Trabajo de fin de grado

Diseño y desarrollo de Readers Bluetooth Low Energy para IoT

José Molina Machado

Grado en Ingeniería Telemática

Universidad Politécnica de Cartagena

Director: Alejandro Santos Martínez Sala

Fecha: October 9, 2018

Trabajo de fin de grado

Diseño y desarrollo de Readers Bluetooth Low Energy para IoT

José Molina Machado

October 9, 2018

Resumen

La tecnología Bluetooth Low Energy (BLE) tiene un enorme potencial para el desarrollo de servicios de localización en IoT. Una arquitectura posible de localización consiste en una infraestructura de Readers BLE desplegada en un entorno que monitorizan dispositivos beacon BLE y envía los datos a un servidor de localización.
En el presente TFG se pretende diseñar e implementar un Reader BLE basado en Linux y un bus de Readers BLE usando el estándar RS485 que permita reducir el cableado necesario para el despliegue de una infraestructura IoT para servicios de localización de interiores.

Índice

Lista de Figuras

Lista de Tablas

Lista de scripts

Capitulo 0

Introducción

Los sistemas de localización en IoT son de interés para el desarrollo de servicios de localización de objetos y personas en entornos de interior (hospital, supermercado, aeropuerto, universidad, etc.). En los entornos de interior no funciona la tecnología GPS por lo que se necesitan otras tecnologías. En particular la tecnología Bluetooth Low Energy (BLE) 4.0 es una evolución del estándar bluetooth diseñado para el IoT, con un enfoque en el bajo consumo, y tiene un enorme potencial para el desarrollo de servicios de localización.

Existen varias arquitecturas de localización pero una de interés para el mundo IoT se basa en tener una red de scanners o Readers BLE desplegados en un entorno indoor que monitorizan los beacons transmitidos por los dispositivos BLE (asociados a personas u objetos móviles) y enviar los datos en bruto a un servidor para su procesamiento y la estimación de la posición.

Un Reader BLE tiene que tener un cableado mínimo para alimentación y comunicaciones con un servidor. Una forma de ahorrar cableado es teniendo un único bus con hilos para alimentación y datos. En el presente TFG se pretende diseñar e implementar un Reader BLE basado en Linux y un bus de Readers BLE usando el estándar RS485 que permita reducir el cableado necesario para el despliegue de una infraestructura IoT para servicios de localización de interiores.

Para llegar al objetivo, hemos seguido las siguientes fases:

1. Estudio de la tecnología BLE 4.0, describiendo su arquitectura de protocolos y el estándar RS485 para las comunicaciones para tener un cableado mínimo junto con la alimentación.

2. En este capitulo describimos los procedimientos de Advertising, sus tipos, funcionamiento.

3. En este capitulo nos centramos en el procedimiento de Scanning, sus tipos, funcionamiento.

4. En este capitulo describiremos los procedimientos de configuración tanto del Advertising como el Scanning.

5. Ahora pasamos en este capitulo a crear un prototipo con el estándar de Advertising iBeacon y módulo bluetooth SENA, realizaremos pruebas de cobertura así como el envío de información mediante el protocolo UDP.

6. En este capitulo describimos cómo sería un sistema final basado en BLE, con un maestro, que es el que enviara la información al servidor de localización y múltiples esclavos que a través del bus enviaran la información de escaneo al maestro.

Capitulo 1

Bluetooth Low Energy y RS485

Para conseguir nuestro objetivo que es diseñar un sistema de readers Bluetooth Low Energy para IoT usaremos Bluetooth low energy para obtener la información transmitida de los beacons y RS485 para enviar la información recopilada desde los readers al servidor maestro.

1.1 Bluetooth Low Energy

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

BLE 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. De esta idea surgen aplicaciones novedosas en el cuidado de la salud, fitness y sobre todo para este proyecto beacons que son los que objetivos de nuestros readers.

1.1.1 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.

BLE está formado por diferentes capas que interactúan entre sí. Cada una de ellas tiene una función con ciertas limitaciones y requerimientos. Estas diferentes capas forman la pila de protocolos. La pila de protocolos de divide en tres partes; controlador, host y la aplicación. Vamos a ver de qué está compuesta para después profundizar en ellas.

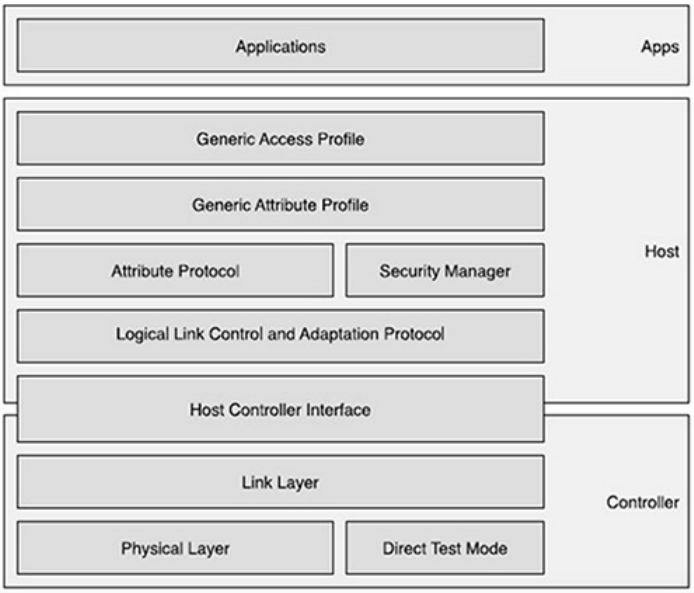


Figura 1.1: ***Pila del protocolo BLE***

Cómo se describe en la figura 1.1 la pila tiene la siguiente estructura:

- Controlador:
  - Interfaz de control del host (HCI), en el lado del controlador.
  - Capa de enlace (LL).
  - Capa física (PHY).
- Host:
  - Perfil genérico de acceso (GAP).
  - Perfil genérico de atributo (GATT).
  - Protocolo de atributo (ATT).
  - Protocolo de gestión de seguridad (SMP).
  - Protocolo de control lógico y adaptación de enlace (L2CAP).
  - Interfaz de control del host (HCI), en el lado del host.
- Aplicación:
  Es la capa superior y encargada de manipular los datos. Su arquitectura depende de la implementación particular de cada caso.

Ahora entraremos en mas profundidad en las capas mas importantes para nuestro proyecto.

1.1.1.1 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. Se definen 40 canales de 2MHz de ancho. Dos tipos de canales: de anuncio y de datos. Los canales de anuncio se utilizan para descubrimiento de dispositivos, transmisión de mensajes de broadcast y configuración de conexiones.

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

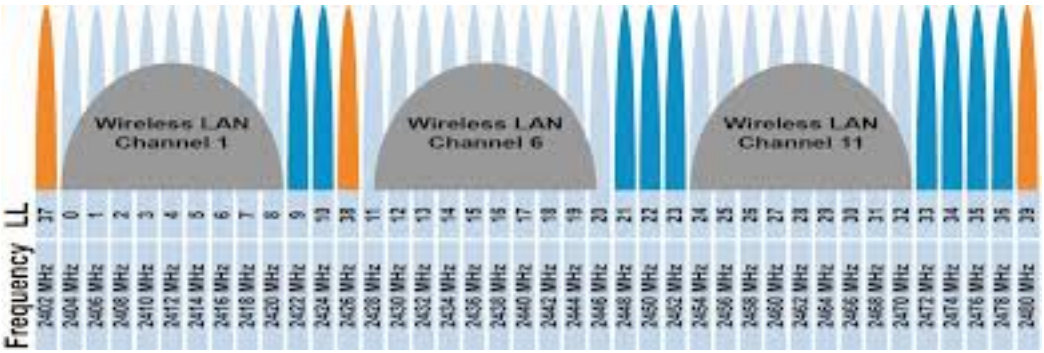


Figura 1.2: ***Distribución de canales BLE***

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.

1.1.1.2 Capa de enlace

Interactúa directamente con la capa física. Es la responsable de cumplir con todos los requisitos de tiempo definidos por la especificación así como de los estados de Advertising, Scanning, creación y mantenimiento de las conexiones. También es la responsable de la estructura de los paquetes.

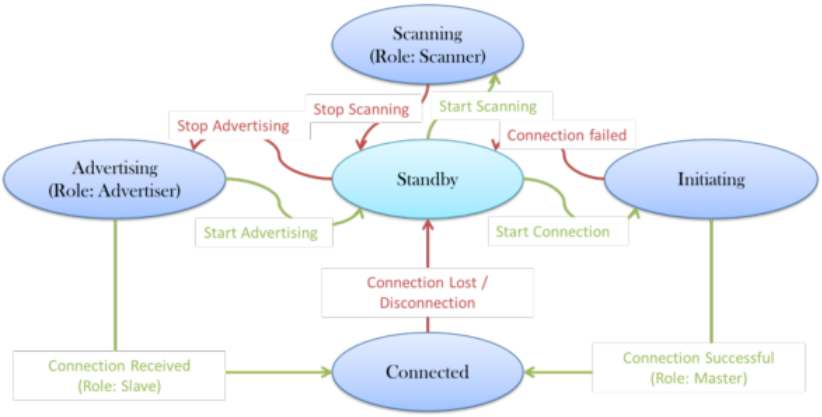


Figura 1.3: ***Estados y roles de BLE***

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

Se definen 4 roles en la capa de enlace:

- Advertiser: el dispositivo envía mensajes de anuncio.
- Scanner: el dispositivo escanea paquetes de anuncio.
- Master: el dispositivo que inicia la conexión y la gestiona después.
- Slave: el dispositivo que acepta peticiones de conexión.

Así como los siguientes estados:

- Standby: No transmite o recibe paquetes. A este estado se puede acceder desde cualquier otro estado, llamado también estado de espera.
- Advertising: Transmite paquetes de anuncio y cuando sea necesario escucha y contesta a los posibles mensajes de petición y repuesta de escaneo. Correspondería al rol de Advertiser. A este estado se puede acceder desde el estado de espera.
- Scanning: Escucha a paquetes de anuncio en los canales de anuncio. Correspondería al rol de Scanner. A este estado se puede acceder desde el estado de espera.
- Initiating: Escucha a paquetes de anuncio y responde para iniciar una conexión con otro dispositivo. A este estado se puede acceder desde el estado de espera.
- Connected: A este estado se puede acceder desde el estado de Scanning o Initiating. Se definen dos roles:
  - Rol maestro: cuando se accede desde Initiating.
  - Rol esclavo: cuando se accede desde Advertising.

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.

- Connectable Undirected Advertising: es el más común. El dispositivo escáner puede recibir los anuncios o ir a hacía una conexión.
- Connectable Directed Advertising 1 : 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 si 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.

1.1.1.3 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, y los veremos en mas profundidad en próximos capítulos.

1.1.2 Comparación con otros estándares

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

Tabla 1.1: Comparación entre estándares

En esta figura podemos comparar diferentes tecnologías que operan en la misma banda de frecuencias, a la hora de elegir la batería o la fuente de alimentación, es importante fijarse en el consumo de corriente. BLE tiene el valor mas bajo y en su estado Standby su consumo es prácticamente nulo. Por lo tanto hemos elegido BLE para este proyecto por las siguientes razones:

- Bajo consumo de potencia: Es importante para todos los proyectos el consumo de energía ya que repercute en el coste de las infraestructuras.
- Fácil desarrollo e instalación: Ya que los dispositivos son pequeños, es decir ocupan poco espacio y pueden ser instalados como tarjetas o pulseras y son configurables.
- Larga vida de batería: Es importante para reducir el mantenimiento de los equipos que la batería de los dispositivos dure lo mas posible.

1.2 RS485

La interfaz RS485 ha sido desarrollada junto a la interfaz RS422 para la transmisión en serie de datos de alta velocidad a grandes distancias y encuentra creciente aplicación en el sector industrial. Pero mientras que la RS422 sólo permite la conexión unidireccional de hasta 10 receptores en un transmisor, la RS485 está concebida como sistema Bus bidireccional con hasta 32 participantes. Físicamente las dos interfaces sólo se diferencian mínimamente. El Bus RS485 puede instalarse tanto como sistema de 2 hilos o de 4 hilos.

Dado que varios transmisores trabajan en una línea común, tiene que garantizarse con un protocolo que en todo momento esté activo como máximo un transmisor de datos. Los otros transmisores tienen que encontrarse en ese momento en estado

de espera. La norma RS485 define solamente las especificaciones eléctricas para receptores y transmisores de diferencia en sistemas de bus digitales. La norma ISO 8482 estandariza además adicionalmente la topología de cableado con una longitud máx. de 500 metros.

1.2.1 Modos de instalación

El Bus RS485 puede instalarse tanto como sistema de 2 hilos o de 4 hilos. Nosotros para este proyecto hemos elegido el de 2 hilos por su simpleza ya que usaremos un protocolo maestro esclavo para la transmisión de datos.

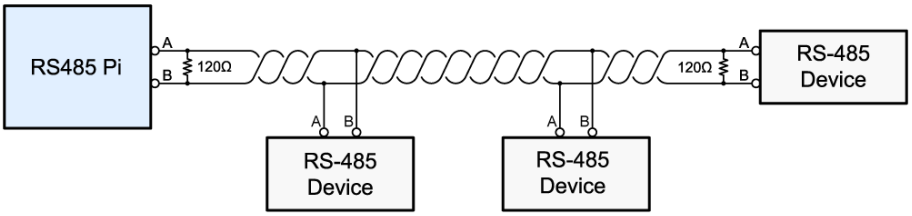


Figura 1.4: ***Cableado de RS485***

1.2.2 Método físico de transmisión

Los datos en serie, como en interfaces RS422, se transmiten sin relación de masa como diferencia de tensión entre dos líneas correspondientes. Para cada señal a transmitir existe un par de conductores que se compone de una línea de señales invertida y otra no invertida. La línea invertida se caracteriza por regla general por el índice ”A” o ”-”, mientras que la línea no invertida lleva ”B” o ”+”. El receptor evalúa solamente la diferencia existente entre ambas líneas, de modo que las modalidades comunes de perturbación en la línea de transmisión no falsifican la señal útil. Los transmisores RS485 ponen a disposición bajo carga un nivel de salida de ±2V entre las dos salidas; los módulos de recepción reconocen el nivel de ±200mV como señal válida.

1.2.3 Longitud de líneas

Usando un método de transmisión simétrico en combinación con cables de pares de baja capacidad y amortiguación (twisted pair) pueden realizarse conexiones muy eficaces a través de una distancia de hasta 500m con ratios de transmisión al mismo tiempo altas. El uso de un cable TP de alta calidad evita por un lado la diafonía entre las señales transmitidas y por el otro reduce adicionalmente al efecto del apantallamiento, la sensibilidad de la instalación de transmisión contra señales perturbadoras entremezcladas. En conexiones RS485 es necesario un final de cable con redes de terminación para obligar al nivel de pausa en el sistema de Bus en los tiempos en los que no esté activo ningún transmisor de datos.

Capitulo 2

Advertising en BLE

Como hemos visto en el capitulo anterior, para obtener los datos que necesitamos vamos a usar el método de Scanning y Advertising, donde los paquetes de Advertising contienen campos donde se proporciona la información, al ser este método broadcast no existe seguridad, la emisión de datos es pública pero todos los dispositivos que detectan el advertising pueden recibir la información al mismo tiempo. En este capitulo nos centraremos en el procedimiento de advertising, explicando sus tiempos y el tipo y tamaño de los paquetes, asi como la configuración de los datos incluidos en ellos.

2.1 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.

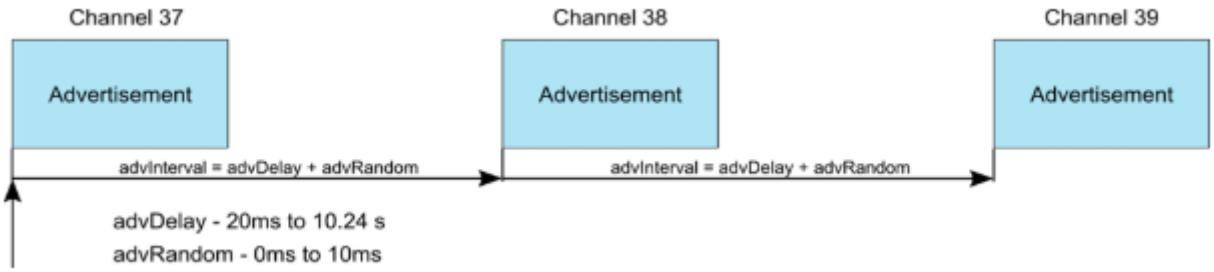


Figura 2.1: ***Intervalo de Advertising***

El intervalo fijo que se llama Advertising Interval, 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 aleatorio es un valor entre 0 ms y 10 ms que se añade a continuación. Este 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.

En la siguiente figura podemos ver un cronograma del proceso de Advertising.

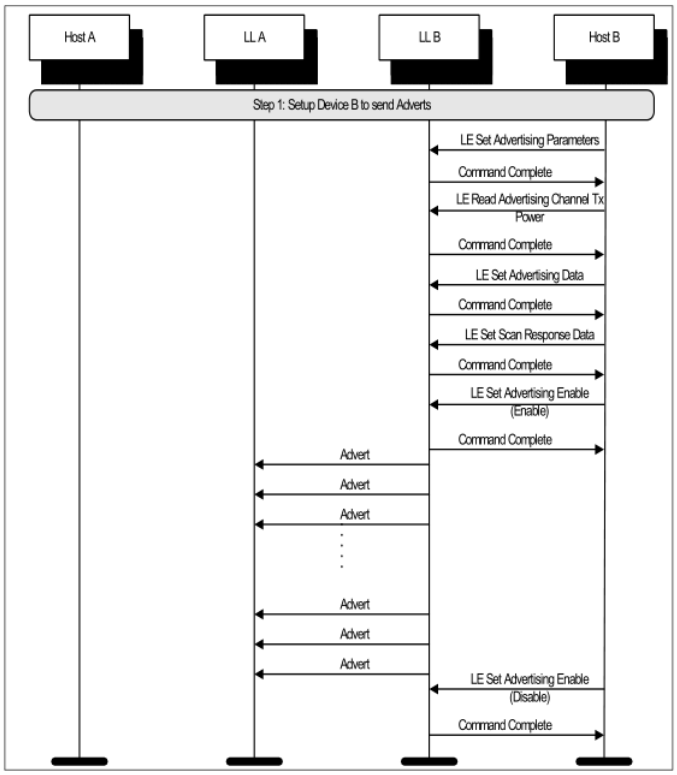


Figura 2.2: ***Cronograma de Advertising***

En este cronograma podemos ver el proceso de configuración y el de Advertising. Entraremos mas en detalle en el proceso de configuración en este capitulo 4.3.

2.2 Paquetes Advertising PDU

Hay varios tipos de paquetes Advertising según la función que se requiera:

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

Tabla 2.1: Tipos de paquetes de Advertising

2.2.1 Elección del tipo de paquete

En este proyecto como vamos a usar beacons que no requieren conexión directa, vamos a usar el Non-connectable Undirected.

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.

El tiempo entre dos paquetes ADV_NONCONN enviados en canales consecutivos tiene que ser como máximo 10 ms. Lo podemos ver en mas detalle en la siguiente figura 2.3

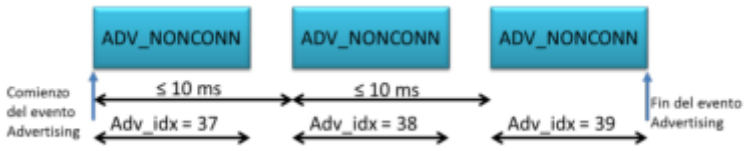


Figura 2.3: ***Nonconnectable Advertising***

2.3 Estructura de datos del paquete

A la hora de mandar paquetes los beacon se pueden configurar según una serie de estándares, como pueden ser:

- IBeacon de Apple ha estandarizado el formato para BLE. Bajo este formato, un paquete de publicidad consta de cuatro piezas principales de información:
  - UUID: Esta es una cadena de 16 bytes utilizado para diferenciar un gran grupo de balizas relacionados.
  - Major: Esta es una cadena de 2 byte utilizado para distinguir un subconjunto más pequeño de balizas dentro del grupo más grande.
  - Menor: Esta es una cadena de 2 byte destinado a identificar balizas individuales.
  - Tx alimentación: Esta se utiliza para determinar la proximidad (distancia) de la baliza. ¿Cómo funciona esto? Potencia de TX se define como la intensidad de la señal exactamente 1 metro desde el dispositivo.
- Eddystone: es un proyecto de código abierto desarrollado por Google. A diferencia de iBeacon, tiene soporte oficial para iOS y Android. Un beacon configurado con este protocolo puede emitir uno de los siguientes tipos de paquetes:
  - Eddystone-UID: contiene un identificador de un beacon.
  - Eddystone-URL: contiene una URL.
  - Eddystone-TLM: es emitido con los paquetes anteriores y contiene el estado de salud de un beacon, como el nivel de batería por ejemplo.
  - Eddystone-EID: contiene un identificador encriptado que cambia periódicamente.

Para nuestro proyecto necesitamos transmitir poca información por lo que hemos elegido iBeacon y este sería el paquete ilustrado en la siguiente figura.

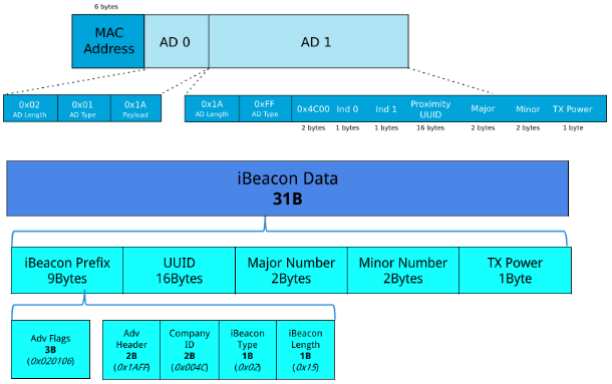


Figura 2.4: ***PDU iBeacon***

Capitulo 3

Scanning en BLE

En este estado es en el que escuchamos los paquetes de Advertising, existen dos tipos de escaneo:

- Passive Scanning: Este tipo de escaneo pasivo solo recibe paquetes de Advertising.
- Active Scanning: Este tipo de escaneo activo a parte de recibir los paquetes de Advertising, puede enviar paquetes de respuesta para solicitar mas información de los dispositivos.

3.1 Intervalo de Scanning

El proceso de escaneo depende de dos parámetros:

- Ventana de escaneo: Duración de la escucha de paqutes Advertising.
- Intervalo de escaneo: Frecuencia que ocurre la escucha.

En esta figura podemos ver claramente los tiempos.

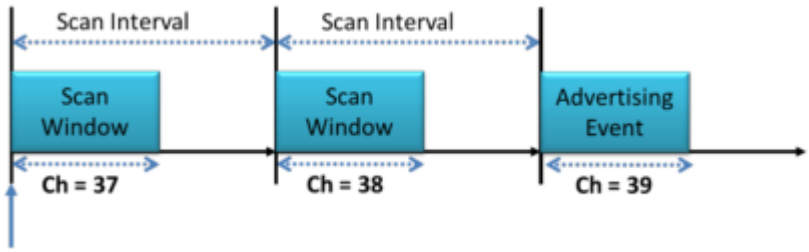


Figura 3.1: ***Intervalo de Scaning***

3.2 Elección del tipo de escaneo

Para este proyecto hemos elegido el tipo de escaneo pasivo ya que, solo necesitamos los datos de advertising de los dispositivos y nada mas. Nuestro dispositivo de escaneo usa el escaneo pasivo para encontrar paquetes de Advertising en su área de cobertura. En esta figura 3.2 podemos ver un cronograma de un escaneo.

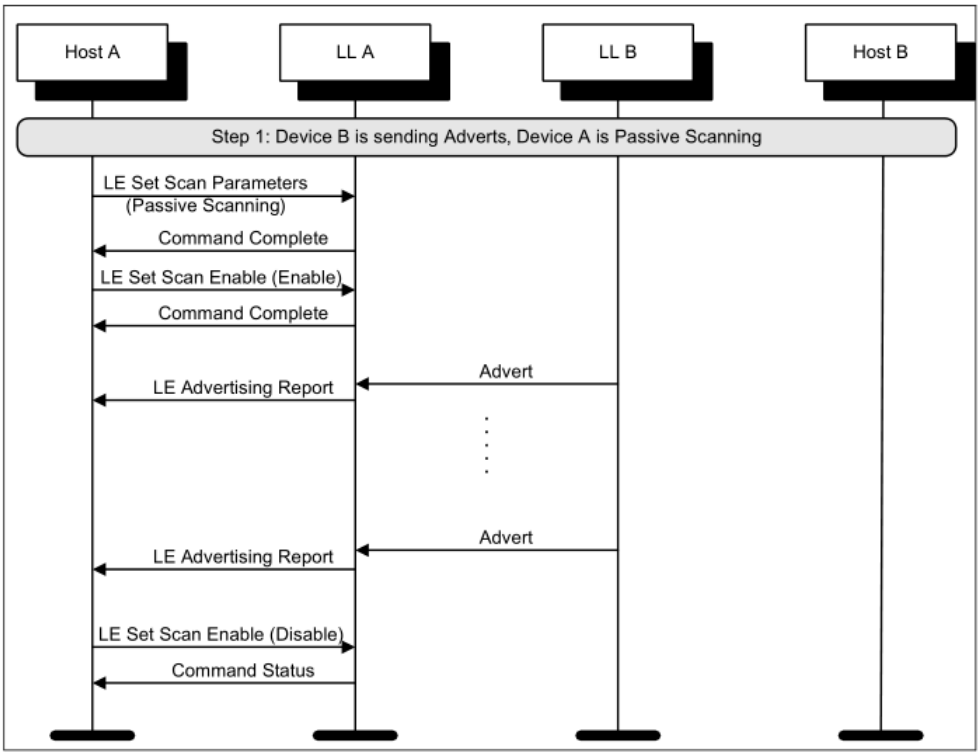


Figura 3.2: ***Cronograma de Escaneo***

En este cronograma podemos observar como el dispositivo B envía paquetes Advertising y el dispositivo A se configura en modo escaneo pasivo, recibe los paquetes y los envía al Host. Entraremos mas en detalle en el proceso de configuración en el capitulo 4.2.

3.3 Estructura de datos del paquete

El paquete de Scanning tendría esta forma:

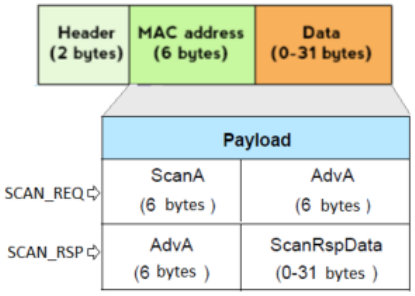


Figura 3.3: ***PDU de Scanning***

Este paquete tiene dos tipos de carga útil:

- SCAN_REQ: La capa de enlace del dispositivo escaneador, este paquete lo recibe el dispositivo Advertiser que envió el paquete de Advertising. Este paquete no contiene datos del host.
- SCAN_RSP: Este paquete lo envía el dispositivo Advertiser como contestación a un paquete SCAN_REQ. Este paquete puede contener datos del host en el campo ScanRspData.

Capitulo 4

Configuración de Advertising y Scanning

En este capitulo vamos a describir como configurar el Advertising y el Scanning en los dispositivos correspondientes así como la plataforma de desarrollo y el software utilizado.

4.1 Plataforma de desarrollo y software

Para este proyecto se ha elegido Linux Ubuntu ya que posee la pila de bluetooth llamada Bluez, como Windows no tiene el software adecuado para el desarrollo de las aplicaciones necesarias, lo he descartado para cumplir el objetivo de este proyecto. Como lenguaje de programación usaremos Bash para las pruebas de configuración y funcionamiento, y Python para el prototipo final, ya que nos aporta funciones de alto nivel que necesitamos.

4.1.1 BlueZ

BlueZ ofrece soporte para el núcleo de Bluetooth y sus protocolos, es flexible, eficiente y usa un modo modular de implementación, entre sus características principales, están:

- Diseño modular completo.
- Soporte para múltiples dispositivos bluetooth.
- Interfaz socket para todas las capas.

Sus principales módulos son:

- Subsistema bluetooth del kernel.
- HCI UART, USB, PCMCIA y drivers para dispositivos virtuales.
- Librerías generales para bluetooth.
- Herramientas de análisis y decodificación del protocolo.

4.1.1.1 Instalación

Para poder instalar BlueZ primero necesitamos instalar los siguientes paquetes linux, y se harían con los siguientes comandos 1:

```
apt-get -y install python-dev
apt-get -y install libglib2.0-dev
apt-get -y install libdbus-1-dev
apt-get -y install libudev-dev
apt-get -y install libical-dev
apt-get -y install libreadline-dev
apt-get -y install python-pip
python -m pip install pybluez
```

Código 1: Instalación de dependencias

Una vez instalamos las dependencias, podemos proceder a instalar BlueZ con estos comandos:

```
wget http://www.kernel.org/pub/linux/bluetooth/bluez-5.50.tar.xz
tar -xvf bluez-5.50.tar.xz
cd bluez-5.50.tar.xz
./configure
make
make install
```

Código 2: Instalación de BlueZ

Después de instalar BlueZ, si la instalación se ha realizado correctamente 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 siguiente comando:

```
hcitool dev
```

Código 3: Comando para ver dispositivos conectados

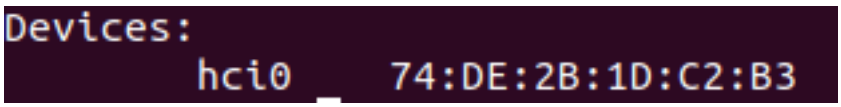


Figura 4.1: ***Resultado del comando hcitool dev***

El resultado muestra todos los dispositivos BLE que están conectados al PC con sus respectivas direcciones BD ADDR (Bluetooth Device ADDRess), únicas como las conocidas MAC.

4.2 Configuración de Scanning

Para configurar el escaneo pasivo vamos a usar comandos HCI que son aquellos con los que BlueZ se comunica con el dispositivo para configurarlo u obtener información de él. 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. Mirando la documentación haremos uso del comando "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

Figura 4.2: ***LE Set Scan Parameters Command***

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

Código 4: Comando de configuración de escaneo pasivo

hci0 es el identificador del adaptador Bluetooth que tenemos conectado a nuestro ordenador.

A partir del OCF 0x000B se configuran los siguientes campos:

- LE_Scan_Type (1 octeto):
  - 0x00: Escaneo pasivo, no se envía paquetes SCAN_REQ.
  - 0x01: Escaneo activo, se pueden enviar paquetes SCAN_REQ.
- LE_Scan_Interval (2 octetos):
  - Rango: 0x0004 a 0x4000.
  - $T = N \cdot 0.625$ siendo N el valor introducido.
- LE_Scan_Window (2 octetos):
  - Rango: 0x0004 a 0x4000.
  - $T = N \cdot 0.625$ siendo N el valor introducido.
    Este valor tiene que ser menor o igual que LE_Scan_Interval.
- Own_Address_Type: (1 octeto):
  - 0x00: Dirección del dispositivo pública.
  - 0x01: Dirección del dispositivo aleatoria.
- Scanning_Filter_Policy: (1 octeto):
  - 0x00: Sin filtro, se aceptan todos los paquetes, menos los paquetes dirigidos a otros dispositivos específicos.
  - 0x01: Se ignoran todos los paquetes de los dispositivos que no están en la lista blanca.

Si el comando de configuración es correcto se recibe el código 0x00 si nó se recibirá un código que va desde el 0x01 al 0xFF.

Para iniciar o parar usaremos los siguientes comandos:

Command	OCF	Command parameters	Return Parameters
HCI_LE_Set_Scan_ Enable	0x000C	LE_Scan_Enable Filter_Duplicates	Status

Figura 4.3: ***LE Set Scan Enable Command***

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

Código 5: Comando para parar el escaneo

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

Código 6: Comando para iniciar el escaneo

A partir del OCF 0x000C que es el LE Set Scan Enable Command se configuran los siguientes campos:

- LE_Scan_Enable (1 octeto):
  - 0x00: Parar el escaneo.
  - 0x01: Iniciar el escaneo.
- Filter_Duplicates:
  - 0x00: Filtro de duplicados desactivado.
  - 0x01: Filtro de duplicados activado.

4.3 Configuración de Advertising con formato iBeacon

Al igual que con la configuración de escaneo vamos a utilizar comandos HCI para configurar la transmisión de paquetes Advertising con formato iBeacon, para ello haremos uso del comando "LE Set Scan 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

Figura 4.4: ***LE Set Advertising Parameters Command***

- Advertising_Interval_Min (2 octetos):
  - Rango: 0x0020 a 0x4000.
  - T = N $*$ 0.625 siendo N el valor introducido.
- Advertising_Interval_Max (2 octetos):
  - Rango: 0x0020 a 0x4000.
  - T = N $*$ 0.625 siendo N el valor introducido.
- Advertising_Type (1 octeto):
  - 0x00: Connectable undirected advertising (ADV_IND).
  - 0x01: Connectable directed advertising (ADV_DIRECT_IND).
  - 0x02: Scannable undirected advertising (ADV_SCAN_IND).
  - 0x03: Non connectable undirected advertising (ADV_NONCONN_IND).
- Own_Address_Type (1 octeto):
  - 0x00: Dirección del dispositivo pública.
  - 0x01: Dirección del dispositivo aleatoria.
- Direct_Address_Type (1 octeto):
  - 0x00: Dirección del dispositivo pública.
  - 0x01: Dirección del dispositivo aleatoria.
- Direct_Address (6 octetos):
  - 0xXXXXXXXXXXXX: Dirección pública o aleatoria.
- Advertising_Channel_Map (1 octeto):
  - xxxxxxx1: Habilitamos el canal 37.
  - xxxxxx1x: Habilitamos el canal 38.
  - xxxxx1xx: Habilitamos el canal 39.
  - 00000111: Habilitamos todos los canales.
- Advertising_Filter_Policy (1 octeto):
  - 0x00: Permite peticiones de escaneo y de conexión de cualquiera.
  - 0x01: Permite peticiones de escaneo de la lista blanca pero peticiones de conexión de cualquiera.
  - 0x02: Permite peticiones de escaneo de cualquiera pero de conexión solo de la lista blanca.
  - 0x03: Solo permite peticiones de escaneo y conexión de la lista blanca. Si el comando el correcto recibiremos un 0x00.

Y para configurar los datos que van dentro del paquete usaremos el OCF 0x0008 LE Set Advertising Data Command:

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

Figura 4.5: ***LE Set Advertising Data Command***

- Advertising_Data_Length (1 octeto):
  - 0x00 a 0x1F: El número de datos significativos.
- Advertising_Data (31 octetos).

Estos serían los pasos de configuración de un beacon que transmite cada 100 ms los paquetes en formato iBeacon, para ello seguimos los siguientes pasos:

- Primero necesitamos un UUID valido, esto lo podemos conseguir con Python con el siguiente comando:

```
python -c 'import sys,uuid;sys.stdout.write(uuid.uuid4().hex)'
```

Código 7: Comando para obtener un UUID valido

En este caso: 636f3f8f64914bee95f7d8cc64a863b5

- Los comandos serían para enviar un Ibeacon cada 100 ms en todos los canales:

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

Código 8: Comando para configurar los parámetros del Advertising

```
hcitool -i hci0 cmd 0x08 0x0008 1E 02 01 1A 1A FF 4C 00 02 15 63 6F
↪  3F 8F 64 91 4B EE 95 F7 D8 CC 64 A8 63 B5 00 01 00 02 C8 00
```

Código 9: Comando para configurar los datos del Advertising iBeacon

- Iniciamos el Advertising:

```
hcitool -i hci0 cmd 0x08 0x000A 01
```

Código 10: Comando para iniciar el Advertising iBeacon

El formato de iBeacon sigue la siguiente estructura, después del 1E que indica que se van a usar los 30 bits de datos:

- Configuración de las Flags:
  - 1E Uso de los 30 bits de datos.
  - 02: Número de bytes que siguen la primera estructura.
  - 01: Tipo.
  - 1A: Valor de las Flags, 0x1A = 000011010
    - bit 0 (OFF) LE Limited Discoverable Mode
    - bit 1 (ON) LE General Discoverable Mode
    - bit 2 (OFF) BR/EDR Not Supported
    - bit 3 (ON) Simultaneous LE and BR/EDR to Same Device Capable (controller)
    - bit 4 (ON) Simultaneous LE and BR/EDR to Same Device Capable (Host)
  - 1A: Número de bytes que siguen la segunda y última estructura.
- Valores específicos de Apple:
  - FF: Tipo específico de datos.
  - 4C 00: Código de identificación de Apple (0x004C == Apple)
  - 02: Byte 0 del indicador del iBeacon advertisement.
  - 15: Byte 1 del indicador del iBeacon advertisement.
- Nuestro valor de UUID, los valores Major, Minor y la potencia de TX:
  - 63 6F 3F 8F 64 91 4B EE 95 F7 D8 CC 64 A8 63 B5: UUID.
  - 00 01: Major.
  - 00 02: Minor.
  - C8 00: Potencia de TX a 1 metro.

Aquí podemos ver una captura hecha con Wireshark de un paquete iBeacon:

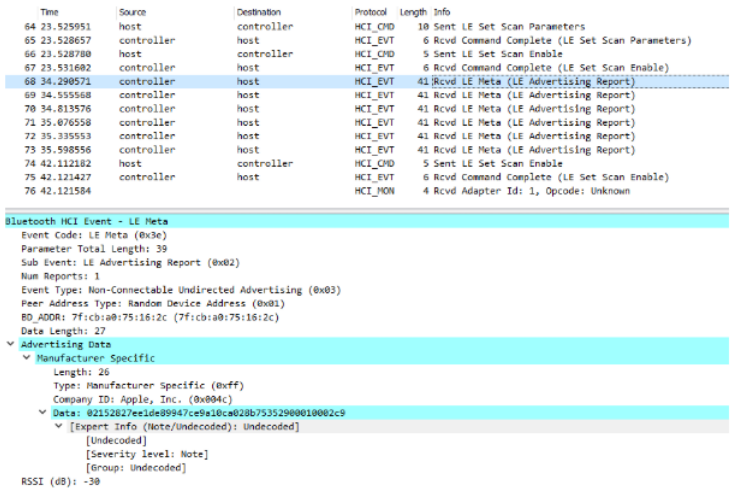


Figura 4.6: ***Captura de un paquete iBeacon***

Capitulo 5

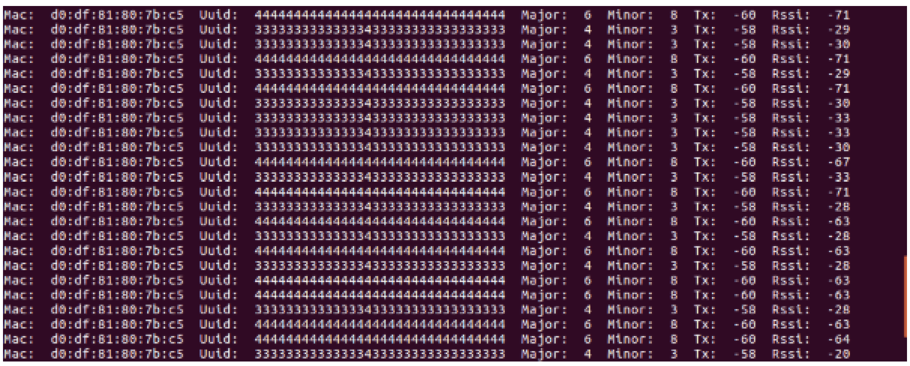
Prototipo con módulo bluetooh Sena

Para este prototipo vamos a unsar un pc Linux con un módulo bluetooth Sena como escaner y dispositivos iBks como beacons, para luego enviar los datos obtenidos por el escaner por Ethernet en con el protocolo UDP a otro ordenador donde se procesaran y almacenaran los datos.

Figura 5.1: ***Módulo bluetooth sena***

Figura 5.2: ***Dispositivo beacon iBks***

A partir de este punto dejamos de usar comandos en bash y pasamos a usar Python para los scripts usados para el escaner, ya que necesitamos funciones de alto nivel para decodificar los paquetes de Advertising y poder enviar los datos por ethernet. A continuación podemos ver los datos recibidos por el servidor.



```
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -71
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -29
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -30
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -71
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -29
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -71
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -30
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -33
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -33
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -30
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -67
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -33
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -71
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -28
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -63
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -28
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -63
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -28
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -63
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -63
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -28
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -63
Mac:  d0:df:81:80:7b:c5  Uuid:  44444444444444444444444444444444  Major:  6  Minor:  8  Tx:  -60  Rssi:  -64
Mac:  d0:df:81:80:7b:c5  Uuid:  33333333333333433333333333333333  Major:  4  Minor:  3  Tx:  -58  Rssi:  -20
```

Figura 5.3: ***Datos recibidos por el servidor***

En la figura 5.3 podemos ver que de los paquetes de Advertising el escaner decodifica los siguientes campos, construye un String y lo envía al servidor donde lo muestra. Los campos son:

- Mac
- UUID
- Major
- Minor
- Tx a un metro.
- RSSI

5.1 Prueba capacidad de scanner para recibir advertisments sin pérdida de datos

El objetivo es evaluar mediante experimentos qué cantidad de advertisments es capaz de monitorizar el scanner y enviar a un servidor por unidad de tiempo y saber si se envían todos los datos o se pierden muestras.

Mediante las pruebas también se quieren entender cómo funciona la implementación del método de escaneo en BlueZ (LE Report) y saber qué tiempos de configuración son los óptimos.

5.1.1 Planteamiento experimentos

- Configuración beacon iBKS
  - Sólo un slot con intervalo de transmisión de 1000 mseg y 500 mseg.
  - Se asume que según el estándar cuando el beacon le toca transmitir envía 3 advertisment (1 por canal 37,38, 39).
- Configuración scanner
  - Cuando se genera un LE Report se envía mensaje al servidor (con número de secuencia).
  - Escaneo pasivo con scan interval y window parametrizable: 1000 mseg, 500 mseg, 200 mseg.
  - Tiempo inicio y fin de la prueba de escaneo configurable a X segundos.
    - En X segundos se estima y redondea el número de advertisment transmitidos por el beacon iBKS y que serían los máximos que podría recibir el servidor.
  - P.e $X = 10s \rightarrow 10 \cdot 3 = 30$ advertisment entre los canales 37,38,39. Por probabilidad el scanner podrá capturar un máximo de 10 advertisment de cualquiera de los canales por que en un instante sólo puede estar monitorizando un canal.

5.1.2 Resultados

Duración prueba (s)	Scan Window (ms)	Numero máximo Advertising	Número BLE	Advertisment recibidos
10	1000	1 seg / $10 \cdot 3$	1	10
10	500	1 seg / $10 \cdot 3$	1	9
10	250	1 seg / $10 \cdot 3$	1	8
20	1000	1 seg / $10 \cdot 3$	1	20
20	500	1 seg / $10 \cdot 3$	1	19
20	250	1 seg / $10 \cdot 3$	1	19

Tabla 5.1: Resultados de pruebas con 1 beacon iBKS

Por las pruebas que hecho para todos los valores de Window se reciben un número alto de paquetes siendo con tiempos de 250 ms o superiores los que más se acercan a 10 paquetes para 10 segundos de simulación o 20 para 20 segundos de simulación.

5.2 Prueba de capacidad del scanner para varios beacons en cobertura

La del probabilidad de éxito de recibir el advertisment de un BLE depende de la calidad de la señal recibida del mensaje en la radio del scanner, el ruido RF que haya y la probabilidad que haya interferencias o colisiones porque otros beacons transmitan a la vez.

Asumiendo que todos los beacons están cercanos al scanner (asunción de buena calidad de la señal) y que tienen los mismos intervalos de transmisión configurados pero que no están sincronizados, el objetivo es comprobar la capacidad del scanner de muestrear advertisment de varios beacons en cobertura.

5.2.1 Planteamiento de los experimentos

- Usar tiempos de escaneo óptimos deducidos en la prueba de 1 beacon.
- Beacons configurados a la misma potencia y slot cada 250mseg.
- Se colocan a la misma distancia del escaner.
- Lanzar pruebas de 10 segundos y generar fichero resultados. Repetir la prueba de captura de 10 segundos 2-3 veces para confirmar que se obtiene lo mismo.

5.2.2 Resultados

Número de BLE	Número mínimo advert. de un BLE	Número medio adver. por BLE	Número máximo adver.	Se detectan todos los BLE
2	40	78	80	Si
4	40	157	160	Si
8	40	308	320	Si
10	40	390	400	Si

Tabla 5.2: Resultados de pruebas con múltiples beacon iBKS

5.3 Prueba de cobertura y alcance

La probabilidad de éxito de la recepción de un advertisment depende de la calidad de la señal recibida (SNR) en el receptor. A su vez, la calidad de la señal es función de la propagación de la señal RF, potencia de transmisión, ganancias de las antenas, distancia entre beacon y scanner, etc.

El objetivo de los experimentos es estimar la métrica de probabilidad de éxito de recepción de un beacon variando la distancia y obstáculos entre el beacon y el scanner y analizar la RSSI medida en cada caso.

5.3.1 Planteamiento de los experimentos

- BLE configurado a potencia -58, 1 slot cada 250mseg.
- Prueba de 10 segundos (máximo 40 advertisment)

Las pruebas han sido realizadas en un piso con estas características:

Figura 5.4: ***Plano de las pruebas***

5.3.2 Resultados

Número de prueba	Distancia (m)	RSSI promedio	Exito (recibidos/40)	Comentarios
1	1	-45	40	Sin obstáculos
2	2	-53	40	Sin obstáculos
3	6	-70	35	Tres paredes finas
4	8	-67	35	Cuatro paredes finas
5	9	-72	24	Cuatro paredes finas
6	11	-73	23	Una pared fina más dos armarios
7	13	-	-	En el balcón

Tabla 5.3: Resultados de pruebas de cobertura

Capitulo 6

Prototipo final de Readers BLE

La idea principal de este proyecto es la de crear una red de Readers BLE en entornos cerrados para la localización de objetos y/o personas en dichos entornos ya que no funciona la tecnología GPS, por lo que es necesario aplicar otras formas de localización. Estos readers estarán escaneando zonas concretas del sitio a monitorizar, enviando los datos de los beacons obtenidos primero a un servidor maestro donde se procesaran los datos en bruto y les dará el formato adecuado para después enviar dichos datos a un servidor de localización para su evaluación.

Para que esto funcione vamos a aplicar un protocolo de petición respuesta entre el maestro y los esclavos en la red, ya que vamos a conectar estos readers en un bus serie RS485. Tiene que ser petición respuesta porque este bus es half-duplex y solo puede estar ocupando el canal un dispositivo.

6.1 Consideraciones y requisitos

A continuación se explican las condiciones de funcionamiento tanto del maestro como de los readers, para poder definir claramente su propósito

6.1.1 Maestro

1. Hay un listado de zonas de un entorno donde se quiere tener información de la localización:

    - Id zona

2. Sabemos el número total de beacons/personas existentes (alta en el sistema): ID beacon - ID persona.

3. Los beacons tienen una misma configuración preestablecida: formato iBeacon, intervalo, Potencia tx.

4. Las personas se mueven por todas las zonas pero no sabemos en qué zona real están y durante cuánto tiempo.

5. Se considera que se tienen N Readers que se instalan en puntos estratégicos para detectar los beacons (personas) en cobertura.

6. No sabemos cuántos beacons habrá en cobertura de un Reader y durante cuánto tiempo.

7. Fijamos ventanas temporales en el maestro.

6.1.2 Readers

1. Los Reader tienen que recopilar los datos de beacons de una forma optimizada y enviarlos al maestro.

2. Se busca detectar todos los beacons en cobertura en el menor tiempo posible.

3. No se van a poder tener todos los datos porque hay un compromiso entre las colisiones entre beacons, el intervalo de transmisión de los beacons, el tiempo de escaneo, la implementación protocolo bus RS485, latencia del Linux, etc.

4. Hay que ver qué se puede lograr y cuántos datos de calidad y cantidad se pueden conseguir.

6.2 Descripción del procolo petición respuesta

A continuación describiremos el protocolo petición respuesta diferenciando el funcionamiento de los readers y el maestro.

6.2.1 Maestro

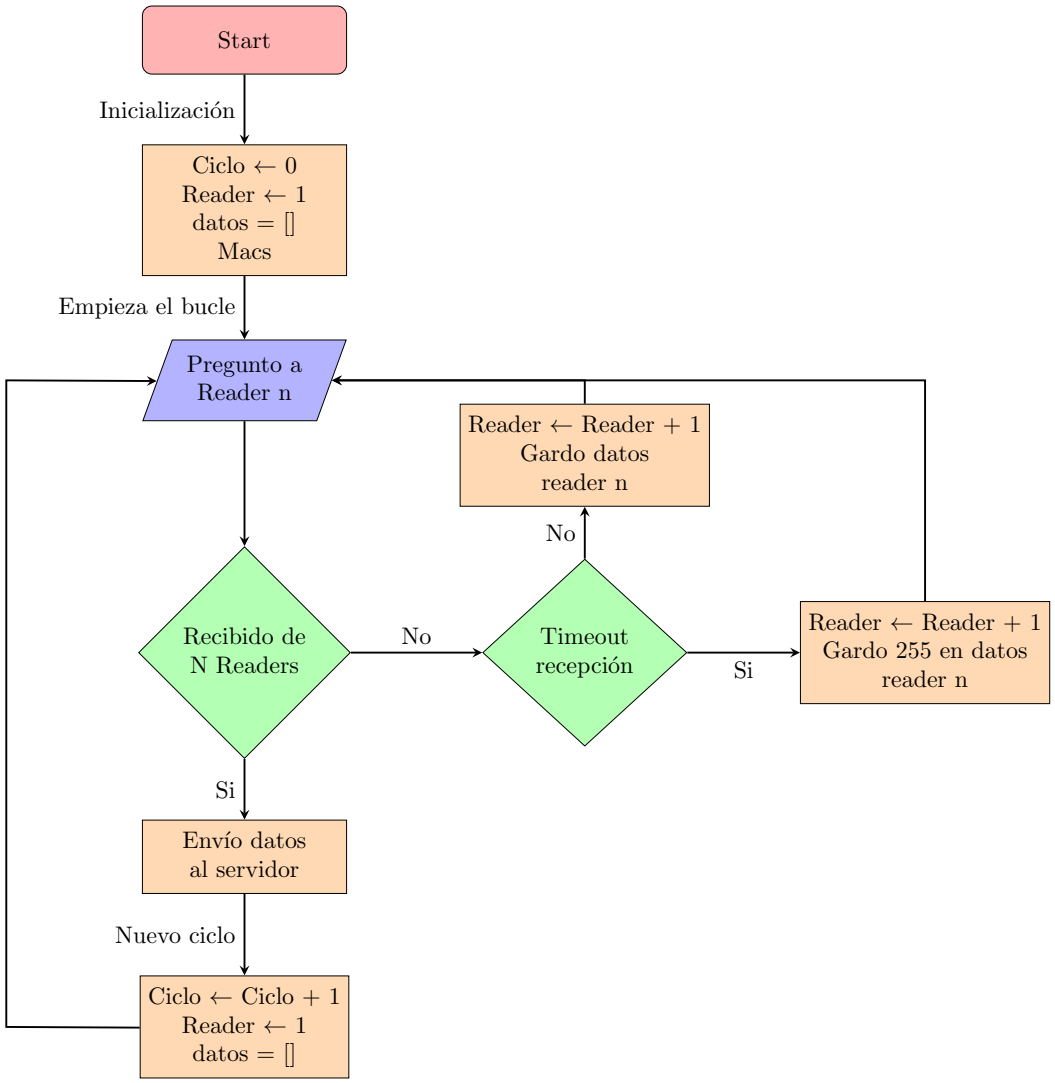


Figura 6.1: ***Algoritmo del dispositivo maestro***

El maestro hace slots temporales (slot 1, slot 2, slot 3, slot i, $\cdots$ slot N) para consultar a los N Readers del bus qué datos nuevos tienen. Un ciclo dura N slots y sirve para consultar a los N Reader. Durante el ciclo j, slot i:

- Maestro pide datos al Reader i.
- Reader i envía datos nuevos disponibles. Si no tiene datos nuevos envía 0.
- Maestro pasa al slot i+1...
- Reader i+i envía datos nuevos disponibles.
- Maestro pasa al slot N
- Reader N envía datos nuevos disponibles.
- Maestro empieza ciclo j+1, slot 1...

Ejemplo datos esperados para enviar al servidor: Supongamos que hay 3 beacon MB1, MB2 y MB3 y hay dos Reader R1 y R2, MB1, MB2 en cobertura de R1 y MB3 en cobertura sólo de R2.

- Empieza ciclo 1 en instante t1
- Pregunto Reader 1, como el Reader 2 esta en el bus también recibe el mensaje pero lo descarta:
- Resp R1: MB1;-80;04;05;-50; y MB2;-60;06;06;-65;
- Pregunto Reader 2, el Reader 1 recibe el mensaje pero lo descarta.
- Respuesta R2: MB1;-51;5;05;-50; / MB2;-61;7;06;-65; / MB3;-30;02;05;-44;
- Fin ciclo 1 - formateo datos para enviar al servidor (para motor localización)
  - MB1;1;1;-80;04;05;-50;2;-51;5;05;-50;
  - MB2;1;1;-60;06;06;-65;2;-61;7;06;-65;
  - MB3;1;1;000;00;2;-30;02;05;-44;
- Empiezo ciclo 2 en instante t2

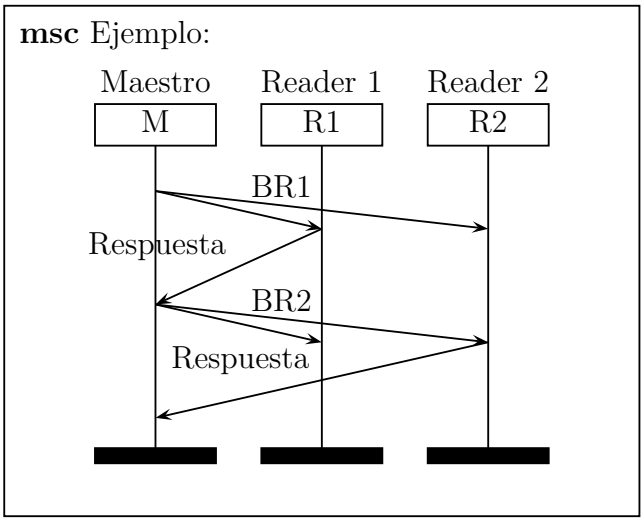


Figura 6.2: ***Cronograma de un ciclo del algoritmo***

6.2.2 Reader esclavo

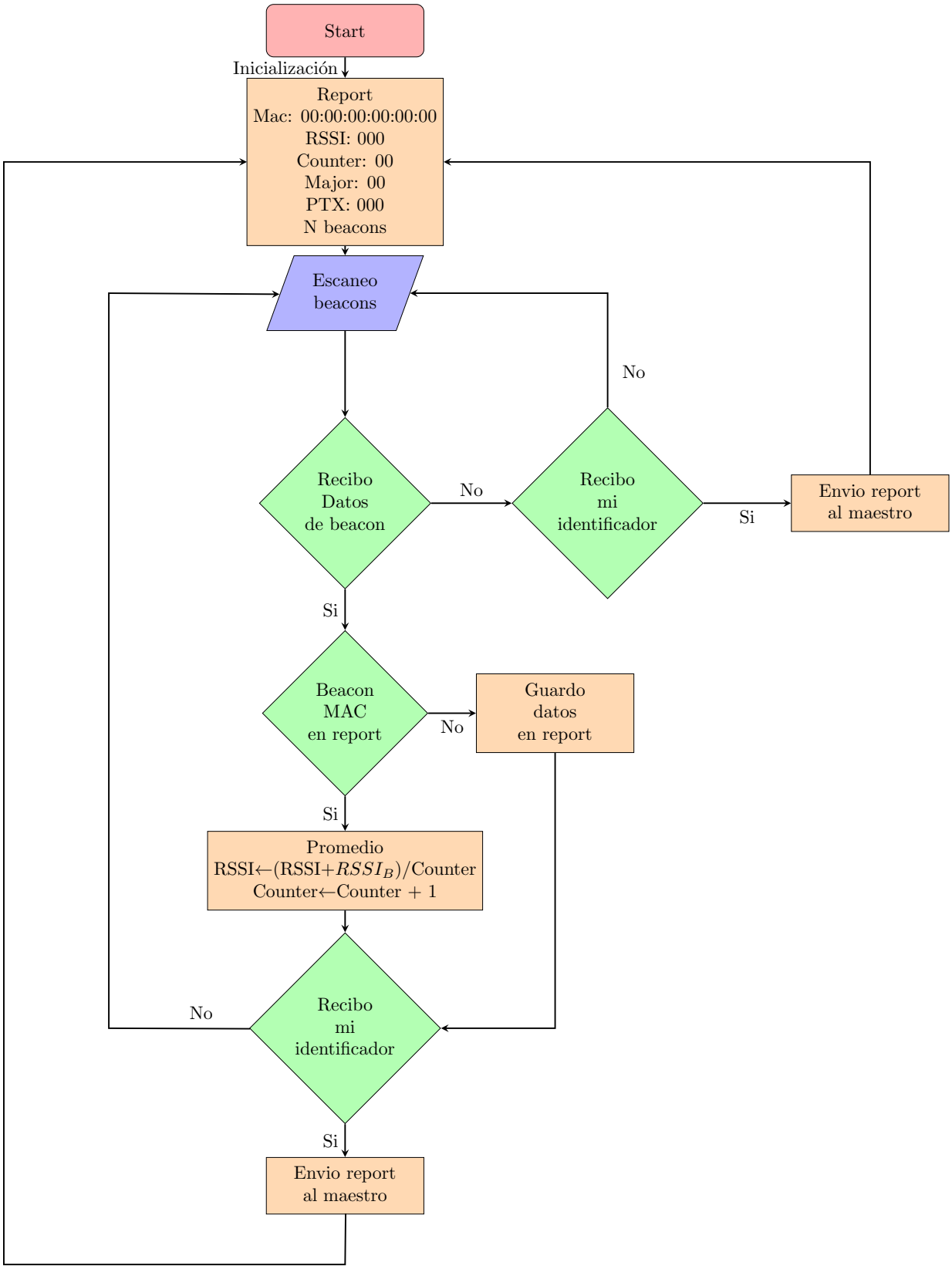


Figura 6.3: ***Algoritmo del dispositivo reader slave***

Se tiene una cola FIFO (matriz de B filas por 5 columnas) donde las columnas MAC;RSSI;contador;Major;PTX;

- Contador es el número de mensajes recibidos del beacon.
- La cola FIFO está inicializada a todo 0.

Cuando se recibe dato de un beacon se comprueba si está en la lista:

- No está → se añade MAC; RSSI;contador++;Major;PTX
- Está MAC → se promedia la RSSI y contador ++.

Cuando el Maestro le pide datos al Reader i en el slot i, le envía los datos de la cola FIFO y la resetea a 0.

6.2.3 Formato y tipo mensajes protocolo controlador maestro-readers

Para la comunicación entre el maestro y los readers esclavos se necesita configurar que tipo y que mensajes se tienen que intercambiar para su funcionamiento.

6.2.3.1 Maestro→Esclavo reader

- Tipo: String
- Formato:
  - Preámbulo (1B)
  - Id reader (2B)
  - Type code (1B)

Tipos de mensaje según campo Type:

- Type: eco request.
- Type: data request.

Un ejemplo, si quisiéramos preguntar al reader 1 el mensaje sería:

- Data request: S01B
- Echo request: S01E

6.2.3.2 Esclavo reader→Maestro

- Tipo: String
- Formato:
  - Data request: Data($27B \cdot N$), si recibo un data request la longitud del mensaje de datos dependerá de la cantidad configurada de beacons a capturar. Por ejemplo si configuramos que escanee 3 beacons, el tamaño será de $27B \cdot 3 = 81B$
  - Echo request: Data(4B), si recibo un echo request de devolverá al maestro la palabra echo.

6.3 Construcción del prototipo

Como hemos descrito en anteriores apartados, tenemos dos partes diferenciadas, el maestro que va a ser un pc con Linux y los readers esclavos que serán los que escaneen los beacons. A continuación describiré los componentes que están involucrados en cada elemento del prototipo. En resumen, estos serán los dispositivos usados:

- Maestro: PC Linux conectado por Ethernet a la red donde esta el servidor de localización, donde se estará ejecutando el script que ejecuta el algoritmo 6.1.
- Tres reader esclavos, con las siguientes partes:
  - Raspberry Pi Zero, que ejecuta el algoritmo 6.3.
  - Módulo USB para RS485
  - Transformador de corriente CUI INC
- Fuente de alimentación industrial que da alimentación a los readers.
- Beacons de la marca iBKS.

6.3.1 Circuito Reader Esclavo

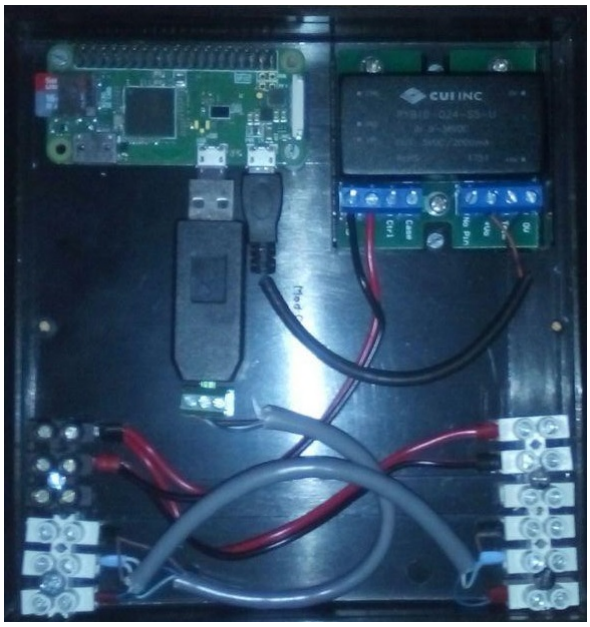
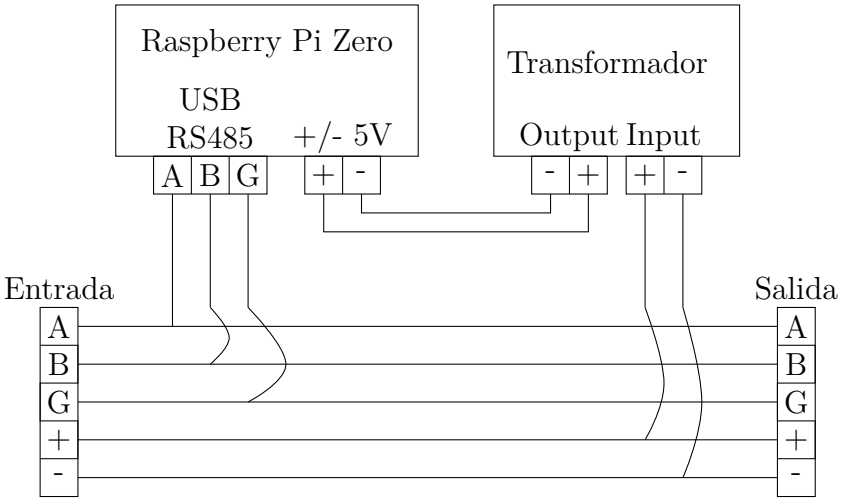


Figura 6.4: ***Prototipo Reader esclavo***

A continuación podremos ver en mas profundidad los elementos y conexiones que componen el reader esclavo



6.3.1.1 Elementos

- Al transformador le puede llegar una corriente de hasta 36VDC, que transformará a 5V para alimentar la Raspberry Pi Zero.
- Raspberry Pi Zero que tiene linux y actuará de escaner, ejecutando el algoritmo visto anteriormente.
- Dos puertos, uno de entrada y otro de salida, se pueden usar indistintamente los dos. Estos puertos tienen:
  - Conexiones del bus RS485, con los pines A,B y GND.
  - Alimentación que puede variar de 9-36VDC.

6.3.2 Bus RS485

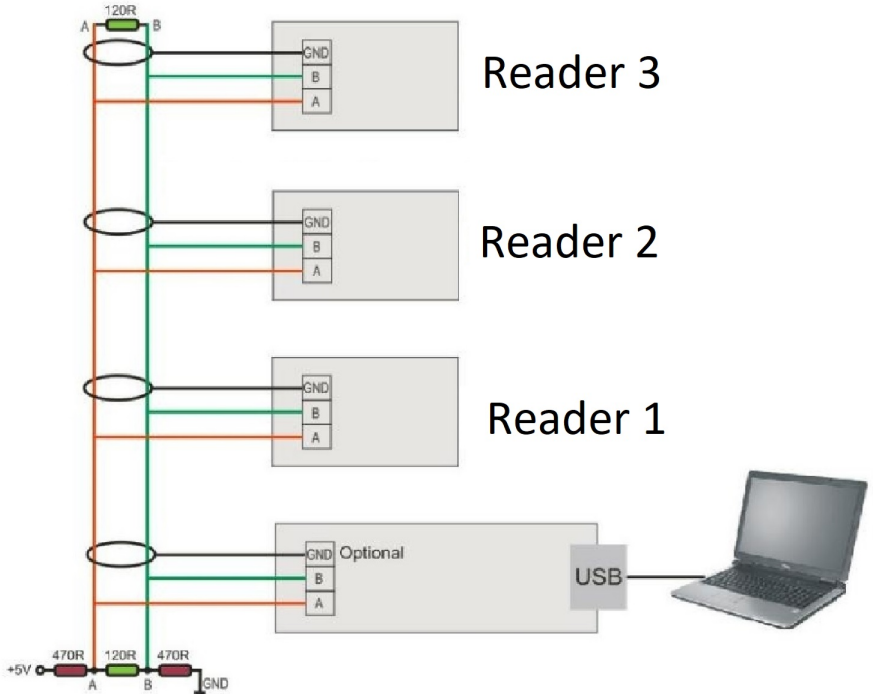


Figura 6.5: ***Bus RS485***

Las partes que componen este bus son las siguientes:

- Maestro con modulo USB RS485
- Readers
- Resistencias de terminación.

Necesitamos esas resistencias de terminación si el bus tiene mucha longitud, para evitar señales perturbadoras entremezcladas y obligar al nivel de pausa en el sistema en los tiempos donde no este activo ningún transmisor de datos.

6.3.3 Beacons iBKS

Figura 6.6: ***Dispositivo beacon iBks***

Estos beacon los usaremos para definir los usuarios a capturar, en las zonas donde se colocan los readers. Estos dispositivos tienen las siguientes características:

- Mac conocida de cada uno de ellos.
- Datos en formato iBeacon, con:
  - Potencia de transmisión conocida a 1 metro del escaner.
  - Campo Mayor para dar mas información.

Capitulo 7

Conclusión

El objetivo de este proyecto es estudiar la viabilidad de un sistema basado en readers bluetooth low energy para la localización de personas u objetos en entornos cerrados donde la cobertura de sistemas tradicionales como el GPS no alcanza. Para cumplir dicho objetivo después de haber presentado las diferentes tecnologías involucradas y sus características, se han seguido los siguientes pasos:

- Los procedimientos de Scanning y Advertising, donde se explica su funcionamiento y configuración. Analizando los diferentes tipos de Advertising y los modos Scanning, se han seleccionado los tipos adecuados para la implementación de un sistema Maestro-Esclavo.
- La configuración del Scanner y del Advertiser, donde se ha visto la facilidad de su configuración en los modos Scanner y Advertiser.
- Medidas de tiempo de descubrimiento, donde se han probado varias configuraciones. También medidas de cobertura en un entorno real para comprobar su efectividad con obstáculos así como medidas de volumen de beacons a detectar para detectar umbrales de funcionamiento.
- Construcción de los readers, teniendo en cuenta el tipo de bus, alimentación y protocolo Maestro-Esclavo para la recepción y procesado de los datos, así como el envío por parte del Maestro al servidor de localización.

Viendo los resultados obtenidos, podemos decir que la tecnología Bluetooth low energy es valida para su implementación como sistema de localización en entornos cerrados donde se requiere un cableado mínimo. Debido a los materiales con los que contaba he propuesto un escenario donde hay un maestro y tres readers, pero este escenario se podría ampliar con mas readers, incluso con mas buses controlados por el maestro, para poder tener cubierto con mas cobertura el lugar de instalación.

Desde el punto de vista económico, tenemos ante nosotros una tecnología de muy bajo consumo, con muchos dispositivo BLE que se podrían usar como beacons para poder localizarse, como pulseras, tarjetas que serían llevadas por los usuarios o colocadas en objetos. Como punto negativo, debido a su cobertura las zonas de cobertura por los readers estarían limitadas a habitaciones o pasillos, pero habrían que hacerse mas pruebas para determinar su correcta colocación, en un entorno realista.

Finalmente decir que existen otras tecnologías para la localización en interiores como el WIFI, pero este trabajo propone una alternativa con un coste de infraestructura y equipos mas económica, dado a su bajo consumo, cableado reducido y equipos fáciles de instalar.

Bibliografia

[1] Bluetooth SIG, https://www.bluetooth.com

[2] What is iBeacon, a guide to beacons, http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/

[3] Especificación RS485, http://en.wikipedia.org/wiki/RS485

[4] Group, Bluetooth Special Interest, Bluetooth Core Specification v4.0

[5] Bluetooth stack for linux, Bluez, http://www.bluez.org/

Anexos

Anexo A

Componentes usados

A.1 Raspberry pi Zero

Figura A.1: ***Raspberry pi Zero***

Especificaciones técnicas:

- Hardware:
  - 1GHz single-core CPU
  - 512MB RAM
  - Mini HDMI port
  - Micro USB OTG port
  - Micro USB power
  - Bluetooth LE 4.0
  - HAT-compatible 40-pin header
  - Composite video and reset headers
  - CSI camera connector (v1.3 only)
- Software:
  - Linux Raspbian

A.2 Módulo bluetooth Sena

Figura A.2: ***Módulo bluetooth sena***

Standards	Bluetooth 4.0+EDR 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	Basic 1Mbps: -90 dBm EDR 2Mbps: -88 dBm EDR 3Mbps: -86 dBm
Antenna Connector	RP-SMA
Antenna Gain	Default Stub Antenna: 1 dBi Optional Dipole Antennas: 3 dBi & 5 dBi Optional Patch Antenna: 9 dBi
Working Distance (In Open Field)	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 * working distance can vary depending on install environment
Bluetooth Stack Software	BlueSoleil
Bluetooth Profiles	DUN, FAX, SPP, HID, FTP, OPP, A2DP, AVRCP, HSP, HFP, PAN, BPP
Computer OS Support	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	-20 ~ +70℃
Storage Temperature	-40 ~ +85℃
Humidity	90% Non-condensing
Regulatory Approvals	FCC, CE, TELEC, KCC, Bluetooth SIG
Warranty	1 year limited warranty

Figura A.3: ***Especificaciones técnicas del módulo Sena***

A.3 Beacon iBKS

Figura A.4: ***Dispositivo beacon iBks***

Dimensions	Ø52.6 x 11.3 mm	**Case material**	ABS
Weight	24g	**Case finish**	Matte white
Core	Nordic nRF51822	**Fixing method**	Double side sticker
Radio Protocol	Bluetooth® Low Energy	**Operating Temperature**	-25 to +60°C
Distance Range	Up to 50m	**Storage Temperature**	0 to +35°C
Battery	Coin Cell CR2477 3V – 1000mAh	**Beacon Protocols**	iBeacon Eddystone: UID, URL, TLM & EID
Optional Sensors	Hall Accelerometer	**Firmware Update**	OTA (Over The Air)
Idle Current Consumption	2.4µA	**Certifications**	CE, FCC, Anatel

Figura A.5: ***Especificaciones técnicas del beacon iBks***

En la siguiente tabla se muestra la potencia de recepción en comparación con la distancia y la potencia de transmisión configurable.

Distance (m)	TX Power (dBm)							
	-30	-20	-16	-12	-8	-4	0	+4
0	-63	-47	-42	-40	-40	-34	-27	-23
1	-92	-79	-71	-68	-66	-58	-59	-57
3	-99	-84	-82	-74	-73	-70	-67	-62
5	-101	-90	-88	-84	-82	-74	-71	-67
10		-95	-95	-89	-87	-81	-79	-75
15		-98	-97	-95	-92	-90	-84	-82
20		-102	-102	-97	-94	-92	-87	-85
30				-102	-96	-93	-92	-88
40					-99	-99	-94	-91
50					-102	-102	-100	-99

Figura A.6: ***Características de potencia***

A.4 Transformación de corriente CUI INC

Figura A.7: ***Transformador PYB10-Q24-S5-U***

Características:

- Up to 10 W isolated output
- Compact chassis mount package
- 4:1 input range
- Over voltage and short circuit protections
- -40 to +85°C temperature range
- Efficiency up to 88
- Input Voltage 9-36Vdc
- Power 10W
- Output Voltage 5Vdc
- Output Current 2A

A.5 Adaptador USB RS485

Figura A.8: ***Adaptador USB RS485***

Características:

- CP2104, SN75176, FUSE + TVS
- Protección contra sobretensiones TVS
- No se necesita una fuente de alimentación externa o un cable de datos