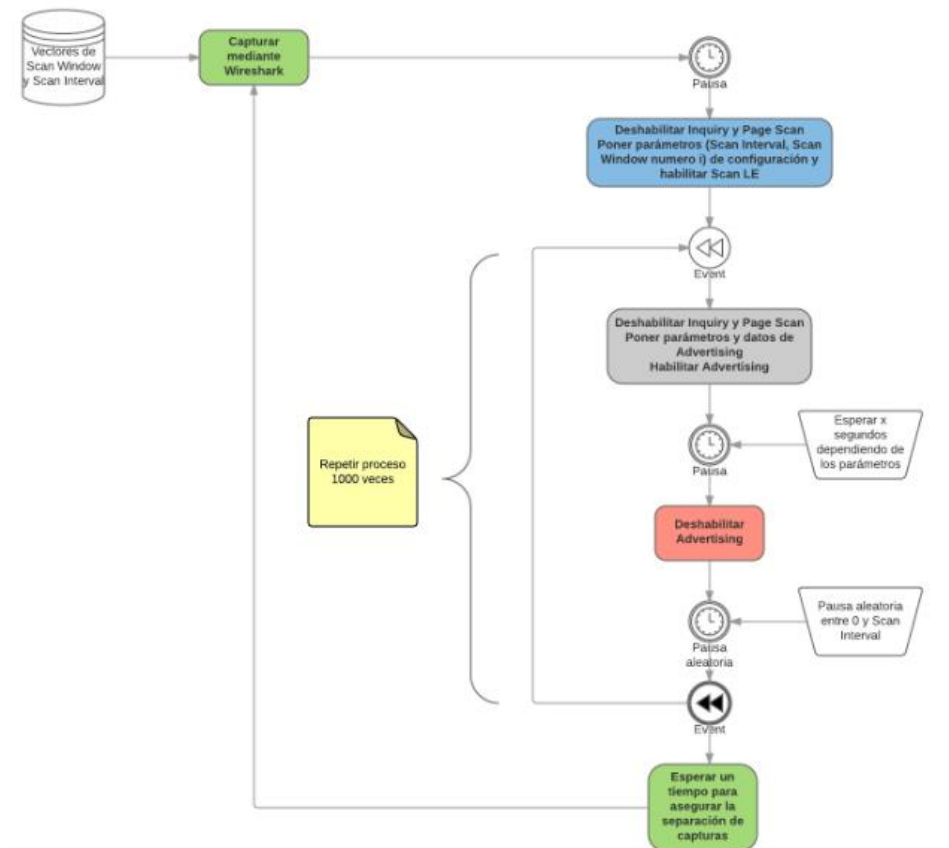


Fig. 6.1 Algoritmo para obtener las medidas

Para cada configuración de *Scan Interval* (10, 50, 100, 150, 250, 500, 1000 ms) se habilita y deshabilita el *Advertising* mil veces con el fin de obtener mil muestras de tiempos de descubrimiento.

La duración del estado de *Advertising* (tiempo entre habilitar y deshabilitar *Advertising*) tiene que ser coherente con la configuración planteada. Por ejemplo, para *Scan Interval* de 10 ms que sería un *Scan* continuo que corresponde al máximo ciclo de trabajo, poniendo unos segundos de *Advertising* sería suficiente para obtener las muestras correctamente. No obstante, a medida que vaya aumentando el *Scan Interval* (el ciclo de trabajo disminuye), la duración del estado de *Advertising* tiene que ser más grande, por lo contrario se perderán muestras.

También se tiene que dejar una pausa aleatoria entre los eventos de *Advertising* comprendida entre 0 y *Scan Interval*, para asegurar desincronizaciones independientes entre *Advertiser* y *Scanner*.

En el Anexo A2.5 se muestra cómo quedaría un script con propósitos iguales pero para configuraciones diferentes.

A continuación se muestra la distribución del tiempo de descubrimiento para las dos primeras configuraciones (*Scan Interval* = 10 ms y *Scan Interval* = 50 ms):

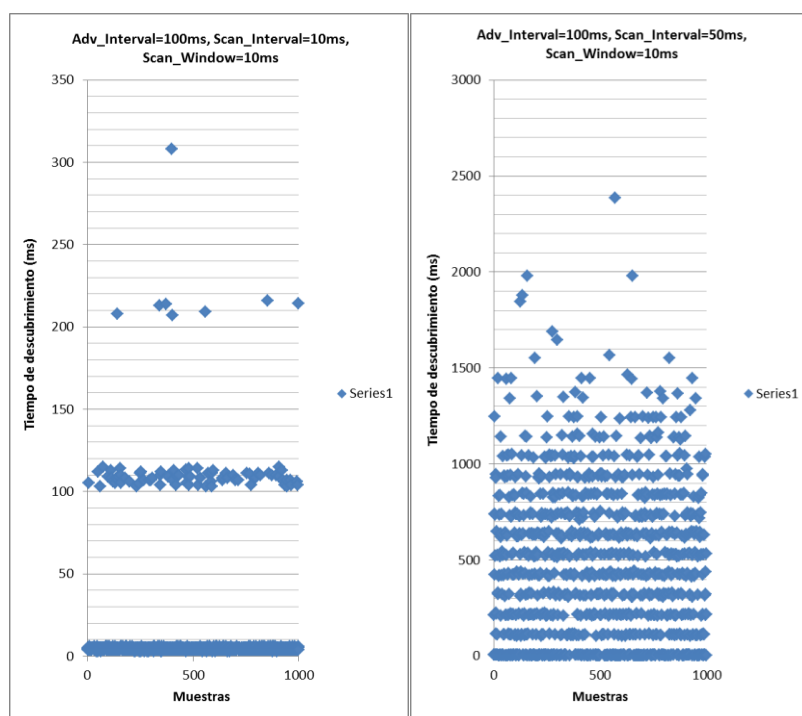


Fig. 6.2 Distribución del tiempo de descubrimiento con *Advertising Interval* de 100 ms: *Scan* continuo de 10 ms (Izquierda), *Scan Interval* 50 ms y *Scan Window* 10 ms (Derecha)

En las dos distribuciones (**Fig. 6.2**) se observan claramente nubes de puntos cada intervalo de *Advertising* que en este caso es de 100 ms. En el caso de *Scan Interval* de 50 ms que implica un ciclo de trabajo de 0.2, se observa que hay tiempos de descubrimiento superiores a 2 s.

En el caso de *Scan* continuo se observan cuatro nubes de puntos separados por 100 ms aproximadamente. Idealmente, el tiempo de descubrimiento debería ser siempre casi inmediato (unidades de milisegundos) ya que el *Scanner* no para de buscar paquetes de *Advertising*. La aparición de valores mayores a unos pocos milisegundos, implica que hay paquetes que no se están detectando. Esto puede ser debido a interferencias o a que el dispositivo que escanea realice pausas para cambiar de frecuencias.

Para comprobar el funcionamiento del *Scan* continuo se han realizado capturas con el osciloscopio. En la figura (**Fig. 6.3**), se observa que el *Scan* continuo del dispositivo Trust no es totalmente continuo, sino que se pueden apreciar pausas entre cada *Scan Window/Interval*.

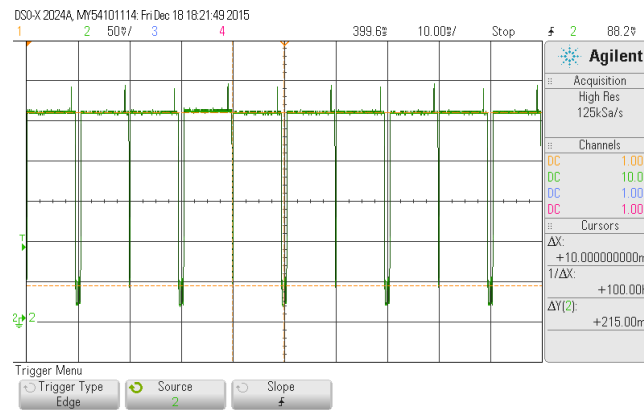


Fig. 6.3 Comportamiento *Scan* continuo de Trust

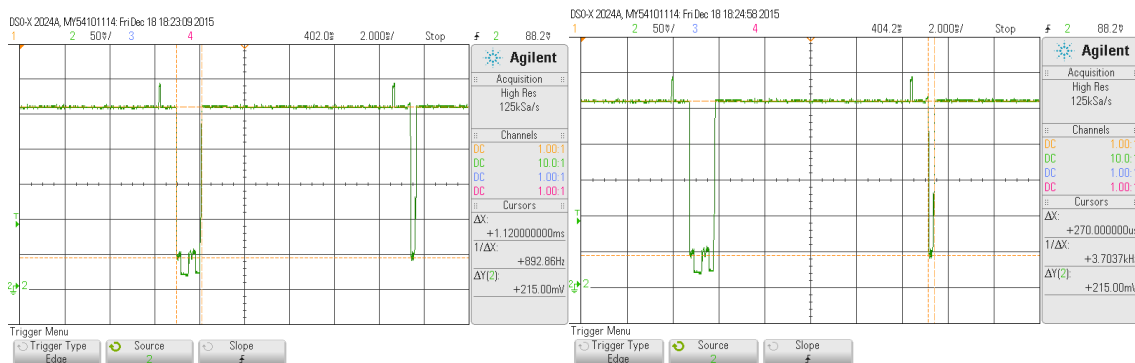


Fig. 6.4 Duración de pausas *Scan* continuo de Trust

Se observan dos pausas de duración diferente (**Fig. 6.4**), una de 1.12 ms y otra de 270 μ s que se alternan cada *Scan Window/Interval*. Para ver si estas pausas podrían afectar a la detección de los paquetes de *Advertising*, se ha medido los tiempos de transmisión de los paquetes de *Advertising* enviados por el dispositivo Belkin y el resultado es el siguiente:

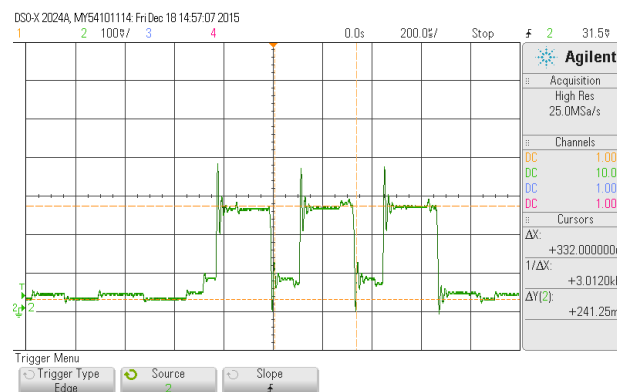


Fig. 6.5 Duración pulsos de *Advertising* de Belkin

Se observa que el tiempo que dura la transmisión de un paquete de *Advertising* es de 250 μ s, y el tiempo total de los tres paquetes enviados en los tres canales de *Advertising* es de 900 μ s aproximadamente (**Fig. 6.5**).

En consecuencia, estas pausas podrían explicar las medidas de tiempos de descubrimiento superiores a 100 ms ya que existe la posibilidad de coincidencia de hasta tres paquetes de *Advertising* en las pausas de *Scan* continuo del dispositivo Trust previamente descritas.

A continuación se presenta una ilustración que muestra cómo afectan estas pausas de *Scan* continuo en la detección de los *Advertising*.

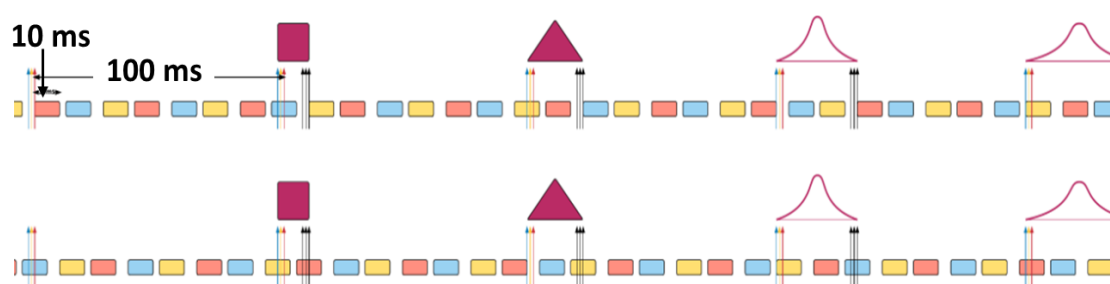


Fig. 6.6 Efecto de las pausas de Scan continuo en la detección de los Advertising

En la figura (**Fig. 6.6**) se muestran dos casos, el primero es del caso cuando no se detectan los primeros paquetes de *Advertising*, y el segundo es cuando coincide el escaneo con los paquetes de *Advertising* por lo tanto se detecta en pocos milisegundos. En el Anexo 4, se muestran las figuras que ilustran el efecto de las pausas y el ciclo de trabajo con mejor resolución.

Los colores azul, amarillo y rojo representan los tres canales de *Advertising* (Ch37, Ch38 y Ch39). Las flechas representan paquetes de *Advertising* enviados en los tres canales y los rectángulos los instantes de escaneo.

Las flechas negras representan los eventos de *Advertising* añadiendo el máximo *random* (10 ms) respecto al periodo fijo de tiempo de *Advertising*. Las primeras tres flechas negras están a una distancia de 10 ms y la función dibujada justo encima es una función de probabilidad uniforme entre 0 y 10 ms. El segundo grupo de tres flechas negras está a una distancia de 20 ms y la función dibujada justo encima es una función de probabilidad triangular que es la suma de dos variables aleatorias uniformes (la convolución). El tercer grupo de flechas está a una distancia de 30 ms y la función de probabilidad es la convolución de una uniforme y la triangular. Y así sucesivamente.

Por lo tanto, se puede decir que existe la posibilidad de que no se detecten los paquetes de *Advertising* de forma inmediata con un *Scan* continuo, sin embargo es poco probable que no se detecte en ninguno de los tres siguientes periodos de *Advertising*.

Por otro lado para una configuración de ciclo de trabajo de 0.2 se puede extender el tiempo de descubrimiento hasta dos segundos tal como se ha visto en la figura (Fig. 6.2). Esta configuración se ilustra en la siguiente figura (Fig. 6.7) que representa un posible caso donde se ve la nula o poca probabilidad de detección en los cinco primeros periodos de *Advertising*. (Ver Anexo 4).



Fig. 6.7 Posible caso de *Scanning* al 20% con *Advertising* de intervalo 100 ms

Se puede intuir que a medida que se vaya aumentando el valor *Scan Interval*, aparecerán tiempos de descubrimiento más grandes, por lo tanto, aumenta el promedio del tiempo de descubrimiento. A continuación se muestra el resultado del tiempo de descubrimiento máximo, mínimo y promedio de las medidas del resto de configuraciones:

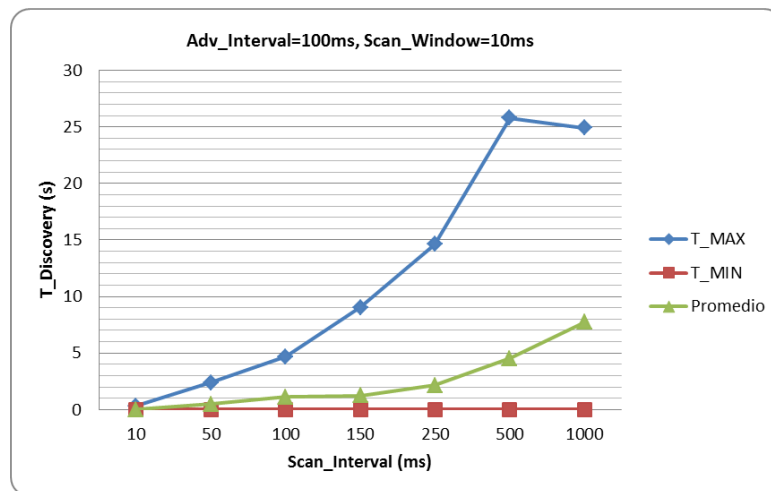


Fig. 6.8 Evolución del tiempo de descubrimiento en función de *Scan Interval* con *Advertising Interval* de 100 ms

En la gráfica se observa cómo evoluciona el tiempo de descubrimiento respecto a las configuraciones de *Scan Interval*. Respecto a la última configuración se nota claramente que los datos no son coherentes a dicha evolución, eso pasa porque en la captura se pierden muchas muestras (muestras cuyos tiempos de descubrimiento son superiores a 25 s) debido al ciclo de trabajo tan bajo en el cual está operando el *Scanner* (es de 10 ms/1000 ms = 0.01). La razón por la cual no se detectan paquetes con tiempos de descubrimiento superiores a 25 s es, el tiempo que se deja emitiendo el *Advertiser* puesto en el código implementado.

6.2. Influencia de *Scan Interval* con *Advertising Interval* de 200 ms

El siguiente experimento que se ha realizado consiste en repetir la misma prueba anterior pero configurando el dispositivo Belkin a un intervalo de *Advertising* mayor, de 200 ms.

A continuación se muestra las distribuciones de tiempo de descubrimiento para dos configuraciones (*Scan Interval* = 10 ms y *Scan Interval* = 100 ms):

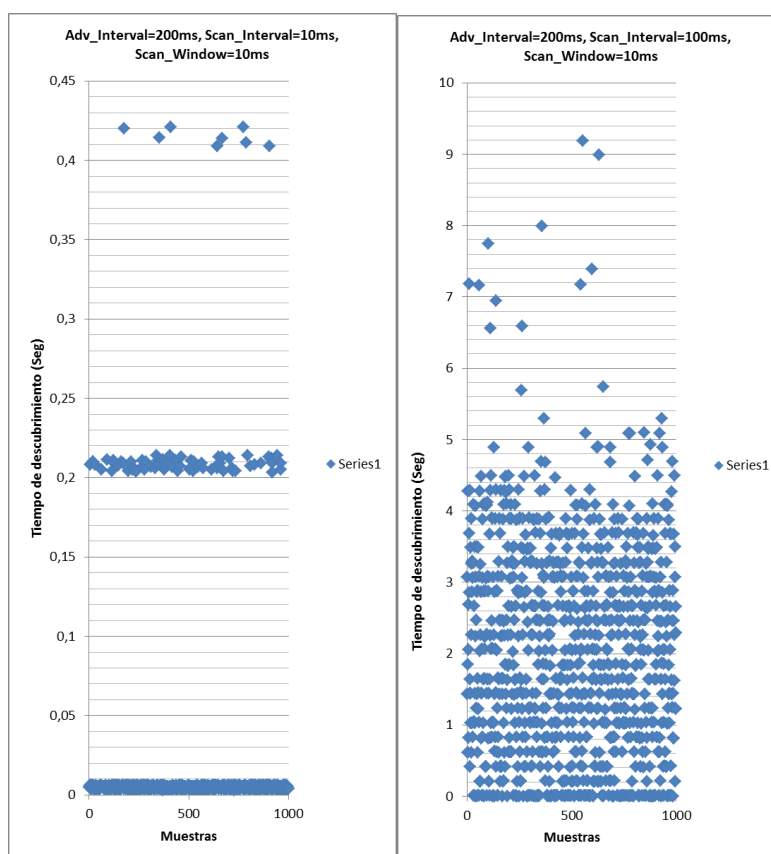


Fig. 6.9 Distribuciones de tiempo de descubrimiento con *Advertising Interval* de 200 ms: *Scan* continuo de 10 ms (Izquierda), *Scan Interval* 100 ms y *Scan Window* 10 ms (Derecha)

Se observa que las nubes de puntos en este caso se repiten cada 200 ms aproximadamente que es el valor del intervalo de *Advertising*. El tiempo máximo de descubrimiento en el caso de *Scan Interval* de 100 ms supera los 9 s debido a su bajo ciclo de trabajo ($10 \text{ ms}/100 \text{ ms} = 0.1$). Sin embargo, si comparamos el mismo ciclo de trabajo con el intervalo de *Advertising* de 100 ms, experimento anterior, donde su tiempo de descubrimiento es de 4.68 s, se puede llegar a la conclusión de que el intervalo de *Advertising* también afecta al tiempo máximo de descubrimiento y por lo tanto a su promedio.

A continuación se presenta el resultado de las medidas de todas las configuraciones:

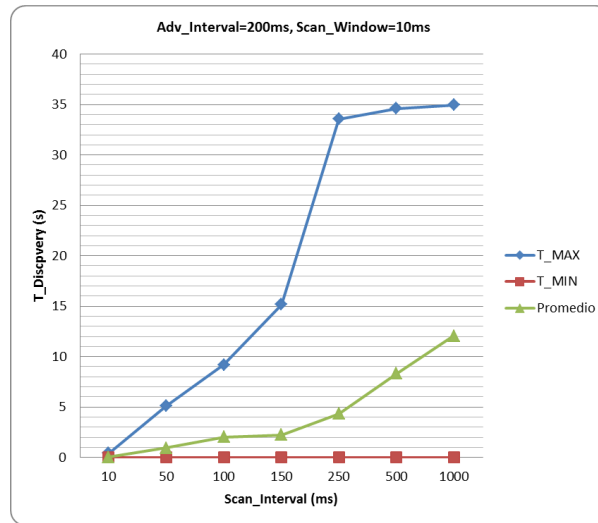


Fig. 6.10 Evolución del tiempo de descubrimiento en función de *Scan Interval* con *Advertising Interval* de 200 ms

Se percibe el aumento esperado del tiempo máximo y promedio de descubrimiento que es inversamente proporcional al ciclo de trabajo. Se destaca también el aumento notable en todas las configuraciones para un intervalo de *Advertising* de 200 ms respecto al de 100 ms.

En este caso se ha aumentado el tiempo de espera (tiempo de *Advertising*) a 35 s para cada muestra, aun así, se observa que en la última configuración se han perdido muestras, es decir, que ha habido tiempos de descubrimiento mayores a 35 s debido al bajo ciclo de trabajo.

6.3. Influencia del ciclo de trabajo

Para ver mejor el efecto del ciclo de trabajo, se han realizado medidas de veinte configuraciones con dos porcentajes de escaneo diferentes.

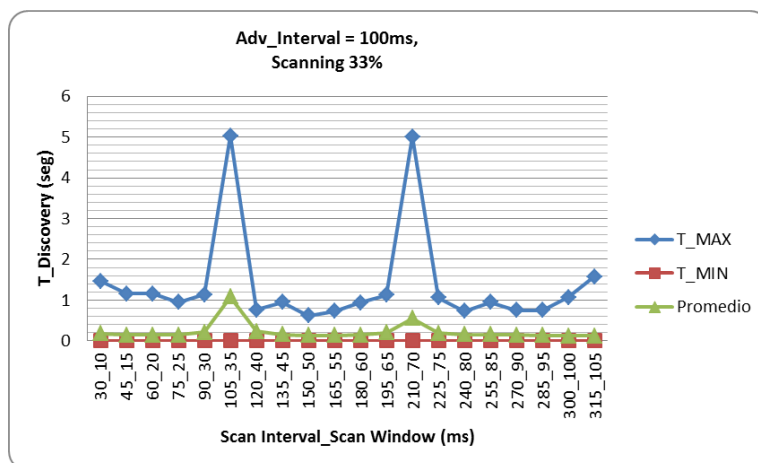


Fig. 6.11 Evolución del tiempo de descubrimiento en función de la combinación *Scan Interval-Window* de 33%

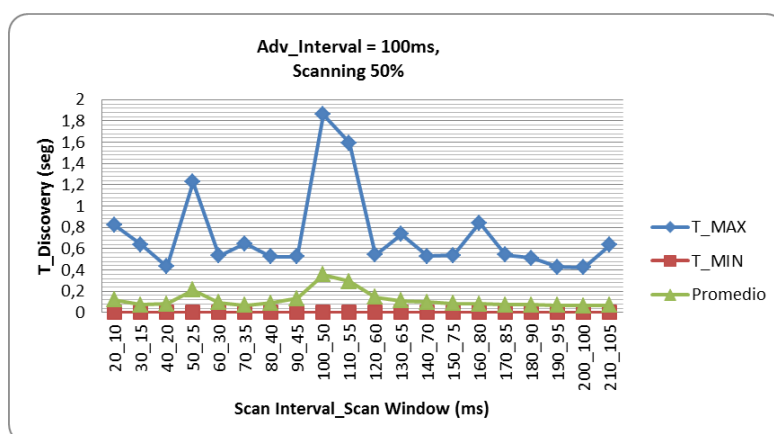


Fig. 6.12 Evolución del tiempo de descubrimiento en función de la combinación *Scan Interval-Window* de 50%

En este experimento, se observa claramente que a mayor ciclo de trabajo se reduce el tiempo de descubrimiento. También se puede añadir que existen combinaciones de *Scan Interval-Scan Window* donde se dispara el tiempo máximo de descubrimiento, como los casos 100-50 y 110-55 del *Scanning* 50% y los casos 105-35 y 210-70 del *Scanning* 33%.

6.4. Influencia del *chipset*

Respecto al *Scan* continuo, debido a los resultados obtenidos, se han realizado pruebas que confirman el efecto de las pausas entre *Scan Intervals* consecutivos sobre el tiempo de descubrimiento.

Definimos delta del *Scan* continuo como la duración de las pausas que aparecen en el *Scan* continuo cada *Scan Window/Interval*. Se ha medido la evolución de delta cambiando parámetros del *Scan* continuo para diferentes fabricantes. Se observa que a partir de intervalos de 25 ms, delta aumenta en los dispositivos Sena y Trust. Mientras que en el dispositivo Belkin permanece constante para todas las configuraciones.

En el caso del dispositivo Trust, y para el caso concreto de *Scan Window/Interval* de 10 ms, el valor de delta se alterna entre 1.12 ms y 270 μ s.

Si ampliamos la captura de la tensión durante el proceso de escaneo de cada dispositivo, para escaneos superiores a 10 ms, aparecen otras pausas. Se han obtenido las siguientes capturas:

En el caso del dispositivo Sena:



Fig. 6.13 Comportamiento de *Scan* continuo del dispositivo Sena

Se observan pequeñas pausas de 470 μ s aproximadamente, que se repiten cada 17 ms a partir de cada delta.

En el caso del dispositivo Trust:

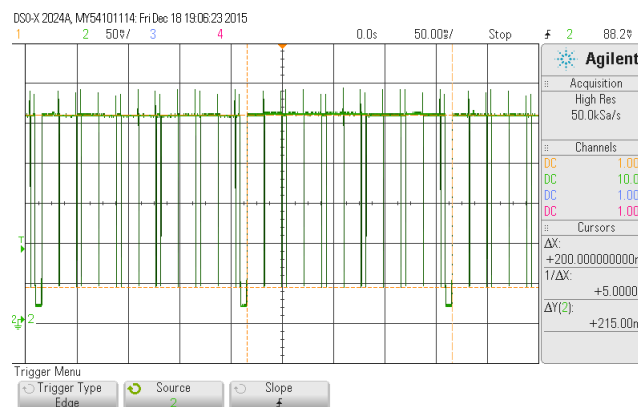


Fig. 6.14 Comportamiento de *Scan* continuo del dispositivo Trust

Se observan pequeñas pausas de 270 μ s, aproximadamente, que siguen un patrón de separación entre sí de 17/17/4.6 ms a partir de cada delta.

Y por último, en el caso del dispositivo Belkin y para todas las configuraciones de escaneo continuo no se encontró ninguna pausa dentro de las ventanas de escaneo. Como se ha visto anteriormente, este dispositivo tampoco tiene un delta variable en función de la configuración del *Scan* continuo.

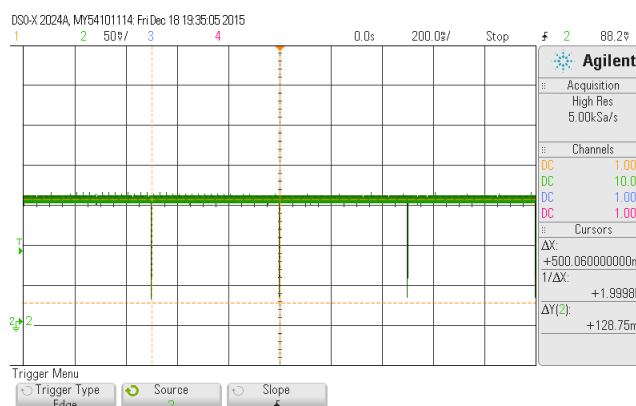


Fig. 6.15 Comportamiento de *Scan* continuo del dispositivo Belkin

Finalmente, se han realizado pruebas de configuraciones diferentes de *Scanning* continuo, para los tres dispositivos, enviando paquetes de *Advertising* con el mismo dispositivo *Advertiser* Sena, con intervalo de *Advertising* de 100 ms.

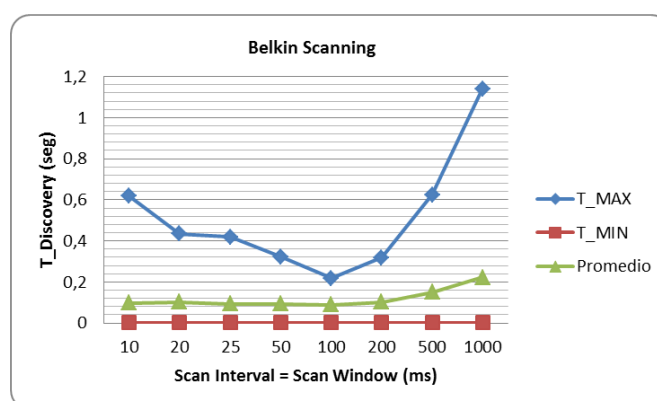


Fig. 6.16 Evolución del tiempo de descubrimiento con configuraciones *Scan* continuo diferentes del dispositivo Belkin

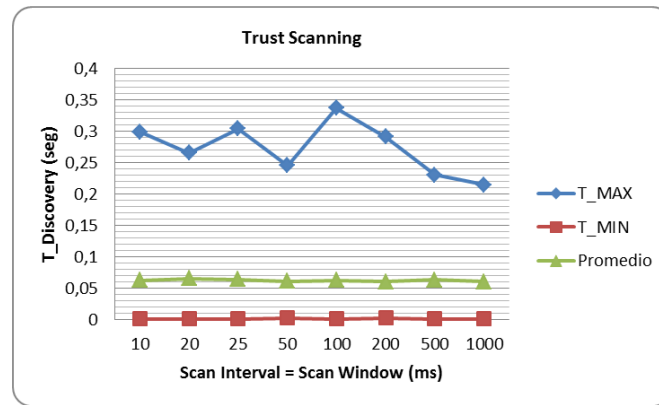


Fig. 6.17 Evolución del tiempo de descubrimiento con configuraciones *Scan* continuo diferentes del dispositivo Trust

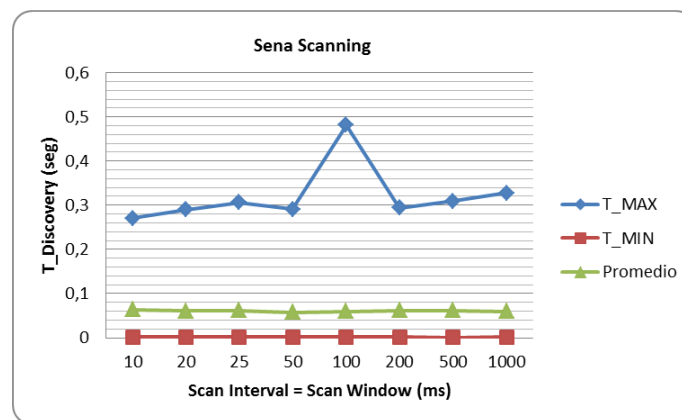


Fig. 6.18 Evolución del tiempo de descubrimiento con configuraciones *Scan* continuo diferentes del dispositivo Sena

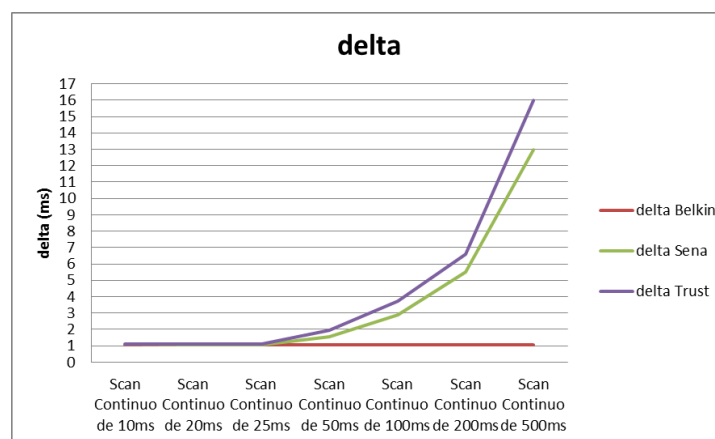


Fig. 6.19 Evolución delta vs *Scan Interval/Window* en *Scan* continuo de diferentes chipsets

CAPÍTULO 7. CARACTERIZACIÓN DE ENERGÍA

Respecto a las medidas de corriente realizadas en este trabajo, se ha utilizado el diseño de un sensor de corriente de Texas Instrument [5] para obtener mejor precisión, sobre todo cuando se trata de pulsos de microsegundos, como el caso de las pausas mostradas en el capítulo anterior o los pulsos de *Advertising*. A continuación se presenta la función V_{OUT} en función de I_{IN} teórica de este sensor:

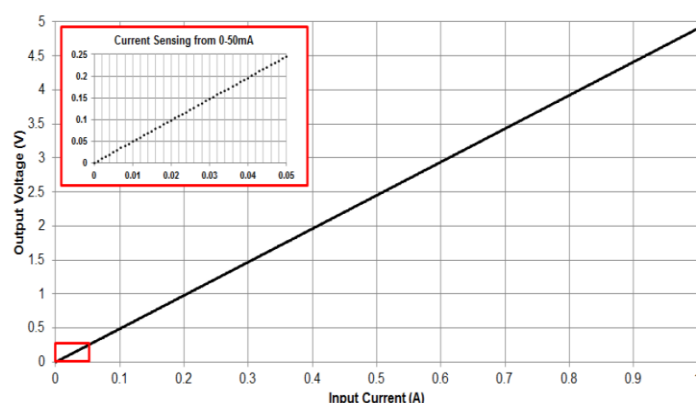


Fig. 7.1 Función transformada V_{OUT} vs I_{IN} del sensor de corriente utilizado

7.1. Consumo de *Scanning* al 50% de diferentes fabricantes

Primero se presentan las medidas de corriente de dispositivos de fabricantes diferentes en modo *Scanning* de 50%, probando dos configuraciones diferentes, *Scan Window* de 10 ms y 500 ms.

En el caso del dispositivo Belkin:

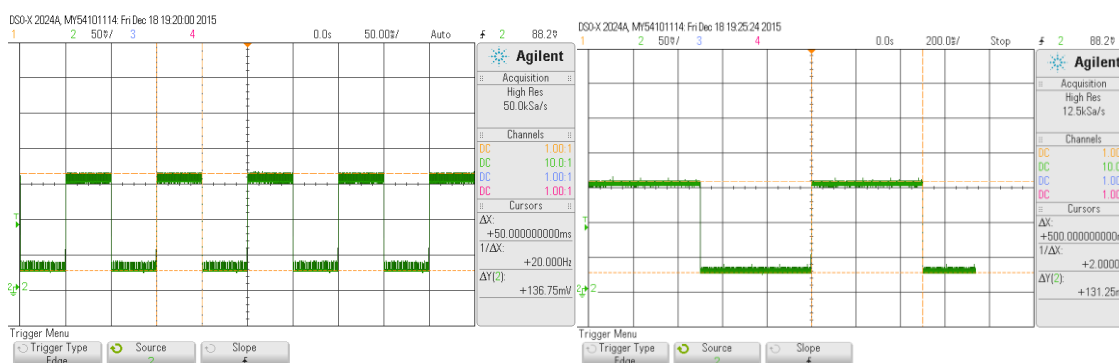


Fig. 7.2 Consumo de *Scanning* al 50% del dispositivo Belkin (Izquierda *Scan Window* de 50 ms y *Scan Interval* 100 ms, derecha 500 ms y 1000 ms consecutivamente)

Se observa que, para las dos configuraciones, se obtienen los mismos niveles de tensión (mínimo = 22 mV, máximo = 150 mV) y, por lo tanto, los mismos valores de corriente: $I_{\min} = 3.5$ mA, $I_{\max} = 30$ mA.

Para el caso del dispositivo Sena:

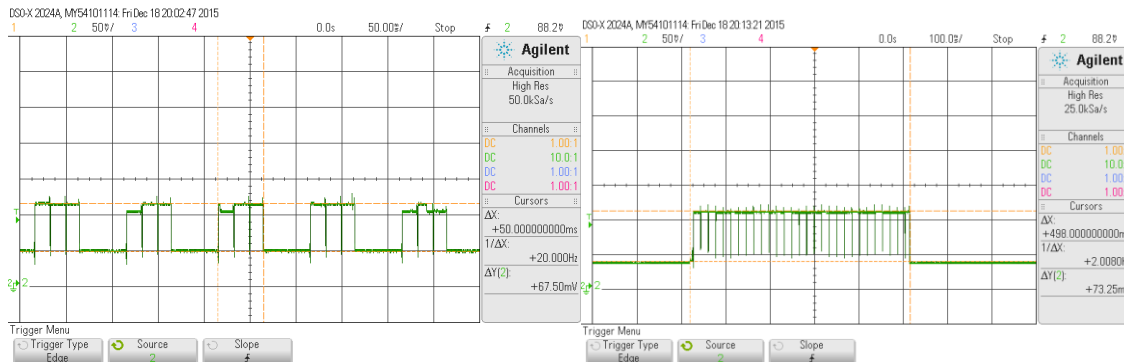


Fig. 7.3 Consumo *Scanning* al 50% del dispositivo Sena (Izquierda *Scan Window* de 50 ms y *Scan Interval* 100 ms, derecha 500 ms y 1000 ms consecutivamente)

Se observa que la configuración de *Scan Interval* = 100 ms (*Scan Window* = 50 ms) presenta un nivel de tensión mínimo ligeramente superior a la configuración *Scan Interval* = 1 s (*Scan Window* = 500 ms), mientras que el nivel de tensión máximo es el mismo en las dos. $I_{\min} = 6$ mA para la captura derecha, $I_{\min} = 8$ mA para la captura izquierda. La tensión máxima es de 121 mV en las dos configuraciones, lo que equivale a 24 mA según la figura (Fig. 7.1).

Para el caso del dispositivo Trust:

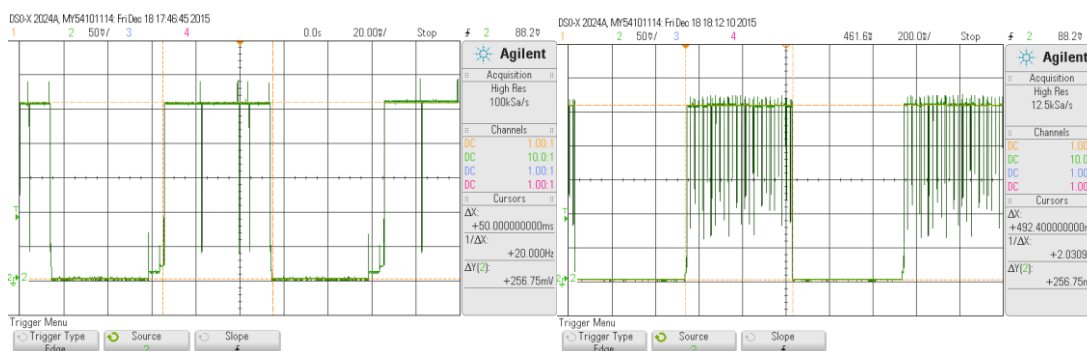


Fig. 7.4 Consumo *Scanning* al 50% del dispositivo Trust (Izquierda *Scan Window* de 50 ms y *Scan Interval* 100 ms, derecha 500 ms y 1000 ms consecutivamente)

Para este dispositivo, se ve que la tensión mínima es prácticamente nula, mientras que la máxima es de 256 mV, que corresponde a 56 mA y que se mantiene igual en las dos configuraciones.

7.2. Consumo de pulsos de *Advertising* de diferentes chipsets

También se han medido los niveles de tensión de los pulsos generados en modo *Advertising*, configurando los mismos parámetros de *Advertising* en los tres dispositivos:

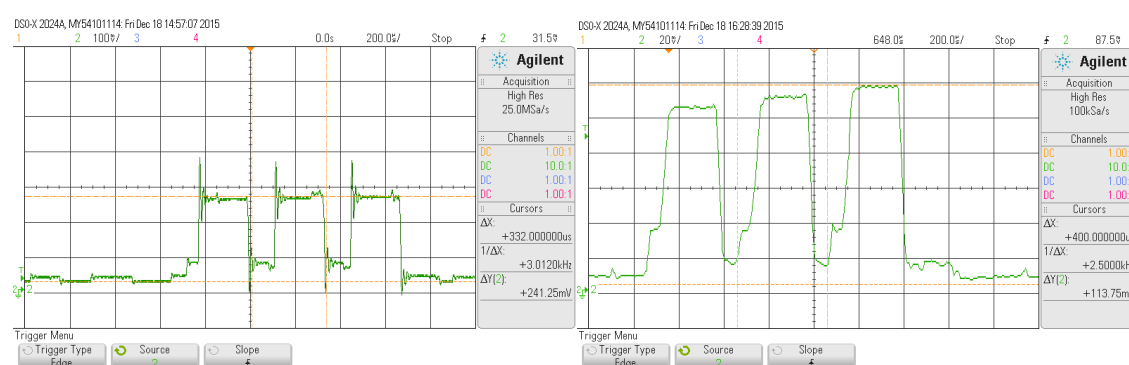


Fig. 7.5 Consumo de los pulsos de *Advertising* del dispositivo: Derecha Belkin, Izquierda Sena

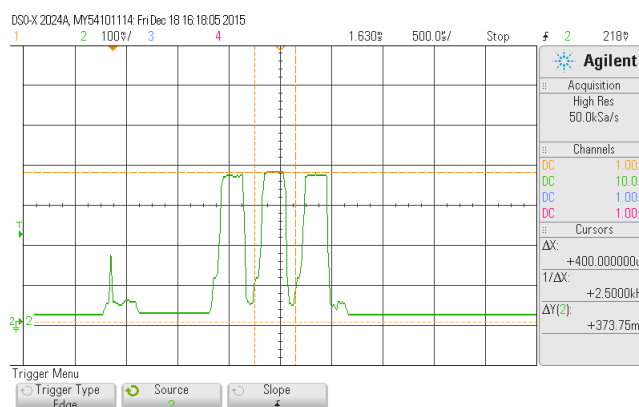


Fig. 7.6 Consumo de los pulsos de *Advertising* del dispositivo Trust

En este modo (*Advertising*), para el dispositivo Belkin se observa un nivel de tensión prácticamente nulo cuando no se envían los paquetes de *Advertising*. Sin embargo, la máxima tensión es de 260 mV, que corresponde a unos 56.5 mA aproximadamente.

En el caso del dispositivo Sena, se mide una tensión mínima de 29 mV y una máxima de 120 mV que son aproximadamente las mismas medidas obtenidas en las configuraciones de escaneo de 50% (6 mA y 24 mA consecutivamente).

Para el caso del dispositivo Trust, se mide una tensión mínima de 22 mV que equivale a unos 3.5 mA de corriente. Sin embargo, la máxima medida de tensión de los pulsos de *Advertising* es muy alta respecto a la de los otros dispositivos, 357 mV que corresponde a unos 70 mA, aproximadamente.

Por otro lado, hay que destacar que los dispositivos que se alimentan a través de puertos USB consumen más energía que las placas de desarrollo BLE, como por ejemplo BLE Nano (Anexo A3.4).

A continuación se muestra la captura de niveles de tensión de los pulsos de *Advertising* generados por BLE Nano:

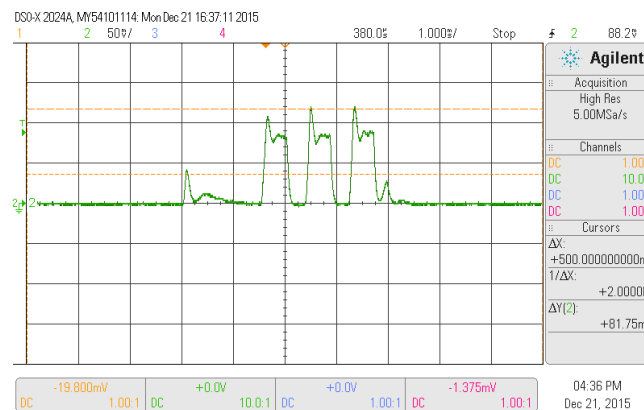


Fig. 7.7 Consumo de los pulsos de *Advertising* del dispositivo BLE Nano

Para este dispositivo, se observa que el nivel mínimo de tensión es nulo, comportamiento parecido al que se había encontrado en las medidas realizadas para el dispositivo Belkin. Sin embargo, en este caso la tensión máxima no supera los 120 mV, lo que equivale según la gráfica de la función del sensor de corriente a unos 23 mA aproximadamente.

7.3. Consumo de *Scan* continuo de diferentes chipsets

En lo que se refiere al *Scanning* continuo, se han realizado medidas de siete configuraciones diferentes para los tres dispositivos USB y se han obtenido los resultados de las figuras (**Fig. 7.8** y **Fig 7.9**).

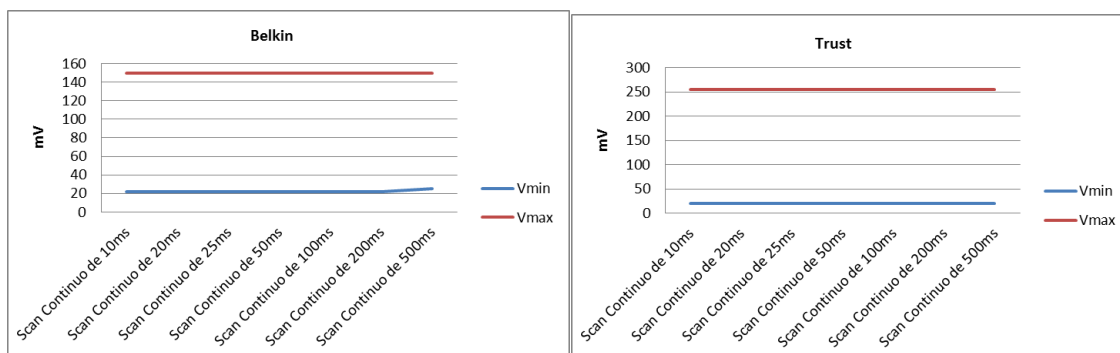


Fig. 7.8 Consumo modo *Scan* continuo del dispositivo: Izquierda Belkin, derecha Trust

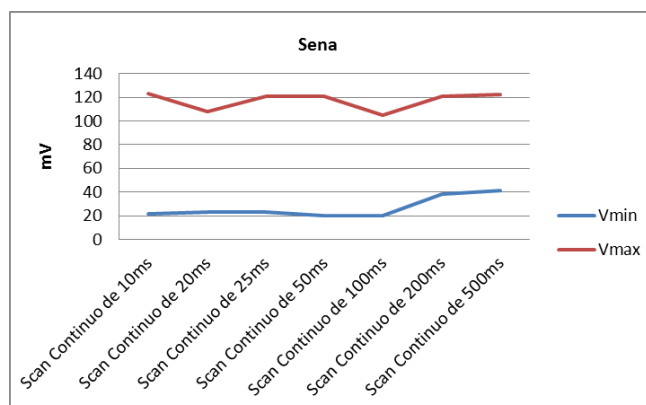


Fig. 7.9 Consumo modo *Scan* continuo del dispositivo Sena

Los niveles de tensión se mantienen constantes para todas las configuraciones excepto algunas ligeras desviaciones en el caso del dispositivo Sena. La máxima tensión del dispositivo Trust es la mayor de los tres, y es de unos 255 mV que equivale a 55 mA, mientras que para los dispositivos Sena y Belkin, su máxima tensión es de 120 mV y 150 mV respectivamente (24 mA y 30 mA).

Respecto a los niveles mínimos de tensión, el dispositivo Trust marca la menor de los tres con 20 mV que equivale a 3 mA. En el dispositivo Belkin se miden 22 mV que equivale a una corriente de 3.5 mA. La mínima tensión que se observa en el dispositivo Sena es la misma que la del dispositivo Belkin aproximadamente, excepto en las configuraciones de *Scan Interval* de 200 ms y 500 ms que aumenta a 40 mV y que implica unos 8 mA de corriente.

CAPÍTULO 8. PROPUESTAS PARA EL RECONOCIMIENTO DE SEÑALES DE TRÁFICO

En este capítulo se presentan varios ejemplos de escenarios donde se podría utilizar el sistema Bluetooth LE como alternativa al sistema de detección de señales de tráfico basado en cámaras. También se introducen nuevos métodos de aviso de posibles imprevistos en las carreteras como podrían ser accidentes, obras, congestiones o auxilio de averías...

A continuación se muestran cuatro situaciones genéricas de las funcionalidades que podría adoptar este sistema.

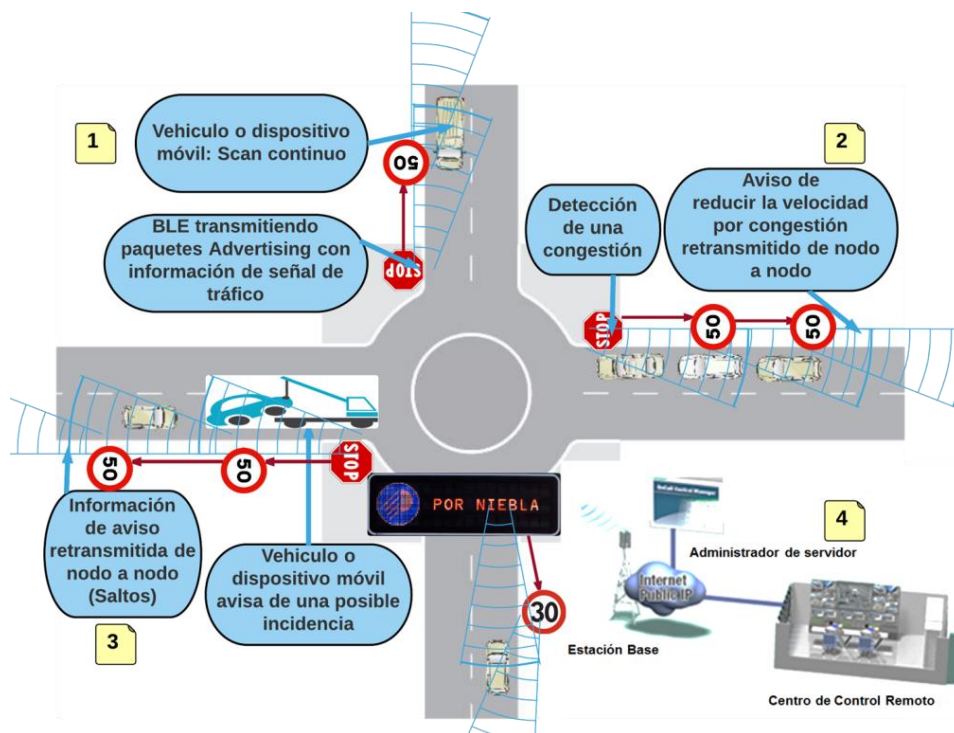


Fig. 8.1 Croquis de posibles escenarios de señales de tráfico

El primero es el más simple y permite a los vehículos detectar la existencia de una señal de tráfico fija. Esta última debe tener instalado un dispositivo BLE en modo *Broadcaster* que envía paquetes de *Advertising* que contienen como datos: código de la señal de tráfico, identificador de la carretera y su sentido y las coordenadas de la propia señal de tráfico. El *Advertising Interval* se configura dependiendo del contenido de la señal. Si es una señal que limita la velocidad por ejemplo, el *Advertising Interval* debe ser menor cuanto mayor sea el límite de velocidad. A cada dispositivo BLE en modo *Broadcaster* se le instala una antena directiva para obtener un mayor alcance y controlar la propagación de la señal hacia los vehículos interesados en la información de dicha señal. Respecto al alcance, se han realizado pruebas, previas a este TFG, que demuestran que la señal de *Advertising* emitida desde una antena directiva

puede ser detectada a distancias superiores a 1 km. La regla general para colocar las señales de tráfico es: tienen que ser visibles a 100 metros de donde se colocan, ya que es la distancia que recorre un vehículo a 90 km/h desde el momento en que el conductor se da cuenta de la señal y reacciona para frenar. Por lo cual, el alcance de BLE junto a una antena directiva cumpliría de sobra este requisito. En el hipotético caso de captar paquetes de una señal de tráfico de otra carretera o del otro sentido de la misma carretera, el *Scanning* del dispositivo móvil o del vehículo debe ser capaz de descartar dicha información gracias al identificador de la carretera y su sentido incluidos en el mismo paquete de *Advertising*.

La segunda situación muestra un caso de congestión de tráfico. El primer dispositivo que detecta esta situación difunde la información necesaria junto a un aviso para reducir la velocidad. Los siguientes dispositivos BLE (señales de tráfico) que reciben dicho aviso lo retransmiten con un límite de saltos (por ejemplo tres saltos). De esta manera, los vehículos reciben la información con antelación. Para implementar esta funcionalidad, los dispositivos BLE instalados en las señales de tráfico deben estar escaneando y enviando paquetes de *Advertising* simultáneamente. Las pruebas realizadas confirman la viabilidad de esta configuración.

La tercera situación es la de un caso de avería, como podría ser también un accidente, obras o una grúa de auxilio. En este caso, es el propio vehículo averiado o la grúa quien transmite los paquetes de *Advertising* con la información de aviso de cada situación. Por lo tanto, la aplicación Android o IOS del dispositivo móvil, que podría estar también integrada en el propio vehículo, debe tener la opción de transmitir información, incluso seleccionar qué tipo de aviso quiere transmitir, avería, obras, accidente..., en los paquetes de *Advertising*. Estos avisos también se pueden difundir mediante un método de saltos retransmitiendo la información de señal a señal de tráfico para avisar de posibles imprevistos y reducir la velocidad. En casos de avería y accidentes, esta información podría llegar hasta un centro de auxilio para el rescate.

La cuarta situación es parecida a la primera, solo que esta vez no se trata de una señal de tráfico fija, sino de una variable. El centro de control remoto puede variar el contenido de la señal LED dependiendo de la situación del tráfico o condiciones meteorológicas. Existen varias formas implementadas para la realización de esta tarea, mediante redes de fibra óptica o enlaces radio y servidores. Sobre las infraestructuras ya instaladas para el cambio de las señales de tráfico variables, se podría implementar código para actualizar la información en los paquetes de *Advertising*, con el fin de difundir el contenido correcto de la señal de tráfico variable cada vez que se modifique. En este caso también se puede utilizar una estrategia de retransmisión entre señales de tráfico, con el fin de avisar a los vehículos con antelación del cambio realizado.

En lo que se refiere a la información que se puede enviar en los paquetes de *Advertising*, los ejemplos de datos propuestos sumarían unos 24 bytes como máximo. Teniendo en cuenta que los paquetes de *Advertising* pueden incluir hasta 32 bytes de datos, la suma propuesta encajaría bien. Además sobran 8 bytes que se rellenarían con ceros (no se transmiten), así se respeta el compromiso entre la cantidad de información y el tiempo de transmisión. EL

SIG Bluetooth se encarga de asignar los números de tipos de datos dependiendo de la aplicación. Por lo tanto, las tablas que se muestran a continuación son simplemente propuestas para ilustrar una posible distribución de los bytes de datos que se envían en los paquetes de *Advertising*.

Tabla 8.1. Ejemplo de asignación de números para los tipos de datos propuestos

Valor	Descripción	Información
0x77	Señal de tráfico	Descripción de la señal de tráfico
0x88	Vía	Sentido + código o Id de la carretera
0x66	Ubicación	Coordenadas del origen del mensaje

Tabla 8.2. Ejemplo de distribución de bits para el tipo de dato “Estado”

Valor	Descripción	Bit	Información
0x99	Estado	0	Accidente
		1	Vehículo averiado
		2	Grúa de auxilio
		3	Congestión
		4	Obras
		5..7	Reservados para futuras implementaciones

Un ejemplo de uso para los bits que han quedado sin especificar sería definir un tiempo *Timeout*, que indicaría durante cuánto tiempo debe estar activo el estado indicado. En este caso como sobran 3 bits tendríamos 8 posibles tiempos, que se pueden interpretar como pasos de unidades de tiempo (por ejemplo pasos de un minuto: de 1 minuto a 8).

El tipo de datos “Ubicación” tiene dedicados 8 bytes siguiendo el formato utilizado por el protocolo Trimble Estándar Interface⁵. Cuatro bytes para la latitud puestos en radianes con signo, el positivo significa el norte y el negativo el sur. Para la longitud otros 4 bytes y también en radianes con signo, el signo positivo equivale al este y el negativo al oeste [6].

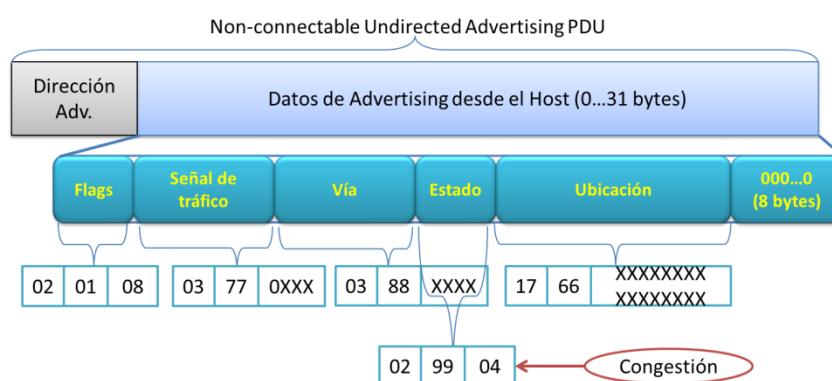
⁵ <ftp://ftp.trimble.com/pub/sct/timing/palisade/pal-appa.pdf>

Tabla 8.3. Ejemplo de distribución de bytes y descripción de los tipos de datos propuestos

Nombre	Longitud (bytes)	Descripción
Vía	2	Sentido e identificador de la vía, calle o carretera...
Señal de tráfico	2	Contenido de la señal de tráfico 0x0000 – 0x01FF daría lugar a 512 códigos. 0x02FF – 0xFFFF reservados para futuras implementaciones
Estado	1	Situación temporal del tráfico
Ubicación	8	Coordenadas de la señal de tráfico

Existen otros formatos de otros protocolos como el NMEA que necesitan más bytes para su utilización, y que no encajarían en el paquete de *Advertising*. Si es necesaria la utilización de este tipo de protocolos, la solución es utilizar otro tipo de evento de *Advertising* para difundir los datos, que es el *Discoverable Advertising*. Con este tipo de eventos, como ya se ha visto en el capítulo 5, se puede enviar información añadida a parte de la que se envía en los paquetes de *Advertising* a través de los paquetes *Scan Response*. Estos paquetes siguen la misma estructura de datos y se configuran de la misma manera.

Por último, la figura (**Fig. 8.2**) muestra cómo quedaría la estructura de los datos propuestos en el campo de datos del paquete de *Advertising*

**Fig. 8.2** Ejemplo estructuras de tipos de datos propuesto.

CAPÍTULO 9. CONCLUSIÓN E IMPACTO SOCIAL

El objetivo de este trabajo es estudiar la viabilidad de un sistema basado en BLE para la detección de señales de tráfico. Para ello, después de presentar la tecnología y sus características, se han analizado los siguientes aspectos:

- Los procedimientos de *Scanning* y *Advertising*, donde se explica su funcionamiento y los parámetros que se deben tener en cuenta para una correcta configuración. Analizando los diferentes tipos de *Advertising* y los modos *Scanning*, se han seleccionado los tipos adecuados para la implementación de un sistema *Broadcaster-Scanner*.
- La configuración del *Scanner* y del *Advertiser*, donde se ha visto la facilidad y la flexibilidad que presenta BLE para su configuración en los modos *Scanner* y *Advertiser*.
- Medidas de tiempo de descubrimiento, donde se han probado varias configuraciones y el efecto que tienen algunos parámetros en este tiempo fundamental. También el efecto de la forma del escaneo, que se ha mostrado que depende del *chipset*.
- Medidas de energía, donde se ha confirmado la característica de bajo consumo de esta tecnología y además la influencia de la configuración de los parámetros y el *chipset*.

A partir del análisis de los resultados obtenidos se puede concluir que BLE es una tecnología válida para su uso en automoción. Concretamente, como sistema de detección de señales de tráfico. Teniendo en cuenta los resultados de las pruebas y las medidas realizadas, se han propuesto cuatro escenarios de ejemplo de esta tecnología para la detección de señales de tráfico y su configuración adaptativa a las condiciones de la vía en tiempo real.

Dicho lo cual, desde el punto de vista económico nos encontramos ante una tecnología de muy bajo consumo y extendida en muchos dispositivos de uso cotidiano, los cuales se pueden utilizar directamente para esta propuesta instalando una sencilla aplicación habilitada para BLE. Como contrapartida se tiene que desplegar una infraestructura de dispositivos BLE en modo *Advertiser/Scanner* en todas las señales de tráfico de todas las carreteras, calles, vías,... etc, lo que implica una gran inversión económica.

Respecto al impacto sobre el medio ambiente, la característica principal de esta tecnología es su bajo consumo de energía, una característica que permite implementar un sistema de alimentación con paneles fotovoltaicos. Esto implica una reducción de la producción de baterías (material contaminante). También se puede considerar como un ahorro de energía para otras necesidades.

Finalmente remarcar que hoy en día, siguen muriendo muchas personas en las carreteras, entre otros motivos, por la falta de visibilidad de las señales, distracción o por la falta de adaptabilidad de las señales de tráfico. Este trabajo propone un nuevo mecanismo que podría salvar la vida de personas mejorando los sistemas de alerta e información.

BIBLIOGRAFÍA

- [1] ADAC, COLABORACIÓN RACC en *SISTEMAS DE RECONOCIMIENTO DE SEÑALES DE TRÁFICO EN TURISMOS*
- [2] Group, Bluetooth Special Interest. *Bluetooth Core Specification v4.0*
- [3] Group, Bluetooth Special Interest. *BLUETOOTH SPECIFICATION Version 4.0 [Vol 6], Physical Layer*
- [4] Group, Bluetooth Special Interest. *BLUETOOTH SPECIFICATION Version 4.0 [Vol 2], Core System Package*
- [5] Instrument, Texas. *0-1A, Single-Supply, Low-Side, Current Sensing Solution.*
- [6] Trimble Standard Interface Protocol, A.14.41. *Palisade NTP Synchronization Kit User Guide.*
 - Kevin Townsend, Carles Cufí, Akiva and Robert Davidson. *Getting Started with Bluetooth Low Energy.*
 - Heydon., R. *Bluetooth Low Energy. The Developer's Handbook.*

ANEXOS

Anexo 1: List Of Error Codes

Error Code Name	0x00 Success
0x01 Unknown HCI Command	0x02 Unknown Connection Identifier
0x03 Hardware Failure	0x04 Page Timeout
0x05 Authentication Failure	0x06 PIN or Key Missing
0x07 Memory Capacity Exceeded	0x08 Connection Timeout
0x09 Connection Limit Exceeded	0x0A Synchronous Connection Limit To A Device Exceeded
0x0B ACL Connection Already Exists	0x0C Command Disallowed
0x0D Connection Rejected due to Limited Resources	0x0E Connection Rejected Due To Security Reasons
0x0F Connection Rejected due to Unacceptable BD_ADDR	0x10 Connection Accept Timeout Exceeded
0x11 Unsupported Feature or Parameter Value	0x12 Invalid HCI Command Parameters
0x13 Remote User Terminated Connection	0x14 Remote Device Terminated Connection due to Low Resources
0x15 Remote Device Terminated Connection due to Power Off	0x16 Connection Terminated By Local Host
0x17 Repeated Attempts	0x18 Pairing Not Allowed
0x19 Unknown LMP PDU	0x1A Unsupported Remote Feature / Unsupported LMP Feature
0x1B SCO Offset Rejected	
0x1C SCO Interval Rejected	0x1D SCO Air Mode Rejected
0x1E Invalid LMP Parameters	0x1F Unspecified Error
0x20 Unsupported LMP Parameter Value	0x21 Role Change Not Allowed
0x22 LMP Response Timeout / LL Response Timeout	0x23 LMP Error Transaction Collision
0x24 LMP PDU Not Allowed	0x25 Encryption Mode Not Acceptable
0x26 Link Key cannot be Changed	0x27 Requested QoS Not Supported
0x28 Instant Passed	0x29 Pairing With Unit Key Not Supported
0x2A Different Transaction Collision	0x2B Reserved
0x2C QoS Unacceptable Parameter	0x2D QoS Rejected
0x2E Channel Classification Not Supported	0x2F Insufficient Security
0x30 Parameter Out Of Mandatory Range	0x31 Reserved
0x32 Role Switch Pending	0x33 Reserved
0x34 Reserved Slot Violation	0x35 Role Switch Failed
0x36 Extended Inquiry Response Too Large	0x37 Secure Simple Pairing Not Supported By Host.

0x38 Host Busy - Pairing	0x39 Connection Rejected due to No Suitable Channel Found
0x3A Controller Busy	0x3B Unacceptable Connection Interval
0x3C Directed Advertising Timeout	0x3D Connection Terminated due to MIC Failure
0x3E Connection Failed to be Established	0x3F MAC Connection Failed

Anexo 2: Scripts

A2.1: Habilitar Scan Pasivo

```
#!/bin/bash

#Deshabilitamos el Scan para poder poner los parámetros que nos interesan
sudo hcitool -i hci0 cmd 0x08 0x000C 00 00

#Despues ponemos los parámetros: 00 (pasivo= sin paquetes Scan_req),
LE_Scan_Interval 0xFFFF, LE_Scan_Window 0xFFFF, ...
sudo hcitool -i hci0 cmd 0x08 0x000B 00 80 00 40 00 00 00

#Por último habilitamos el scanning con los parámetros puestos
sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

A2.2: Habilitar Scan Activo

```
#!/bin/bash

#Deshabilitamos el Scan para poder poner los parámetros que nos interesan
sudo hcitool -i hci0 cmd 0x08 0x000C 00 00

#Despues ponemos los parámetros: 01 (activo= con paquetes Scan_req),
LE_Scan_Interval 0xFFFF, LE_Scan_Window 0xFFFF, ...
sudo hcitool -i hci0 cmd 0x08 0x000B 01 80 00 40 00 00 00

#Por último habilitamos el scanning con los parámetros puestos
sudo hcitool -i hci0 cmd 0x08 0x000C 01 00
```

A2.3: Habilitar Advertising tipo Nonconnectable

```
#!/bin/bash
```

```
#Primero ponemos los parámetros (OGF=0x08 OCF=0x0006
Adv_Interval_Min=0xFFFF Adv_Interval_Max=0xFFFF
Adv_Type=0x03...)

sudo hcitool -i hci1 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00
00 00 00 00 00 07 00

#Despues ponemos los datos (OGF=0x08 OCF=0x0008 Name=09 "STOP=73
74 6F 70" Flags=01 y el relleno en ceros para completar los 32B)

sudo hcitool -i hci1 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02
01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00

#Habilitamos los Advertising con los parámetros previamente
puestos

sudo hcitool -i hci1 cmd 0x08 0x000A 0x01
```

A2.4: Habilitar Advertising tipo Discoverable

```
#!/bin/bash

#Primero ponemos los parámetros (OGF=0x08 OCF=0x0006
Adv_Interval_Min=0xFFFF Adv_Interval_Max=0xFFFF
Adv_Type=0x03...)

sudo hcitool -i hci1 cmd 0x08 0x0006 A0 00 A0 00 02 00 00 00
00 00 00 00 00 07 00

#Despues ponemos los datos de Advertising (OGF=0x08 OCF=0x0008
Name=09 "STOP=73 74 6F 70" Flags=01 y el relleno en ceros para
completar los 32B)

sudo hcitool -i hci1 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02
01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00

#Despues ponemos los datos de Scan Response (OGF=0x08 OCF=0x0009
Name=09 "STOP=73 74 6F 70" Flags=01 y el relleno en ceros para
completar los 32B)

sudo hcitool -i hci0 cmd 0x08 0x0009 09 05 09 73 74 6F 70 02
01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00

#Habilitamos los Advertising con los parámetros previamente
puestos

sudo hcitool -i hci1 cmd 0x08 0x000A 0x01
```

A2.5: Script capturar medidas de configuraciones diferentes

```
#!/bin/bash

#Primero definimos Array de variables que vamos a usar
IntervaloMs=(150 250 500 1000)
Intervalos=("F0 00" "90 01" "20 03" "40 06")

#Entramos en un bucle que genera 20 medidas de 1000 muestras
for k in `seq 0 3`; do

    tshark -i bluetooth-monitor -a duration:35600 -w Adv_200_`$k`
    &

    sleep 5

    echo mypassword | sudo -S command

#Deshabilitamos el Inquiry Scan y Page Scan

    sudo hciconfig -a hci0 noscan

#Primero deshabilitamos el Scan para poder poner los parámetros
que nos interesan

    sudo hcitool -i hci0 cmd 0x08 0x000C 00 00

#Despues ponemos los parámetros: 00 (pasivo= sin paquetes
Scan_req), LE_Scan_Interval 0xFFFF, LE_Scan_Window 0xFFFF, ...

    sudo hcitool -i hci0 cmd 0x08 0x000B 00 ${Intervalos[$k]} 10
00 00 00

#Por último habilitamos el scanning con los parámetros puestos

    sudo hcitool -i hci0 cmd 0x08 0x000C 01 00

#Este bucle es para generar 1000 muestras de tiempo Discovery
for i in `seq 1 1000`; do

    sudo hciconfig -a hci1 noscan

#Primero ponemos los parámetros (OGF=0x08 OCF=0x0006
Adv_Interval_Min=0xFFFF Adv_Interval_Max=0xFFFF
Adv_Type=0x03...)

    sudo hcitool -i hci1 cmd 0x08 0x0006 40 01 40 01 03 00 00 00
00 00 00 00 00 07 00

#Despues ponemos los datos (OGF=0x08 OCF=0x0008 Name=09 "STOP=73
74 6F 70" Flags=01 y el relleno en ceros para completar los 32B)
```

```

    sudo hcitool -i hci1 cmd 0x08 0x0008 09 05 09 73 74 6F 70 02
01 08 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00

```

#Habilitamos los Advertising con los parámetros previamente
puestos

```

    sudo hcitool -i hci1 cmd 0x08 0x000A 0x01

```

```

    echo $i

```

#Lo dejamos durante un X tiempo advirtiendose (Anunciandose)

```

    sleep 35

```

#Y Despues lo paramos (Desactivar 0x00)

```

    sudo hcitool -i hci1 cmd 0x08 0x000A 0x00

```

#Generamos un numero aleatorio DECIMAL 0.00X [0,0.1]

```

    PAUSE=$(echo "scale=4; $((($RANDOM%${IntervaloMs[$k]}))/1000"
| bc)

```

```

    echo "La pausa es " $PAUSE "Segundos"

```

#Wait for X milisegons PLEASE (Esperamos unos milisegundos antes
de lanzar la siguiente muestra)

```

    sleep $PAUSE

```

done

```

    sleep 700

```

done

Anexo 3: Dispositivos utilizados

A3.1: Trust



Ultra small Bluetooth 4.0 USB adapter to wirelessly connect your notebook with your smartphone, tablet or other Bluetooth device

Key features

- ✓ Ultra small USB adapter to add Bluetooth 4.0 (Smart Ready) to your notebook
- ✓ Latest low energy technology and 15m wireless range *
- ✓ Ultra small; plug in once and never unplug again
- ✓ Also works with older Bluetooth versions
- ✓ * works with devices that support Bluetooth 4.0 Low Energy Mode

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hcitool dev
```

```
Devices:
```

```
hci0 00:1A:7D:DA:71:06
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU: 64:8
```

```
UP RUNNING PSCAN
```

```
RX bytes:612 acl:0 sco:0 events:37 errors:0
```

```

TX bytes:942 acl:0 sco:0 commands:37 errors:0

Features: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87

Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3

Link policy: RSWITCH HOLD SNIFF PARK

Link mode: SLAVE ACCEPT

Name: 'ubuntu-EasyNote-TS44HR-0'

Class: 0x6c0100

Service Classes: Rendering, Capturing, Audio, Telephony

Device Class: Computer, Uncategorized

HCI Version: 4.0 (0x6) Revision: 0x22bb

LMP Version: 4.0 (0x6) Subversion: 0x22bb

Manufacturer: Cambridge Silicon Radio (10)

```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 features
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:
64:8
```

```

Features page 0: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87

<3-slot packets> <5-slot packets> <encryption> <slot
offset>

<timing accuracy> <role switch> <hold mode> <sniff
mode>

<park state> <RSSI> <channel quality> <SCO link> <HV2
packets>

<HV3 packets> <u-law log> <A-law log> <CVSD> <paging
scheme>

<power control> <transparent SCO> <broadcast encrypt>

<EDR ACL 2 Mbps> <EDR ACL 3 Mbps> <enhanced iscan>

<interlaced iscan> <interlaced pscan> <inquiry with
RSSI>

<extended SCO> <EV4 packets> <EV5 packets> <AFH cap.
slave>

```

```

    <AFH class. slave> <LE support> <3-slot EDR ACL>
    <5-slot EDR ACL> <sniff subrating> <pause encryption>
    <AFH cap. master> <AFH class. master> <EDR eSCO 2
Mbps>
    <EDR eSCO 3 Mbps> <3-slot EDR eSCO> <extended
inquiry>
    <LE and BR/EDR> <simple pairing> <encapsulated PDU>
    <non-flush flag> <LSTO> <inquiry TX power> <EPC>
    <extended features>

```

```

Features page 1: 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x00

```

```

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 version

```

```

hci0: Type: BR/EDR  Bus: USB

```

```

    BD Address: 00:1A:7D:DA:71:06  ACL MTU: 310:10  SCO MTU:
64:8

```

```

    HCI Version: 4.0 (0x6)  Revision: 0x22bb

```

```

    LMP Version: 4.0 (0x6)  Subversion: 0x22bb

```

```

    Manufacturer: Cambridge Silicon Radio (10)

```

```

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 reversion

```

```

Warning: unknown command - "reversion"

```

```

hci0: Type: BR/EDR  Bus: USB

```

```

    BD Address: 00:1A:7D:DA:71:06  ACL MTU: 310:10  SCO MTU:
64:8

```

```

    UP RUNNING PSCAN

```

```

    RX bytes:935 acl:0 sco:0 events:43 errors:0

```

```

    TX bytes:962 acl:0 sco:0 commands:43 errors:0

```

```

    Features: 0xff 0xff 0x8f 0xfe 0xdb 0xff 0x5b 0x87

```

```

    Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3

```

```

    Link policy: RSWITCH HOLD SNIFF PARK

```

```

    Link mode: SLAVE ACCEPT

```

```

    Name: 'ubuntu-EasyNote-TS44HR-0'

```

```

    Class: 0x6c0100

```

```
Service Classes: Rendering, Capturing, Audio, Telephony  
Device Class: Computer, Uncategorized  
HCI Version: 4.0 (0x6) Revision: 0x22bb  
LMP Version: 4.0 (0x6) Subversion: 0x22bb  
Manufacturer: Cambridge Silicon Radio (10)
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 name
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:  
64:8
```

```
Name: 'ubuntu-EasyNote-TS44HR-0'
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 class
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:  
64:8
```

```
Class: 0x6c0100
```

```
Service Classes: Rendering, Capturing, Audio, Telephony
```

```
Device Class: Computer, Uncategorized
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 inqtpl
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:  
64:8
```

```
Inquiry transmit power level: 4
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 inqmode
```

```
hci0: Type: BR/EDR Bus: USB
```

```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:  
64:8
```

```
Inquiry mode: Inquiry with RSSI or Extended Inquiry
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 inqparms
```

```
hci0: Type: BR/EDR Bus: USB
```



```
BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:
64:8

Inquiry interval: 4096 slots (2560.00 ms), window: 18 slots
(11.25 ms)

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 pageparms

hci0: Type: BR/EDR Bus: USB

BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:
64:8

Page interval: 2048 slots (1280.00 ms), window: 18 slots
(11.25 ms)

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci0 pageto

hci0: Type: BR/EDR Bus: USB

BD Address: 00:1A:7D:DA:71:06 ACL MTU: 310:10 SCO MTU:
64:8

Page timeout: 8192 slots (5120.00 ms)

ubuntu@ubuntu-EasyNote-TS44HR:~$
```

A3.2: Belkin



ADD BLUETOOTH® WIRELESS TECHNOLOGY TO YOUR DESKTOP, NOTEBOOK OR TABLET

Wirelessly connect up to seven Bluetooth-enabled devices. The Belkin Mini Bluetooth V4.0 Adapter provides cable-free connections to devices up to 10 meters away indoors. This compact, stylish adapter is unobtrusive when plugged into a USB port, and draws minimal energy, so it's perfect to leave plugged in and take anywhere. It is also backward-compatible with legacy Bluetooth technology.

WORKS WITH: MacBook Air, MacBook Air 11", MacBook Air 13", MacBook Pro

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 version

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

HCI Version: 4.0 (0x6) Revision: 0x1000
```

```

LMP Version: 4.0 (0x6) Subversion: 0x220e

Manufacturer: Broadcom Corporation (15)

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 features

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Features page 0: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87

offset> <3-slot packets> <5-slot packets> <encryption> <slot
offset>

<timing accuracy> <role switch> <sniff mode> <RSSI>

<channel quality> <SCO link> <HV2 packets> <HV3
packets>

<u-law log> <A-law log> <CVSD> <paging scheme> <power
control>

<transparent SCO> <broadcast encrypt> <EDR ACL 2
Mbps>

<EDR ACL 3 Mbps> <enhanced iscan> <interlaced iscan>

<interlaced pscan> <inquiry with RSSI> <extended SCO>

<EV4 packets> <EV5 packets> <AFH cap. slave>

<AFH class. slave> <LE support> <3-slot EDR ACL>

<5-slot EDR ACL> <sniff subrating> <pause encryption>

<AFH cap. master> <AFH class. master> <EDR eSCO 2
Mbps>

inquiry> <EDR eSCO 3 Mbps> <3-slot EDR eSCO> <extended

<LE and BR/EDR> <simple pairing> <encapsulated PDU>

TX power> <err. data report> <non-flush flag> <LSTO> <inquiry

<EPC> <extended features>

Features page 1: 0x03 0x00 0x00 0x00 0x00 0x00 0x00 0x00

ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 reversion

Warning: unknown command - "reversion"

hci1: Type: BR/EDR Bus: USB

```

```
BD Address: 5C:F3:70:6C:9B:F9  ACL MTU: 1021:8  SCO MTU:
64:1

UP RUNNING PSCAN ISCAN

RX bytes:658  acl:0  sco:0  events:40  errors:0

TX bytes:953  acl:0  sco:0  commands:40  errors:0

Features: 0xbf 0xfe 0xcf 0xfe 0xdb 0xff 0x7b 0x87

Packet type: DM1 DM3 DM5 DH1 DH3 DH5 HV1 HV2 HV3

Link policy: RSWITCH SNIFF

Link mode: SLAVE ACCEPT

Name: 'ubuntu-EasyNote-TS44HR-0'

Class: 0x6c0100

Service Classes: Rendering, Capturing, Audio, Telephony

Device Class: Computer, Uncategorized

HCI Version: 4.0 (0x6)  Revision: 0x1000

LMP Version: 4.0 (0x6)  Subversion: 0x220e

Manufacturer: Broadcom Corporation (15)
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 name
```

```
hci1:Type: BR/EDR  Bus: USB
```

```
BD Address: 5C:F3:70:6C:9B:F9  ACL MTU: 1021:8  SCO MTU:
64:1
```

```
Name: 'ubuntu-EasyNote-TS44HR-0'
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 class
```

```
hci1:Type: BR/EDR  Bus: USB
```

```
BD Address: 5C:F3:70:6C:9B:F9  ACL MTU: 1021:8  SCO MTU:
64:1
```

```
Class: 0x6c0100
```

```
Service Classes: Rendering, Capturing, Audio, Telephony
```

```
Device Class: Computer, Uncategorized
```

```
ubuntu@ubuntu-EasyNote-TS44HR:~$ hciconfig -a hci1 inqtpl
```

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Inquiry transmit power level: 0

ubuntu@ubuntu-EasyNote-TS44HR:~\$ hciconfig -a hci1 inqmode

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Inquiry mode: Inquiry with RSSI or Extended Inquiry

ubuntu@ubuntu-EasyNote-TS44HR:~\$ hciconfig -a hci1 inqparms

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Inquiry interval: 4096 slots (2560.00 ms), window: 18 slots
(11.25 ms)

ubuntu@ubuntu-EasyNote-TS44HR:~\$ hciconfig -a hci1 pageparms

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Page interval: 2048 slots (1280.00 ms), window: 18 slots
(11.25 ms)

ubuntu@ubuntu-EasyNote-TS44HR:~\$ hciconfig -a hci1 pageto

hci1: Type: BR/EDR Bus: USB

BD Address: 5C:F3:70:6C:9B:F9 ACL MTU: 1021:8 SCO MTU:
64:1

Page timeout: 8192 slots (5120.00 ms)

A3.3: Sena



Parani-UD100	
Standards	<ul style="list-style-type: none"> • Bluetooth 4.0 Class 1 • USB 2.0
Max Transfer Rate	3 Mbps (EDR)
Frequency Range	2.402 ~ 2.480GHz
Transmit Output Power	+19dBm (+6dBm EDR) E.I.R.P
Receive Sensitivity	<ul style="list-style-type: none"> • Basic 1Mbps: -88 dBm • EDR 2Mbps: -87dBm • EDR 3Mbps: -82dBm
Antenna Connector	RP-SMA
Antenna Gain	<ul style="list-style-type: none"> • Default Stub Antenna: 1 dBi • Optional Dipole Antennas: 3 dBi & 5 dBi • Optional Patch Antenna: 9 dBi
Working Distance (In Open Field)	<ul style="list-style-type: none"> • Stub antenna – Stub antenna: 300 m • Dipole (3 dBi) – Dipole (3 dBi) : 400 m • Dipole (5 dBi) – Dipole (5 dBi): 600 m • Patch antenna – Patch antenna: 1 km <p>* working distance can vary depending on install environment</p>
Bluetooth stack software	Latest Device Driver: BlueSoleil
Bluetooth Profiles	DUN, FAX, SPP, HID, FTP, OPP, SDP, HCRP, LAN, OBEX FTP, OBEX OPP, OBEX BIP, BIP, AVRCP, A2DP, HSP, HFP, PAN, BPP, Headset, AVCTP, AVDTP, HDP, Find Me, Proximity, Health Thermometer, Heart Rate, HID OVER GATT
Computer OS Support	<ul style="list-style-type: none"> • Windows XP/Vista/7/8 (32/64bit) • Linux (3rd party driver required) • MAC OS X (MAC OS X driver required)
Size	72(L) x 22(W) x 10(H) mm

Operating Temperature	-40 ~ +85°C
Storage Temperature	-40 ~ +85°C
Humidity	90% Non-condensing
Regulatory Approvals	FCC, CE, TELEC, KCC, IC, Bluetooth SIG
Warranty	1 year Limited Warranty

A3.4: BLE Nano

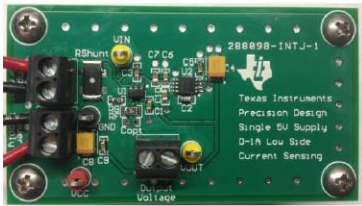


BLE Nano Specification:

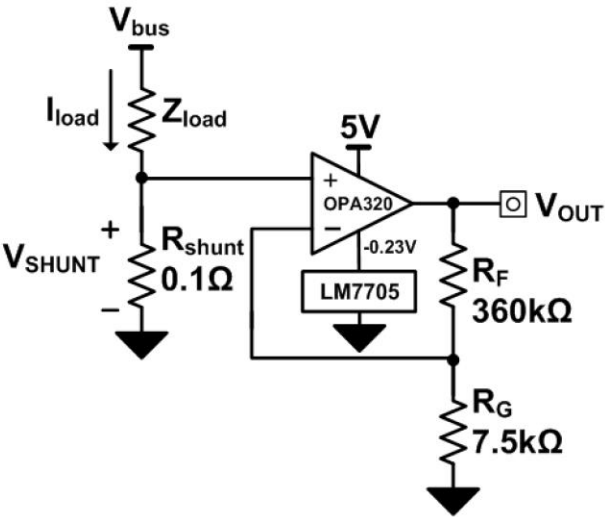
Microcontroller	Nordic nRF51822			
CMSIS-DAP Chip	Freescale MK20			
Operating Voltage	1.8V to 3.3V			
Input Voltage	1.8V-3.3V (VDD) 3.3V-13V (VIN) Caution: Use either one power source at a time, otherwise you will damage the board.			
Clock Speed	16MHz			
Connectivity	Bluetooth	4.0	Low	Energy (TX/RX)
	Serial			

	I2C SPI	
Flash Memory	256KB	
SRAM	32KB 16KB (v1.0)	(v1.5)
Dimensions	18.5 x 21.0mm	
I/O Pins	11	

A3.5: Sensor de corriente



PCB is shown to scale.
Dimensions are 2.45" (L) x 1.35" (W)



Anexo 4: Figuras de posibles casos de tiempo de descubrimiento

