ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Universidad Politécnica de Cartagena

Trabajo Fin de Grado

GRADO EN INGENIERÍA TELEMÁTICA

Diseño y desarrollo de un servicio de mensajes SMS en una red IoT

AUTORA: Encarna Mª Buitrago Dato

DIRECTOR: Alejandro Martínez Sala

Diciembre / 2020

Autor	Encarna María Buitrago Dato
E-mail del autor	ebuitragodato@gmail.com
Director	Alejandro S. Martínez Sala
E-mail del director	alejandros.martinez@upct.es
Título del TFM	Diseño y desarrollo de un servicio de mensajes SMS es una red IoT
Resumen	Diseño de un escenario para la solución de falta de cobertura telefónica 3G o 4G mediante el uso de módulos LoPy, móviles emisores de tramas beacon y un servidor receptor y transmisor de mensajes SMS.
Titulación	Grado en Ingeniería de Telemática
Departamento	Tecnología de la Información y las Comunicaciones
Fecha de presentación	Diciembre - 2020

Índice

Índice de figuras

Capítulo 1. Introducción

1.1 Introducción

El caso de uso del TFG es un escenario donde los dispositivos móviles no tienen cobertura de telefonía 3G o 4G y se quiere poder enviarle mensajes de texto tipo SMS desde un servidor. Un escenario de uso podría ser un parque natural o dentro de una zona subterránea donde no hay cobertura de telefonía móvil. Se requiere una red IoT formada por varios nodos que en zonas delimitadas permitan la asociación de un dispositivo móvil de un usuario y el intercambio de mensajes de texto con un servidor. Las tecnologías IoT utilizadas son LoRa y Bluetooth Low Energy (BLE).

-LoRa es una tecnología IoT de comunicación punto a punto de larga distancia y de bajo consumo.

-Bluetooth low Energy permite la comunicación de corto alcance.

La solución adoptada para este problema de falta de cobertura ha sido la de crear un escenario, mediante tres nodos LoPy un PC que hace de server y un móvil, que resuelva el inconveniente de la ausencia de cobertura en telefonía 3G o 4G.

Para ellos se han emitido señales beacon desde un móvil, para que dos de los tres módulos LoPy puedan recibir dicha señal con la intención de generar un mensaje de detección que sea enviado mediante LoRa al nodo Master (módulo LoPy). Siendo este último el que reenvíe el mensaje hacia el server. Una vez aquí, el server sería el encargado de generar un mensaje tipo SMS en respuesta a la señal beacon detectada por los nodos LoPy y enviarla de vuelta por el mismo camino de llegada.

Esto sería una posible solución ante la ausencia de cobertura ya que se podría seguir enviando datos evitando la pérdida de estos.

Este dibujo mostraría un esquema de la arquitectura empleada. En este caso el dispositivo móvil estaría en cobertura con el nodo B, no siendo detectado por el nodo C.

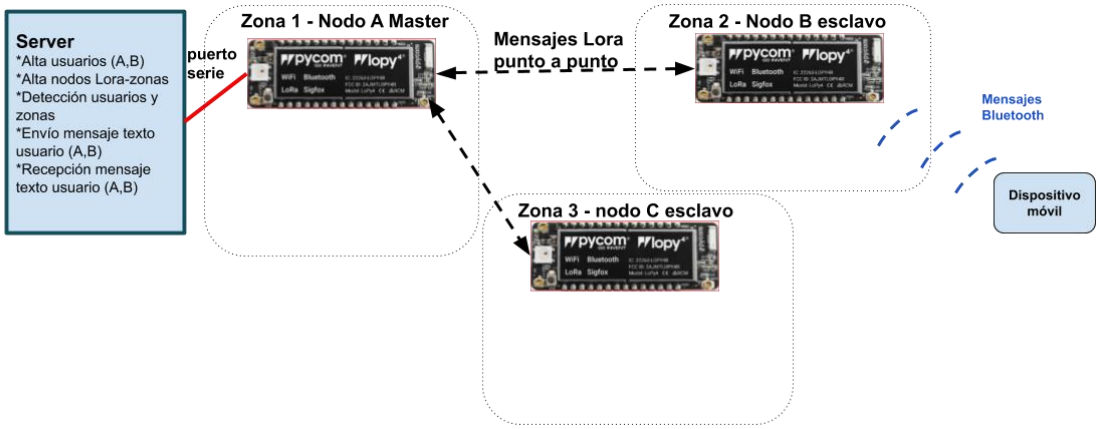


Figura 1. Esquema de la solución. Usuario en cobertura nodo B

Por el contrario, en este esquema sería el nodo C el que estaría en cobertura con el móvil, no siendo detectado por el nodo B.

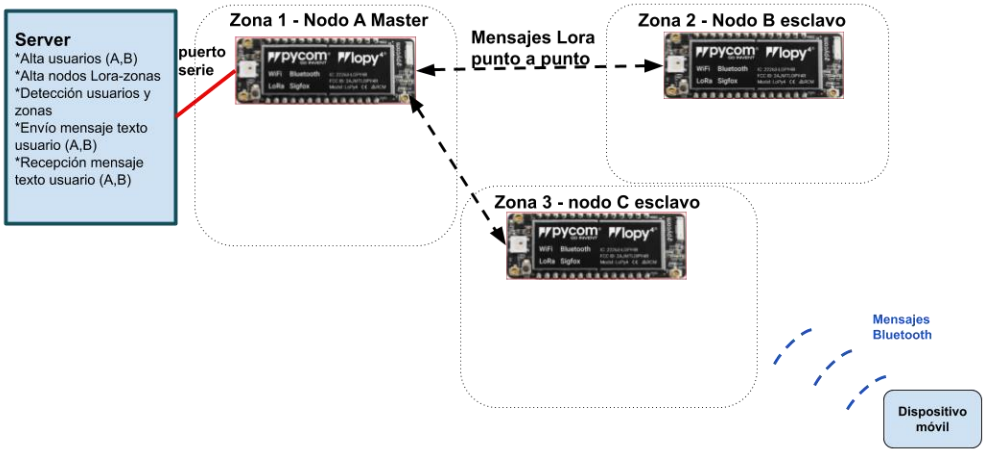


Figura 2. Esquema de la solución. Usuario en cobertura nodo C

1.2 Objetivos y organización del TFG

El TFG es una prueba de concepto y primer paso para diseñar e implementar un prototipo IoT de envío de mensajes tipo SMS. Se espera que este prototipo se mejore y amplíe en sucesivos TFG. Los objetivos que se han marcado han sido:

1) Diseñar un prototipo de infraestructura de red IoT básica con topología en estrella formada por un nodo maestro y dos nodos esclavos.
2) Diseñar el protocolo de comunicación del servicio de transmisión de mensajes de texto, desde el Server hasta el dispositivo móvil emulado.
3) Implementación de Servidor y los nodos IoT maestro y esclavos.
4) Pruebas básicas de validación del prototipo. Asegurar que se detecta e informa al servidor de la presencia de un dispositivo móvil emulado y que el server le envía un mensaje de texto que recibe dicho dispositivo móvil.

1.3 Herramientas, aplicaciones y entornos de desarrollo

Para el desarrollo de este proyecto se han usado diferentes herramientas, lenguajes y programas que se describen de forma resumida a continuación.

- **Atom:** Atom es un editor de texto multiplataforma creado por GitHub el cual podemos utilizar para programar nuestros scripts en Python. Atom soporta la instalación de paquetes que añaden funcionalidades extra al programa. Además, la instalación de estos paquetes es muy sencilla ya que el mismo programa incluye un buscador que nos facilita esta tarea.

  Este será nuestro entorno de desarrollo junto con la librería pymakr 1.4.18. Y usaremos el lenguaje de Micro Python para configurar los dispositivos LoPy. Para más información https://atom.io/

- **NotePad++:** Notepad++ es un editor de texto y de código fuente libre con soporte para varios lenguajes de programación. Con soporte nativo para Microsoft Windows. Se parece al Bloc de notas en cuanto al hecho de que puede editar texto sin formato y de forma simple. Nosotros lo hemos utilizado para editar los scripts de Python. Más información https://notepad-plus-plus.org/

- **Python:** Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Se usará la versión 3.7.9 para programar el server.

- **Micro Python:** Micro Python es una implementación del lenguaje de programación Python 3, escrita en C, optimizada para poder ejecutarse en un microcontrolador. Micro Python es un compilador completo del lenguaje Python y un motor e intérprete en tiempo de ejecución, que funciona en el hardware del microcontrolador.

- **232Analyer**: 232Analyer es un programa que nos ha permitido controlar, monitorear y revisar errores en la comunicación serie. Este programa permite ver todos los datos de la comunicación entre dos dispositivos que utilicen el puerto serie. Mas información http://www.commfront.com/

- **Beacon Simulator:** Esta aplicación transforma su dispositivo Android en un anunciante y transmisor de baliza BLE virtual. Puede crear su propia colección de configuraciones de balizas y utilizarlas en cualquier momento y en cualquier lugar para emular una baliza física. Esta app la hemos usado para enviar las señales del móvil 2 y escanear las señales procedentes del LoPy.

- **BeaconScope:** Es una aplicación que trasmite más de 10 señales de advertisment simultáneamente (en dispositivos compatibles). Esta se ha usado para transmitir señales del móvil 1.

1.4 Estructura del contenido del proyecto

El proyecto a presentar se compone de varios capítulos, exactamente 5. Y son los siguientes:

- **Capítulo 1. Introducción.**
  En este capítulo se realiza una pequeña introducción sobre el proyecto, explicando brevemente su funcionalidad con esquemas y también se explican las herramientas utilizadas para su desarrollo.

- **Capítulo 2. Tecnología Lora y Bluetooth Low Energy**.
  Aquí se hará una explicación teórica sobre las tecnologías usadas en el proyecto, que son Lora y BLE.

- **Capítulo 3. Especificación de los protocolos entre los nodos de la red.**
  Se explicará las máquinas de estado asociadas al sistema. Se hará una breve descripción de la funcionalidad de este. Así como el tipo y formato de los mensajes que existen en la comunicación y, por último, se mostrarán unos diagramas de bloques de la arquitectura.

- **Capítulo 4. Arquitectura del servicio de mensajes de texto en red IoT.**
  En este capítulo se van a explicar el funcionamiento de los módulos LoPy y del server mediante diagramas de flujo. Y se hará una explicación sobre el desarrollo software.

- **Capítulo 5. Pruebas y resultados.**
  En este último capítulo se mostrará el resultado de las pruebas llevadas a cabo una vez terminado del desarrollo del proyecto y probado que todo funciona correctamente.

- **Capítulo 6. Conclusiones y trabajos futuros.**

Capítulo 2. Tecnología LoRa y Bluetooth Low Energy

2.1 Tecnología LoRa

La tecnología LoRa de comunicación permite el envío y recepción de información punto-a-punto. LoRa se caracteriza por ser un protocolo de comunicación para largas distancias a baja potencia. Para ello emplea la técnica de espectro ensanchado (SSM), donde la señal a mandar utiliza más ancho de banda que el necesario teóricamente pero que permite una recepción de múltiples señales a la vez que tengan distinta velocidad.

Establecer una comunicación punto-a-punto con LoRa es relativamente sencillo. Para distancias muy cortas (menos 1 KM) no suele haber mucho problema. El problema surge cuando se quiere establecer comunicación a grandes distancias. Para ello es necesario el uso de antenas con mayor ganancia, una visibilidad adecuada entre ellas sin obstáculos de por medio, una correcta configuración del canal, velocidad, etc.

Con Lora se pueden llegar a alcanzar distancias de decenas de KM con poca energía. Pero para conseguir comunicaciones satisfactorias es recomendable y en ocasiones necesario cumplir ciertos requisitos:

1) **Línea de visión**: desde los nodos hasta las pasarelas debe haber una línea de visión libre o con pocos obstáculos. Si entre puntos hay una colina, es casi imposible que se consiga comunicar dichos puntos.
2) **Potencia**: es recomendable usar la máxima potencia legal disponible, +20 dBm.
3) **Alto spreading factor**: un valor alto de SF (10-12) permite que potencialmente lleguen más mensajes al destino a coste de reducir la velocidad.

Esta tecnología se encuentra en el nivel 1 de OSI, nivel físico.

2.1.1 Capa física LoRa

SSM codifica la señal base con una secuencia de frecuencia variable, que deliberadamente extiende la señal en banda base a mayor ancho de banda, reduciendo así el consumo de energía y aumentando la resistencia a las interferencias electromagnéticas. Esto permite que LoRa pueda coexistir en entornos con señales en banda estrecha (entornos rurales), ya que sólo les aporta un pequeño incremento en el ruido.

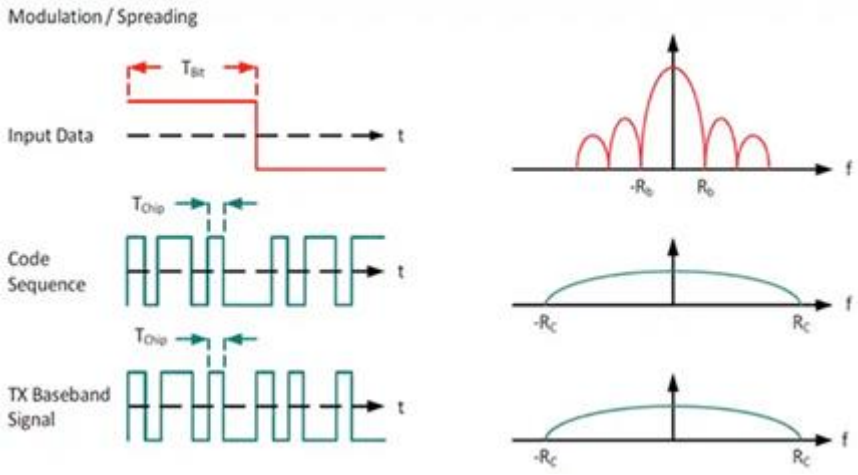


Figura 3. Modulación

En la imagen se puede apreciar como un sistema de espectro ensanchado multiplica los datos de entrada en una secuencia de código mucho más rápida que propaga la señal de ancho de banda.

El protocolo LoRa se basa en una variante de SSM llamada modulación CHIRP de espectro ensanchado. CSS codifica los datos con un "pitido" o chirp, que es esencialmente una señal sinusoidal de frecuencia modulada (variable) en banda ancha que aumenta o disminuye con el tiempo.

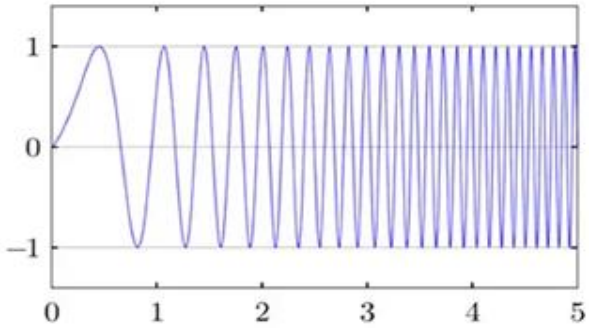


Figura 4. Modulación chirp

Los parámetros de comunicación LoRa son:

1. **"Spreading factor" (SF) o Factor de ensanchamiento**: define el número de bits usados para codificar un símbolo, es decir, nos permite escoger el ensanchamiento deseado para la señal a transmitir. A mayor SF, menor velocidad de transferencia tendremos, pero mayor inmunidad a interferencias. LoRa permite 6 valores de SF (SF7-SF12), siendo SF12 el máximo ensanchamiento y por tanto mayor robustez.
2. **"Coding rate" (CR):** indica la forma de codificar para corrección de errores. Es decir, según la técnica especificada, añade símbolos de control para saber si los datos son correctos o no e incluso poder determinar los valores correctos. Los valores permitidos son 4/5, 4/6, 4/7 y 4/8 lo que quiere decir que codifica 4 bits en 5, 6, 7 u 8 bits de datos. Este parámetro también afecta a la tasa de bit:

$$Tb = SF \frac{\left[\frac{4}{4+CR}\right]}{\left[\frac{2^{SF}}{BW}\right]}$$

3. **"Bandwidth" (BW) o ancho de banda**: LoRa permite, como norma general, tres anchos de banda distintos para la comunicación (125,250 y 500 kHz). A mayor ancho de banda, menor será la robustez de la comunicación, es decir, mayor será la tasa de bit y por lo tanto disminuirá la tasa de éxito en recepción.

4. **Tasa de bit (Tb)**: Este valor dependerá de la configuración escogida, se puede calcular de la siguiente manera:

$$Tb = SF \frac{BW}{2^{SF}}$$

5. **Periodo de símbolo (Ts)**: La señal LoRa transmitida estará compuesta por un número de símbolos y el periodo de cada uno dependerá del ancho de banda y el SF escogidos:

$$Ts = \frac{2^{SF}}{BW}$$

6. **Frecuencia de transmisión**: LoRa permite distintas bandas de frecuencia según la legislación de cada país. Además, dispone de 10 canales frecuenciales para cada uno de los enlaces (EU).

    - 169 MHz
    - 433 MHz
    - 868 MHz
    - 915 MHz

	Europa	América del Norte
Banda de frecuencia	867-869 MHz	902-928 MHz
Canales	10	64 + 8 + 8
Canal banda ancha ascendente	125/250 kHz	125/500 kHz
Canal banda ancha descendente	125 kHz	500 kHz
TX encendido	+14 dBm	+20 dBm típ (+30 dBm permitidos)
TX desconectar	+14 dBm	+27 dBm
SF Up	7-12	7-10
Velocidad de datos	250 bps - 50 kbps	980 bps - 21.9 kbps
Link Budget Up	155 dB	154 dB
Link Budget Dn	155 dB	157 dB

Figura 5. Especificaciones Lora para Eurpa y América

2.1.2 Transmisión y recepción de mensajes LoRa en el LoPy

Para poder ver la aplicación de los conceptos anteriormente explicados se va a mostrar trozos de código donde se aplica todo lo descripto. Para empezar, se muestra la forma de configurar el protocolo LoRa en un LoPy, definiendo los parámetros anteriores que son necesarios:

```
lora = LoRa(mode=LoRa.LORA,
            frequency=868100000,
            tx_power=14,
            bandwidth=LoRa.BW_125KHZ,
            sf=12,
            coding_rate=LoRa.CODING_4_5)

lora.preamble(8)
s = socket.socket(socket.AF_LORA, socket.SOCK_RAW)
```

Por otro lado, debemos crear el mensaje que se quiera para ser enviado mediante Python:

```
mensaje_send=struct.pack("BBB16s", IDtx, ID_L2, tipo, str(mensaje)
```

Para ello se usa struct.pack que lo que hace es devolver una cadena que contiene los valores empaquetados de acuerdo con el formato dado.

El primer campo indica el formato de los caracteres que se quiere enviar. En este caso los 3 siguientes campos son de tipo unsigned char que se representan con una B y son de tamaño de 1 byte. Y el último campo es de tipo char y se le indica que tiene una longitud de 16 bytes.

Una vez generado el mensaje se procede al envío de este mediante LoRa:

```
s.send(mensaje_send)
```

Ya enviado, el receptor debe escuchar para ver si recibe datos.

```
rxLopy=s.recv(255)
```

Una vez que se ejecuta este comando, se comprueba si hay datos, es decir, si la longitud de rxLopy es mayor que cero, en caso de ser que si es porque se ha recibido datos y si es que no será que no se han recibido datos.

2.2 Tecnología Bluetooth Low Energy

2.2.1 BLE o Bluetooth low energy

Bluetooth es el nuevo estándar Bluetooth 4.0 (versiones 4.1, 4.2) que nos permite desplegar redes inalámbricas de área personal (PAN); es decir, vincular sin cables varios dispositivos que están dentro de un radio cercano. Este tipo de conexiones penalizan mucho la

autonomía de los terminales y, prácticamente, podían agotar la batería de un *smartphone*. Por ello, se creó el Bluetooth Low Energy, que es una variante del Bluetooth y forma parte de la especificación de Bluetooth 4.0

Bluetooth LE es un sistema inalámbrico que sirve para conectar varios aparatos electrónicos e intercambiar información entre ellos vía bluetooth. Se fundamenta en la reducción del consumo y, por tanto, en minimizar la potencia de transmisión de la señal radio utilizada y el radio de cobertura. La sustancial reducción del consumo, y la consiguiente extensión de la autonomía, no solo redunda en una mejora para los *smartphones* o *tablets,* sino que también es importante para el desarrollo de sensores. Permite emitir durante meses sin cambiar la batería del emisor o del receptor.

Un ejemplo de esto podría ser un sensor alimentado por una "pila de botón" y con soporte Bluetooth LE. Este podría estar encendido durante meses o, incluso, llegar al año de funcionamiento sin necesidad de reemplazar la batería.

BLE o Bluetooth Low Energy es una variante de la tecnología Bluetooth. Se diferencia del Bluetooth común en el bajo consumo de batería básicamente y es una buena opción para aplicaciones que solo necesitan intercambiar datos de vez en cuando.

Bluetooth de baja energía tiene 40 canales físicos en la banda ISM de 2,4 GHz, cada uno separado por 2 MHz. Bluetooth define dos tipos de transmisiones: transmisiones de datos y de publicidad (advertising). Como tal, 3 de estos 40 canales están dedicados a publicidad y 37 a datos.

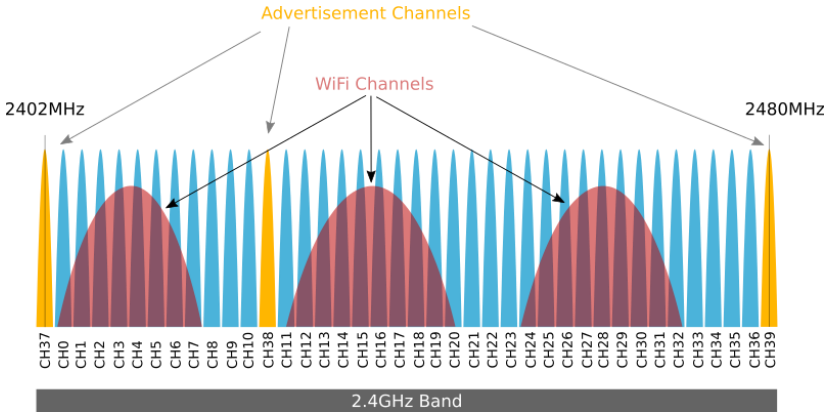


Figura 6. Canales advertisement

Uso del canal de publicidad:

Descubrimiento de dispositivos
Establecimiento de la conexión
Transmisiones de radiodifusión

Uso del canal de datos: para la comunicación bidireccional entre los dispositivos conectados. En BLE se usa la técnica del salto en frecuencia por lo que se usan una secuencia aleatoria de canales de datos entre los dispositivos enlazados.

2.2.2 Tipos de paquetes BLE según el estándar BT BLE 4.X

Se distinguen dos tipos básicos de mensajes: Advertising channel PDU y Data Channel PDU. Los campos preámbulo (1B), access address (4B), header(2B) - payload (variable) y CRC (3B) son comunes para ambos mensajes. Para cada tipo básico (adv o data), después hay subtipos donde cambia la estructura y contenido del payload.

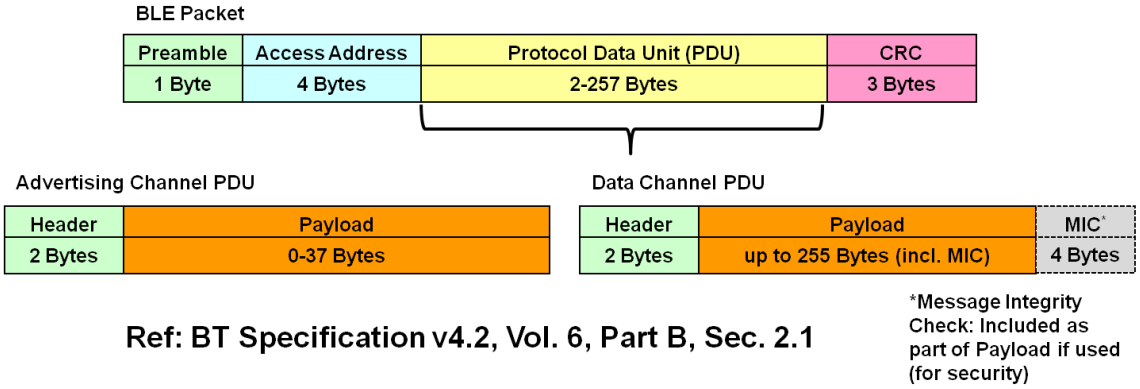


Figura 7. Tipo paquetes estándar BT BLE 4.X

En este proyecto se usan los mensajes de advertising de BLE para la comunicación a corta distancia entre un nodo Lopy y un dispositivo móvil del usuario. En el payload del mensaje de advertising se codifican los campos para un protocolo de envío de mensajes SMS y el propio mensaje de texto limitado a una corta cadena de caracteres ASCII.

2.2.3 Proceso de Advertising

Un periférico emite su Advertising Data payload (también denominado beacons) a intervalos regulares configurables para que un dispositivo sea detectado por otro. Cada vez que el intervalo pasa, el periférico emite su Advertising Data payload.

Intervalos altos, permiten ahorrar batería, mientras que intervalos cortos, permiten ser más reactivos.

Si un dispositivo central necesita más datos y el periférico lo permite, puede solicitar adicionalmente el scan response payload, y el este contestara con la información adicional.

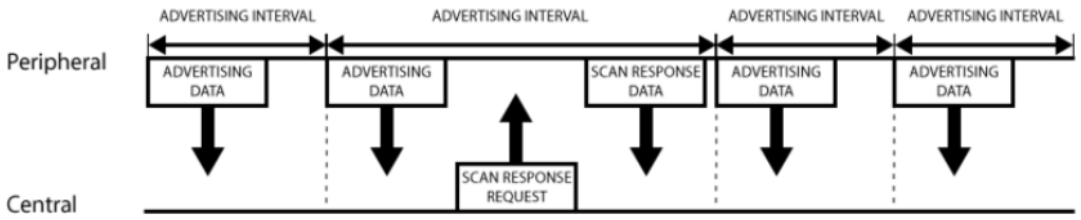


Figura 8. Proceso de advertising

Hay cuatro tipos de advertising que se pueden realizar: general, dirigida, no relacionada y descubrible. Cada vez que un dispositivo publicitario transmite el mismo paquete en cada uno de los tres canales de publicidad. Esta secuencia de eventos se llama advertising evento.

ADV Tipo de paquete → Tipo de publicidad soportada
ADV_IND → Publicidad no dirigida de conexión
ADV_DIRECT_IND → Publicidad Dirigida Conectada
ADV_NONCONN_IND Publicidad no conectable no dirigida
ADV_SCAN_IND → Publicidad escaneable no dirigida

- Paquete ADV_IND: Este tipo de paquete soporta publicidad conectable y no dirigida y se utiliza cuando un dispositivo esclavo/periférico se enciende por primera vez (es decir, nunca se ha conectado con un maestro) y busca conectarse con cualquier nodo (es promiscuo). Esto representa típicamente un estado de fábrica por defecto.

2.2.4 Modo Broadcaster o Beacon

El modo beacon BLE es un transmisor hardware que transmite periódicamente mensajes de advertising BLE permitiendo a smartphones, tablets y otros dispositivos electrónicos, realizar acciones específicas cuando se encuentren cerca. La información transmitida incluye un UID o universally unique identifier y varios bytes que pueden ser utilizados para determinar la localización física del dispositivo**.**

Los beacons pueden ser dispositivos muy sencillos (formados por un microcontrolador con una batería) o un smartphone puede actuar como beacon o broadcaster.

Todos los beacons BLE poseen las siguientes características configurables independientemente del fabricante o el protocolo utilizado:

- **Tx power**: es la potencia con la que los beacons transmiten una señal que viaja por el aire y disminuye con la distancia. Lógicamente, cuanto más grande sea esta potencia, más batería consumirá el beacon.
- **Intervalo de emisión**: este valor define la frecuencia con la que emite un beacon. A diferencia del tx power, cuanto más bajo sea este intervalo de emisión, más consumirá el beacon, pero es importante pensar un poco antes de establecer este intervalo muy alto, ya que el dispositivo receptor tendría más dificultades para detectar el beacon en este caso.

En el payload del mensaje de advertising se pueden codificar campos y distinguir protocolos iBeacon de Apple y Eddystone de Google. Se habla del formato iBeacon o formato Eddystone pero todos se basan en el estándar BLE y en los mensajes de advertising.

- iBeacon (Apple): fue el protocolo que introdujo la tecnología BLE mundialmente y define 3 parámetros:
    - UUID: identifica un grupo.
    - Major: identifica un subgrupo de beacons dentro de un grupo más grande.
    - Minor: identifica un beacon específico.
- Eddystone: es un proyecto de código abierto desarrollado por Google. Un beacon configurado con este formato 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 este proyecto, usaremos el protocolo Eddystone. Se usarán dos apps para Android que harán de emisores beacon y de escáner. Exactamente usaremos el tipo de paquete Eddystone-UID con identificador de beacon aleatorio asignado por la app.

2.2.5 Escaneo

El escaneo es una tarea importante que completa el proceso de descubrimiento. Se requiere para poder recibir paquetes publicitarios. Hay dos tipos de escaneo: pasivo y activo.

- Escaneo pasivo:

En este tipo de exploración, el escáner simplemente escucha los paquetes publicitarios. El anunciante nunca se da cuenta de que se recibieron paquetes:

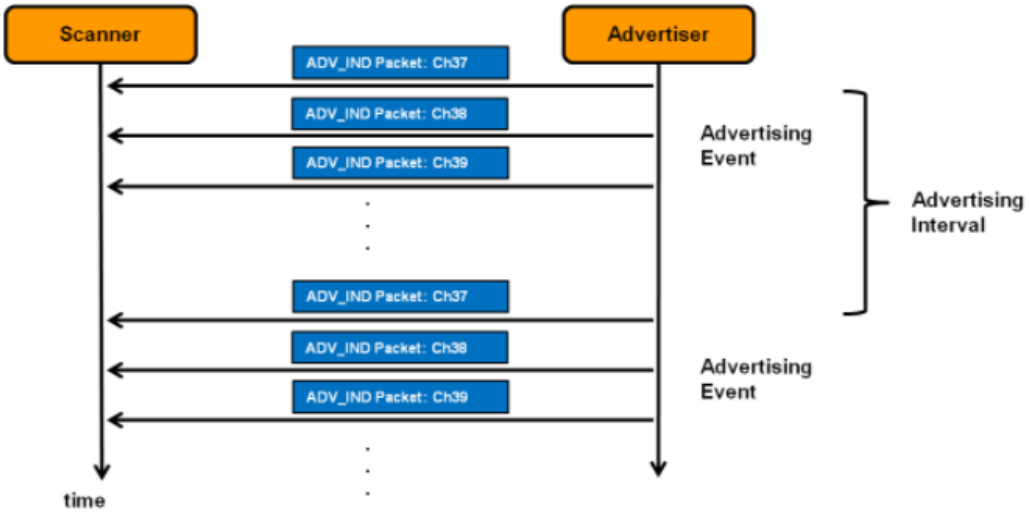


Figura 9. Escaneo pasivo

- Escaneo activo:

Este tipo de exploración se utiliza normalmente cuando el dispositivo central potencial desea obtener más información de la que se puede proporcionar en un paquete ADV_IND, antes de tomar la decisión de conectarse a él.

En un intervalo de publicidad, el escáner emite un paquete SCAN_REQ. El anunciante responde con más información en un paquete SCAN_RSP.

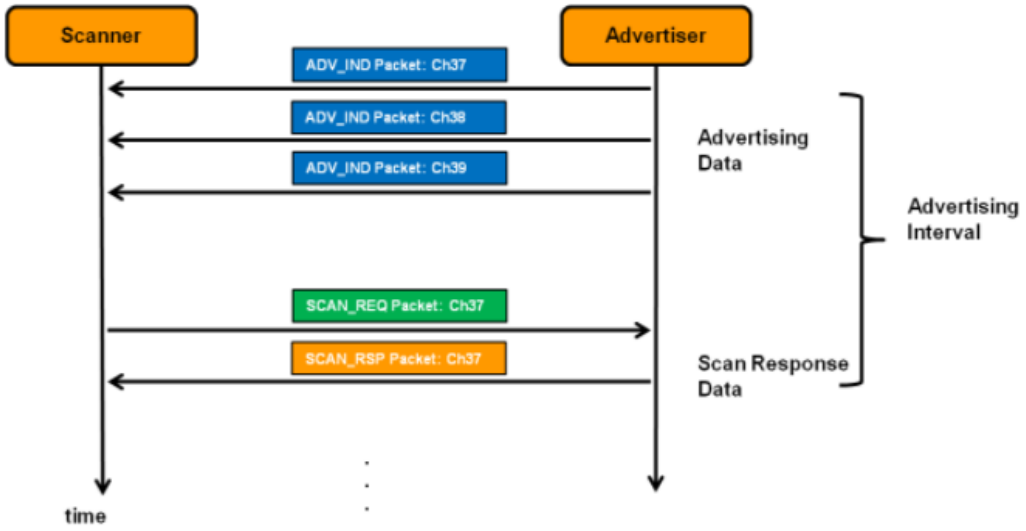


Figura 10. Escaneo activo

2.2.6 Transmisión y escaneo de advertising en el LoPy

Una vez finalizada la introducción del BLE se pasa a mostrar el código donde se usan dichos conceptos.

Primero se comienza configurando el bluetooth

```
print("Configuramos BLE...")
bt=Bluetooth()
bt.start_scan(-1)
```

El método start_scan comienza a realizar un escaneo escuchando dispositivos BLE que envían anuncios. El argumento es:

- Un timeout que especifica la cantidad de tiempo en segundos para buscar anuncios, no puede ser cero. Si el tiempo de espera es> 0, la radio BLE escuchará anuncios hasta que transcurra el valor especificado en segundos. Si el tiempo de espera es <0, entonces no hay tiempo de espera en absoluto, y se debe llamar a stop_scan () para cancelar el proceso de escaneo.

Luego se procederá a recuperar los mensajes publicitarios escaneados mediante la función get_adv():

```
adv=bt.get_adv()
```

Tras convertir los datos a ASCII para poder tratarlos se comprueba si la MAC recibida es la de algún móvil conocido. En caso de conocerse se envía un mensaje LoRa al LoPyA, quien enviará dicho mensaje al server.

Una vez que se recibe el mensaje de respuesta del server, se rellena con ceros hasta la longitud de 31bytes para poder enviarlo mediante BLE y ser detectado por la app de escaneo.

```
respuesta_relleno=str(2) + str(8) + str(3)+ str(respuesta)+ '000000000'
print('Mensaje enviado mediante BLE: ', respuesta_relleno, '\n')
bt.set_advertisement(name='RESPUESTA SERVER', manufacturer_data=respuesta_relleno)
bt.set_advertisement_params(adv_int_min=0x20, adv_int_max=0x30, adv_type=bt.ADV_TYPE_IND)
bt.advertise(True)
```

La función bluetooth.set_advertisement(name, manufacturer_data) Configura los datos que se enviarán durante la publicidad. Los parámetros son:

Name: es el nombre de la cadena que se mostrará en los anuncios.

Manufacturer_data: datos del fabricante que se anunciarán.

La función bluetooth.set_advertisement_params configura los parámetros que se usan al hacer la publicidad:

- Adv_int_min: es el intervalo de publicidad mínimo para publicidad dirigida no dirigida y de ciclo de trabajo bajo.
- Adv_int_max: es el intervalo publicitario máximo para publicidad dirigida no dirigida y de ciclo de trabajo bajo.
- Adv_type: es el tipo de publicidad.

Capítulo 3. Especificación de los protocolos entre los nodos de la red.

3.1 Máquinas de estados del sistema

Para conocer en detalle el funcionamiento del sistema, es conveniente conocer cuál es la máquina de estados del sistema y tener una idea simple de cómo funciona.

En la siguiente figura se puede observar la máquina de estados del sistema:

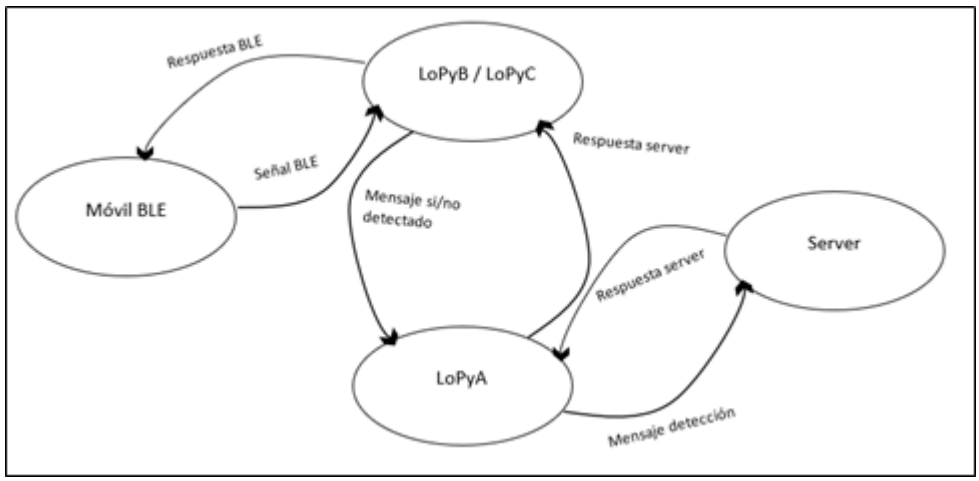


Figura 11. Máquina de estados del sistema

Esta máquina muestra el cambio entre los diferentes estados en los que se podrá encontrar el sistema, conforme vaya funcionando. Se han considerado los estados los diferentes módulos LoPy, el móvil y el server.

Lo que hace que se cambie de uno a otro son:

- **Señal BLE**: Si el LoPy detecta una señal Beacon o no.
- **Mensaje si/no detectado:** Mensaje que envía el LoPyB o LoPyC al LoPyA para informarle de que ha detectado o no trama beacon.
- **Mensaje detección:** El LoPyA envía mensaje al Server para informar de que un LoPy ha detectado o no una trama beacon.
- **Respuesta server:** Respuesta que envía el server al LoPyA, para que este la reenvíe al LoPy que ha detectado la trama beacon.
- **Respuesta BLE:** Mensaje de respuesta del server que es enviada al móvil por el LoPy que detectó la trama beacon.

3.2 Descripción del funcionamiento del sistema

El funcionamiento principal del sistema es el de detectar una trama beacon mediante un LoPy, el cual envía un mensaje hacía el server para que este genere un mensaje de respuesta hacia el móvil. Más claramente sería:

Primero, tanto el LoPyB como el LoPyC estarían escuchando señales beacon emitidas por aparatos varios. Si alguno de ellos detecta una señal beacon procedente del móvil cuya MAC es conocida, generará un mensaje de detección que irá dirigido hacía el server. Que un LoPy detecte una trama beacon significa que se encuentra en cobertura con dicho móvil emisor de tramas beacon.

Seguidamente, mediante LoRa, el LoPy que detecta la trama envía un mensaje con el texto 'BLE SI DETECTADO' hacía el LoPyA. Es necesario enviar este mensaje al LoPyA ya que es este el que tiene comunicación directa con el server mediante comunicación serie. Una vez recibido el LoPyA el mensaje de detección, lo reenviará hacia el server.

Ya en el server, el mensaje es recibido del LoPyA. A continuación, se generará un mensaje de respuesta con el mensaje 'HOLA USUARIO X'. Este mensaje, de nuevo por comunicación serie, será enviado al LoPyA.

El LoPyA reenviará la respuesta del server hacia el LoPy que detectó la señal beacon para proceder a su envío por BLE.

Por último, la respuesta del server ya ha llegado al LoPy que detectó la señal, se procede al envío, mediante BLE, de la respuesta del server al móvil.

Por el contrario, también cabe la posibilidad de que algún LoPy no detecte señales beacon procedentes del móvil con MAC conocida. En este caso el proceso sería algo parecido.

Ahora en vez de enviar un mensaje especificando que se ha detectado una señal BLE, se envía un mensaje tal que así 'BLE NO DETECTADO'. De nuevo sería enviado al LoPyA y de este pasaría al server.

Hasta aquí todo seguiría igual, a excepción del contenido del mensaje enviado. Pero una vez que el mensaje llega al server, éste comprueba que no se ha detectado trama beacon, por lo que, en vez de enviar un mensaje de respuesta, este imprime por pantalla un 'BLE NO DETECTADO'.

3.3 Tipos y formatos de tramas entre módulos LoPy

La comunicación entre módulos LoPy se realiza mediante el protocolo LoRa explicado en puntos anteriores.

Existen varios tipos de mensajes para comunicarse entre los módulos LoPy:

- **Mensaje de detección**: Se envía cuando el LoPyB o LoPyC ha detectado una señal BLE.

IDtx (1byte)	**IDrx (1byte)**	**Tipo (1byte)**	**Datos (16bytes)**

CAMPO	DESCRIPCIÓN
IDtx	Es el ID del móvil que transmite la señal BLE
IDrx	Es el ID del LoPy que recibe la señal BLE
Tipo	Indica si el mensaje es BLE (3), Lora (2) o Serie (1)
Datos	Contiene el mensaje enviado, para este caso sería 'BLE SI DETECTADO'

- **Mensaje de no detección:** Se envía cuando el LoPyB o LoPyC no ha detectado una señal BLE.

IDtx (1byte)	IDrx (1byte)	Tipo (1byte)	Datos (16bytes)

CAMPO	DESCRIPCIÓN
IDtx	'0'. No se recibe ninguna señal
IDrx	Es el ID del LoPy que no recibe la señal BLE
Tipo	Indica si el mensaje es BLE (3), Lora (2) o Serie (1)
Datos	Contiene el mensaje enviado, para este caso sería 'BLE NO DETECTADO'

- **Mensaje de respuesta:** Se envía cuando el LoPyA ha recibido respuesta del server.

IDtx (1byte)	IDrx (1byte)	Tipo (1byte)	Datos (16bytes)

CAMPO	DESCRIPCIÓN
IDtx	Es el ID del server
IDrx	Es el ID del LoPy que tiene que enviar la respuesta BLE
Tipo	Indica si el mensaje es BLE (3), Serie (2) o Lora (1)
Datos	Contiene el mensaje de respuesta, para este caso sería 'HOLA USUARIO X'

- El campo 'tipo', como se muestra en la descripción de cada trama puede tener 3 valores diferentes:
- BLE → '3': Indica que el mensaje que se envía es de tipo BLE. Este se usará para la comunicación entre el LoPyB/LoPyC y el móvil.
- Serie →'2': Indica que el mensaje que se envía es de tipo Serie. Este será usado para la comunicación entre el server y el LoPyA.
- LoRa →'1': Indica que el mensaje que se envía es de tipo LoRa. Este será usado en la comunicación entre LoPy-LoPy.

3.4 Tipos y formatos de tramas entre Server y nodo LoPy

La comunicación entre el server y el nodo LoRa se realiza por puerto serie. Dicho puerto es una interfaz de comunicación de datos digitales, donde la información es transmitida bit a

bit, enviando un solo bit a la vez. En este caso se han creado puertos serie virtuales mediante USB, lo que se conocen como puertos COM.

Estos puertos son interfaces de E/S. Sustituyen a los puertos serie, debido a que muchas máquinas no tienen ninguna interfaz serie instalada.

El puerto serie utiliza un dispositivo UART. Estas son las siglas de Receptor/Transmisor Asíncrono Universal. Es un dispositivo hardware que traduce los datos entre los puertos serie y paralelo de un ordenador. El procesamiento UART transmite un byte de datos serie bit a bit. Luego, los datos se vuelven a convertir a un byte completo en el otro extremo de la conexión. Usar esta tecnología hace que sea irrelevante si se utiliza puerto paralelo o serie para la transmisión de datos.

En la comunicación entre el server y el nodo LoRa hay dos tipos de mensajes:

- **Mensaje de detección:** Se envía desde el LoPyA al server informando sobre si se ha detectado o no un móvil BLE

IDtx (1byte)	**IDrx (1byte)**	**Tipo (1byte)**	**Datos (16bytes)**

CAMPO	DESCRIPCIÓN
IDtx	Es el ID del móvil que emite beacon
IDrx	Es el ID del LoPy que recibe la señal BLE
Tipo	Indica si el mensaje es BLE (3), Lora (2) o Serie (1)
Datos	Contiene el mensaje enviado por el LoPy detector del beacon. Puede ser 'BLE SI DETECTADO' o 'BLE NO DETECTADO'

- **Mensaje de respuesta**: Es la respuesta del server al mensaje de detección de señal BLE.

IDtx (1byte)	**IDrx (1byte)**	**Tipo (1byte)**	**Datos (14bytes)**

CAMPO	DESCRIPCIÓN
IDtx	Es el ID del server.
IDrx	Es el ID del LoPy que ha recibido la señal BLE
Tipo	Indica si el mensaje es BLE (3), Lora (2) o Serie (1)
Datos	Contiene el mensaje de respuesta 'HOLA USUARIO X'

3.5 Tipos y formatos de tramas entre nodo LoPy y dispositivo móvil

La comunicación entre el nodo LoPy y el dispositivo móvil emisor de tramas beacon se realiza mediante BLE. Este protocolo ha sido explicado en puntos anteriores.

Existen dos tipos de mensaje entre el LoPy y el móvil:

- **Mensaje beacon:** Es el mensaje ADV. enviado por el móvil para ser detectado en el LoPy.

MAC	PAYLOAD

CAMPO	DESCRIPCIÓN
MAC	Es la MAC del móvil
Payload	Es el mensaje ADV. que emite el móvil

- **Mensaje de respuesta:** Es el mensaje que envía el LoPy en respuesta al mensaje recibido del móvil.

IDtx (1byte)	IDrx (1byte)	Tipo (1byte)	Datos (14bytes)

CAMPO	DESCRIPCIÓN
IDtx	Es el ID del server.
IDrx	Es el ID del LoPy que ha recibido la señal BLE
Tipo	Indica si el mensaje es BLE (3), Lora (2) o Serie (1)
Datos	Contiene el mensaje de respuesta 'HOLA USUARIO X'

En este caso, el mensaje cuando es enviado mediante BLE se convierte a hexadecimal, que es el tipo de dato que recibe el móvil.

En primer lugar, el móvil 1 se configura para que emita señales beacon continuamente para que sea detectado por un LoPy con el que se encuentre en cobertura. A la vez, el móvil 2 está configurado para escanear beacon.

3.6 Diagrama de bloques de la arquitectura en conjunto

El siguiente cronograma muestra la secuencia de mensajes cuando el LoPyB/C detecta una trama beacon. El LoPy estaría en cobertura con el móvil:

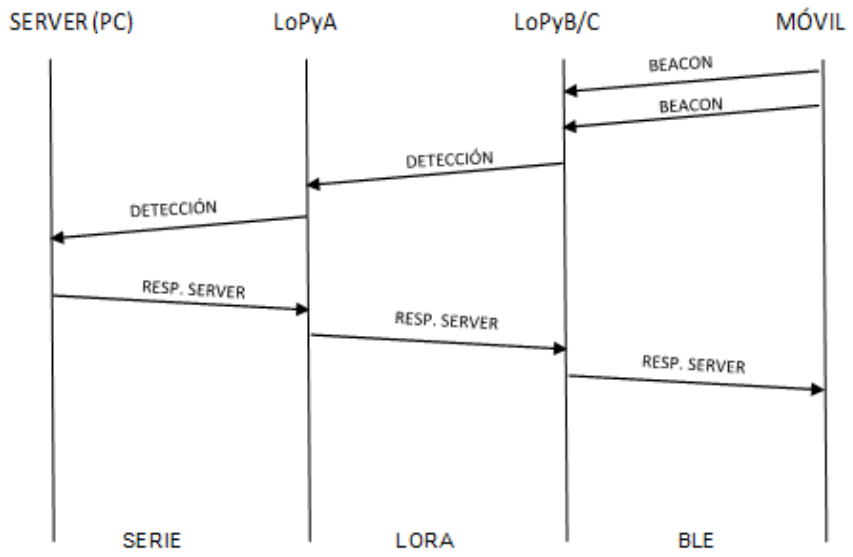


Figura 12. Cronograma mensajes BLE detectado

En este caso, el cronograma muestra la secuencia de mensajes cuando el LoPyB/C no detecta tramas beacon. Por lo que el LoPyB/C no estaría en cobertura con el móvil.

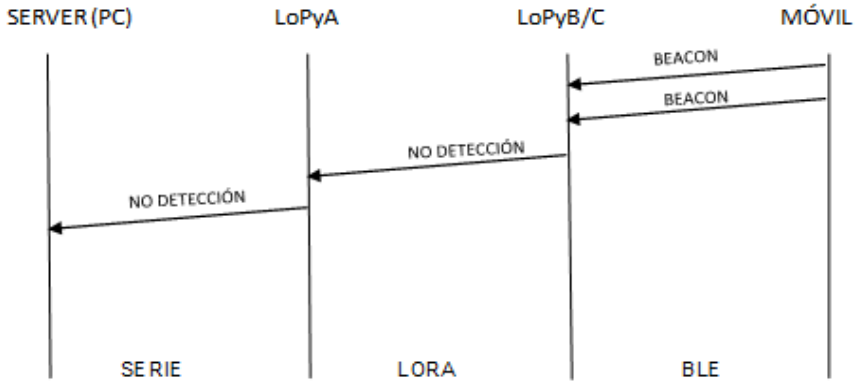


Figura 13. Cronograma mensajes BLE no detectado

Para este caso en concreto, como no se detecta trama beacon no se genera ningún mensaje de respuesta. Por el contrario, el server muestra por pantalla que no se ha detectado ningún mensaje BLE.

Capítulo 4. Arquitectura del servicio de mensajes de texto en red IoT

4.1 Diagrama de bloques arquitectura en conjunto

Este esquema de bloques representa la arquitectura en conjunto del proyecto. Se aprecia una distribución formada por tres LoPys, de los cuales B y C son esclavos, y un tercero A que es master. Dicho LoPy es el encargado de establecer la comunicación por puerto serie con el server. Además, también se muestra los móviles encargados de transmitir tramas beacon y escanear.

Las flechas indican que, la comunicación entre los LoPy es bidireccional, pueden tanto transmitir como enviar mensajes LoRa entre ellos. Y pasa lo mismo con el LoPyA y el server. Mediante comunicación puerto serie pueden enviar y recibir mensajes.

Por último, para el caso de los móviles, tenemos dos, uno que emite y otro que escanea. Ambos no pueden transmitir y escanear beacons a la vez, de ahí que sea necesario trabajar con dos móviles.

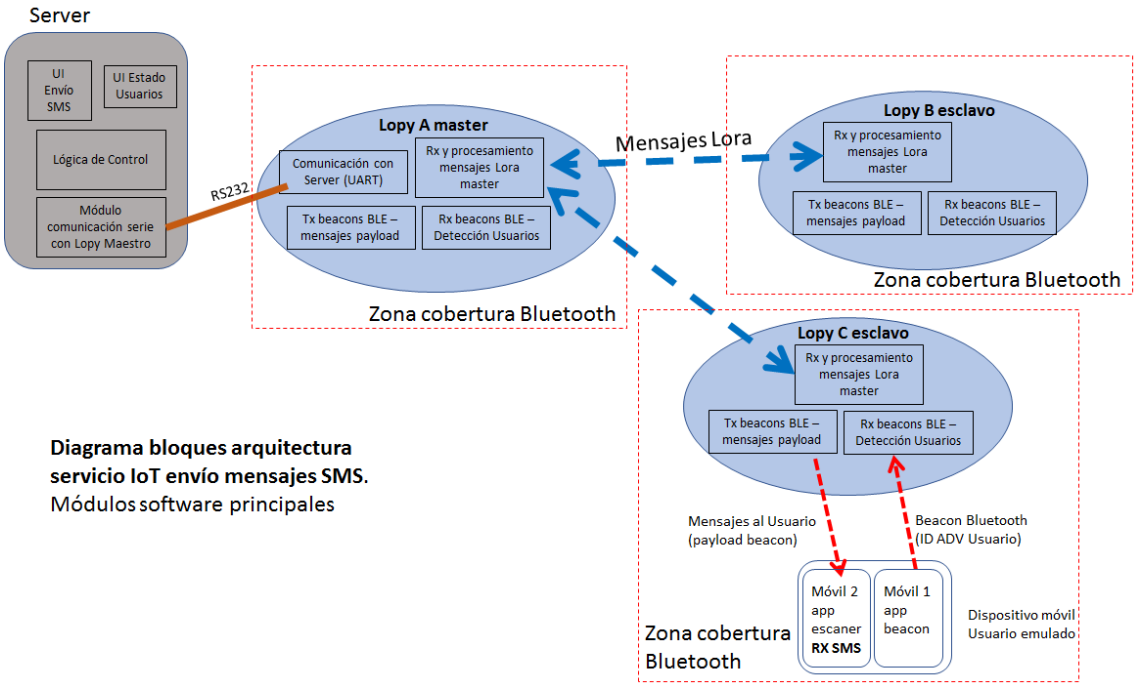


Diagrama bloques arquitectura servicio IoT envío mensajes SMS.
Módulos software principales

Figura 14. Diagrama de bloques de la arquitectura

4.2 Desarrollo software

Para el desarrollo de las pruebas de cobertura se utilizarán los dispositivos IoT LoPy, compuestos por un procesador ESP32 con distintos chips de comunicaciones como LoRa **(SX1276)**, WiFi, BLuetooth y Sigfox.

Figura 15. Chip LoPy

Este chip se programa en lenguaje Micro Python. El entorno de desarrollo será Atom, junto con la librería pymakr 1.4.18.

Se dispone de 3 dispositivos de este tipo. Todos ellos son tanto transmisores como receptores.

También se hace uso de un PC con la versión 3.7.9 de Python que hará de server. El entorno de desarrollo será Notepad++ y para ejecutarlo se usará el cmd que es un intérprete de comandos.

Por último, se usa también dos móviles con sistema Android que hacen, uno de emisor de tramas beacon y otro de escáner. Dichos dispositivos tienen instalado dos apps llamadas 'Beacon Simulator' y 'BeaconScope'. Estos programas emiten tramas beacon y también actúan de escáner.

Un esquema básico sería es el siguiente:

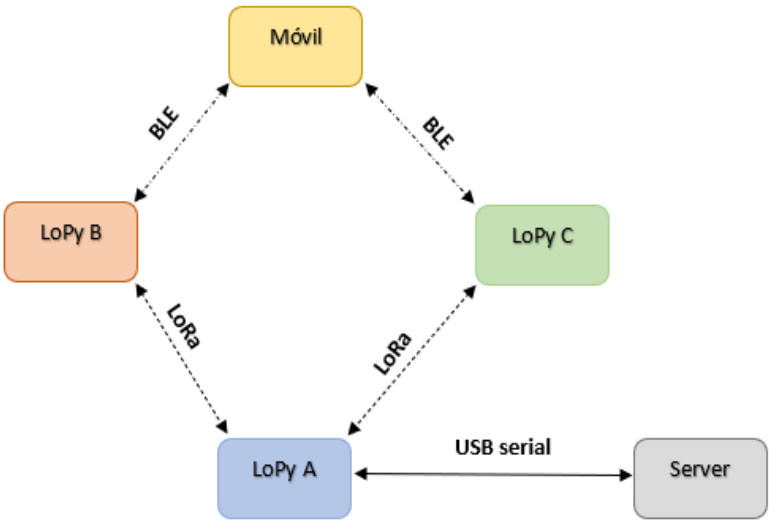


Figura 16. Esquema del sistema

4.2.1 Software LoRa

Se utilizan varios scripts de micro Python, uno para cada LoPy, donde se reciben los mensajes LoRa de cada uno de los transmisores y a su vez, cuando es necesario se generan también los mensajes LoRa a transmitir entre ellos.

Los mensajes recibidos por los LoPyB o C tienen un tamaño de 19 bytes. La distribución sería 1 byte para el IDtx, 1 byte para el IDrx, 1 byte para el tipo y los 16 restantes para el payload.

En cambio, los mensajes que se envían de respuesta, procedentes del server, tienen una longitud de 17. 1 byte para el IDtx, 1 byte para el IDrx, 1 byte para el tipo y los 14 restantes para el payload de respuesta. Este es de menor tamaño debido a que el mensaje incluido en el payload es menor.

4.2.2 Software puerto serie

Hay un único script de Python que escucha por puerto serie y recibe los mensajes LoRa transmitidos por el LoPyA. A la vez también genera un mensaje de respuesta que envía por puerto serie en respuesta al LoPyA.

El puerto serie lo que hace es leer byte a byte para identificar el IDrx y averiguar si el mensaje recibido fue generado por el LoPyB o por el LoPyC. Una vez identificado el origen del mensaje comprueba el contenido del mensaje para generar la respuesta adecuada a cada tipo de mensaje. En caso de no venir ninguno de los dos continúa escuchando.

Para ejecutar el script se escribe el siguiente comando en el cmd:

>>Python el_nombre_del_archivo.py -u “puerto”

Para poder ejecutar dicho comando y que funcione correctamente se debe previamente situarse en la carpeta donde se encuentre el script mediante el comando **cd.**

El puerto debe ser el COM específico del LoPy con el que se mantiene la comunicación por puerto serie. En este caso sería el puerto 4.

4.3 Máquina de estados y emulación del dispositivo móvil

El desarrollo de una app de recepción de SMS queda fuera del alcance del TFG. Para la prueba de concepto se emula un dispositivo virtual de un usuario empleando una app Android gratuita que se instala en un móvil. Dicha app se puede configurar para que actúe como:

- **Modo Beacon**: se envían mensajes de advertising tipo iBeacon en el periodo y potencia de transmisión configurados.
- **Modo Scanner**: se reciben y muestran por pantalla los advertising de dispositivos Bluetooth en cobertura. Si se selecciona un mensaje de advertising concreto se puede visualizar el payload expresando los bytes en hexadecimal. Las MAC Bluetooth de los módulos Lopy son conocidas y se asume que están disponibles en un fichero de configuración. Por tanto, se va a aprovechar esta funcionalidad para seleccionar los

advertising recibidos de un nodo Lopy y, viendo su payload, identificar el mensaje SMS codificado en hexadecimal.

Por tanto, para la emulación de un dispositivo virtual de un usuario se usarán dos móviles simultáneos con la app Android BLE:

- Móvil 1 con app beacon. MAC BLE del móvil se asocia a un usuario en el fichero de configuración.
- Móvil 2 con app escáner, donde se emula la recepción de los SMS a partir del payload de los advertising de un nodo Lopy.

En cuanto al dispositivo móvil 1 la funcionalidad básica sería la de emitir señales beacon continuamente, dentro de un rango, para que el LoPy que más cerca esté de este pueda captar su señal y detectar la trama beacon.

En este momento el móvil 1 se encontraría en estado de emisión de beacons. Y se mantendrá en este estado siempre.

En cuanto la señal beacon sea detectada por un LoPy, el dispositivo móvil 2 que está en el estado de escáner, estará escuchando señales beacon. Detectará muchas de ellas y entre todas encontrará la procedente del LoPy que anteriormente haya detectado la señal beacon emitida por el móvil 1.

Existen dos posibilidades, la primera es que el dispositivo móvil esté en cobertura con el LoPyB, por lo que será este el que detecte las tramas beacon. Esto significa que será el encargado tanto de detectar como de enviar la respuesta emitida por el server. Así pues, el dispositivo móvil 2 detectará, mediante el modo escáner, la señal que proceda de este LoPy con su id correspondiente.

La segunda posibilidad que existe es que el dispositivo móvil esté en cobertura con el LoPyC, por lo que el proceso llevado a cabo será el mismo que el anterior, cambiando únicamente el id.

Para ambos casos, el LoPy que no se encuentre en cobertura con el dispositivo móvil intentará encontrar tramas beacon procedentes de dicho móvil sin éxito ninguno. Enviando un mensaje con payload ‘BLE NO DETECTADO’ al server.

La siguiente figura muestra la máquina de estado correspondiente al dispositivo móvil. (Para simplificar la complejidad de la máquina de estado se ha tenido en cuenta que un mismo móvil transmite y recibe beacon):

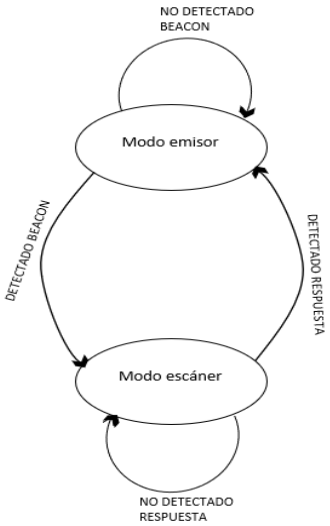


Figura 17. Máquina de estado del móvil

4.4 Diagrama de flujo y máquina de estados de los módulos LoPy

En este apartado se va a explicar el funcionamiento de los módulos LoPy. Para empezar, se muestra el diagrama de flujo del LoPy A. Este LoPy es el encargado de establecer una conexión serie con el server. Todos los mensajes BLE recibidos por el LoPyB y LoPyC serán enviados a este LoPy para establecer la comunicación con el server.

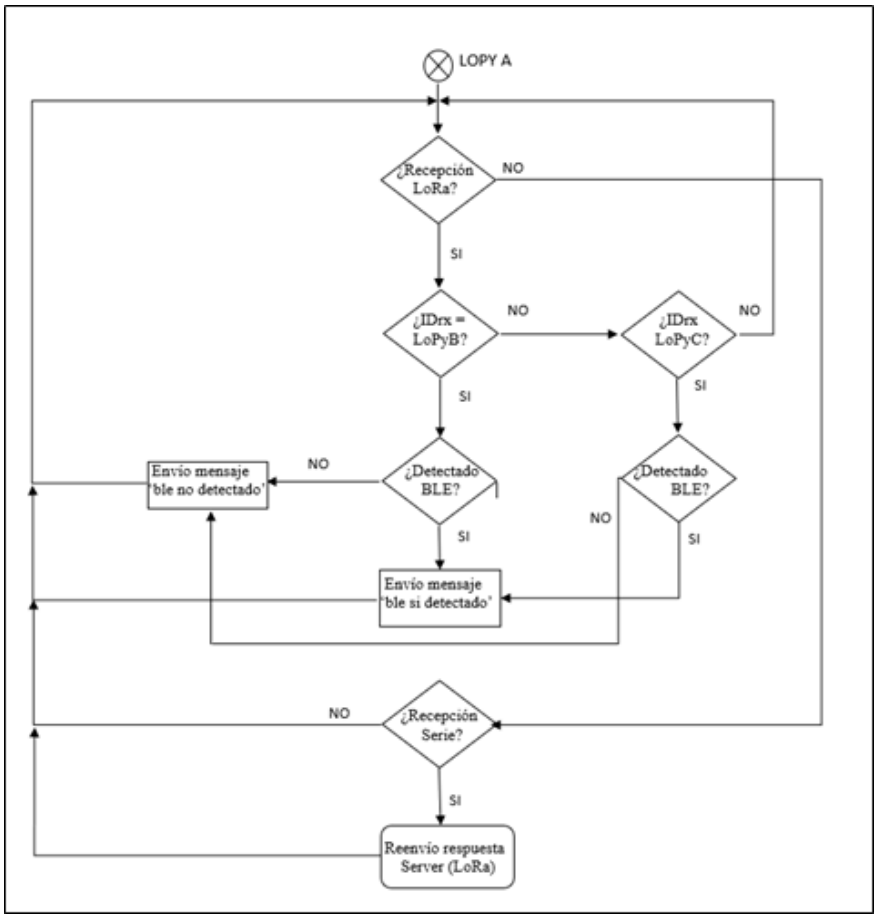


Figura 18.Diagrama de flujo LoPyA

El diagrama de flujo empieza con una decisión de si el mensaje recibido ha sido Lora, es decir que viene del LoPyB o LoPyC, o serie, que vendría entonces del servidor. Una vez decidido de quien viene el mensaje, en el caso de ser LoRa se decide sobre que LoPy lo ha enviado, fijándose en el id de recepción. Por último, hay un bloque de decisión que su función es averiguar si se ha detectado un mensaje BLE o no. Todo ello se envía encapsulado en un mensaje al servidor mediante comunicación serie.

Para el caso de recibir un mensaje serie, este querría decir que el server ha contestado a nuestro mensaje enviado desde el LoPyA. Este mensaje lo tendríamos que reenviar mediante Lora al receptor indicado en el mensaje serie.

Para el caso del LoPyB tenemos el siguiente diagrama de flujos:

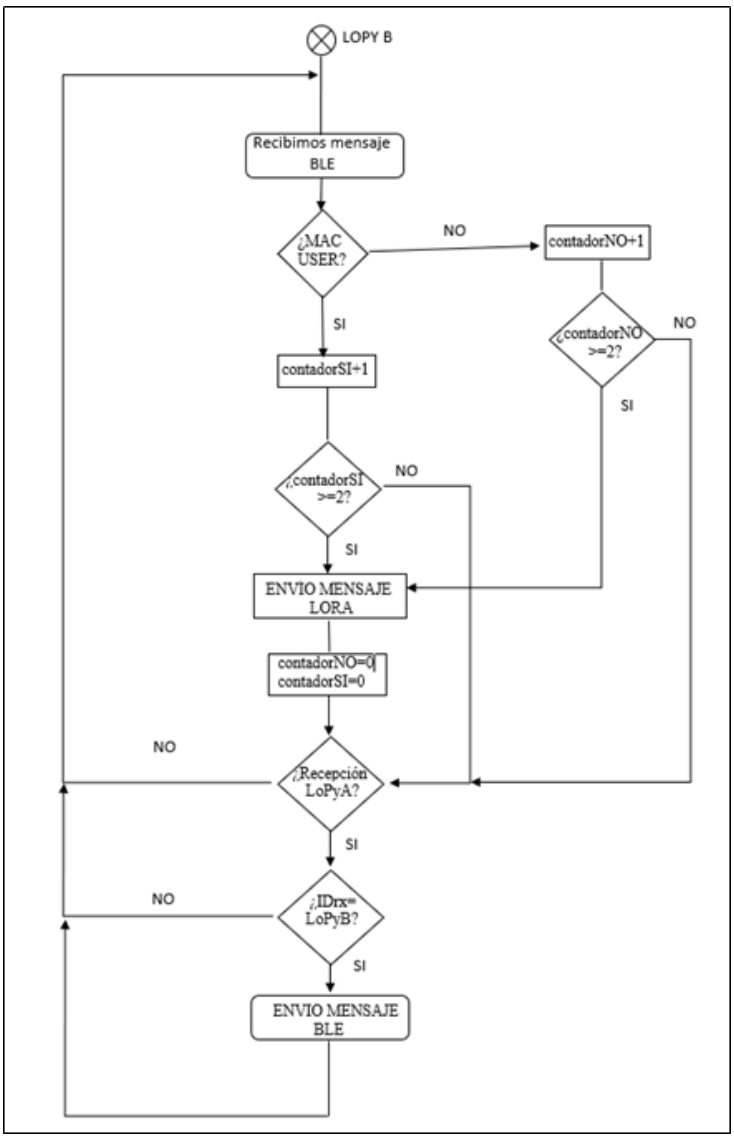


Figura 19. Diagrama de flujo LoPyB

En este caso el diagrama de flujo empieza recibiendo un mensaje BLE de un móvil que emite señales beacon. Una vez que se recibe dicha señal se debe comprobar si viene del móvil que conocemos, de ahí que se compruebe la MAC. Si es el móvil correcto incrementaremos la variable contadorSI, y seguidamente comprobaremos si dicha variable es mayor que dos. Si es

así, quiere decir que el móvil sigue en cobertura con el LoPyB y enviaremos un mensaje al LoPyA mediante Lora.

Tanto si el contador es mayor que dos o no, el siguiente paso sería comprobar si hemos recibido respuesta del LoPyA, en caso de ser así enviaremos un mensaje BLE con la respuesta del servidor. Si no recibimos respuesta volveremos a comprobar la señal recibida BLE.

El otro caso es cuando el mensaje recibido no es del móvil que conocemos. En este caso incrementaremos la variable contadorNO, la cual se comprobará para saber si es mayor de dos. En caso de ser mayor de dos, se enviará un mensaje diciendo que el móvil no es detectado. Esto significa que el LoPyB está fuera de cobertura con el móvil y no se recibe ninguna señal beacon de este.

Tanto si la variable es menor como si es mayor se comprueba también si hay respuesta del LoPyA. Una vez se llega a este punto, se actuaría de la misma forma que se ha explicado anteriormente.

Esta sería la máquina de estado asociada al LoPyB:

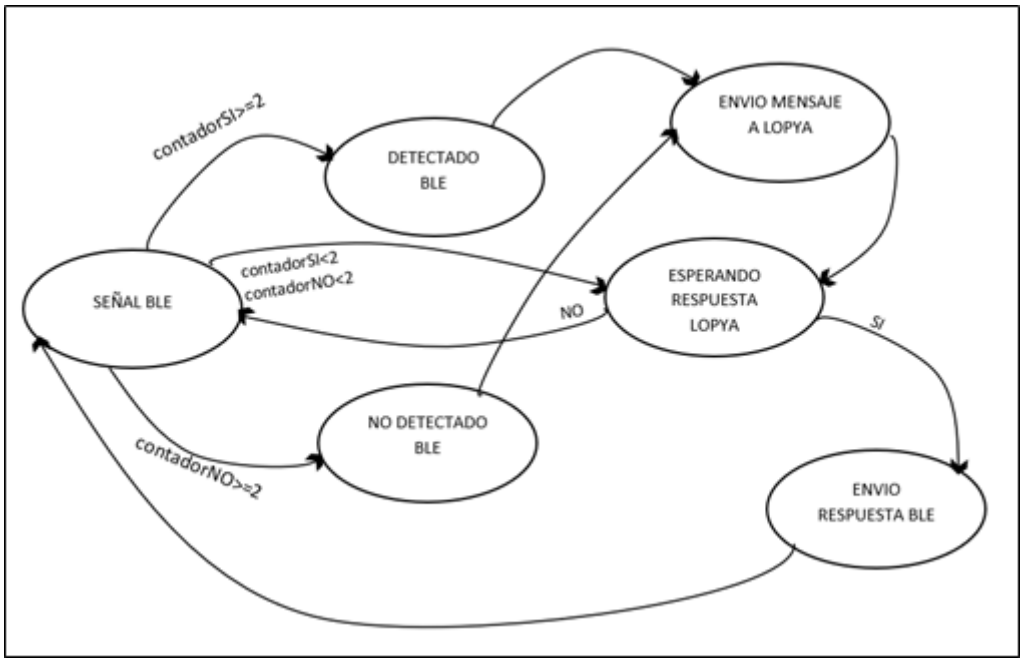


Figura 20. Máquina de estados LoPyB

4.5 Diagrama de flujo del server

A continuación, se pasará a explicar el diagrama de flujo del server. Este es el encargado de generar un mensaje de respuesta que será enviado al LoPy que detectó la trama beacon.

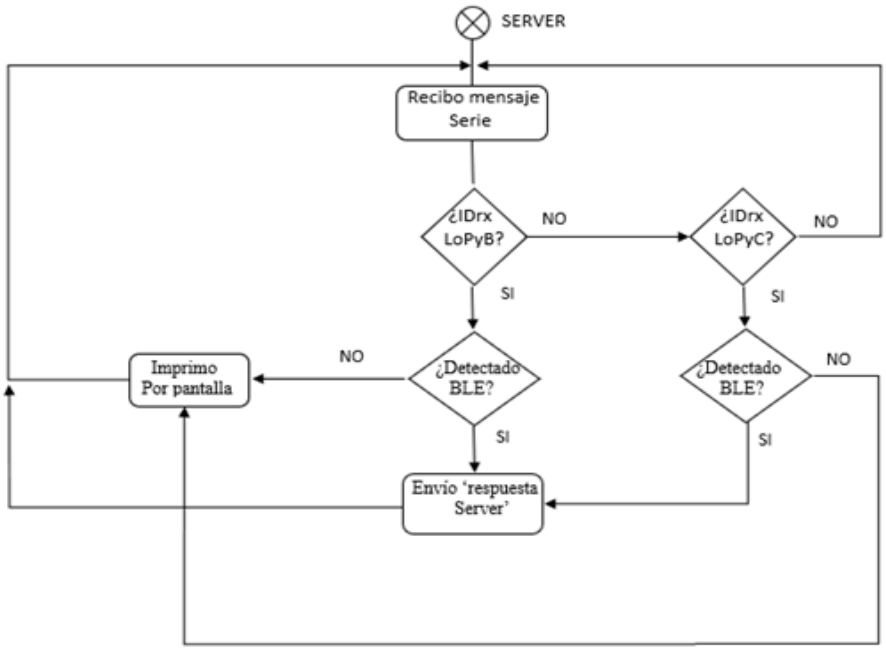


Figura 21. Diagrama de flujo del Server

Este diagrama comienza con la recepción por puerto serie de un mensaje enviado desde el LoPyA. Una vez recibido el mensaje, se comprueba si viene del LoPyB o del LoPyC, comprobando si el IDrx es 2 (LoPyB) o 3 (LoPyC). Una vez averiguado de quien procede el mensaje, el siguiente paso será comprobar cuál es el contenido del payload, es decir, si el mensaje recibido contiene un mensaje de detección o de no detección de trama beacon. Si el contenido es 'sí', se generará un mensaje de respuesta que será enviado por puerto serie al LoPyA. Este será el encargado de redirigir dicha respuesta al LoPyB o LoPyC dependiendo de quien haya enviado el mensaje.

En caso contrario, cuando el contenido del mensaje sea 'no', el server imprimirá un mensaje diciendo que no se ha detectado señal BLE.

Y, por último, está el caso en el que el mensaje recibido no venga de ninguno de los dos LoPy, en este momento el server no realizará ninguna acción, si no que pasará a escuchar de nuevo.

4.6 Parámetros de configuración en server y nodos LoPy

Para una correcta comunicación entre los dispositivos que componen la red es necesario definir previamente unos parámetros de configuración que servirán de referencia, tanto al server como a los LoPy, para identificar la procedencia de los mensajes recibidos.

En por ello que se han definido, al principio de cada script, unas variables con la información necesaria a tener en cuenta y la cual será usada para la identificación de los dispositivos.

```
MAC_USER1 = ‘1aff4c000215'

ID_USER1= 57


MAC_USER2 = '0303aafe1516'

ID_USER2 = 56


ID_L1 = 49

ID_L2 = 50

ID_L3 = 51

ID_S = 48
```

Para el caso de los LoPy y del server no ha sido necesario la definición de variables con la MAC asociada a cada uno de ellos ya que con el ID es suficiente para realizar las comprobaciones de identificación.

Capítulo 5. Pruebas y resultados

Para la implementación del proyecto se empezó estudiando teóricamente el protocolo LoRa y su uso en los módulos LoPy. También se estudió su funcionamiento comenzando por la creación de aplicaciones simples, aumentando el grado de dificultad hasta llegar al objetivo final. Las primeras pruebas ayudaron a comprender y a manejar el protocolo LoRa.

Una vez controlado, se pasa al estudio del Bluetooth Low Energy (BLE). Se usa el mismo procedimiento que el utilizado para LoRa. Primero se realiza una toma de contacto estudiando teóricamente su funcionamiento y creando aplicaciones desde más sencillas hasta más complejas.

Por último, una vez estudiado y controlado todo lo necesario para la realización del proyecto propuesto, se fusionan todos los conceptos y todos los procedimientos aprendidos hasta ahora, dando lugar al resultado final mostrado en este TFG.

A continuación, se describen las pruebas de validación del correcto funcionamiento del sistema y los resultados obtenidos.

5.1 Usuarios y dispositivo virtual emulados en las pruebas

Para las pruebas se emula que hay dos usuarios: user 1 con ID=9 y user 2 con ID=8

Por simplicidad se va a usar un dispositivo de usuario virtual formado por dos móviles. Cada móvil tiene instaladas una app Android BLE (configurable en modo beacon o en modo scanner). Según el modo que se ejecute en cada dispositivo se estará emulando al usuario 1 o al usuario 2.

Emulación del usuario 1:

- Móvil 1 con app beacon. MAC BLE 0303aafe1716aafe00bf asociada al user1.
- Móvil 2 con app escáner (donde se emula la recepción de los SMS).

Emulación del usuario 2:

- Móvil 1 con app escáner (donde se emula la recepción de los SMS).
- Móvil 2 con app beacon. MAC BLE 0303aafe1516aafe00bf asociada al user2.

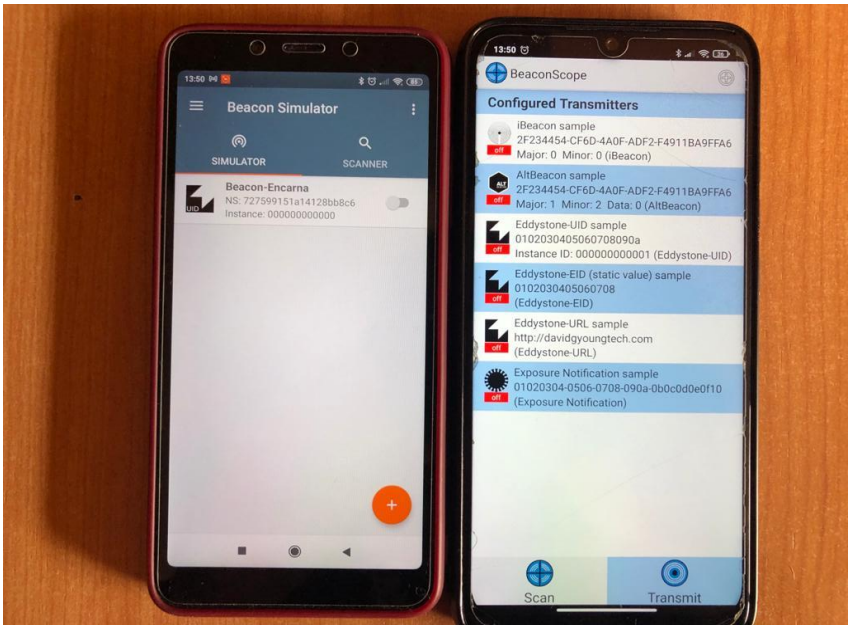


Figura 22. Dispositivos móviles usados

5.2 Prueba 1. Usuario 1 entra en cobertura de un nodo LoPy y recibe SMS

La prueba de validación del sistema consiste en que el dispositivo de usuario entre en cobertura con un nodo Lopy, seguidamente sea detectado en el server como asociado al nodo Lopy, el server le envíe un SMS y que el dispositivo de usuario reciba el SMS.

Antes de pasar a mostrar imágenes sobre la secuencia de pruebas se va a definir los diferentes colores en los que los leds integrados en los LoPy se pueden encontrar conforme se vayan ejecutando las pruebas.

Para ambos LoPys (B y C) existen tres tipos de colores para los leds, color azul, verde y rojo que significan lo siguiente:

Para el LoPyB/C:

· Verde: significa que está en modo escucha de tramas beacon o de recepción de mensajes LoRa.

Figura 23. Led verde LoPyB

· Rojo: significa que ha detectado una trama beacon procedente de una MAC conocida, y que se encuentra a la espera de una respuesta LoRa. De aquí pasaría de nuevo al color verde.

Figura 24. Led rojo LoPyB

· Azul: significa que el LoPy no ha detectado ninguna trama beacon y que pasa a esperar algún mensaje LoRa. De aquí pasaría de nuevo al color verde.

Figura 25. Led azul LoPyB

Para el LoPyA los colores de los leds son los mismo, pero tienen significado diferente:

· Verde: significa que está esperando la recepción de un mensaje LoRa o un mensaje por puerto serie.

Figura 26. Led verde LoPyA

- Rojo: significa que ha recibido un mensaje LoRa procedente del LoPyB o del LoPyC. Mientras se mantiene en este color también envía un mensaje por puerto serie al server.

Figura 27. Led rojo LoPyA

- Azul: significa que ha recibido un mensaje por el puerto serie procedente del server y a la vez, también envía la respuesta del server mediante LoRa al LoPy específico.

Figura 28. Led azul LoPyA

Para esta primera prueba hemos usado dos móviles. Uno de ellos su función será la de emitir beacon que serán detectados por uno de los LoPy configurados para ello, en este caso el encargado de recibirlos será el LoPyB. El segundo móvil será utilizado para escanear la respuesta procedente del servidor, un SMS cuyo contenido es 'HOLA USUARIO 1'.

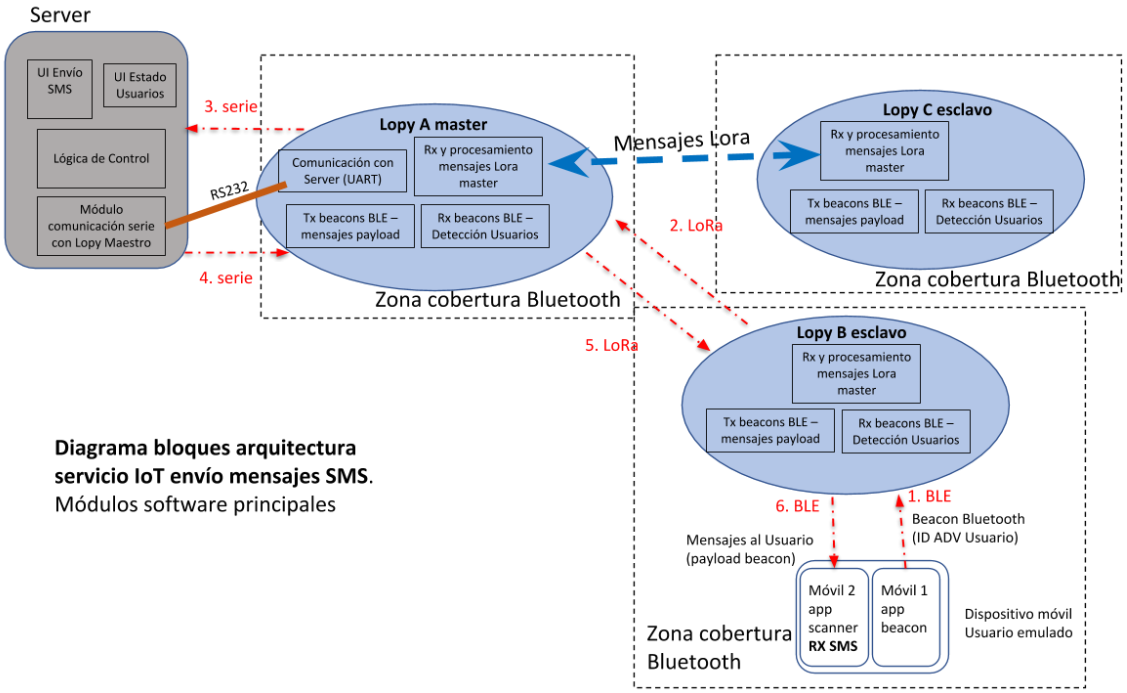


Diagrama bloques arquitectura servicio IoT envío mensajes SMS. Módulos software principales

Figura 29. Secuencia de mensajes móvil 1 emisor

En este esquema se puede ver la secuencia de mensajes, la dirección y el tipo de mensaje que se transmite a lo largo de una comunicación. A continuación, se pasará a explicar cada paso definido en la imagen, añadiendo en los casos que sea necesario fotos y/o capturas.

1) Primeramente, la app del móvil 1 se configura para emitir señales beacon.
2) Una vez puesto en marcha el LoPyB empieza a detectar diferentes MAC procedentes de diversos dispositivos que podemos encontrar hoy día. Una vez que el nodo LopyB detecta dos veces la misma MAC (que coincide con la MAC asociada a los usuarios registrados según fichero de configuración), en este caso la del móvil 1, muestra lo siguiente por consola:

```
MAC: 0303aafe1716aafe00bf
MAC: 03036ffd17166ffd4e0a
MAC: 0303aafe1716aafe00bf

DISPOSITIVO BLE DETECTADO
Mensaje enviado al LoPyA:  b'921BLE SI DETECTADO'
```

Y envía un mensaje LoRa al LoPyA.

Se hace un inciso para explicar el significado de lo mostrado en la captura anterior:

MAC: Muestra la MAC que detecta el LoPyB. Que puede ser de la app del móvil 1 u otro emisor bluetooth como pueden ser unos cascos inalámbricos.

Mensaje 'DISPOSITIVO BLE DETECTADO': Mensaje que muestra el LoPyB cuando detecta la MAC del dispositivo móvil por segunda vez.

Mensaje enviado al LoPyA b'921BLE SI DETECTADO':

Nº 9: Corresponde al ID del dispositivo móvil 1.

Nº 2: Corresponde al ID del LoPy que ha detectado la trama beacon. En este caso LoPyB

Nº 1: Corresponde al tipo de mensaje enviado. LoRa para este mensaje.

Resto de mensaje: Mensaje que envía el LoPyB al server para confirmar que ha detectado una trama beacon.

3) El LoPyA ha recibido el mensaje anterior, procedente del LoPyB, y ahora es él el que envía un mensaje por puerto serie al server.
4) El server ha recibido el mensaje que le ha enviado el LoPyA procedente del LoPyB y muestra por consola lo siguiente:

```
Mensaje recibido: 922BLE SI DETECTADO

Mensaje recibido del LoPyB

LoPyB detectó trama beacon

Envío mensaje de respuesta
b'022HOLA USUARIO 1'
```

Mensaje recibido 992BLE SI DETECTADO:

Nº 9: ID del móvil emisor de tramas beacon.

Nº 2: ID receptor de beacons.

Mensaje confirmando que es el LoPyB quien detecta los beacon del móvil 1.

Y por últmo, el mensaje que envía el server de respuesta al LoPyB, donde:

Nº 0: ID del server.

Nº 2: ID del LoPyB

Nª 2: Tipo de mensaje. Puerto serie en este caso.

Resto de mensaje: El SMS enviado por el server para ser enviado del LoPyB al móvil 1.

5) De nuevo, mediante puerto serie el server envía el mensaje de respuesta el LoPyA quien se encarga de reenviarlo al LoPyB
6) Ya en el LoPyB el mensaje de respuesta se genera para ser enviado mediante BLE al móvil 2, el cual esta en modo escáner preparado para escanear la trama procedente del LoPyB.

```
Mensaje recibido del server:  021HOLA USUARIO 1
Mensaje enviado mediante BLE:  293HOLA USUARIO 1000000000
```

Por último, en la siguiente imagen se puede apreciar como el móvil 1 se encuentra emitiendo beacons y el móvil 2 en modo escáner. Se aprecia un beacon recibido donde pone 'RESPUESTA SERVER', dicho mensaje contiene, en hexadecimal, el mensaje enviado por el LoPyB mediante BLE.

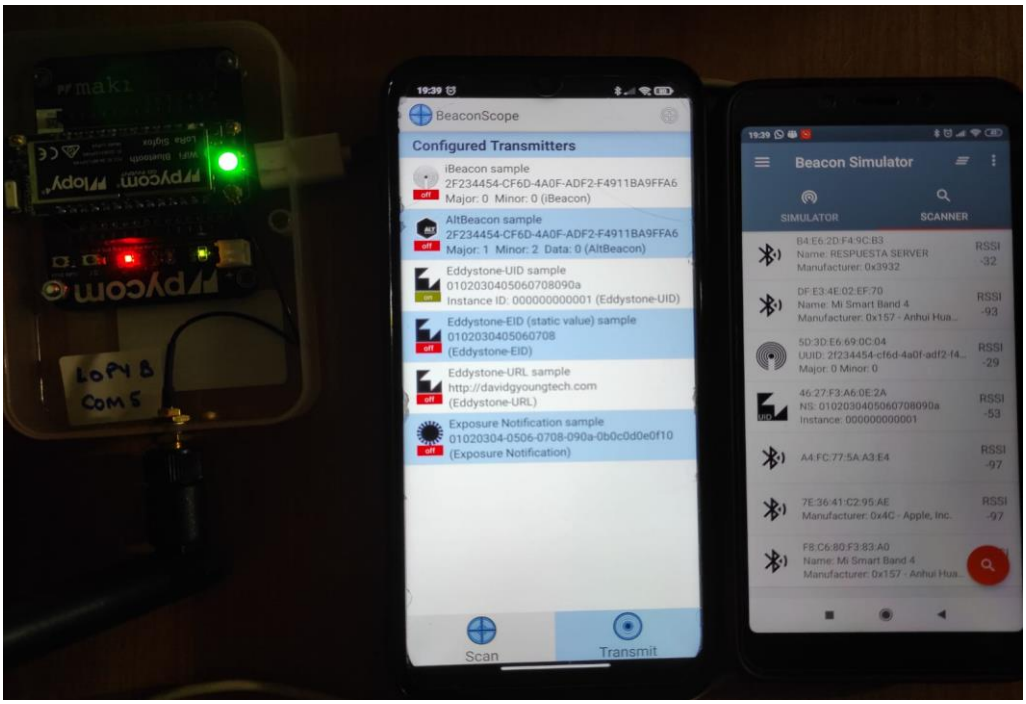
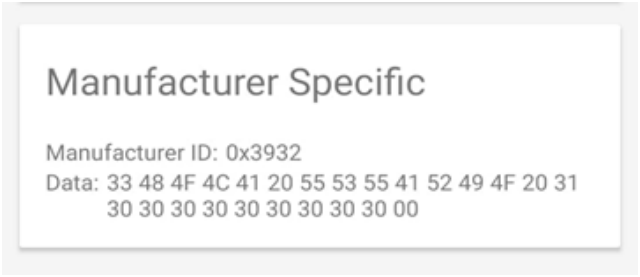
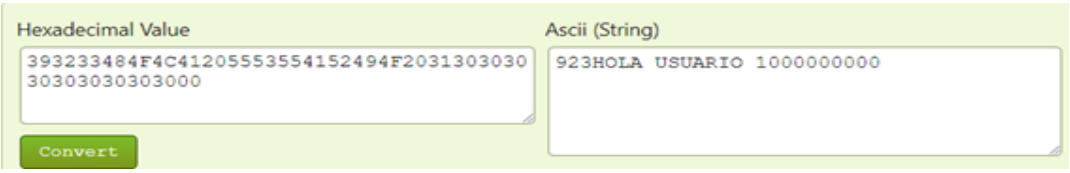


Figura 30. Detección de respuesta server por movil 2

Este sería el mensaje en hexadecimal:



Y si se realiza la conversión a ascii se puede apreciar que el mensaje coincide con el enviado por el LoPyB, con una ligera modificación de orden, puesto que el mensaje enviado es '293HOLA USUARIO 1000000000' y el recibido es '923HOLA USUARIO 1000000000'.



5.3 Prueba 2. Usuario 2 entra en cobertura de un nodo LoPy y recibe SMS

Para la segunda prueba intercambiamos los usos de las apps. Ahora el dispositivo móvil 1 será el que escaneé las tramas beacon y el dispositivo 2 será el que emita beacons. En este caso el SMS que deberá enviar el server tendrá el contenido 'HOLA USUARIO 2'.

Para esta prueba los LoPy siguen actuando de la misma forma y con los mismos colores led que se describieron en el punto anterior.

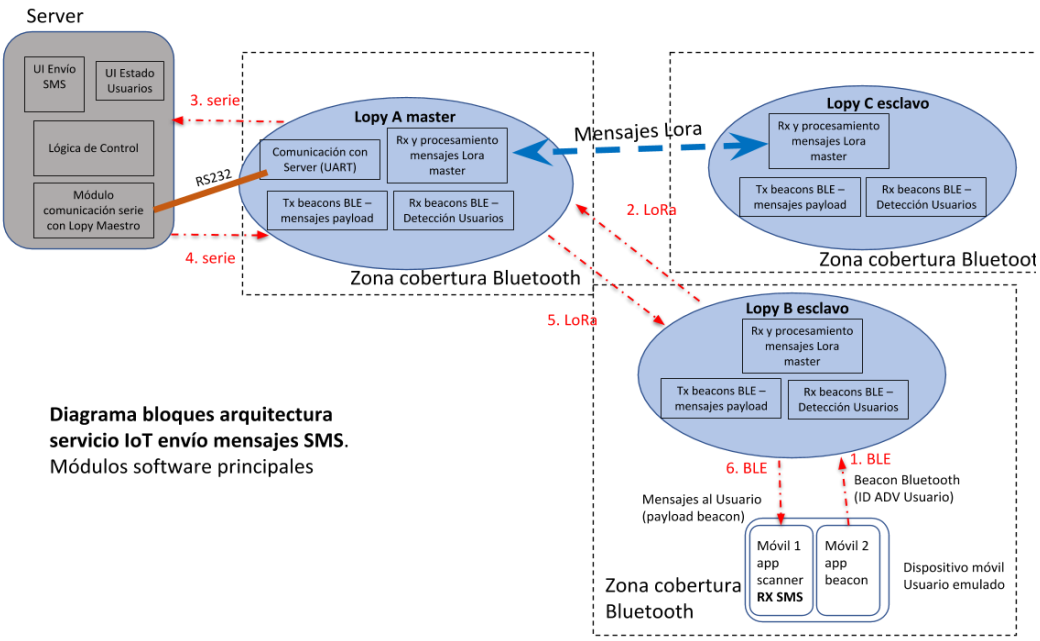


Diagrama bloques arquitectura servicio IoT envío mensajes SMS. Módulos software principales

Figura 31. Secuencia de mensajes móvil 2 emisor

De nuevo un esquema como el anterior, donde se puede ver la secuencia de mensajes, la dirección y el tipo de mensaje que se transmite a lo largo de una comunicación. A continuación, se pasará a explicar cada paso definido en la imagen, añadiendo en los casos que sea necesario fotos y/o capturas.

1) Primeramente, la app del móvil 1 se configura para emitir señales beacon.
2) Una vez puesto en marcha el LoPyB empieza a detectar diferentes MAC. Una vez que el nodo LopyB detecta dos veces la misma MAC (que coincide con la MAC asociada a los usuarios registrados según fichero de configuración), en este caso la del móvil 2, muestra lo siguiente por consola:

```
MAC: 0303aafe1516aafe00bf
MAC: 03036ffd17166ffd4e0a
MAC: 03039ffe17169ffe0000
MAC: 0303aafe1516aafe00bf

DISPOSITIVO BLE DETECTADO
```

Y envía un mensaje LoRa al LoPyA.

3) El LoPyA ha recibido el mensaje anterior, procedente del LoPyB, y ahora es él el que envía un mensaje por puerto serie al server.
4) El server ha recibido el mensaje que le ha enviado el LoPyA procedente del LoPyB y muestra por consola lo siguiente:

```
Mensaje recibido: 822BLE SI DETECTADO

Mensaje recibido del LoPyB

LoPyB detectó trama beacon

Envío mensaje de respuesta
b'022HOLA USUARIO 2'
```

Mensaje recibido 822BLE SI DETECTADO:

Nº 8: ID del móvil 2 emisor de tramas beacon.

Nº 2: ID receptor de beacons, LoPyB.

Mensaje confirmando que es el LoPyB quien detecta los beacon del móvil 2.

Y, por último, el mensaje que envía el server de respuesta al LoPyB, donde:

Nº 0: ID del server.

Nº 2: ID del LoPyB

Nº 2: Tipo de mensaje. Puerto serie en este caso.

Resto de mensaje: El SMS enviado por el server para ser enviado del LoPyB al móvil 2

5) De nuevo, mediante puerto serie el server envía el mensaje de respuesta el LoPyA quien se encarga de reenviarlo al LoPyB.
6) Ya en el LoPyB el mensaje de respuesta se genera para ser enviado mediante BLE al móvil 1, el cual está en modo escáner preparado para escanear la trama procedente del LoPyB.

```
Mensaje recibido del server:  021HOLA USUARIO 2
Mensaje enviado mediante BLE:  283HOLA USUARIO 2000000000
```

Por último, en la siguiente imagen se puede apreciar como el móvil 2 se encuentra emitiendo beacons y el móvil 1 en modo escáner. Se aprecia un beacon recibido donde pone 'RESPUESTA SERVER', dicho mensaje contiene, en hexadecimal, el mensaje enviado por el LoPyB mediante BLE.

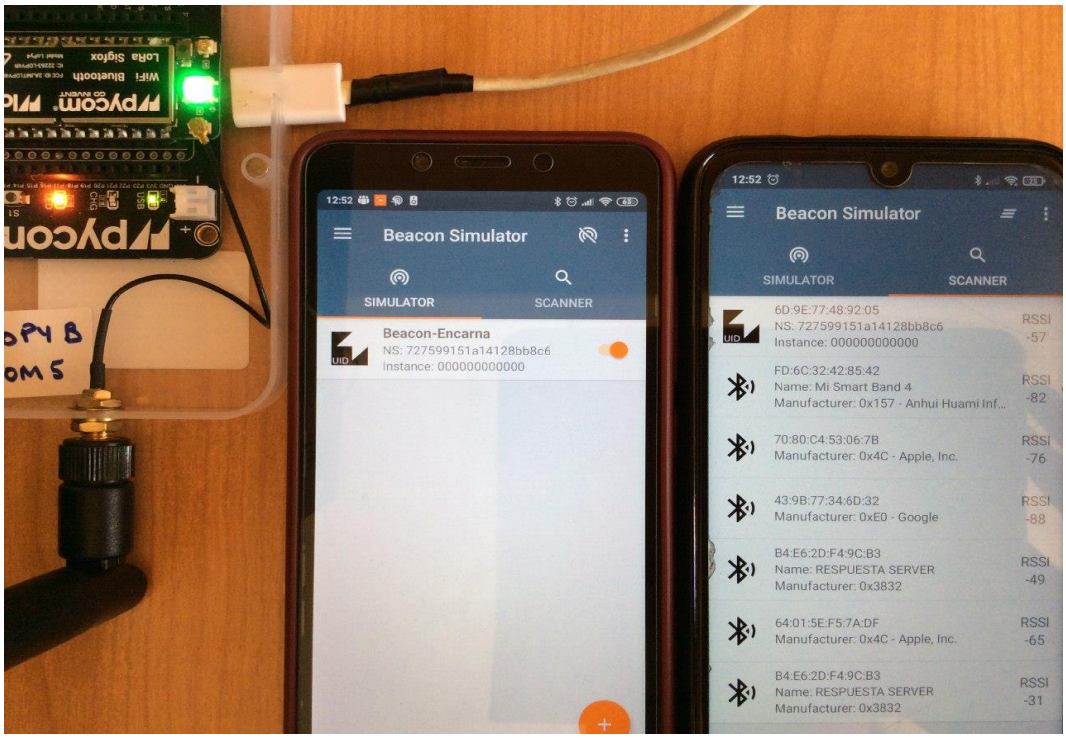
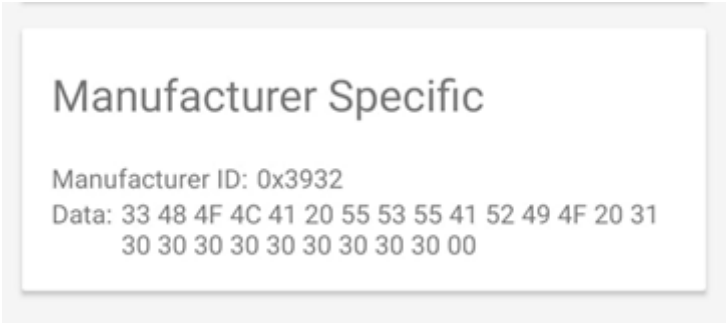
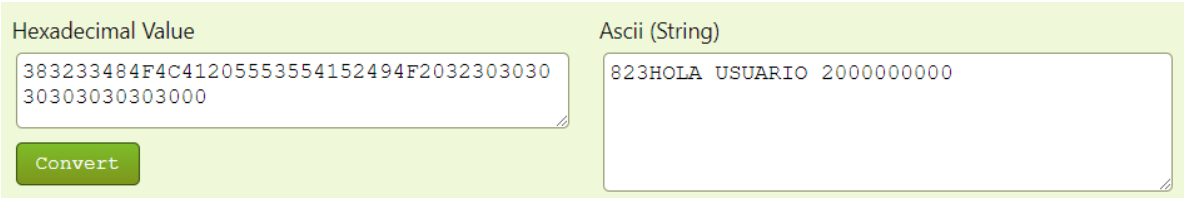


Figura 32. Detección de respuesta server por móvil 1

Este sería el mensaje en hexadecimal:



Y si se realiza la conversión a ascii se puede apreciar que el mensaje coincide con el enviado por el LoPyB, con una ligera modificación de orden, puesto que el mensaje enviado es '283HOLA USUARIO 1000000000' y el recibido es '823HOLA USUARIO 1000000000'.



5.4 Prueba 3. Emulación salida de cobertura de usuarios y aviso al Server

Para emular que un usuario sale de cobertura se apaga el app beacon de un móvil. En pocos segundos el server recibe la notificación de usuario fuera de cobertura. En la siguiente

prueba no se van a enviar mensajes SMS. Sólo se van a detectar usuarios y emular la pérdida de cobertura. Esta información se verá en la consola del server.

1) Detección usuarios en nodos Lopy

El móvil 1 en modo beacon (usuario 1) se coloca cerca del lopyB y en segundos sale mensaje en el server de detectado usuario 1 en lopy B.

Figura 33. LoPyB detecta trama móvil 1

Como se puede apreciar en la imagen el LoPyB tiene el led de color rojo lo que significa que ha detectado una trama beacon. Seguidamente en la consola del server aparece el siguiente mensaje:

```
Mensaje recibido: 922BLE SI DETECTADO

Mensaje recibido del LoPyB

LoPyB detectó trama beacon

DETECTADO USUARIO 1 EN LOPY B
```

Lo importante de la imagen anterior es el último mensaje impreso por pantalla ‘DETECTADO USUARIO 1 EN LOPY B’ que lo que quiere decir es que la trama beacon emitida por el dispositivo 1 (móvil a la derecha del LoPyB) ha sido detectada por el LoPyB.

Ahora se pone el móvil 2 en modo beacon (usuario 2) se coloca cerca del lopyC y en segundos sale mensaje en el server de detectado usuario 2 en lopyC.

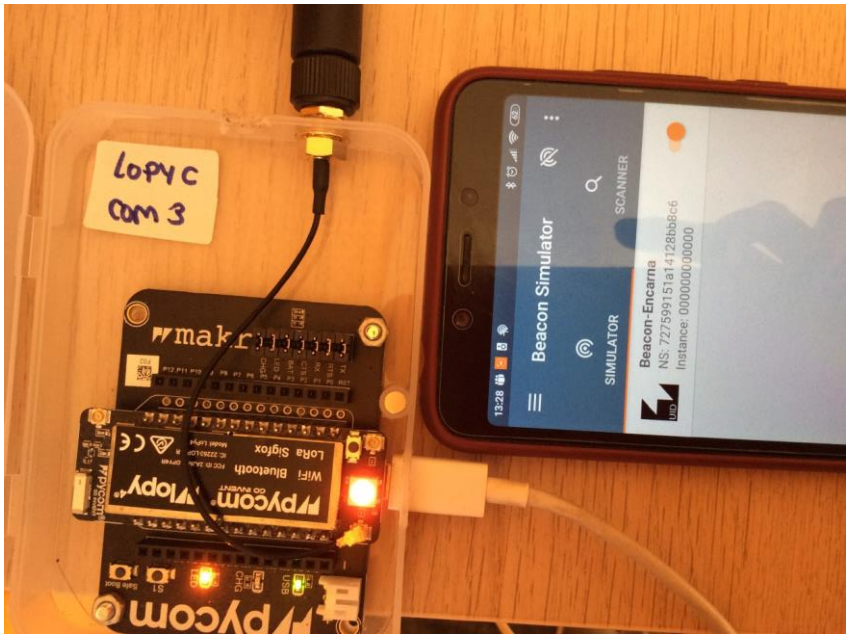


Figura 34. LoPyC detecta trama móvil 2

De nuevo en esta imagen se puede ver como el led del LoPyC indica que ha detectado una trama beacon procedente del móvil 2.

```
Mensaje recibido: 832BLE SI DETECTADO

Mensaje recibido del LoPyC

LoPyC detectó trama beacon

DETECTADO USUARIO 2 EN LOPY C
```

De igual modo, en esta imagen lo importante es el último mensaje que imprime el server 'DETECTADO USUARIO 2 EN LOPY C', confirmando que el LoPyC ha detectado una trama beacon que procedía del móvil 2.

2) Salida cobertura usuario 1.

Para emular que el usuario 1 sale de cobertura del lopyB se apaga el app beacon del móvil 1. El LopyB envía mensaje al server de usuario no detectado. En el server sale mensaje usuario1 no detectado.

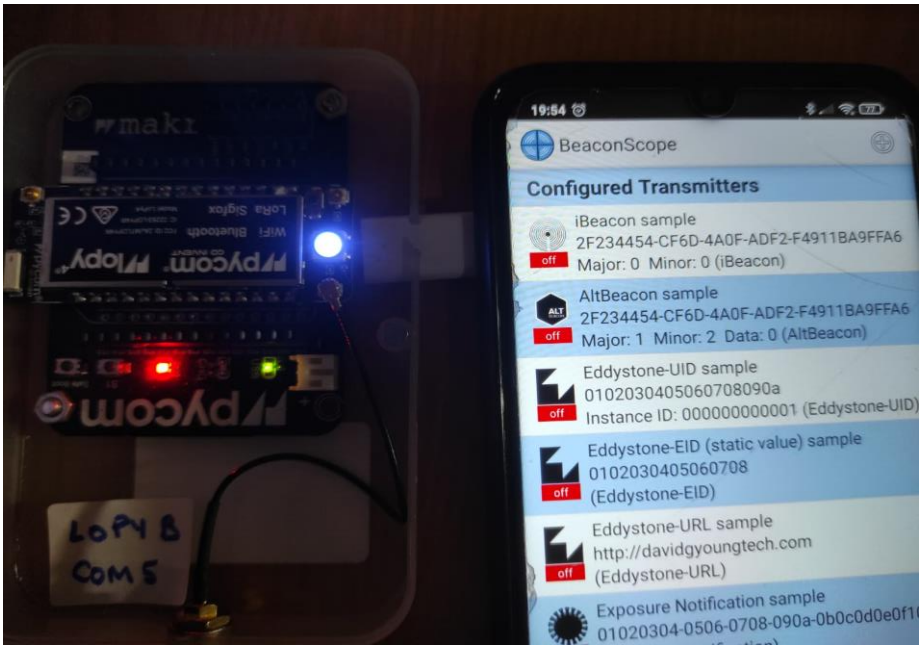


Figura 35. LoPyB no detecta trama móvil 1

En esta imagen se puede observar como el móvil 1 tiene apagado la opción de transmisión de beacons y la luz del LoPyB, azul, nos confirma que el LoPy no ha detectado ninguna trama.

```
Mensaje recibido del LoPyB

USUARIO NO DETECTADO
```

La confirmación de la ausencia de trama se puede ver en la consola del server, cuyo mensaje 'USUARIO NO DETECTADO' nos aclara que el LoPyB no ha detectado trama beacon.

2) Salida cobertura usuario 2.

Para emular que el usuario 2 sale de cobertura del lopyC se apaga el app beacon del móvil 2. LopyC envía mensaje al server de usuario no detectado. En el server sale mensaje usuario2 no detectado.

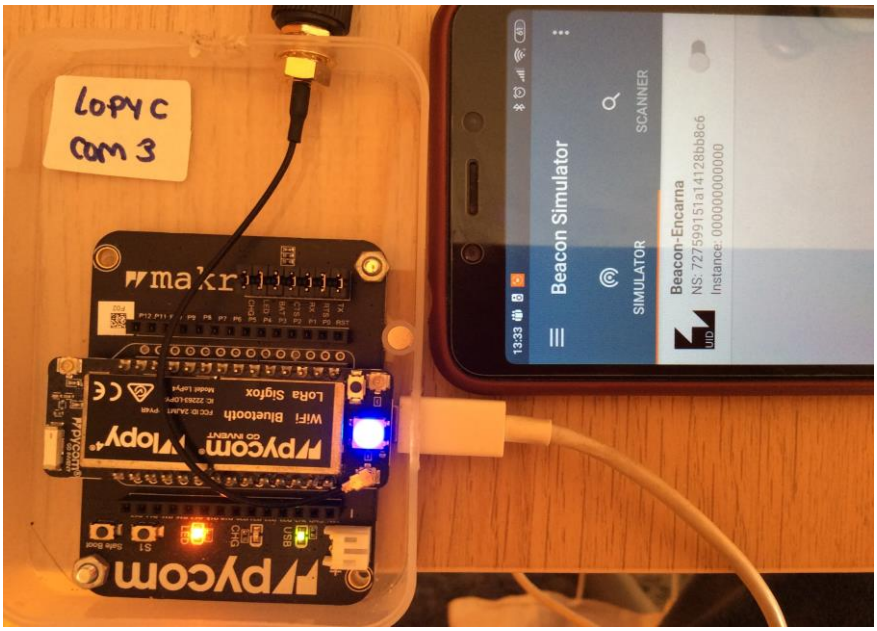


Figura 36. LoPyC no detecta trama móvil 2

La imagen anterior viene a mostrar lo mismo que la mostrada para el LoPyB. Led azul en el LoPyC que confirma que no se ha detectado ninguna trama beacon. También se puede observar que la transmisión de beacon de la app está apagada.

```
Mensaje recibido del LoPyC

USUARIO NO DETECTADO
```

Misma respuesta para diferente LoPy. En este caso el LoPyC es el que no detecta la trama beacon y el server muestra mensaje confirmando la ausencia de tramas.

Capítulo 6. Conclusiones y trabajos futuros

Con respecto a los objetivos que se marcaron al comienzo de la memoria e inicio del proyecto, se puede decir que han sido alcanzados. Dichos objetivos se enumeran a continuación:

1) Diseño de un esquema de red IoT básico en estrella, compuesto por un nodo maestro y dos nodos esclavos.
2) Diseño de un protocolo para la transmisión de mensajes de texto en el servidor.
3) Implementación del servidor y nodos IoT, tanto maestro como esclavos.
4) Realización de pruebas para la validación del prototipo propuesto, las cuales se han llevado a cabo con éxito.

Tras una última valoración de los resultados obtenidos durante todo el proceso de desarrollo, se puede afirmar que dichos objetivos planteados al inicio del proyecto, que en conjunto forman el resultado final del proyecto, han sido resueltos de forma satisfactoria.

Durante el proceso de desarrollo llevado a cabo he abordado el aprendizaje de estas tecnologías y conocimientos:

- Protocolo LoRa
- Protocolo BLE
- Módulos LoPY
- Lenguaje de programación como es Python y microPython

Como posibles trabajos futuros y continuación del TFG, que servirán para mejorar el ya expuesto, se propone:

- Pasar de una arquitectura en estrella maestro-esclavo a una red mesh donde los nodos Lora colaboran entre sí.
- Desarrollo de una app Android de recepción de SMS basada en BLE para un smartphone o tablet.
- Mejorar y ampliar el protocolo de comunicaciones con control de errores, reenvíos y ACK de confirmación de recepción de mensajes.
- Permitir el envío de SMS desde el dispositivo móvil al Server o a otros usuarios registrados.

Bibliografía y enlaces

- Web Pycom fabricante del módulo de IoT Lopy

https://pycom.io/

https://docs.pycom.io/firmwareapi/pycom/pycom/

- Python

https://www.python.org/

- Manuales LoPy

https://docs.pycom.io/gettingstarted/#step-1-setting-up-the-hardware

https://docs.pycom.io/datasheets/expansionboards/expansion3/

- API y funciones LoRa en Lopy

https://docs.pycom.io/firmwareapi/pycom/network/lora/

- API Puerto serie en Lopy

https://docs.pycom.io/firmwareapi/pycom/machine/uart/#app

- API Bluetooth en Lopy

https://docs.pycom.io/firmwareapi/pycom/network/bluetooth/

- [BLE] Standard Bluetooth Low Energy:

https://www.bluetooth.com/specifications/bluetooth-core-specification/legacy-specifications

- Atom

https://atom.io/

- 232Analyzer

https://232analyzer.com/

- NotePad++

https://notepad-plus-plus.org/downloads/

- APP Android modo beacon o scanner BLE

https://play.google.com/store/apps/details?id=net.alea.beaconsimulator&hl=es_419&gl=US

- Segunda APP Android modo beacon

https://play.google.com/store/apps/details?id=com.davidgyoungtech.beaconscanner&hl=es&gl=US