ESCUELA DE INGENIERÍA DE

TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

Implementación de un sistema de detección acústica empleando los nodos de una red inalámbrica en zonas rurales

Titulación: **Grado en Ingeniería en Tecnologías de la Telecomunicación**
Mención: **Telemática**
Autor: **Francisco Carlos Felipe Rodríguez**
Tutor: **Álvaro Suárez Sarmiento**
Fecha: **julio de 2022**

Índice

1 Introducción

En este capítulo se presenta la idea general del sistema desarrollado y su entorno, los objetivos perseguidos, algunos conceptos básicos relacionados con las tecnologías usadas y la estructura de la memoria.

1.1 Antecedentes

Actualmente, el senderismo está viviendo un gran crecimiento ya que, cada vez, son más las personas que apuestan por la práctica de esta actividad dentro de España [1]. Al hablar de este tema es imposible no nombrar el archipiélago canario que cuenta con 8 islas dotadas de numerosas rutas de senderismo de diferentes características, que son un reclamo para muchas personas en cualquier época del año.

Este turismo rural puede convertirse en un complemento ideal para el modelo de desarrollo turístico sustentado sobre Sol y Playa que siempre ha sido el motor principal dentro de la economía canaria [2]. Aunque este tipo de turismo ya ha sufrido un notable crecimiento durante los últimos 20 años [3] es necesario seguir trabajando en la actualización, renovación y adaptación, mediante estrategias innovadoras, de estas zonas rurales para continuar con el desarrollo de dicho turismo rural.

Una posible estrategia para seguir sería el uso de las tecnologías inalámbricas, las cuales son una parte imprescindible de nuestro día a día, tanto en el trabajo [4] como en nuestra vida personal y su despliegue no es abundante en zonas rurales. Las redes de sensores se presentan como una opción ideal que permiten monitorizar cualquier entorno por inaccesible que pueda parecer. Estas redes se basan en el uso de dispositivos electrónicos autónomos equipados con baterías que incluso pueden ser recargadas mediante placas solares si fuera necesario [5].

En nuestro caso, estamos interesados en aplicar las comunicaciones inalámbricas en sendas rurales convirtiendo el senderismo en una experiencia mucho más enriquecedora y atractiva para la captación de un nuevo tipo de turismo. Por lo tanto, en este *Trabajo de Fin de Grado (TFG)* se pretende: estudiar en profundidad las tecnologías inalámbricas y más concretamente las redes de sensores, para utilizar los nodos de dicha red para la detección de voz humana de senderistas.

1.2 Objetivos

El objetivo principal es la detección de sonido mediante micrófonos instalados en una red inalámbrica capaz de registrar el paso de personas ubicada en zonas rurales. Para ello se va a diseñar y configurar una red Ad-Hoc sobre la que montar la red acústica y un sistema de registro de personas basado en el uso de *Bluetooth Low Energy* (*BLE*).

Este objetivo general se lleva a cabo mediante la consecución de los diferentes objetivos operativos detallados a continuación:

- Diseño e implantación de una red Ad-Hoc capaz de registrar el paso de personas en ámbitos rurales.
- Diseño e implantación de una red acústica mediante los nodos de la red inalámbrica en caminos de senderismo.
- Realización de pruebas para verificar el correcto funcionamiento del sistema desarrollado.

Para el desarrollo de los diferentes programas se ha usado el lenguaje de programación Python [6], el cual viene instalado en varios de los sistemas compatibles con RaspBerry Pi [7], concretamente con la versión Debian Bullseye [8] que se ha usado en este TFG.

1.3 Estructura de la memoria

En el capítulo 1 se explican los conceptos básicos que se tratan en la memoria, así como la motivación y objetivos propuestos.

El capítulo 2 se compone de un análisis de la problemática que se pretende solucionar con este proyecto.

En el capítulo 3 se detallan todas las tecnologías que han sido necesarias para el desarrollo del proyecto.

En el capítulo 4 se muestra la implementación y el diseño de las diferentes partes del proyecto, detallando como ha sido posible completar los objetivos y como se han desarrollado los diferentes programas que permiten el funcionamiento del sistema.

En el capítulo 5 se detallan las diferentes pruebas a las que ha sido sometido el sistema con el objetivo de comprobar su funcionamiento en diferentes escenarios. Además, se comprueba el correcto desempeño tanto de la red acústica como de la red inalámbrica analizando el tráfico que cursa.

En el capítulo 6 se comentan las conclusiones después del desarrollo del TFG y se muestran posibles ampliaciones que se podrían llevar a cabo en futuros proyectos.

Para finalizar, se concluye con el pliego de condiciones y el presupuesto donde se expone la posible problemática del uso del sistema y un análisis del coste económico que supone su implantación.

2 Análisis del problema

Con el paso de los años las tecnologías inalámbricas han cobrado un valor enorme, llegando a estar presente en la mayoría de las comunicaciones que tiene lugar en todo el mundo. En este capítulo se presenta las ideas básicas de las tecnologías inalámbricas y el problema a solventar con la implementación de este proyecto, realizando un modelo sencillo capaz de detectar voz humana y registrar el paso de personas en zonas poco accesible como son las rutas de senderismo.

2.1 Introducción

Las comunicaciones inalámbricas son aquellas que se llevan a cabo sin el uso de cables de conexión entre los elementos que establecen la comunicación [9]. Con lo cual una red inalámbrica es un conjunto de dispositivos electrónicos que se comunican entre sí a través de ondas electromagnéticas.

Para nuestro TFG, se va a trabajar en zonas poco accesibles, como son las rutas de senderismo, donde el despliegue de la infraestructura para este tipo de comunicaciones es muy limitado [10]. Debido a esto, no se desarrolla únicamente un sistema de captación y registros de personas, se requiere diseñar e implementar una red inalámbrica Ad-Hoc que permita la comunicación durante todo el recorrido que constituya la ruta en cuestión.

Para la comunicación de este tipo de redes existen diferentes tipos de tecnologías inalámbricas [11]. Sin embargo, para el desarrollo de nuestro TFG se emplea *Wireless Fidelity* (*WiFi*), basado en el estándar *Institute of Electrical and Electronics Engineers* (IEEE) 802.11 [12]. Se emplea WiFi ya que tiene un buen área cobertura y soporta posibles interferencias causadas por objetos, como ramas o arbustos, mejor que otras tecnologías como *Bluetooth* o *ZigBee*. Además, los equipos que se usan para el desarrollo del TFG, en este caso *Raspberry Pi* (*RPi*) 3 modelo B+ y Raspberry Pi 4, ya cuentan con este tipo de tecnología.

Sin embargo, WiFi no es el único tipo de tecnología inalámbrica que se usa para el desarrollo del proyecto. Se trabaja con BLE, recogido en el estándar IEEE 802.15 [13], concretamente con el uso de *Beacons BLE,* para el registro de personas que realizan la ruta de senderismo. De esta forma, se consigue que el usuario, al finalizar dicha ruta, pueda tener acceso a una serie de datos relacionados con la actividad que acaba de realizar, como por ejemplo el tiempo total que ha tardado en realizar el recorrido.

Con este proyecto se intenta seguir fomentando el turismo rural en Canarias y acercar la digitalización a estas rutas de senderismo donde en muchas ocasiones no se dispone de los medios necesarios debido a la poca accesibilidad de estas zonas, y dándole así mayor visibilidad a este tipo de actividad. En este proyecto se pueden diferenciar dos partes, la red inalámbrica Ad-Hoc y el sistema de detección y registro de personas, que hace posible su seguimiento durante el recorrido de la ruta.

Este sistema de detección y registro de personas está formado por una red acústica montada sobre los dispositivos de la red inalámbrica, haciendo uso de micrófonos tipo *Universal Serial Bus* (*USB*). De esta forma se consigue monitorizar el medio en busca de voz humana indicando así la presencia de usuarios que puedan ser registrados. Para el proceso de registro se escanea el medio en busca de *Beacons BLE* transmitidos por los usuarios, que están provistos de un dispositivo transmisor de *Beacons*.

2.2 Sistema de detección acústica y registro de personas cercanas

El sistema de detección acústica se basa en el despliegue de una red acústica sobre una serie de estaciones, en este caso sobre los nodos de la red inalámbrica, capaces de captar información sobre las condiciones del entorno físico circundante [14]. Este entorno sería una zona rural en la que la contaminación acústica es prácticamente nula. Esto facilita en gran medida el tratamiento de las muestras de audio captadas ya que los únicos sonidos no deseados que pueden ser captados son los originados en la propia naturaleza, como por ejemplo el cantar de alguna especie de pájaro [15]. Sin embargo, estos sonidos son impercetibles si la calidad de los micrófonos a utilizar no fuera elevada (elevada sensibilidad).

Para poder implementar la red acústica, a cada nodo de la red se le instala un micrófono tipo USB capaz de captar muestras de sonido del exterior y almacenarlas para su posterior análisis a través de un programa *Python* a ser diseñado e implantado. En principio, mediante el adecuado tratamiento del audio recogido y almacenado se puede determinar si se ha captado voz humana dentro del área de cobertura de la red o no.

Los equipos que se emplean para el diseño de este TFG son dispositivos electrónicos autónomos que tienen que ser equipados con baterías para su funcionamiento en zonas donde no se tiene acceso a la red eléctrica. Gracias a este sistema de detección acústica se consigue un gran ahorro en el nivel de las baterías de los equipos. Esto se debe a que las RPi no tienen que estar escaneando el medio en busca de Beacons constantemente, lo cual conlleva un alto consumo de energía, sino que únicamente inician este proceso de escaneo en el momento que una persona es detectada.

El proceso de escaneo de Beacons permite el registro de la hora a la que el usuario fue detectado en ese nodo de la red. El dispositivo transmisor de Beacons que la persona utiliza, aporta la información necesaria para la creación de un archivo vinculado a dicha persona. En el archivo se registran las marcas de tiempo que realicen los nodos de la red al detectar a la persona.

Una vez se cree el archivo en el primer nodo, se envía al siguiente nodo de la red Ad-Hoc, de esta forma se consigue que toda la red trabaje con un único archivo por usuario. Los nodos de la red Ad-Hoc tienen una topología líneal unidireccional de tamaño *n*, entendiendo por nodo siguiente al que está en la posición relativa *i+1* repecto al que le envía la información que está en la posición *i*.

El usuario al finalizar la ruta y llegar al nodo final (nodo *n*) podría tener acceso al archivo generado. De esta forma puede ver el seguimiento que se le ha realizado y obtener la lista de tiempos registrados por cada nodo (*i=1..n*) y el tiempo total que ha tardado en realizar la ruta.

2.3 Idea básica del sistema propuesto y posibles usos reales

La idea principal de la topología que se plantea para la red Ad-Hoc de este TFG se expone gráficamente en la Figura 2.1, con un ejemplo en el que n=8 y además existe un *Punto de Acceso* (*PA*) WiFi conectado al nodo *i=8* (en realidad esta función de PA la puede llevar a cabo el mismo nodo *i=8*).

Los nodos se implantan a partir de las RPi modelos 3B+ y 4, y a nivel de aplicación se comunican entre ellos mediante una conexión tipo *socket* empleada para el envío de archivos tipo *JavaScript Object Notation* (*JSON*).

Los nodos de la red se equipan tal y como se muestra en la Figura 2.2. Cuentan con un micrófono tipo USB, empleado para la creación de un sistema de detección acústica de voz humana, y con un programa diseñado en Python para el registro de personas a través del escaneo de Beacons.

Como se puede ver en la Figura 2.2, el nodo final se diferencia del resto de nodos de la red, trabajando de forma diferente al resto. Este nodo también forma parte de la red inalámbrica Ad-Hoc, sin embargo, es configurado como un servidor, encargado de almacenar todos los archivos de los usuarios generados por el resto de los nodos de la red. Además de servidor, también actúa como PA, creando una red *WiFi* a la que los usuarios se conectan al finalizar la ruta para tener acceso a su archivo de datos.

2.4 Redes inalámbricas y protocolos de encaminamiento

Como habíamos comentado, las comunicaciones inalámbricas se caracterizan por utilizar ondas de radio para conectar dispositivos entre sí, sin la necesidad de utilizar ningún tipo de

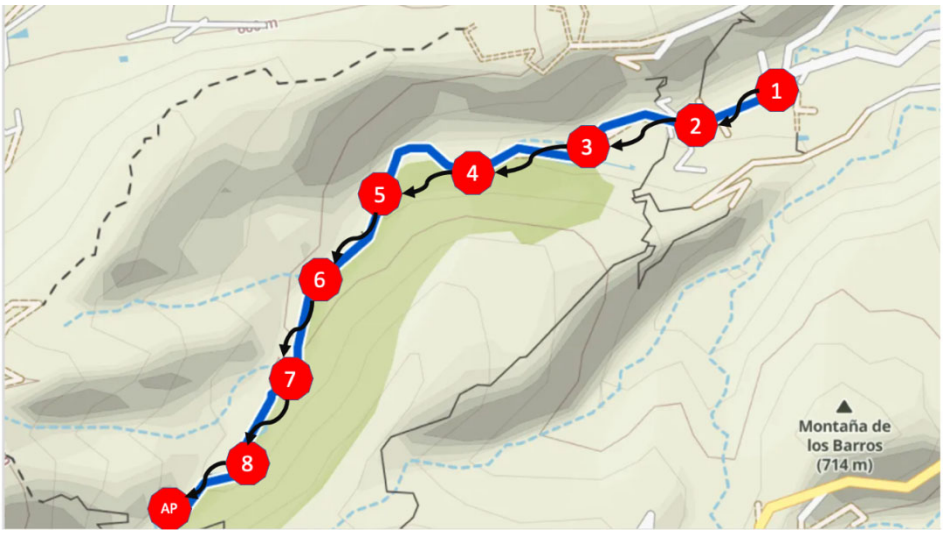

Figura 4.1. Despliegue de los equipod en la ruta.

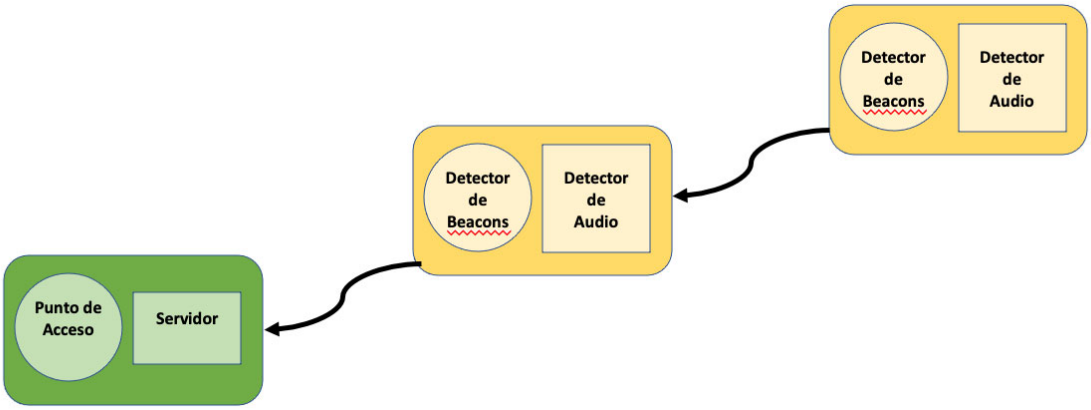

Figura 4.2. Nodos de la red Ad-Hoc.

cableado [16]. Esta es una de sus principales ventajas, porque los nodos pueden desplazarse en una zona amplia sin ataduras a un cable (que suele restringir drásticamente la zona de desplazamiento). Lasredes inalámbricas se pueden clasificar según el área de aplicación y el alcance en: *Wireless Personal Area Network* (*WPAN*), *Wireless Local Area Network* (*WLAN*), *Wireless Metropolitan Area Network* (*WMAN*) y *Wireless Wide Area Network* (*WWAN*) [17]. En la

Figura 2.3 se muestra un esquema gráfico de la clasificación anterior, indicando qué tecnologías inalámbricas son las más utilizadas para cada tipo de red.

En las WLAN se definen dos modos de configuración, el modo Ad-Hoc y el modo infraestructura [18]. El modo que se emplee para la topología de la red determina la manera en la que se comunican los dispositivos entre sí y los protocolos de encaminamiento que se deben utilizar para poder hacer posible dicha comunicación. Debido a las características de nuestro TFG, a continuación, se explica con mayor detalle el modo Ad-Hoc.

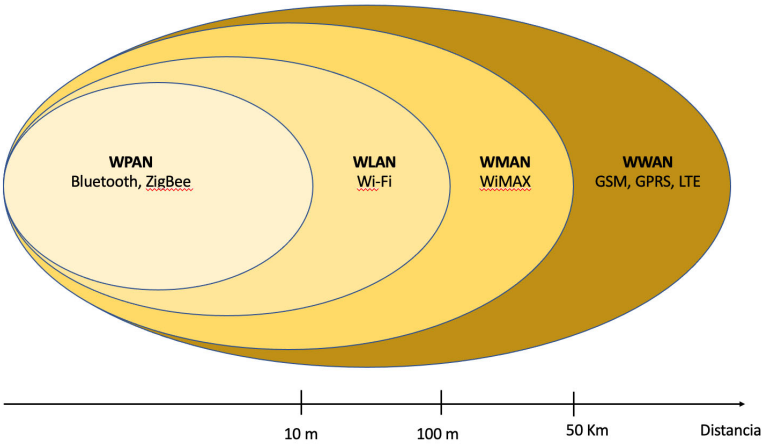

Figura 4.3. Cladificación de las redes inalámbricas según su alcance y principales tecnologías usada para cada tipo.

Fuente modificada: https://upcommons.upc.edu/bitstream/handle/2117/100918/LM01_R_ES.pdf

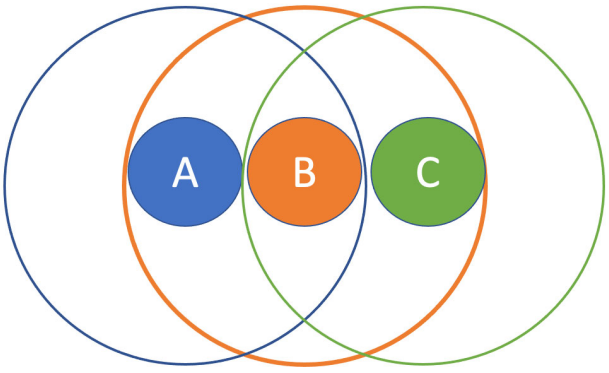

Figura 4.4. El nodo A se comunica con el nodo C a través del nodo B.

Las redes Ad-Hoc se caracterizan por el hecho de que no poseen elementos fijos o administración centralizada de ningún tipo. Están formadas por estaciones móviles capaces de conectarse entre sí definiendo diferentes topologías que pueden cambiar con frecuencia. Cada estación puede actuar como encaminador de los paquetes recibidos hacia el destino correspondiente [19]. De esta forma se permite la comunicación entre dos nodos sin necesidad de que exista un alcance directo entre la fuente y el destinatario como se muestra en la Figura 2.4.

Los protocolos de encaminamiento son una parte importante dentro de la configuración de las redes Ad-Hoc. En una topología de red en la que los nodos pueden estar en constante movimiento, es el protocolo de encaminamiento el responsable de encontrar el camino ideal para la transmisión del mensaje que debe dar varios saltos para alcanzar el nodo destino [20]. Es decir, son el conjunto de reglas que gobiernan la comunicación entre los dispositivos dentro de la red y garantizan que los datos son recibidos correctamente o no.

Un protocolo de encaminamiento para redes Ad-Hoc debe cumplir una serie de propiedades, como ser simple y tener así un tiempo de procesado mínimo, evitar la aparición de bucles, tener un consumo energético reducido o emplear un mecanismo rápido de re-encaminamiento para no introducir retrasos calculando nuevas rutas [21]. De esta forma se garantiza la comunicación a través de un medio que puede llegar a ser vulnerable y cuenta con un ancho de banda limitado. Los protocolos de encaminamiento en general se pueden clasificar de acuerdo con tres criterios [22], a continuación, se va a profundizar en cada uno de ellos y explicar si alguno se adapta a la red Ad-Hoc del TFG:

- *Centralizados o distribuidos:* los centralizados hacen uso de una estación central que se encarga de la toma de decisiones de encaminamiento. Para ello recopila información de cada enlace de la red y generan así una única tabla de encaminamiento que debe consultarse cuando otros nodos necesiten enviar un mensaje. Los distribuidos fuerzan que cada nodo de la red sea responsable de tomar sus propias decisiones de encaminamiento. El protocolo de encaminamiento utilizado para redes Ad-Hoc debe ser de tipo distribuido.

- *Adaptativos o estáticos:* tienen la capacidad de cambiar el comportamiento de la red en función de su estado, pudiendo elegir un camino en función de la congestión de la red u otros posibles problemas, al contrario que un protocolo estático. En una red Ad-Hoc donde los nodos son móviles y la topología de la red puede cambiar con frecuencia es necesario usar un protocolo de tipo adaptativo.

- *Reactivos, proactivos o híbridos* [23]: los reactivos no necesitan que cada nodo de la red mantenga la información de encaminamiento siempre disponible, ya que se actualiza en el momento que sea necesario según la demanda. Los proactivos intentan mantener la información correcta en cada nodo de la red para ya tenerla disponible antes de que se necesite. Los híbridos emplean ambas técnicas, mantiene las rutas de manera proactiva pero únicamente en unos pocos nodos de la red. Actualmente no hay un consenso de que estrategia es la mejor a seguir en una red Ad-Hoc. Por ello podemos encontrar redes de este tipo que usan protocolos reactivos como Ad-Hoc On Demand Distance Vector (AODV) [24] y otras redes que usan protocolos proactivos como Optimized Link-State Routing Protocol (OLSR) [25].

Teniendo todo esto en cuenta y que los nodos de la red a diseñar son estáticos, es decir, no se van a desplazar, y que siempre son visibles entre ellos, se ha decidido no utilizar ningún protocolo de encaminamiento para la red. Esto implica que la red no soporta movimientos de nodos y que los nodos únicamente se comunican con el siguiente nodo de la red.

3 Tecnologías utilizadas

En el presente capítulo se muestra una descripción detallada de los dispositivos físicos (RPi, equipos *WiFi* y *USB*), aplicaciones informáticas instaladas (Raspbian y sus componentes básicos), y protocolos de comunicaciones usados para la correcta implementación de nuestra red se sensores.

3.1 Hardware

En este apartado se presenta una breve descripción de los diferentes elementos físicos utilizados para el desarrollo de este TFG.

3.1.1 Raspberry Pi

RPi es un ordenador de bajo coste y de tamaño reducido [26]. Gracias a su versatilidad se ha convertido en una herramienta para desarrollar todo tipo de proyectos. Actualmente, se pueden encontrar diferentes modelos, con diferentes especificaciones que le permiten adaptarse a todo tipo de escenarios de trabajo. A continuación, en la Tabla 3.1, se realiza una comparativa de las especificaciones técnicas de los modelos existentes [27].

Como se muestra en la Tabla 3.1, la RPi está compuesta por un System on a Chip (SoC), una memoria *Random Access Memory* (RAM), puertos de entrada y salida USB y conectividad de red. Además, cuenta con una ranura para tarjetas micro *Secure Digital* (SD) y con un *General Purpose Input/Output* (*GPIO*). El GPIO permite la conexión de numerosos periféricos ya sean de entrada o de salida. En la Figura 3.1 se muestran los tipos de pines con los que cuenta el GPIO.

En nuestro caso vamos a trabajar con los modelos RPi de la Figura 3.2, y Figura 3.3. Ambos modelos se adaptan a las necesidades de nuestro proyecto ya que cuentan con la potencia necesaria para ejecutar el software y con conexión WiFi integrada, necesaria para las comunicaciones inalámbricas.

Ambos modelos presentan ligeras diferencias que no se recogen en la Tabla 3.1. La RPi modelo 4 se alimenta con un cargador USB tipo C y cuenta con dos entradas micro *High Definition Multimedia Interface* (*HDMI*). Sin embargo, la RPi modelo 3B+ se alimenta con un cargador microUSB y solo dispone de una única entrada HDMI.

3.1.2 Mini micrófono USB

Debido a las características del TFG, se requiere de un micrófono con entrada tipo USB y que tenga un tamaño reducido. Con lo cual, el micrófono utilizado para nuestro proyecto es el modelo MI-305, que se muestra en la Figura 3.4.

	SoC	CPU	RAM	USB	Red
RPi Modelo A+	BCM2835	700 MHz	256 MB	1	NO
RPi Modelo B+	BCM2835	700 MHz	512 MB	4	Ethernet
RPi 2 Modelo B	BCM2836	900MHz	1 GB	4	Ethernet
RPi 3 Modelo B	BCM2837	1,2 GHz	1 GB	4	Ethernet WiFi
RPi 3 Modelo B+	BCM2837 B0	1,4 GHz	1 GB	4	Ethernet WiFi
RPi 4 Modelo B	BCM2711	1,5 GHz	1,2-4 GB	2(3.0) 2(4.0)	Ethernet WiFi
RPi Zero W	BCM2835	1 GHz	512 MB	micro	WiFi Bluetooth

Tabla 3.1. Comparativa de los diferentes modelos de Raspberry Pi.

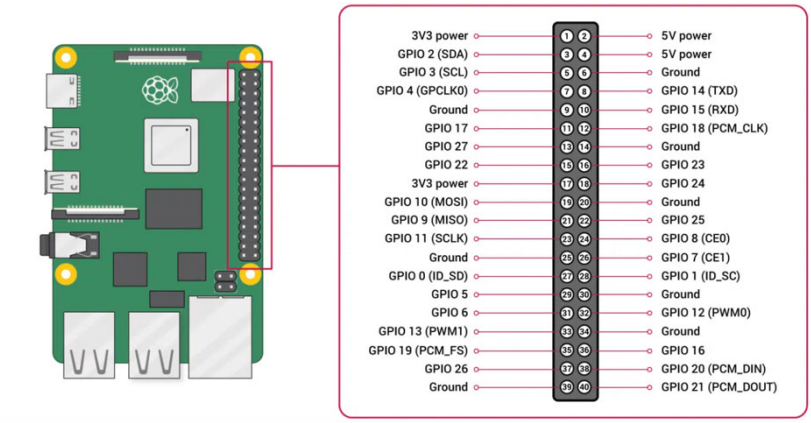

Figura 3.2. Esquema del GPIO.

Fuente: https://www.hwlibre.com/gpio-raspberry-pi/

Figura 3.3. Placa Raspberry Pi 4 Modelo B.

Fuente: https://es.wikipedia.org/wiki/Raspberry_Pi.

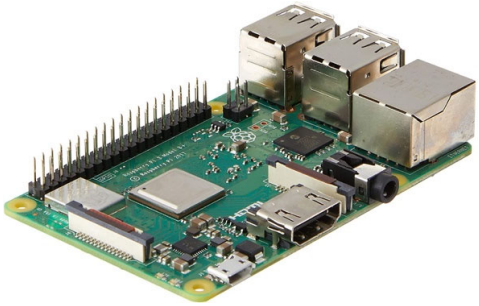

Figura 3.3. Placa Raspberry Pi 3 Modelo B+.

Fuente: https://www.amazon.es/Raspberry-Pi-Modelo-Placa-Color/dp/B07BFH96M3

Este dispositivo USB se caracteriza por ser bastante pequeño y fácil de instalar, ya que cuenta con la ventaja de que es *plug and play,* es decir, no requiere de ningún controlador para su funcionamiento. Es compatible con todos lo sistemas operativos adaptándose así perfectamente a las RPi del TFG.

Figura 3.4. Micrófono de tipo USB.

Fuente: https://www.tiendatec.es/informatica/perifericos/sonido/484-mini-microfono-usb-negro-compatible-raspberry-pi-mi-305-8404840080017.html

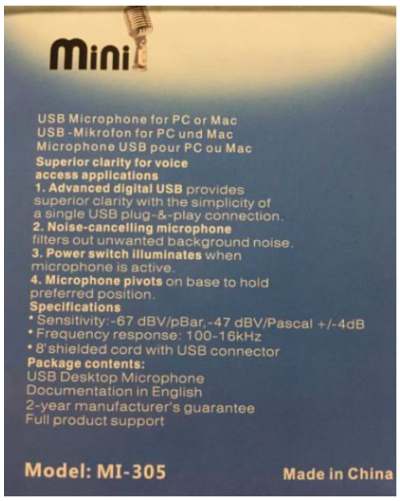

Figura 3.5. Caracterítcas del micrófono USB.

El dispositivo cuenta con los siguientes detalles técnicos:

1) Respuesta en frecuencia: 100-16 KHz.
2) Dimensiones del dispositivo: 22,2 mm x 18,3 mm x 7,0 mm / 0,9” x 0,7” x 0,3”.
3) Peso del dispositivo: 2,2g / 0,1 oz.
4) Sensibilidad: -67 dBV/pBar, -47 dbV/Pascal, +/-4 dB.
5) Cancelación de ruido, filtrando así el ruido de fondo no deseado.

Las especificaciones mencionadas anteriormente han sido obtenidas de la caja del producto, la cual se muestra en la Figura 3.5. Esto se debe a que en la página web del micrófono [28] apenas se recoge información acerca del producto

3.1.3 Adaptador WiFi USB

Debido a las necesidades del TFG, en uno de los equipos es necesario disponer de dos interfaces inalámbricas. Nuestros equipos cuentan con una única interfaz inalámbrica interna, con lo cual se necesita de un dispositivo tipo USB que permita configurar una interfaz adicional.

Para nuestro proyecto se utiliza el dispositivo TP-LINK AC600 Archer T2U Plus, que se muestra en la Figura 3.6. Este dispositivo de posición ajustable cuenta con las siguientes características inalámbricas [29]:

- Estándares inalámbricos: IEEE 802.11 b/g/n 2,4 GHz, IEEE 801.11 a/n/ac 5 GHz.
- Velocidad inalámbrica: 600 Mbps.
- Frecuencia: 2,4 GHz, 5 GHz.
- Modos inalámbricos: Ad-Hoc / Modo infraestructura.
- Seguridad: *Wired Equivalent Privacy (WEP)*, *Wi-Fi Protected Access/Pre Shared Key (WPA/PSK)*, WPA-PSK/WPA2-PSK.
- Tecnología de modulación: *Differential Binary Phase Shift Keying (DBPSK), Differential Quadrature Phase Shift Keying (DQPSK), Complementary Code Keying (CCK), Orthogonal Frequency Division Multiplexing (OFDM), 16-Quadrature Amplitude Modulation (QAM),* 64-QAM, 256-QAM.

La configuración de este equipo requiere de una serie de pasos. En primer lugar, se necesita descargar e instalar el controlador correspondiente. Desde la web oficial [30] únicamente se puede descargar el controlador para Windows y Mac, con lo cual el driver para Linux se descargar mediante los siguientes órdenes:

sudo apt install dkms git

sudo apt-get install raspberrypi-kernel-headers

git clone https://github.com/aircrack-ng/rt8812au.git

Una vez descargado el repositorio, se tiene que acceder a él mediante la orden *cd rtl88** y ejecutar las siguientes órdenes teniendo en cuenta que se trabaja con una RPi modelo 3B+ con sistema de 64 b:

sed -i 's/CONFIG_PLATFORM_I386_PC = y/CONFIG_PLATFORM_I386_PC = n/g' Makefile

sed -i 's/CONFIG_PLATFORM_ARM64_RPI = y/CONFIG_PLATFORM_ARM64_RPI = n/g' Makefile

Figura 3.6. Adaptador WiFi USB.

Fuente: https://www.amazon.es/TP-LINK-Adaptador-Receptor-600Mbps-T2U/dp/B07P681N66.

Gracias a la instalación del controlador, mediante la orden *sudo make dkms_install,* el adaptador es reconocido por la RPi y se puede pasar a su configuración que se explica en detalle en el apartado 4.7.

3.2 Software

En este apartado se resumen los distintos programas que se utilizan para el desarrollo del TFG.

3.2.1 Raspbian

Existe una gran cantidad de sistemas operativos que pueden trabajar con la RPi [31]. Algunos de los más destacados serían: *Windows 10 Intenet of Things* (*IoT*), *Ubuntu Desktop*, *CentOS, Kali Linux* y se pueden encontrar muchas opciones más. Sin embargo, el más destacado y utilizado es Raspberry Pi *Operating System* (RPiOS).

RPiOS, antes conocido como Raspbian, es una distribución de Linux basada en Debian [32]. Este es desarrollado por el mismo fabricante, con lo cual está optimizado para funcionar en procesadores ARM, concretamente en el de la RPi. De esta forma, se consigue generar el menor número de fallos posibles y se puede sacar todo el potencial a la *Central Processing Unit* (*CPU*).

Esta distribución fue desarrollada en 2012 [32] y distribuida como el sistema operativo principal para las RPi desde 2015. Actualmente, se puede trabajar con 3 ediciones diferentes de Raspbian, ajustándose así a los recursos de los que disponga la RPi:

- *Escritorio* PIXEL: incluye una gran cantidad de programas y software recomendados por el distribuidor. Esta es la versión más completa y se dispondría de muchas herramientas desde el primer momento.

- *Básica*: cuenta únicamente con el escritorio y los programas básicos. De esta forma se puede ahorrar memoria en la tarjeta *Secure Digital* (SD) e instalar solo aquellos programas y software a utilizar, consiguiendo así un uso mucho más optimizado. La última versión lanzada es la Debian 11.2 denominada *bullseye.*

- *Lite*: cuenta con lo básico para que el equipo pueda arrancar, siendo necesario que el usuario instale todos los programas. El usuario requiere ciertos niveles de conocimiento ya que se trabaja sin entorno gráfico

En el TFG se usa la versión Debian 11.2 que cuenta con el entorno de escritorio *Lightweight X11 Desktop Environment* (*LXDE*) e incluye numerosos paquetes de software como Python 3, que pueden ser utilizados para el desarrollo de nuestro proyecto.

Para llevar a cabo la instalación del mismo se hace uso del software multiplataforma de RPi llamado *Raspberry Pi Imager*. Este software puede ser descargado desde el sitio web oficial de Raspberry Pi [33]. Una vez descargado, es necesario elegir el sistema operativo que se va a utilizar y la tarjeta SD donde se quiere almacenar, como se muestra en la Figura 3.7.

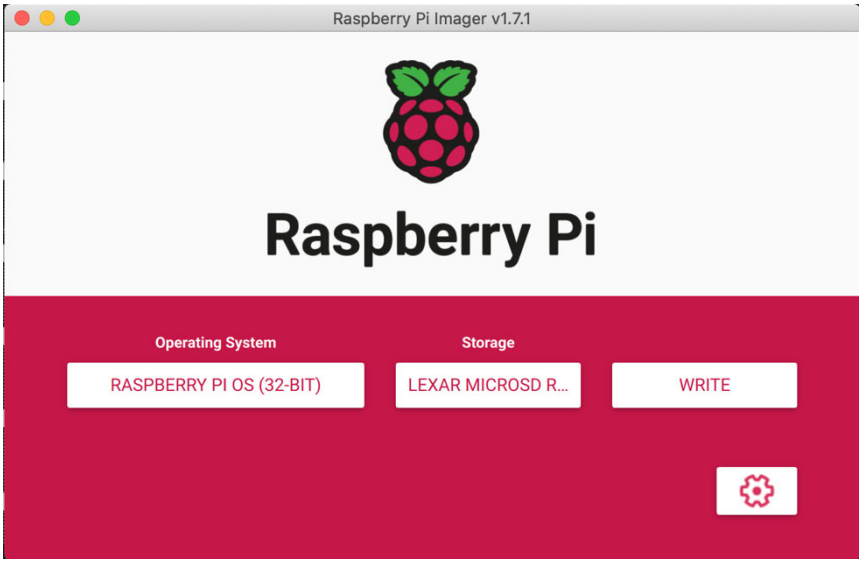

Figura 3.7. Interfaz del software Rasoberry Pi Imager.

Una vez la imagen del SO esté almacenada en la tarjeta SD, únicamente hay que introducirla en la RPi y encenderla. Cuando arranque la RPi se inicia el proceso de instalación y configuración del SO en el propio equipo.

3.2.2 Python

Python es un lenguaje de programación interpretado, es decir, se ejecuta instrucción a instrucción, sin pasar por un proceso de compilación. Se caracteriza por ser multiplataforma, con lo cual puede ser ejecutado en varios sistemas operativos, convirtiéndolo así en un lenguaje de programación utilizado en múltiples escenarios. Además, es posible crear programas usando diferentes estilos de programación ya que soporta varios paradigmas como orientación a objetos, estructurada, programación imperativa y programación funcional [34].

Uno de los lenguajes de programación más utilizado en RPi es Python [35], ya que el propio sistema operativo cuenta con un *Integrated Development Environment* (*IDE*) para trabajar con este lenguaje de programación y es el que se usa en este proyecto para el desarrollo de los diferentes programas.

En nuestro caso la versión de Raspbian instalada en los equipos ya cuenta con Python 3, con lo cual no es necesario instalar o actualizar la versión del lenguaje de programación para el desarrollo del TFG.

3.2.3 Servidor Web Apache

Apache es un servidor web de código abierto que usa el *HyperText Transfer Protocol* (HTTP) [36]. Se caracteriza por su modularidad y su constante actualización por parte de la comunidad, otorgándole un alto nivel de seguridad. Además, es gratuito y multiplataforma, pudiendo ser usado tanto en servidores Windows como Linux.

La funcionalidad principal de este servicio web es servir a los usuarios los archivos alojados en el servidor a través de HTTP. Para ello el cliente realiza una petición HTTP a través de uno de los diversos navegadores como puede ser Chrome, Safari, Firefox… Una vez el servidor reciba esta petición, empleando el mismo HTTP, entregaría los textos, imágenes, estilos… que conforman la web alojada en el servidor.

Este servicio ha llegado a ser muy popular debido a su modularidad, es decir, su capacidad de subdividir sus servicios en partes diferentes [37]. De esta forma el usuario puede activar y desactivar funcionalidades adicionales.

En el TFG se requiere de un servidor instalado en el nodo final de la red. Este se encarga de almacenar archivos de datos de los diferentes usuarios. Además, también atiende las peticiones HTTP realizadas por estos para así poder tener acceso a los datos alojados en el servidor.

3.3 Estándares y protocolos

En esta sección se presentan los diferentes elementos utilizados para la comunicación inalámbrica y los protocolos usados para el desarrollo de este TFG.

3.3.1 Wireless Fidelity

La familia de estándares de comunicación IEEE 802.11 denominado comercialmente WiFi define el uso de los dos niveles más bajos de la arquitectura *Open System Interconnection* (*OSI*): físico y enlace de datos [38].

El nivel físico se divide en dos:

- *Physical Medium Dependent* (*PMD*): define los métodos de modulación y codificación necesarios para poder transmitir y recibir datos a través del medio. Entre mayor eficiencia se consiga en la codificación de los datos, mejores tasas de bits se podrán alcanzar dentro del mismo ancho de banda.

- *Physical Layer Convergence Procedure* (*PLCP*) y la *Physical Medium Dependent* (*PMD*). La función principal es gestionar las tramas de niveles superiores, añadiendo preámbulos y cabeceras en el formato adecuado para la PMD.

A lo largo de los años, han ido surgiendo diferentes estándares, que han ido modificando las técnicas y frecuencias utilizadas, en la Tabla 3.2 se puede ver las principales modificaciones realizadas en cada estándar. Esto provoca que en el IEEE 802.11 se definan diferentes técnicas para ensanchar el ancho de banda, ya que se emplea más del mínimo necesario para conseguir protección contra posibles interferencias. Estas técnicas son [39]:

- *Frequency Hopping Spread Spectrum* (*FHSS*).

- *Direct Sequence Spread Spectrum* (*DSSS*).

- *Orthogonal Frequency-Division Multiplexing* (*OFDM*).

El estándar IEEE 802.11 (versiones anteriores a WiFi 5 como se identifican comercialmente) utiliza 14 canales con un ancho de banda de 22 MHz en la banda de los 2,4 GHz y espaciados 5 MHz entre sí. En la banda de los 5 GHz emplea 25 canales no solapados, con un ancho de banda de 20 MHz.

El nivel de enlace se divide en dos: el *Logical Link Control* (*LLC*) y el *Control de Acceso al Medio* (*MAC*). Debido a que en el medio inalámbrico no se puede emplear mecanismo de detección de colisiones, el MAC se encarga de arbitrar el acceso al medio radio compartido para evitar así posibles colisiones. Para ello se definen dos técnicas de coordinación [40]:

- *Point Coordinated Function* (*PCF*): solo está disponible en el modo infraestructura. Se pueden distinguir dos periodos de tiempo, un periodo con conflicto, el cual es gestionado por *Distributed Coordination Function* (*DCF*) y otro periodo sin conflicto en el que el nodo de coordinación asigna intervalos de tiempos a las estaciones para transmitir. Sin embargo, esta técnica de coordinación nunca llegó a ser utilizada en la práctica.

- *DCF*: es una técnica de coordinación distribuida que emplea el algoritmo *Carrier Sense Multiple Access with Collision Avoidance* (*CSMA/CA*). Este algoritmo se basa en escuchar el canal para verificar que esté libre. En el caso de estar libre, envía una trama *Request To Send* (*RTS*) indicando a las demás estaciones que no retrasmitan. Además, en esta trama se especifican las estaciones destino y origen y el tamaño de la trama a transmitir. Si la estación destino lo recibe correctamente, enviará una trama *Clear To Send* (*CTS*) indicando que está preparada para la emisión. Tras emitir el mensaje, la estación emisora espera una respuesta de confirmación del destinatario indicando si la recepción del mensaje fue correcta (ACK) o incorrecta (NACK). Esta técnica puede provocar un aumento de la probabilidad de que aparezcan colisiones en la red ya que las estaciones tienen que esperar a que el medio esté libre para poder retransmitir. Sin embargo, la nueva versión 802.11ax, también conocida como WiFi 6, emplea *Orthogonal frequency-division multiple Access* (*OFDMA*) [41] para solucionar este problema, dando acceso al medio a varias estaciones de forma simultánea y consiguiendo así una asignación del ancho de banda mucho más eficiente. OFDMA divide un canal en asignaciones de frecuencia más pequeñas, llamadas unidades de recursos (RU) que pueden ser asignadas por el PA a diferentes estaciones. Antes un canal de 20 MHz constaba de 64 subportadoras empleadas todas por una misma estación, gracias a OFDMA el número de subportadoras de ese canal aumenta a 256. De esta forma se consigue reducir el tiempo de transmisión

entre estaciones y una mayor eficiencia. Por las características de la comunicación y número de usuarios del TFG es suficiente con usar versiones de WiFi anteriores a WiFi 6.

Por otra parte, en este estándar se definen dos tipos de topologías, es decir, dos modos de operación: infraestructura y Ad-Hoc. El más común es el modo infraestructura, en el que los nodos se conectan a la red cableada a través de las estaciones bases, lo que se conoce *Basic Service Set* (*BSS*). Sin embargo, el otro modo de trabajo es el que nos interesa a nosotros, es decir, el modo Ad-Hoc. En este modo se trabaja con *Independent Basic Service Set* (*IBSS*) en las que no existen sistemas de distribución. Los dispositivos se comunican entre sí, sin equipos que centralicen la comunicación.

	Estándar							
	802.11	**802.11a**	**802.11b**	**802.11g**	**802.11n**	**802.11ac**	**802.11ad**	**802.11ax**
Fecha	1997	1999	1999	2003	2009	2013	2012	2019
Banda de frecuencia (GHz)	2,4	5	2,4	2,4	2,4/5	5	60	2,4/5
Tecnología de transmisión	DSSS, FHSS	OFDM	DSSS	DSSS, OFDM	OFDM	OFDM	OFDM	OFDMA
Flujo de datos	2 Mbps	54 Mbps	11 Mbps	54 Mbps	600 Mbps	6,93 Gbps	7 Gbps	9,6 Gbps

Tabla 3.2. Versiones del estándar IEEE 802.11.

	Bluetooth 1.2	**Bluetooth 2**	**Bluetooth 3**	**Bluetooth 4**	**Bluetooth 5**
Velocidad	1 Mbps	2,1 Mbps	24 Mbps	24 Mbps	50 Mbps

Tabla 3.3. Comparativa de las velocidades entre las diferentes versiones de Bluetooth.

3.3.2 Bluetooth Low Energy

Bluetooth es una tecnología inalámbrica que usa enlaces de radio de corto alcance que permite la comunicación de datos entre dispositivos digitales, reemplazando así las conexiones de cable entre equipos [42]. Emplea la banda *Industrial Scientific and Medical (ISM)* de 2,4 GHz, la misma frecuencia que WiFi, con lo cual puede transmitir a través de objetos sólidos no metálicos.

En cuanto al alcance, los dispositivos Bluetooth se pueden clasificar en diferentes clases dependiendo de su potencia de transmisión y de su cobertura efectiva. Dispositivos de diferentes clases se pueden comunicar entre sí, siendo el dispositivo más potente el que marca el límite de la comunicación. Por ejemplo, un dispositivo de clase 2, con solo 20 m de alcance y un dispositivo de 100 m de alcance pueden llegar a comunicarse a más de 20 m. A continuación, se detallan las características de cada una de las clases [43]:

- *Clase 1*: tiene un alcance de hasta 100 m, con una potencia de 100 mW.
- *Clase 2*: tiene un alcance de hasta 20 m, con una potencia de 2,5 mW.
- *Clase 3*: tiene un alcance de hasta 1 m, con una potencia de 1 mW.
- *Clase 4*: tiene un alcance de hasta 0,5 m, con una potencia de 0,5 mW.

Actualmente, existen hasta 5 versiones diferentes de Bluetooth [44]. En la Tabla 3.3 se recoge una comparación entre las velocidades de transmisión que se pueden alcanzar con cada una de ellas, siendo la versión Bluetooth 4.0 la que se utiliza en este TFG.

Bluetooth 4.0 fue lanzado en diciembre de 2009 y fue en esta versión donde apareció el estándar de *Bluetooth Low Energy* (*BLE*) destinado para dispositivos que funcionen con Internet de las cosas. En este estándar se especifica por primera vez un mecanismo muy simple para el escaneo y la detección de beacons, que es la base para los servicios de localización y detección de dispositivos.

Para el desarrollo del TFG se ha configurado el dispositivo RPi para escanear beacons. De esta forma se puede registrar el paso de personas, que mediante el uso de un dispositivo transmisor de beacons, podrían ser detectados por la RPi. A continuación, se estudian ambos procesos, tanto el escaneo de beacons por parte de la RPi como la transmisión de beacons por parte del dispositivo.

En la transmisión de beacons, el objetivo principal es la promoción de un dispositivo o en este caso, de personas por medio de las tramas Beacon utilizando la arquitectura de BLE. Estas tramas Beacon o tramas de *Advertising* emplean 3 canales de los 40 en los que se divide el espectro de 2,4 Ghz, estos canales son el 37, 38 y 39 [45]. Durante este proceso de transmisión de tramas Beacon, o también conocido como Intervalo de *Advertising*, se envían paquetes periódicamente en cada uno de los 3 canales comentados.

Las tramas presentan la estructura que se puede ver en la Figura 3.8 y está compuesta por los siguientes campos:

- *Preámbulo*: una serie fija de bits que permiten identificar la trama como una trama Bluetooth.
- *Acces Address*: debe ser siempre 0x8E89BED6 según la especificación de Bluetooth.
- *Código de redundancia cíclica, CRC*: permite comprobar que los datos dentro de la trama son correctos.
- *Packet Data Unit:* este campo se divide a su vez en: cabecera, *Advertiser Address* y *Optional Payload* 31 B

A continuación, se profundiza en cada uno de los diferentes campos del *Packet Data Unit*. La cabecera está compuesta por 2 B. Los 4 primeros bits del primer byte se emplean para identificar el tipo del Advertisements:

- ADV_IND = 0x0000
- ADV_DIRECT_IND = 0x0001
- ADV_NONCONN_IND = 0x0010
- ADV_SCAN_IND = 0x0110
- SCAN_REQ = 0x0011
- SCAN_RSP = 0x0100
- CONNECT_IND = 0x0101

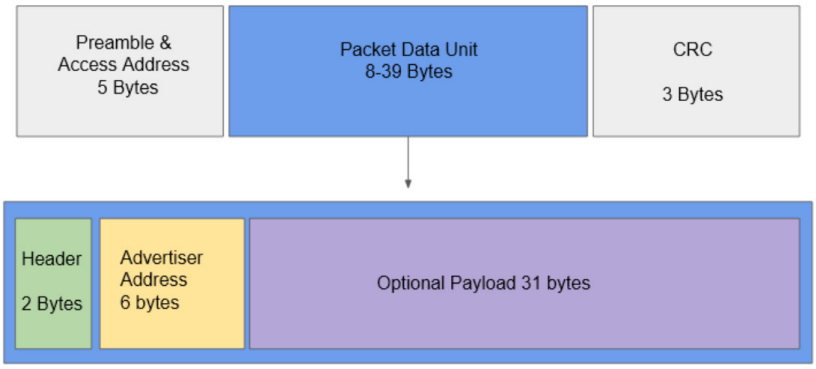

Figura 3.8. Estructura de una trama BLE.
Fuente: https://repositorio.upct.es/bitstream/handle/10317/10081/tfg-ros-dis.pdf?sequence=1&isAllowed=y

Otro de los bits está reservado para usos futuros con lo cual no se utiliza actualmente. Otros dos bits del primer byte identifican el tipo de MAC, es decir, definen si la dirección que acompaña la cabecera es pública o aleatoria.

El segundo byte de la cabecera denota la longitud total entre el campo *Optional Payload* y el *Advertiser Address*, ya que estos pueden variar de tamaño en función del tipo de trama. Debido a esto, los campos *Optional Payload* y *Advertiser Address* pueden presentar diferencias en el uso de los bytes de un tipo de trama a otro.

Debido a estas diferencias entre tramas, se profundiza en el formato de trama especificado por algunas empresas que desarrollaron sus tipos de tramas específicos a partir de la estructura ya creada de BLE, como puede ser iBeacon de Apple o Eddystone de Google [46].

- *iBeacon*: formato desarrollado por Apple. Las señales emitidas pueden llegar a alcanzar los 70 m siempre que la señal no se vea atenuada por barreras arquitectónicas. Un dispositivo al captar esta señal recibirá un identificador (*ID*) y la intensidad de la señal en ese instante. Este ID se compone a su vez de diferentes campos:
    - *UUID*: ID único para la aplicación.
    - *Major*: ayuda a determinar un beacon en concreto. Por ejemplo, permitiría diferenciar las rutas por municipios.

- *Minor*: especifica otra subdivisión. Permite, por ejemplo, diferenciar las rutas de un mismo municipio.

- *Eddystone*: desarrollado por Google y cuenta con algunas similitudes con iBeacon. Sin embargo, en la estructura de su trama cuenta con el campo *Tipo* que especifica el tipo de trama Eddystone que se utiliza. Existen cuatro tipos diferentes de trama:

  - *Eddystone-UID*: para localizar objetos en interiores.

  - *Eddystone-URL*: pensadas para mostrar URLs en los dispositivos que las detecten.

  - *Eddystone-TLM*: para dar información del propio beacon.

  - *Eddystone-EID*: similares a las UID, pero con la diferencia de que el ID está cifrado.

Para el TFG se necesita transmitir poca información, siendo el campo UUID el más importante para su desarrollo, por lo que se utiliza el formato iBeacon.

Como ya se ha visto, un dispositivo BLE puede tanto transmitir beacons, como escanear el medio en busca de estos. Para ello existen dos formas de escanear tramas BLE [47]:

- *Escaneo activo*: en este caso primero se escanea el medio en busca de dispositivos y luego, se les envía un paquete de respuesta a todos los dispositivos que se detecten solicitando más información.

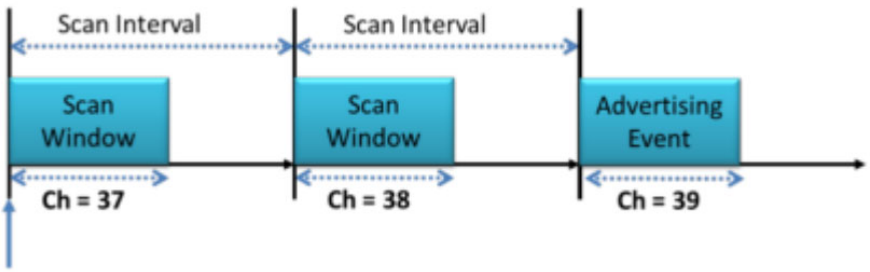

Figura 3.9. Intervalo y ventana de escaneo.
Fuente: https://repositorio.upct.es/bitstream/handle/10317/10081/tfg-ros-dis.pdf?sequence=1&isAllowed=y

- *Escaneo pasivo*: el escáner no realiza envío de tramas. En este caso escanea el medio para ver los tipos de tramas que los dispositivos están enviando.

El escáner comprueba durante un tiempo determinado los tres canales donde puede haber tramas de *Advertising*. Como se muestra en la Figura 3.9, durante la recepción de estas tramas hay un intervalo de escaneo, que es la frecuencia con la que tiene lugar la escucha, y una ventana de escaneo, que es la duración de la escucha de tramas. Además, se debe tener en cuenta que la ventana de escaneo tiene que ser siempre menor o igual al intervalo de escaneo.

Para este TFG se ha elegido el escaneo de tipo pasivo debido a que solo se necesitan los datos de *advertising* de los dispositivos. Con lo cual los equipos de escaneo tienen que buscar este tipo de tramas dentro de su área de cobertura.

3.3.3 Socket de Python

Un socket permite crear una conexión cliente-servidor de forma sencilla, es decir, permite crear un canal de comunicación entre dos equipos a través de la red. En nuestro caso, se configuran los sockets mediante Python, ya que se ha utilizado durante todo el desarrollo del TFG y es un lenguaje sencillo y práctico.

El módulo socket de Python proporciona una interfaz de bajo nivel que permite conexiones *Transmission Control Protocol/Internet Protocol (TCP/IP) y User Datagram Protocol/Internet Protocol (UDP/IP)* para el intercambio de mensajes entre dos máquinas [48]. Concretamente se emplea el TCP, este es un protocolo de la capa de transporte que se caracteriza por ser orientado a conexión [49]. Esto significa que antes de iniciar la transferencia de datos, se realiza una comunicación previa entre los dos equipos para establecer la conexión, garantizando así que la transmisión de datos tiene lugar sin errores.

Además, TCP tiene un mecanismo de control de errores para detectar posibles fallos que se puedan producir y garantizar que todos los datos lleguen en orden. Emplea diferentes métodos como puede ser [50]: numeración de segmentos, confirmaciones de los paquetes recibidos mediante tramas *acknowledgement* (ACK), checksum que permite determinar la integridad de los datos [51], temporizadores y una ventana deslizante tanto en el transmisor como en el receptor que permite determinar si uno de los paquetes no ha llegado en orden, esperando a que se reciba o pidiendo que se retransmita el paquete perdido.

Mediante el socket se crea una comunicación de tipo cliente-servidor. Un servidor es un proceso que espera contactar con un proceso cliente para proporcionarle algún tipo de servicio. El proceso que se debe realizar para que entre ambos se inicien las comunicaciones se muestra en la Figura 3.10. En esta figura se muestran las llamadas para la creación de un socket de comunicación con conexión, conocido como *stream socket (connection-oriented)* [52].

El servidor crea un extremo del socket mediante la llamada *socket()* y le asigna una dirección y un puerto mediante la llamada *bind(),* la dirección y el puerto deben ser conocidos por el cliente para poder establecer la conexión. Luego, mediante la llamada *listen()* el servidor se queda esperando una petición de conexión, la cual será aceptada con la llamada *accept().*

Por otro lado, el cliente crea un socket y le envía la petición de conexión al servidor a través de *connect()*. Si el servidor está disponible, acepta la petición y se completa la creación del socket para el intercambio de datos.

En este TFG se establece una conexión entre los nodos de la red mediante socket para el envío de los archivos de datos de los usuarios. De esta forma, el archivo de un usuario puede pasar de un nodo al siguiente para poder ser utilizado cuando el usuario alcance la altura del siguiente nodo.

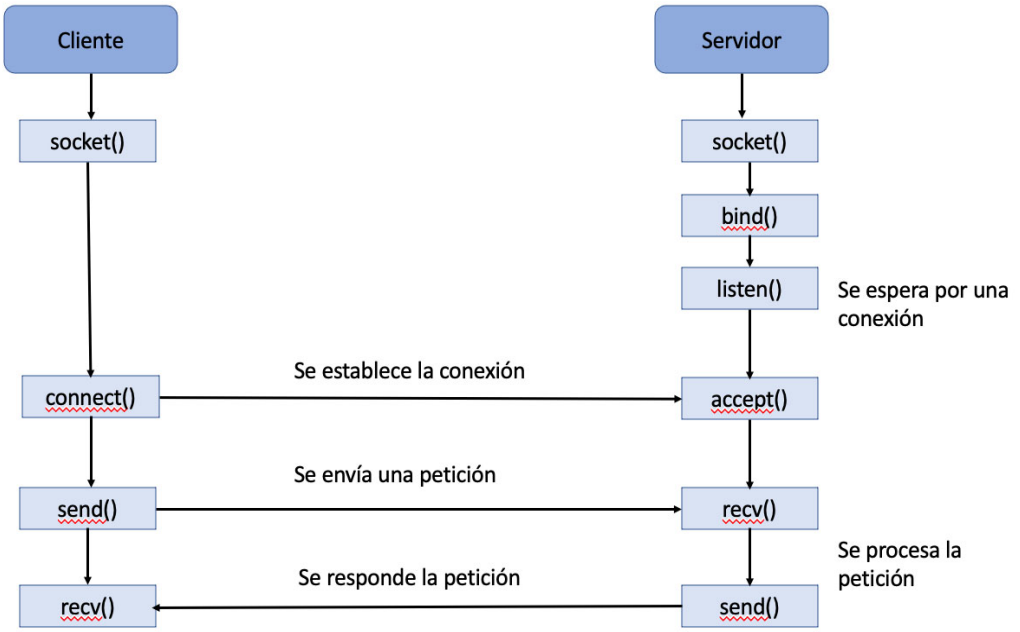

Figura 3.10. Esquema del intercambio de mensajes cliente/servidor.

4 Montaje y configuración del sistema

En este apartado se muestran los pasos seguidos para montar la plataforma de experimentación, así como la configuración de los diferentes dispositivos, sus interfaces, los protocolos empleados, y la topología y arquitectura del sistema.

4.1 Esquema, arquitectura y modelo procesal del sistema

El esquema general del sistema es el que se muestra en la Figura 4.1. El sistema está formado por una red Ad-Hoc que consta de 3 nodos en la que se pueden distinguir dos tipos de nodos.

El primer tipo de nodo se encarga tanto del procesado de audio como de la detección de Beacons de Bluetooth. Primero tiene lugar el procesado de audio, cuyo principal objetivo es la detección de personas que pasan hablando cerca del nodo, para ello el nodo hace uso del micrófono USB instalado. Una vez la persona ha sido detectada, se inicia el proceso de detección de Beacons BLE. En este proceso la persona, provista de un terminal que emite Beacons, puede ser identificada y registrar así la hora a la que pasó por ese nodo en un archivo.

El nodo final actúa como servidor y como PA al mismo tiempo. Como servidor se encarga de almacenar todos los archivos de los usuarios generados por el resto de los nodos de la red. Y como punto de acceso permite a los usuarios conectarse a él y tener acceso a su archivo para poder descargarse los datos de la ruta.

El sistema genera un único archivo por persona que se va enviando entre los nodos de la red Ad-Hoc a través de conexiones socket hasta llegar al nodo final.

4.2 Creación de la red Ad-Hoc

La red Ad-Hoc consta de tres nodos fijos que forman una topología de red, como se muestra en la Figura 4.1. En cuanto al envío y recepción de archivos se pueden diferenciar tres tipos de nodos. El primer nodo, el cliente (*i=1*), que es el encargado de crear el archivo, con lo cual, él solo envía

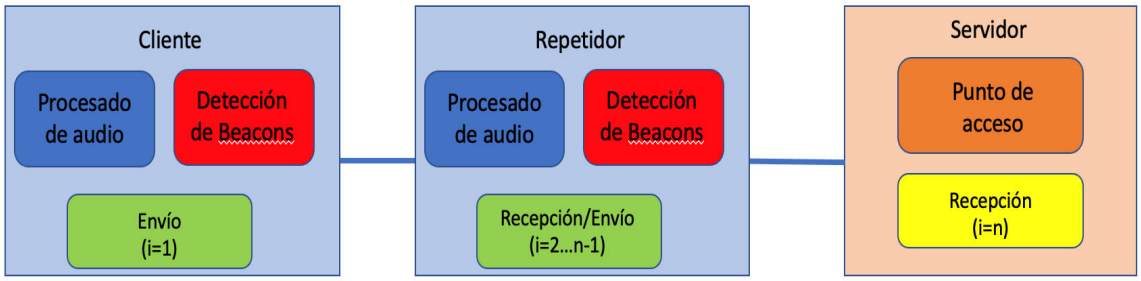

Figura 4.1. Esquema del sistema.

Nodo	Dirección IP (Wlan 0)
Cliente	10.0.0.1
Repetidor	10.0.0.3
Servidor	10.0.0.4

Tabla 4.1. Asignación de direcciones IP a cada nodo de la red.

dicho archivo al siguiente nodo. El segundo tipo de nodo, el repetidor (*i=2..n-1*), que tiene que ser capaz tanto de recibir como de enviar archivos. El envío de estos archivos se realiza mediante sockets creados entre los nodos. Y el último tipo de nodo, el servidor (*i=n*), que actúa tanto como punto de acceso como de receptor de archivos, siendo estos almacenados en el servidor.

Para llevar a cabo la configuración de la red, se empieza asignando una dirección IP estática a cada uno de los nodos de la red. En este caso se van a asignar direcciones IPv4 en el rango de direcciones 10.0.0.0/24, que permite conectar hasta 254 host, siendo más que suficiente para el desarrollo del TFG. Se emplea IPv4 ya que aún es el sistema de direccionamineto predominante en las comunicaciones a través de Internet en la mayoría de los países. En la Tabla 4.1 se muestra la dirección asignada a cada uno de los nodos. Esto es muy importante ya que, al no contar la red con un protocolo de encaminamiento, al realizar la instalación en la zona de campo se debe respetar el orden prestablecido de los nodos.

Para asignarle cada dirección a los equipos se tiene que modificar el archivo */etc/network/interfaces* con la orden sudo nano */etc/network/interfaces* y añadir las líneas que se muestran en la Figura 4.3.

De esta forma, no solo se está asignando la dirección IP estática, sino que se está configurando la interfaz wlan0 para que trabaje en modo ad-hoc. A continuación, se detalla cada línea añadida al archivo /etc/network/interfaces:

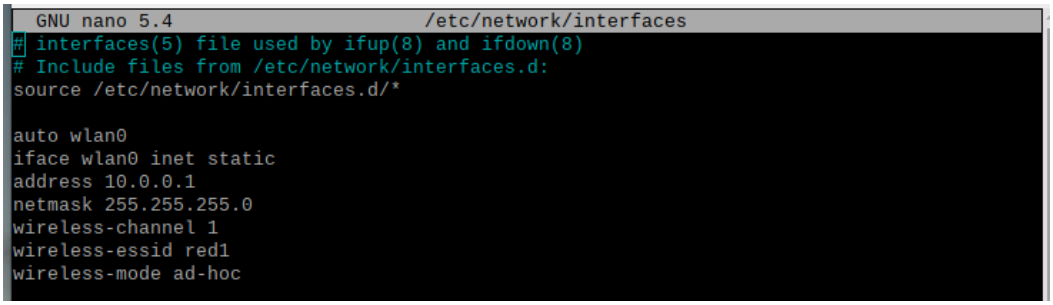

Figura 4.2. Archivo /etc/network/interfaces.

- *auto wlan0*: Interfaz inalámbrica con la que se va a trabajar.
- *address*: Dirección IP que se le quiere asignar a la RPi.
- *wireless-channel*: Canal que se va a utilizar para transmitir. Tienen que ser el mismo para todos los nodos de la red.
- *wireless-essid*: Nombre de la red que van a formar los nodos entre sí. Se tiene que definir el mismo en cada nodo de la red, de esta forma todos pertenecen a la misma red.
- *wireless-mode*: Configura el modo de trabajo del nodo, en este caso, en modo Ad-Hoc.

Después, se debe ejecutar a orden *sudo wpa_cli terminate*, de esta forma nos aseguramos de que se activa el modo Ad-Hoc que se configuró anteriormente. Por último, solo queda ejecutar la orden *sudo service networking restart* para que se apliquen todos los cambios que se han realizado.

Una vez hecho esto, se puede comprobar, mediante la orden *sudo ping 10.0.0.3*, que existe conexión entre dos de los nodos de la red Ad-Hoc, como se muestra en la Figura 4.3.

Como ya se ha comentado, el envío de archivos entre nodos se realiza mediante sockets. Para ello se ejecuta un programa Python en el nodo origen, conocido como cliente, encargado de

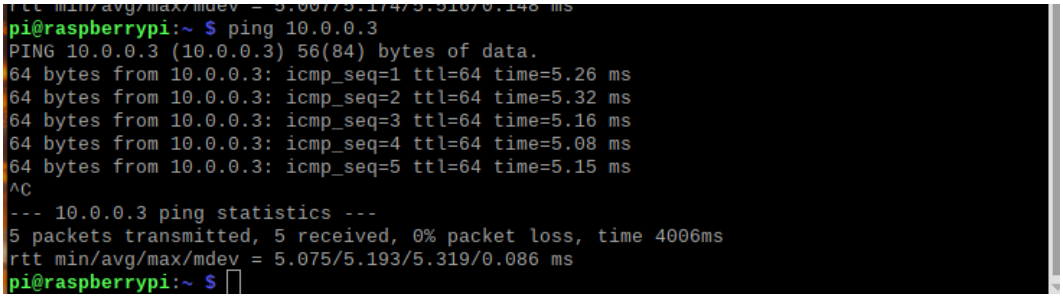

Figura 4.3. Comprobación de la conexión entre nodos.

enviar el archivo y otro programa en el nodo destino, conocido como servidor, encargado de escuchar el puerto definido esperando para recibir el archivo.

4.1.1 Cliente

En primer lugar, para la creación del programa del cliente se debe importar la biblioteca *socket* para crear el socket de comunicación. En nuestro caso, se crea una función llamada *enviar_archivo()* a la cual se le pasa el archivo que se quiere enviar. Esta función se encarga tanto de establecer la conexión como de realizar el envío y será llamada una vez el archivo de datos sea creado. En la Figura 4.4 se muestra el código completo de la función.

```python
def enviar_archivo(rutaDestino):
    SEPARATOR = "<SEPARATOR>"
    BUFFER_SIZE = 4096 # send 4096 bytes each time step
    # Direccion IP del nodo destino
    host = "10.0.0.3"
    # Puerto que se va a utilizar
    port = 5001
    # Ruta del archivo que se quiere enviar
    filename = rutaDestino
    # Size del archivo a enviar
    filesize = os.path.getsize(filename)
    # Se crea el socket pasandole como parametro la direccion IP y el puerto
    s = socket.socket()
    print(f"[+] Connecting to {host}:{port}")
    s.connect((host, port))
    print("[+] Connected.")
    # Se envia tanto el tamano como el archivo
    s.send(f"{filename}{SEPARATOR}{filesize}".encode())
    # comienza el envio
    with open(filename, "rb") as f:
        while True:
            # read the bytes from the file
            bytes_read = f.read(BUFFER_SIZE)
            if not bytes_read:
                # file transmitting is done
                break
            # we use sendall to assure transimission in
            # busy networks
            s.sendall(bytes_read)


    # Se cierra la conexion con el servidorqu
    s.close()
```

Figura 4.4. Código completo de la función.

```python
# Direccion IP del nodo destino
host = "10.0.0.3"
# Puerto que se va a utilizar
port = 5001
```

Figura 4.5. Código del programa de Python.

A continuación, se explican las partes principales del código de la Figura 4.4 para mayor entendimiento de este. Se comienza definiendo la dirección IP del nodo al que se le quiere enviar el archivo y el puerto por el que se envía, como se muestra en la Figura 4.5.

Después, se define la ruta del archivo que se quiere enviar, que es pasada por parámetro, y se obtiene el tamaño del archivo mediante las líneas del código de la Figura 4.6. Esto se debe a que se va a enviar tanto el archivo como el tamaño de este, de esta forma el destinatario sabe cuándo se ha terminado la transferencia.

Por último, se crea el socket y la conexión pasándole la dirección IP y el puerto definido anteriormente, pudiendo realizarse ya el envío, como se muestra en la Figura 4.7. Una vez termine la transferencia, se debe cerrar el socket mediante el método *s.close().*

```
# Ruta del archivo que se quiere enviar
filename = rutaDestino
# Size del archivo a enviar
filesize = os.path.getsize(filename)
```

Figura 4.6. Código del programa de Python.

```
# Se crea el socket pasandole como parametro la direccion IP y el puerto
s = socket.socket()
print(f"[+] Connecting to {host}:{port}")
s.connect((host, port))
print("[+] Connected.")
# Se envia tanto el tamano como el archivo
s.send(f"{filename}{SEPARATOR}{filesize}".encode())
```

Figura 4.7. Código del programa de Python.

```
# IP del servidor y puerto de trabajo
SERVER_HOST = "10.0.0.3"
SERVER_PORT = 5001
```

Figura 4.8. Código del programa de Python.

```
# Se crea el socket pasandole IP y puerto
s = socket.socket()
s.bind((SERVER_HOST, SERVER_PORT))
# Escucha a la espera de alguna conexion
s.listen(5)
print(f"[*] Listening as {SERVER_HOST}:{SERVER_PORT}")
```

Figura 4.9. Código del programa de Python.

4.1.2 Servidor

Para el programa Python del servidor también se debe importar la biblioteca socket. Después, se debe definir la dirección del servidor, es decir, la propia dirección IP de la RPi asignada anteriormente, y el puerto de trabajo, que debe ser el mismo que se definió en el cliente, como se muestra en la Figura 4.8.

Una vez hecho esto, se añade la porción de código de la Figura 4.9, donde se crea el socket pasándole la dirección IP y el puerto de trabajo al método *bind().* Después, el servidor se queda esperando a que se establezca una conexión con algún cliente.

En el momento que una conexión se acepta, porción de código de la Figura 4.10, ya se puede iniciar la recepción de archivos. Se debe tener en cuenta que al finalizar la transmisión se debe cerrar la conexión, como sucedía en el caso del cliente, y que así el servidor se quede esperando nuevas conexiones.

En la Figura 4.11 y la 4.12 se muestra el código completo del programa.

Algunos nodos de la red deben ser capaces tanto de recibir archivos como de enviarlos, como es el caso del nodo denominado repetidor. Eso implica que se debe ejecutar tanto el programa del servidor como del cliente. Esto se consigue ejecutando el código de la Figura 4.13, que nos permite ejecutar ambos programas de forma simultánea.

```
# se acepta la conexion
client_socket, address = s.accept()
print(f"[+] {address} is connected.")
```

Figura 4.10. Código del programa de Python.

```python
while True:
    # device's IP address
    SERVER_HOST = "10.0.0.3"
    SERVER_PORT = 5001
    # receive 4096 bytes each time
    BUFFER_SIZE = 4096
    SEPARATOR = "<SEPARATOR>"


    # create the server socket
    # TCP socket
    s = socket.socket()
    # bind the socket to our local address
    s.bind((SERVER_HOST, SERVER_PORT))
    # enabling our server to accept connections
    # 5 here is the number of unaccepted connections that
    # the system will allow before refusing new connections
    s.listen(5)
    print(f"[*] Listening as {SERVER_HOST}:{SERVER_PORT}")
    # accept connection if there is any
    client_socket, address = s.accept()
    # if below code is executed, that means the sender is connected
    print(f"[+] {address} is connected.")
```

Figura 4.11. Código del programa de Python.

```python
received = client_socket.recv(BUFFER_SIZE).decode()
filename, filesize = received.split(SEPARATOR)
# remove absolute path if there is
filename = os.path.basename(filename)
# convert to integer
#filesize = int(float(filesize))
# start receiving the file from the socket
# and writing to the file stream
rutaDestino = "/home/pi/Documentos/Caminantes/" + filename
with open(rutaDestino, "wb") as f:
    while True:
        # read 1024 bytes from the socket (receive)
        bytes_read = client_socket.recv(BUFFER_SIZE)
        if not bytes_read:
            # nothing is received
            # file transmitting is done
            break
        # write to the file the bytes we just received
        f.write(bytes_read)



# close the client socket
print("Archivo recibido")
client_socket.close()
# close the server socket
s.close()
time.sleep(10)
```

Figura 4.12. Código del programa de Python.

```python
import subprocess

scripts_paths = ("/home/pi/Documentos/SDL_Pi_iBeaconScanner-main/testblescan.py",
'/home/pi/Documentos/SDL_Pi_iBeaconScanner-main/servidor4.py")

ps = [subprocess.Popen(["python", script]) for script in scripts_paths]
exit_codes = [p.wait() for p in ps]

if not any(exit_codes):
    print("Todos los procesos terminaron con éxito")
else:
    print("Algunos procesos terminaron de forma inesperada")
```

Figura 4.13. Programa para la ejecución de múltiples programas.

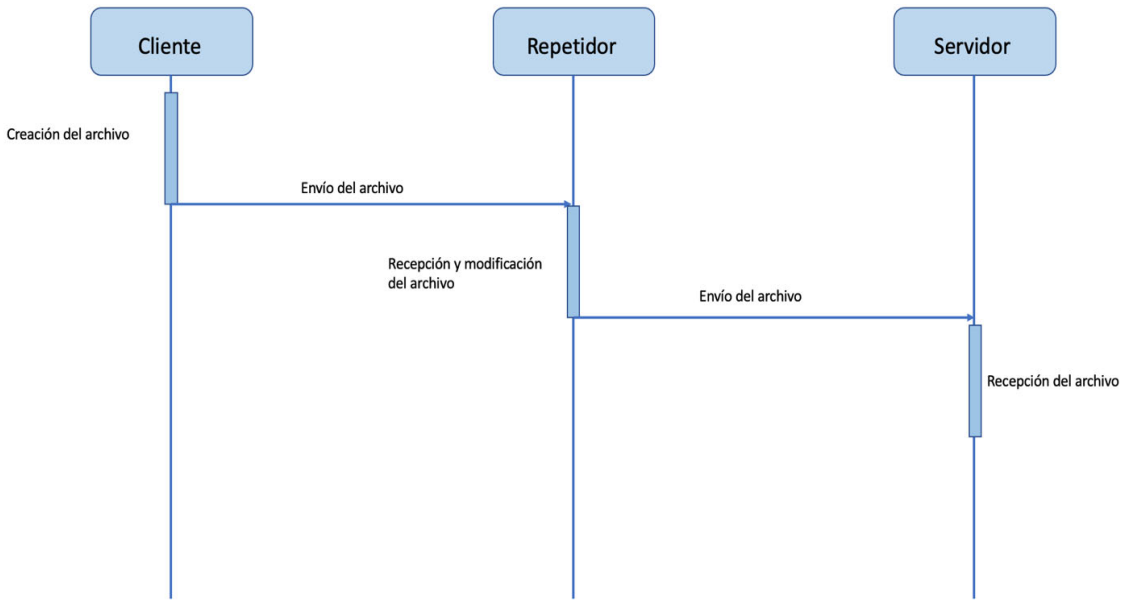

Figura 4.14. Diagrama de secuencia de la comunicación entre equipos.

4.1.3 Secuencia de comunicación

En la Figura 4.14 se muestra un diagrama de secuencia de la comunicación que tiene lugar mediante la ejecución de los programas comentados en los apartados 4.2.1 y 4.2.2. El nodo cliente es el encargado de crear el archivo y de enviarlo mediante la ejecución del programa

cliente al nodo repetidor. Este nodo ejecuta tanto el programa cliente como el programa servidor, de esta forma es capaz tanto de recibir el archivo como de enviarlo al servidor una vez lo haya modificado según corresponda. Finalmente, el servidor recibe el archivo final y lo almacena para que pueda ser accesible por los usuarios al terminar la ruta.

Gracias al uso de socket tipo TCP, orientado a conexión, se consigue una comunicación segura donde se garantiza que los datos van a llegar de un equipo a otro correctamente y que los programas no se van a quedar bloqueados esperando o trasmitiendo datos. Debido a esto no se ha empleado ningún tipo de control de errores adicional para el control de la comunicación. En el apartado 5.3 se estudia en profundidad la comunicación entre equipos mediante el uso de la herramienta Wireshark.

4.3 Procesado de audio

El módulo de procesado de audio mostrado en la Figura 4.1 está formado a su vez por otros dos módulos. Como se muestra en la Figura 4.15 consta de un módulo de recepción de audio equipado con un micrófono encargado de realizar grabaciones del ambiente y de un programa desarrollado en Python que determina si se ha detectado sonidos o no en la grabación realizada. El sistema es diseñado para su uso en caminos de senderismo donde el sonido ambiente es prácticamente indetectable con nuestros micrófonos, con lo cual cuando estos recogen un sonido se considera que este es producido por el paso de personas cercanas.

En la Figura 4.16 se muestra un organigrama del funcionamiento del programa Python desarrollado. El programa comienza definiendo las variables necesarias para su adecuado funcionamiento. Luego, se leen las muestras de audio y se crea un archivo .wav. A este archivo creado, se le aplica la transformada de Fourier y se obtiene la muestra en frecuencia de mayor valor, la cual representa el sonido de mayor amplitud que se registró en la grabación. Esta muestra se compara con el valor umbral para determinar si se detectó voz humana o simplemente sonido ambiente. Para poder ajustar este valor umbral se realizarón varias pruebas para comprobar que valores se obtenían cuando se detectaba voz humana en las grabaciones.

Para llevar a cabo la instalación del micrófono USB se debe insertar el mismo en uno de los puertos USB de la RPi. Luego, se ejecuta la orden *lsub* desde la terminal. De esta forma se muestran todos los dispositivos conectados a la RPi y verificamos que el micrófono ha sido

Figura 4.15. Módulo de procesado de audio.

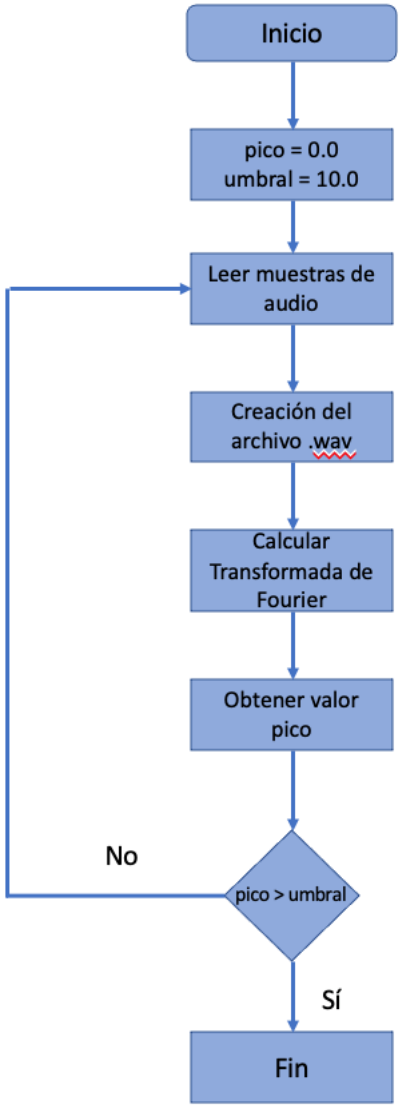

Figura 4.16. Organigrama del programa de detección de audio.

detectado. Una vez que el dispositivo ya es detectado por el equipo puede ser utilizado por diferentes programas para la captación de sonido.

Como ya se ha comentado, el programa de detección de audio tiene que ser capaz de realizar grabaciones y analizarlas de manera que pueda determinar si se ha recogido voz humana en la grabación realizada.

Para empezar, se deben importar las bibliotecas que se van a utilizar, *PyAudio*, la cual nos permite usar Python para reproducir y grabar audio en diferentes plataformas. Antes de poder utilizarla se necesitan instalar los siguientes paquetes mediante la orden: *sudo apt-get install libportaudio0 libportaudio2 libportaudiocpp0 portaudio19-dev* [53]. Por último, se utiliza la orden *sudo pip install pyaudio* para completar la instalación de la biblioteca.

Además, se necesita importar la biblioteca *wave*, que permite crear el archivo .wav donde almacenar la grabación. Las otras dos bibliotecas que se importan son *SciPy* y *NumPy.* Estas bibliotecas permiten pasar la grabación realizada al espectro de la frecuencia mediante el cálculo de la transformada de Fourier del archivo *.wav* generado. De esta forma, se puede analizar la grabación y determinar si se han detectado personas o no. Antes de poder importarlas en nuestro programa de Python se necesita instalarla mediante la siguiente orden: *sudo pip install numpy scipy wave* [54][55].

Después de importar las bibliotecas comentadas, se deben declarar las variables, que se muestran en la porción de código de la Figura 4.17, que necesita la biblioteca *PyAudio* para poder trabajar con el puerto USB que tiene conectado el micrófono:

- *form_1*: número de bits que se utilizan para codificar las muestras de audio.
- *chans*: número de canales, que en nuestro caso es 1 al ser un micrófono monocanal.
- *samp_rate*: frecuencia de muestreo de trabajo. Según el teorema de Nyquist, la frecuencia de muestreo debe ser, al menos, dos veces mayor que la frecuencia más alta que se va a registrar. Como el sonido más alto que el humano puede percibir tiene una frecuencia de 20 kHz, la frecuencia de muestreo mínima debe ser de 40 kHz.
- *chunk*: el tamaño del buffer de recepción, es decir, el número de muestras que se van a tomar del flujo de audio.
- *record_secs*: tiempo que va a durar la grabación es segundos.

- *umbral*: valor umbral a partir del cual la grabación realizada no es considerada como silencio. Este valor es comparado con el valor absoluto de las muestras en el espectro de la frecuencia de la grabación realizada.

- *wav_output_filename:* nombre del archivo .wav que se va a crear.

En la Figura 4.18 se muestra el algoritmo del programa que realiza la grabación del sonido y crea el archivo *.wav*. A través de una llamada a la biblioteca *PyAudio* y las variables definidas anteriormente se puede abrir el canal de audio con el método *audio.open* e iniciar la grabación. Una vez termine el ciclo de lectura de sonido, se detiene y se cierra el canal. De esta forma, se puede crear el archivo *.wav* a partir de la información recopilada mediante dicho ciclo de lectura.

Una vez se ha creado el archivo de audio se usa el método *read()* de la biblioteca *wave* para leer el archivo. Así se recuperan los datos obtenidos anteriormente y se puede calcular la transformada de Fourier usando el método *fft()* de la biblioteca *SciPy*. Cuando se aplica la transformada de Fourier se obteniene un array de números complejos, representados en magnitud y fase. Es necesario calcular el módulo mediante la función *abs()* de la biblioteca *NumPy* para obtener la magnitud de las muestras, este proceso se muestra en la Figura 4.19.

El módulo de todos los valores en frecuencia del archivo de audio se almacena en un array que se recorre mediante la función de la Figura 4.20 para obtener el valor más alto del conjunto de muestras. Finalmente, se compara ese valor con el valor umbral definido anteriormente y se determina si se ha detectado voz de personas cercanas o no. El valor de este número umbral se ha definido mediante una seria de pruebas en las que se tomaron muestras de grabaciones de voz humana y de grabaciones que únicamente captaban el sonido ambiente.

Una vez se ha determinado si se registro voz humana, el archivo *.wav* generado es eliminado para evitar agotar la capacidad de almacenamiento de la RPi.

```
#Variables grabacion
form_1 = pyaudio.paInt16
chans = 1
samp_rate = 44100
chunk = 4096
record_secs = 10
wav_output_filename = 'valor.wav'
umbral = 10.0
```

Figura 4.17. Código del programa de Python.

```python
while True:
    audio = pyaudio.PyAudio()



    stream = audio.open(format = form_1,rate = samp_rate,channels = chans,
                        input = True, frames_per_buffer=chunk)

    frames = []


    for ii in range(0,int((samp_rate/chunk)*record_secs)):
        data = stream.read(chunk)
        frames.append(data)


    print("Grabacion terminada")


    stream.stop_stream()
    stream.close()
    audio.terminate()


    wavefile = wave.open(wav_output_filename,'wb')
    wavefile.setnchannels(chans)
    wavefile.setsampwidth(audio.get_sample_size(form_1))
    wavefile.setframerate(samp_rate)
    wavefile.writeframes(b''.join(frames))
    wavefile.close()
```

Figura 4.18. Código del programa de Python.

```python
archivo = '/home/pi/Descargas/valor.wav'
fsonido, sonido = waves.read(archivo)
sonido = sonido / (2.**15)

from scipy.fftpack import fft
n = len(sonido)
AudioFreq = fft(sonido)

MagFreq = np.abs(AudioFreq)
muestraTop = obtenerMayor(MagFreq)
print(muestraTop)

if muestraTop > umbral:
    print("Se ha detectado sonido")
    #remove("/home/pi/Descargas/valor.wav")
else:
    print("No se ha detectado sonido")
```

Figura 4.19. Código del programa de Python.

```python
#funciones para la grabacion
def obtenerMayor(grabacion):
    valorMasAlto = 0
    for muestra in grabacion:
        if muestra > valorMasAlto:
            valorMasAlto = muestra
    return valorMasAlto
```

Figura 4.20. Código del programa de Python.

4.4 Detección de Beacons

Una vez se detecta voz humana mediante el proceso explicado en el apartado 4.3, la RPi pasa a registrar el paso de personas cercanas, provistas de un terminal que emita beacons de Bluetooth, mediante un proceso de escaneo de beacons. En la Figura 4.21 se muestra el organigrama del programa diseñado. Una vez las variables iniciales están declaradas, se inicia el intervalo de escaneo de Beacons. Todos los dispositivos detectados se almacenan en el array llamado *returnedList*. Este array se tiene que recorrer y comprobar para cada dispositivo si ya ha sido registrado o no anteriormente. En el caso de los dispositivos no registrados, se crea un archivo JSON, se registra la hora y se envía al siguiente nodo de la red. Una vez los dispositivos ya han sido registrados se almacenan en un array llamado *registrados* y así se evita que se creen dos archivos para una misma persona en las diferentes rondas de escaneo que se realizan.

La transmisión de beacons por parte de los usuarios se puede realizar desde cualquier tipo de dispositivo. En nuestro caso se usa una aplicación móvil ya programa instalada en un teléfono móvil.

Por otro lado, para poder configurar la RPi para el escaneo de beacons se debe instalar la biblioteca *bluez*, que será necesaria para poder tener acceso bluetooth a través de Python. Se puede instalar ejecutando las siguientes órdenes:

sudo apt-get install python-pip python-dev ipython

sudo apt-get install bluetooth libbluetooth-dev

sudo pip install pybluez

Además, se debe abrir el archivo *bluetooth.service* mediante la orden *sudo nano bluetooth.service* y añadir la siguiente línea:

ExecStart=/usr/local/libexec/bluetooth/bluetoothd –experimental

Una vez hecho esto, ya se puede crear el programa en Python. En primer lugar, se importan las bibliotecas *blescan* y *bluez* que se acaban de instalar. Luego, se crea una conexión con el microcontrolador del adaptador Bluetooth local especificado, en este caso, el cero, como se puede ver en la Figura 4.22.

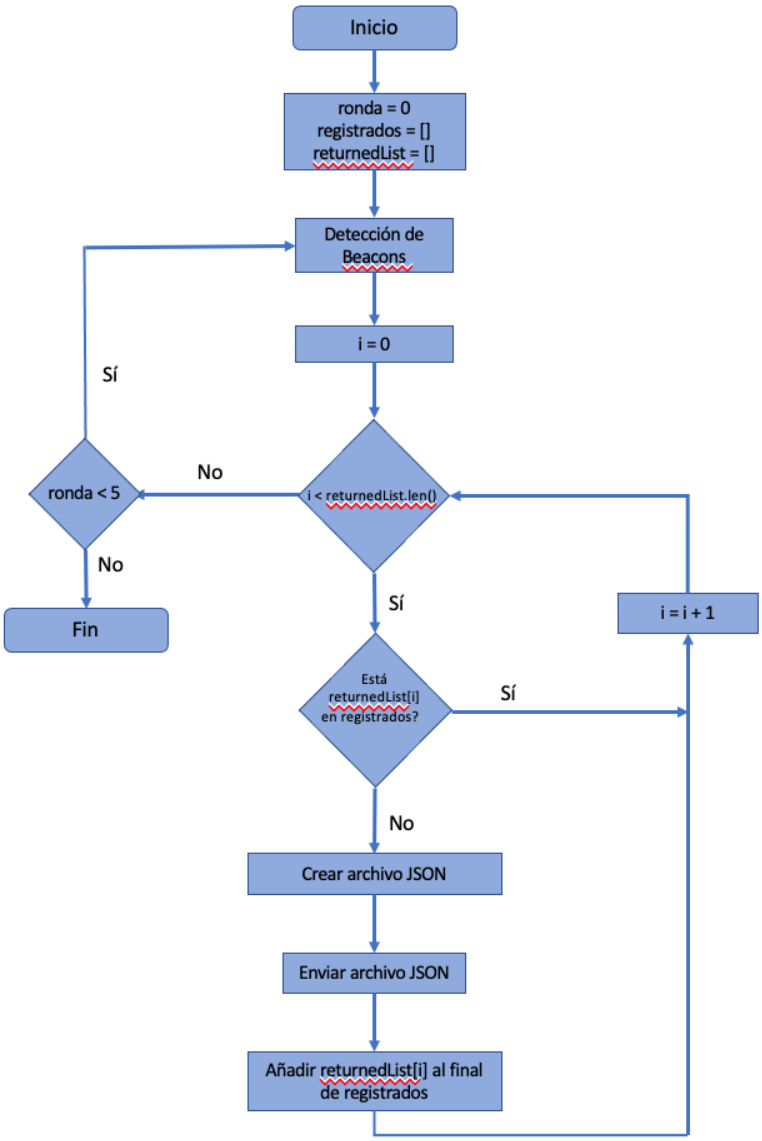

Figura 4.21. Organigrama del programa de detección de Beacons.

```
dev_id = 0
try:
    sock = bluez.hci_open_dev(dev_id)
    print ("ble thread started")

except:
    print ("error accessing bluetooth device...")
    sys.exit(1)

blescan.hci_le_set_scan_parameters(sock)
blescan.hci_enable_le_scan(sock)
```

Figura 4.22. Código del programa de Python.

```
rondaRegistro = 0
registrados = []

while True:
    returnedList = blescan.parse_events(sock, 10)
    print ("----------")
    rango = slice(24,32)
    for beacon in returnedList:
        subeacon = beacon[rango]
        respuesta = subeacon in registrados
        if respuesta == True:
            print("Ya está registrado")
        else:
            rutaDestino = "/home/pi/Documentos/Caminantes/"+subeacon+".json"
            time = datetime.now()
            fecha = time.strftime('%d/%m/%Y %H:%M')
            data = {}
            data['tiempos'] = []
            data['tiempos'].append({
                'nodo_1': fecha})
            with open(rutaDestino,'w') as file:
                json.dump(data,file, indent=4)
            print("Archivo creado")
            registrados.append(subeacon)
            #enviar_archivo(rutaDestino)
    #time.sleep(10)
```

Figura 4.23. Código del programa de Python.

Ahora ya se puede escanear el medio en busca de beacons, almacenando todos los dispositivos encontrados en un array llamado *returnedList*. De cada dispositivo detectado se obtiene diferentes datos como la dirección MAC, el UUID, el número major y minor y el número del *Received Signal Strength Indicator* (*RSSI*).

De todos los datos que se recogen el que nos interesa es el UUID, ya que en la aplicación transmisora de beacons se configura la trama beacons de tal forma que los primeros 8 números de este valor corresponder con el *Documento Nacional de Identidad* (*DNI*) del usuario que utiliza la aplicación. Esto nos permite identificar cada archivo con su usuario y que se lo pueda descargar del servidor, explicado en el apartado 4.5, al finalizar la ruta de senderismo.

En la Figura 4.23 se muestra el código donde se recorre el array de los dispositivos detectados recuperando los 8 primeros números del UUID. Luego se crea el archivo de tipo JSON en la ruta especificada usando esos primeros números y se envía el archivo al siguiente nodo mediante la función explicada en el apartado 4.2.1. Al finalizar el envío del archivo se almacena el DNI en un array creado con el nombre *registrados*. Esto se hace para evitar que se creen varios

archivos para un mismo usuario ya que se realizan varias rondas de escaneo. En la Figura 4.23 se muestra como antes de crear un archivo se comprueba si ya ese usuario fue registrado gracias a este array de usuarios registrados. Se realizan varias rondas de escaneo para poder registrar posibles personas que no fueron detectadas en un primer momento.

4.5 Instalación y configuración del servidor Apache

A continuación, se detallan los pasos a seguir para instalar y configurar el nodo final (servidor). Este servidor es el encargado de alojar todos los archivos de registro de tiempos de cada uno de los usuarios. De esta forma, el usuario podría tener acceso a ellos una vez finalice la ruta.

Es muy habitual el uso del servidor Apache junto a otras tecnologías, como pueden ser *My Structured Query Language (MySQL)* y *Hypertext Preprocessor (PHP)*, lo que se conoce como servidor Linux Apache MySQL PHP (LAMP). Sin embargo, debido a las características de nuestro TFG solo se lleva a cabo la instalación de Apache, ya que no es necesario hacer uso de un servidor de base de datos MySQL ni el servidor PHP para la ejecución de scripts.

En primer lugar, se deben actualizar todos los paquetes ya instalados mediante las siguientes dos órdenes:

sudo apt-get update

sudo apt-get upgrade

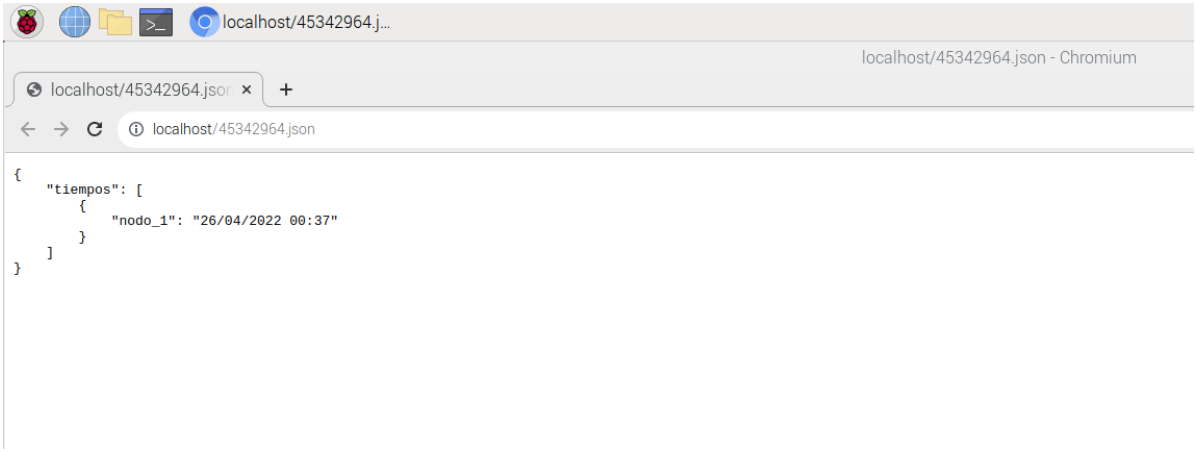

Figura 4.24. Código del programa de Python.

Una vez hecho esto, ya se puede instalar el servidor mediante un único comando *sudo apt-get install apache2* [56].

Cuando ya esté instalado, se crea un directorio, cuya ruta es */var/www/html*, en el cual se almacenan todos los archivos de los usuarios para que puedan ser accesibles desde el navegador. En la Figura 4.24, se muestra el archivo de datos de un usuario alojado en el servidor accediendo a él a través del navegador.

4.6 Instalación y configuración del punto de Acceso

En este apartado se muestra la configuración del PA, el cual permite a los usuarios descargarse sus archivos de datos. El nodo final de la red, aparte de actuar como servidor, también actúa como punto de acceso. De esta forma, cuando el usuario llega al nodo final de la ruta tendrá que conectarse a dicha red WiFi para poder descargarse los datos.

Para llevar a cabo la configuración, en primer lugar, se debe instalar *hostapd*, que permite transformar la tarjeta de interfaz de red en un punto de acceso. Además, se habilita dicho punto de acceso y se configura para que se inicie automáticamente al arrancar la RPi. Para ello se usan las siguientes órdenes [57]:

sudo apt install hostapd

sudo systemctl unmask hostapd

sudo systemctl enable hostapd

A continuación, se instala la biblioteca *dnsmasq* que proporciona un *Sistema de Nombres de Dominio* (*DNS*) y un servidor de *Protocolo de Configuración Dinámica de Host* (*DHCP*), que se encarga de asignarle de manera dinámica una dirección IP a cada uno de los dispositivos que se conecten a la red. Por último, se instala *netfilter-persistent* que guarda y carga las reglas del firewall.

sudo apt install dnsmasq

sudo DEBIAN_FRONTEND=noninteractive apt install -y netfilter-persistent

Ya todas las bibliotecas han sido instaladas y se puede empezar a configurar el equipo. Se debe empezar asignándole una dirección IP estática al servidor DHCP, en nuestro caso se le asigna

la dirección *192.168.4.1/24*. Para ello hay que modificar el archivo */etc/dhcpcd.conf* y añadirle las líneas que se muestran en la Figura 4.25.

La biblioteca *dnsmasq* proporciona un archivo de configuración por defecto con numerosas opciones, las cuales no son todas necesarias. Para facilitar el proceso, se le va a cambiar el nombre al archivo original y se va a crear otro de reemplazo completamente vacío, mediante la orden:

sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig

```
interface wlan0
        static ip_address=192.168.4.1/24
        nohook wpa_supplicant
```

Figura 4.25. Archivo /etc/dhcpcd.conf.

```
  GNU nano 5.4
interface=wlan0
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h
domain=wlan
address=/gw.wlan/192.168.4.1
```

Figura 4.26. Archivo /etc/dnsmasq.conf.

```
country_code=ES
interface=wlan0
ssid=Caminantes
hw_mode=g
channel=7
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
wpa=2
wpa_passphrase=universidad
wpa_key_mgmt=WPA-PSK
wpa_pairwise=TKIP
rsn_pairwise=CCMP
```

Figura 4.27. Archivo /etc/hostapd/hostapd.conf.

Abriendo este nuevo archivo con un editor de texto, se pueden añadir las opciones de configuración de la Figura 4.26, que son las únicas que nos hacen falta. La opción *interface* nos permite definir que interfaz inalámbrica que se va a usar, el *dhcp-range* define el rango de direcciones que puede asignar el DHCP y el tiempo que puede estar activa la dirección, en este caso, 24 horas. Y, por último, en la opción *address* se indica la dirección IP estática asignada con anterioridad al servidor. Por último, falta crear el nombre de la red y la contraseña. Esto se hace modificando el archivo */etc/hostapd/hostapd.conf* y añadiendo las líneas de la Figura 4.27.

En el archivo *hostapd.conf* se definen una serie de parámetros importantes que se detallan a continuación:

- *ssid*: nombre de la red.
- *channel*: canal para la transmitir dentro de la banda de los 2,4 GHz. En nuestro caso no habría interferencias con otros canales de otras redes ya que trabajamos en zonas poco accesibles donde no suelen existir conexiones.
- *wpa_passphrase*: contraseña de la red. Es necesario introducirla para poder conectarse a la misma.
- *wpa*: seguridad de la red. *WiFi Protected Access 2* (*WPA2*) es un método de seguridad que brinda una mayor protección de datos y regula el control de acceso a la red.

Una vez se reinicie la RPi mediante la orden *sudo reboot* todos los cambios realizados se activan y la red WiFi ya es accesible para los usuarios.

4.7 Instalación y configuración del adaptador WiFi USB

La última RPi debe formar parte de la red Ad-Hoc y crear el punto de acceso, con lo cual es necesario usar un adaptador WiFi tipo USB para poder configurar una interfaz inalámbrica adicional. De esta forma la interfaz wlan0 crea la red WiFi como se explica en el apartado 4.6 y la interfaz wlan1 se configura como parte de la red inalámbrica.

En el apartado 3.2.3 se mostró la instalación previa necesaria para que la RPi pueda reconocer el adaptador. Se puede comprobar que se instaló correctamente mediante la orden *sudo lsusb,* que muestra el adaptador como uno de los dispositivos conectados en los puertos usb, como se muestra en la Figura 4.28.

El proceso para configurar la interfaz es el mismo seguido para el resto de los nodos, explicado en el apartado 4.2. Se empieza modificando el archivo */etc/network/interfaces* mediante la orden sudo nano */etc/network/interfaces* y se añaden las líneas que se muestran en la Figura 4.29. De esta forma ya queda configurada la dirección IP, el modo de trabajo, en este caso modo Ad-Hoc, y la red de la que va a formar parte, es decir, *red-1*. Como se muestra en la Figura 4.29, es importante especificar que es la interfaz wlan 1 en la que se aplica los cambios realizados.

Por último, se tiene que ejecutar la orden *sudo wpa_cli terminate* para terminar de configurar el modo Ad-Hoc como modo de trabajo. Después, ejecutando la orden sudo networking restart se terminan de aplicar todos los cambios realizados a la RPi. Como se muestra en la Figura 4.30 ambas interfaces están totalmente configuradas.

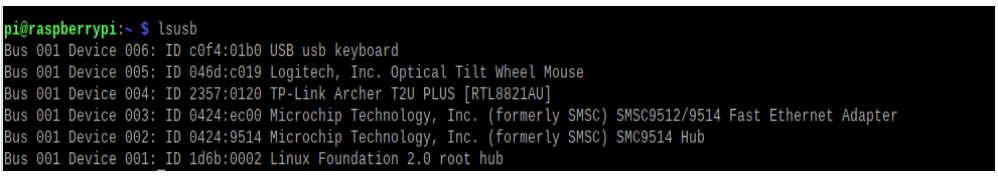

Figura 4.28. Dispositivos conectados a los puertos.

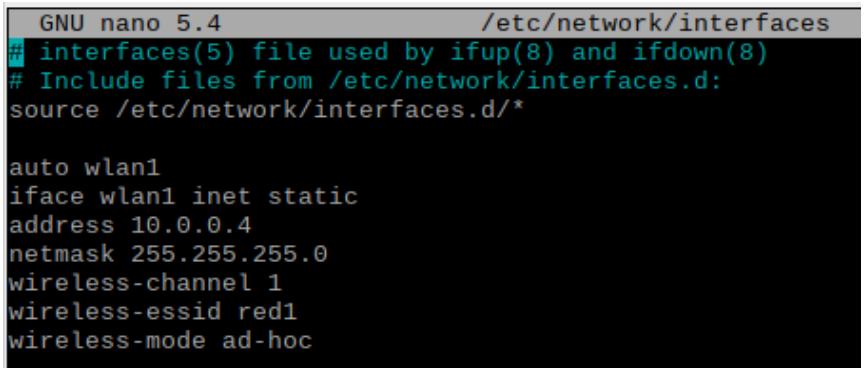

Figura 4.29. Archivo /etc/network/interfaces.

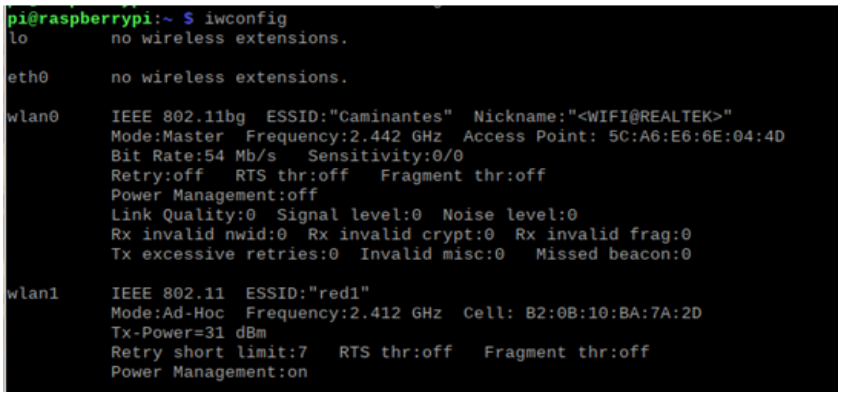

Figura 4.30. Interfaces inalámbricas configuradas.

5 Descripción de experimentos realizados

En este quinto capítulo, se presenta una serie de experimentos realizados a la plataforma creada centrándonos en la red acústica, la red inalámbrica y en el código desarrollado en Python.

3.2 Experimentos realizados

En este apartado se detallan los diferentes experimentos a los que ha sido sometido el sistema para comprobar su correcto funcionamiento. Los experimentos realizados son:

- Pruebas de captación de audio en una ruta de senderismo para comprobar el adecuado funcionamiento de la red acústica.
- Comprobación del tráfico de la red Ad-Hoc mediante el uso de la herramienta *Wireshark*.
- Puesta a prueba del sistema en un caso real donde se compruebe el funcionamiento completo del mismo.

3.3 Experimento para comprobar la red acústica

El siguiente experimento tiene como objetivo comprobar el correcto funcionamiento de la red acústica y comprobar si el sistema es capaz de detectar voz humana. Para ello se realizan una serie de grabaciones en una ruta de senderismo de la isla de Gran Canaria, concretamente en Los Tilos de Moya, en la ruta marcada en rojo que se muestra en la Figura 5.1. Se realizan dos tipos de grabaciones, el primer tipo, que únicamente capta el sonido ambiente del lugar y el segundo, que capta voz humana simulando así posibles personas que realicen dicha ruta. Finalmente, se analizan los resultados obtenidos y se comprueba el funcionamiento del programa diseñado para la detección de voz humana.

Una vez realizas las grabaciones, estas se pasan al dominio de la frecuencia, tal y como se explicó en el apartado 4.3, y además, se representan. De esta forma se puede visualizar fácilmente los valores que se obtienen comprobando así si el valor umbral con el que se trabaja es el adecuado para esa zona, es decir, Los Tilos de Moya.

Se diseñó un nuevo programa de Python que se encarga de realizar las grabaciones y de representarlas en el dominio de la frecuencia. Para poder realizar la representación, primero se debe llevar a cabo la instalación de la biblioteca *matplotlib* mediante la siguiente *orden sudo apt install python3-matplotlib*.

Una vez instalada la biblioteca, ya se puede crear el programa de Python. En la Figura 5.2 se muestra el código encargado de realizar las grabaciones y crear el archivo de audio, declarando las variables necesarias para el funcionamiento del algoritmo de captación de muestras de audio

al inicio de este. La primera parte del código se explicó en el apartado 4.3, con lo cual se profundiza en la segunda parte de este, que es la diseñada para este experimento.

Figura 5.1. Cartel de la ruta de senderismo de Los Tilos.

```
from __future__ import print_function

#import grabacion
import pyaudio
import wave
import numpy as np
import scipy.io.wavfile as waves
import matplotlib.pyplot as plt
from os import remove

#Variables grabacion
form_1 = pyaudio.paInt16
chans = 1
samp_rate = 44100
chunk = 4096
record_secs = 10
wav_output_filename = 'muestra1.wav'

def obtenerMayor(grabacion):
    valorMasAlto = 0
    for muestra in grabacion:
        if muestra > valorMasAlto:
            valorMasAlto = muestra
    return valorMasAlto


audio = pyaudio.PyAudio()



stream = audio.open(format = form_1,rate = samp_rate,channels = chans,
                    input = True, frames_per_buffer=chunk)

frames = []



for ii in range(0,int((samp_rate/chunk)*record_secs)):
    data = stream.read(chunk)
    frames.append(data)


print("Grabacion terminada")


stream.stop_stream()
stream.close()
audio.terminate()


wavefile = wave.open(wav_output_filename,'wb')
wavefile.setnchannels(chans)
wavefile.setsampwidth(audio.get_sample_size(form_1))
wavefile.setframerate(samp_rate)
wavefile.writeframes(b''.join(frames))
wavefile.close()
```

Figura 5.2. Código del programa de Python.

```
archivo = '/home/pi/muestra1.wav'
fsonido, sonido = waves.read(archivo)
sonido = sonido / (2.**15)

from scipy.fftpack import fft
n = len(sonido)
AudioFreq = fft(sonido)

MagFreq = np.abs(AudioFreq)
#muestraTop = obtenerMayor(MagFreq)
#print(muestraTop)

print("Graficando")
plt.plot(MagFreq)
plt.ylabel('Magnitud')
plt.title('FFT total')
plt.show()
```

Figura 5.3. Código del programa de Python.

Una vez realizada la grabación, esta tiene que ser pasada al dominio de la frecuencia mediante la función *fft()* de la biblioteca *SciPy.* Esta función devuelve un array con todos los números complejos obtenidos después de calcular la transformada de Fourier de la grabación realizada. Luego, se calcula el módulo de los números complejos mediante la función *abs()* de la biblioteca *NumPy.* Los resultados se almacenan en un array, el cual ya puede ser representado gracias a la biblioteca *matplotlib* mediante la función *plot()*. El código de Python de este proceso es el que se muestra en la Figura 5.3.

Una vez se tiene el programa diseñado ya se puede llevar a cabo el experimento. En primer lugar, se ubica el equipo en el punto de la ruta donde se realizan las grabaciones, tal y como se muestra en la Figura 5.4, y se conecta la batería portátil a la RPi. Una vez se inicie el equipo completamente, se accede al mismo vía *Secure Shell (SSH)*, el cual fue activado previamente en el menú *Configuración de Raspberry Pi > Interfaces* del equipo. SSH es una aplicación de administración remota que nos permite conectarnos y administrar el equipo de forma remota sin hacer uso de una pantalla, teclado y ratón externos [58]. Finalmente, se ejecuta el programa desarrollado mediante al orden *sudo python muestras.py*.

Figura 5.4. Ubicación de la Raspberry Pi en la ruta.

Primero, se realizan dos grabaciones donde únicamente se capta el sonido ambiente. Así se puede comprobar que valores se alcanzan cuando no se encuentra ninguna persona dentro del área de cobertura de la red acústica. La representación obtenida de estas grabaciones se muestra en las Figuras 5.5 y 5.6.

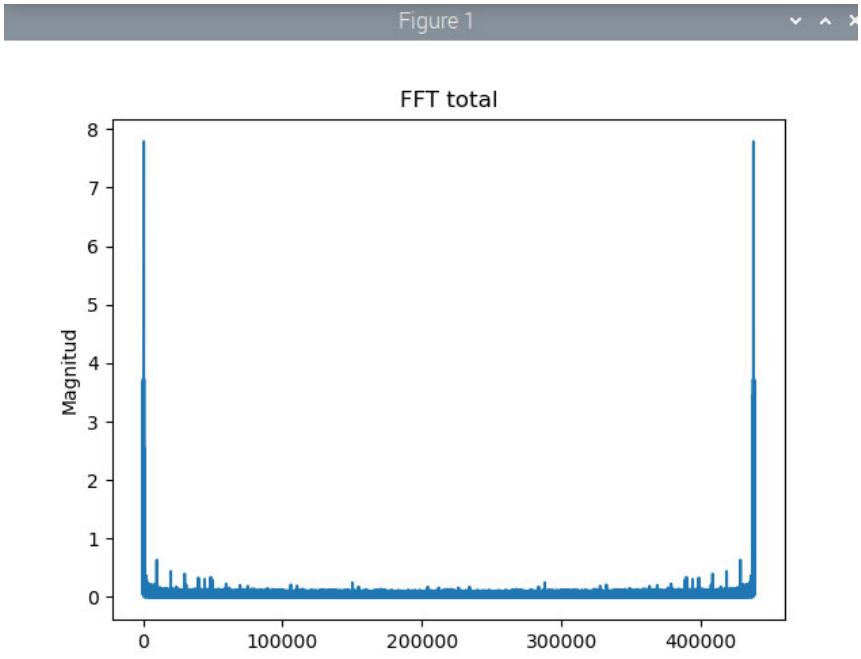

Figura 5.5. Representación 1 de la grabación captando sonido ambiente.

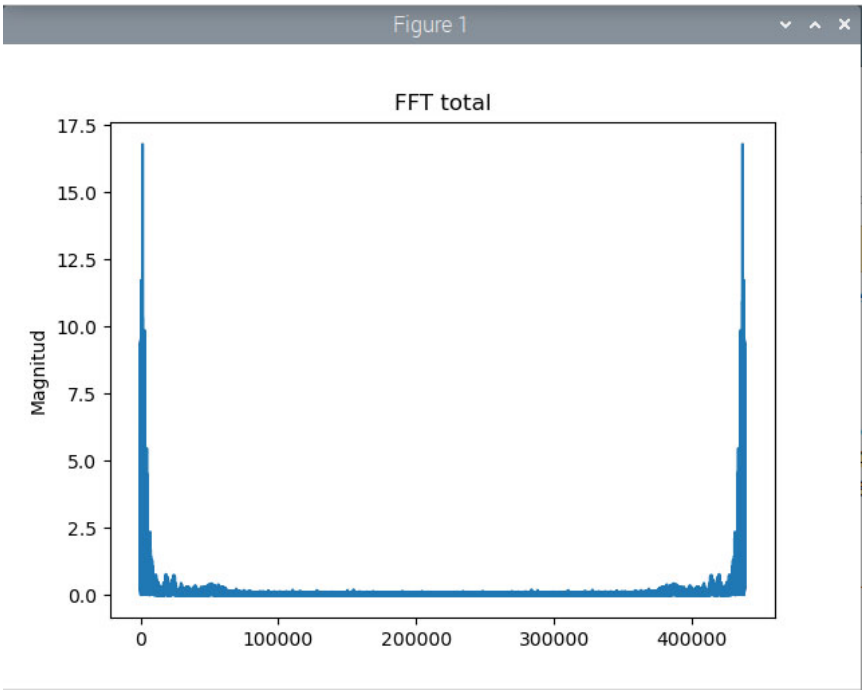

Figura 5.6. Representación 2 de la grabación captando sonido ambiente.

Gracias a la representación de las grabaciones se puede ver que en ambos casos al iniciar y al finalizar las grabaciones se introducen unos valores pico los cuales son valores falsos teniendo en cuenta el resto de los valores que se registran a lo largo de la grabación. Además, estos picos alteran el adecuado funcionamiento del programa diseñado. Por ello, se modifica el código del programa mostrado anteriormente para eliminar estos valores no deseados y únicamente se tiene en cuenta aquellos valores que de verdad corresponden a la grabación realizada. Esto se consigue eliminando los extremos del *array* que contiene los valores en frecuencia de la grabación realizada.

En la Figura 5.7 se muestra la representación nuevamente de una grabación donde se capta el sonido ambiente tras los cambios realizados en el programa de Python. Sin embargo, en la Figura 5.8 se muestra la gráfica obtenida para una grabación donde se capta voz humana, simulando así el paso de una persona dentro del área de trabajo de la red acústica.

Comparando los resultados obtenidos se puede constatar que una grabación donde se capta únicamente el sonido ambiente se obtiene valores menores que 1. Mientras que una grabación donde se recoge voz humana puede alcanzar valores de hasta 14. Esto demuestra que utilizando un valor umbral mayor que 2 se puede diferenciar correctamente grabaciones donde

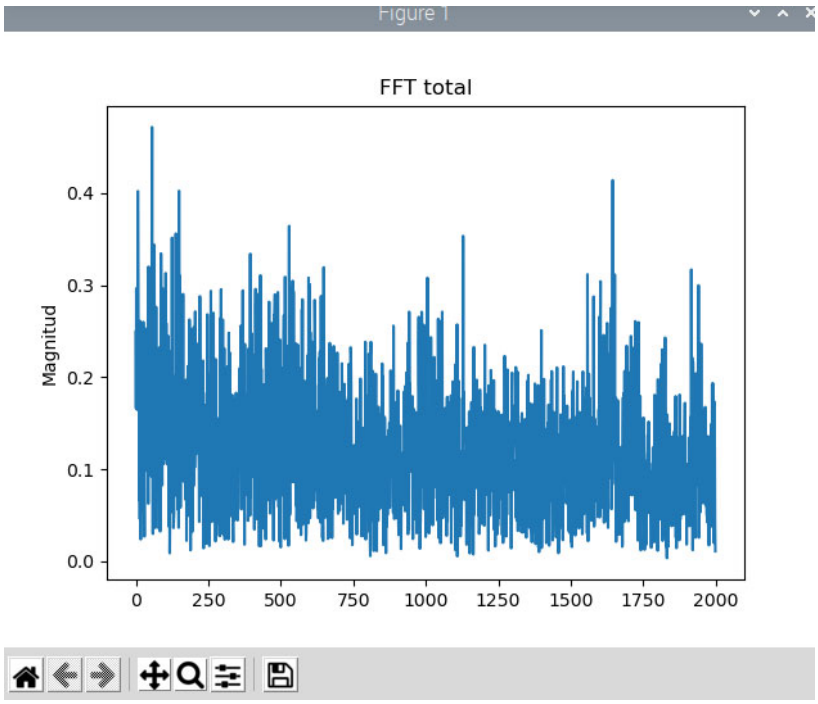

Figura 5.7. Representación del sonido ambiente.

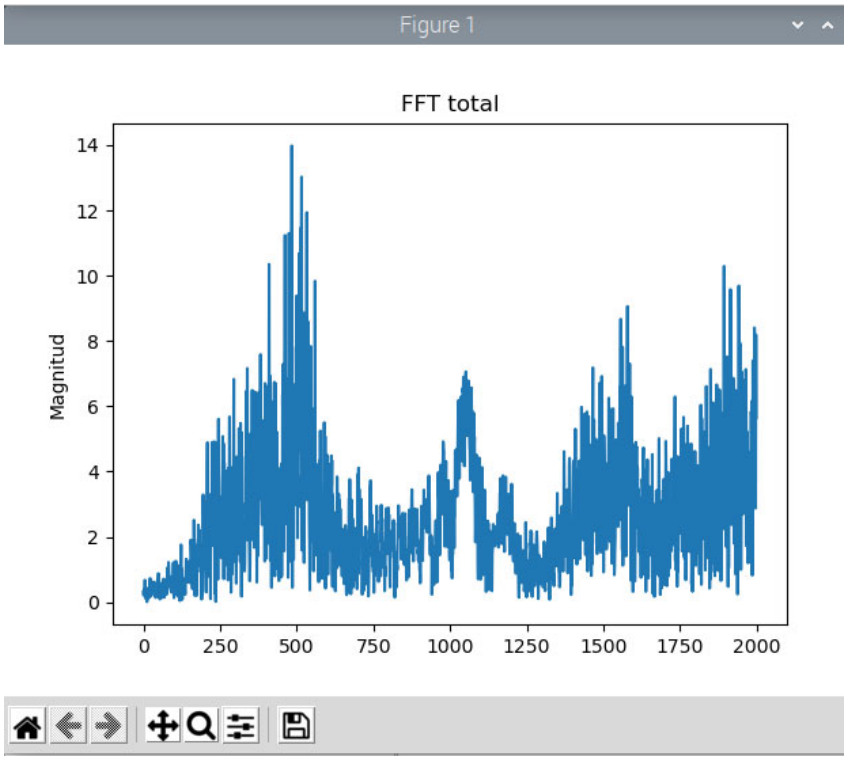

Figura 5.8. Representación de voz humana.

solo se capte sonido ambiente de grabaciones que contengan voz humana en la zona de Los Tilos de Moya. Implicando esto que el proyecto desarrollado funciona correctamente en dicha zona y para la franja horaria y de tiempo atmosférico en la que se realizó. Es importante destacar que en general la red acústica debe ser calibrada para cada una de las zonas rurales en las que se quiera instalar. Esto es, para cada zona rural, franja horaria y caracterización del tiempo atmosférico estos umbrales deben ser determinados y usados para la discriminación de voz humana. En general se debe tener en cuenta incluso el ruido debido a los animales que existen en dicha zona rural. Sin embargo, esos detalles no están en el ámbito del TFG. Pero con el uso de la red acústica y esas calibraciones para determinar los umbrales se puede afirmar que la aplicación del TFG es de ámbito general, puesto que como se ha demostrado su utilidad es real en un entorno concreto.

3.4 Análisis del tráfico de la red Ad-Hoc

Para poder analizar el tráfico que se genera la red acústica se hace uso de la herramienta *Wireshark,* la cual permite capturar todo el tráfico que se genera en la red. En este experimento se captura el tráfico del nodo 2 de la red Ad-Hoc, pudiendo estudiar así los siguientes procesos:

- Establecimiento de la conexión.
- Transmisión de datos entre dos nodos.
- Finalización de la conexión.

Para poder llevar a cabo el experimento, primero se debe instalar la herramienta *Wireshark* en uno de los nodos de la red, en este caso en el nodo 2. Para ello, se inicia la descarga con la siguiente orden *sudo apt-get install wireshark*.

Una vez descargado, se termina de instalar mediante las siguientes órdenes:

sudo groupadd wireshark

sudo usermod -a -G wireshark pi

sudo chgrp wireshark /usr/bin/dumpcap

sudo chmod 750 /usr/bin/dumpcap

Finalmente, una vez se reinicie la RPi con la orden *sudo reboot* ya se tendría instalada la herramienta Wireshark.

Se establece la comunicación entre el nodo 2, cuya dirección IP es la 10.0.0.3, que actúa como cliente y el nodo 3, con dirección IP 10.0.0.2, que actúa como servidor. La herramienta Wireshark se ejecuta en el nodo 2, capturando todo el tráfico que se envíe y se reciba por la *wlan0* de la RPi.

A continuación, se analiza en profundidad cada uno de los diferentes procesos que tienen lugar durante la comunicación de ambos nodos.

5.1.1 Establecimiento de la conexión

Para la transmisión del archivo se usa el TCP, que una de sus principales características es que es un protocolo orientado a conexión. Esto implica que antes de iniciar el envío de datos es totalmente necesario realizar una comunicación previa para crear la conexión entre el cliente y el servidor.

Esta comunicación previa se denomina *3-way handshake,* y consiste básicamente en el intercambio de paquetes que se muestra en la Figura 5.9. En primer lugar, el cliente, que es el que inicia la conexión, envía un paquete *Synchronize* (SYN) al servidor. Si se analiza el campo *Transmission Control Protocol* de este paquete con el *Wireshark,* como se muestra en la Figura 5.10, se puede observar que el puerto de destino es el configurado anteriormente y que el indicador *SYN* está activado, indicando que se quiere establecer una comunicación.

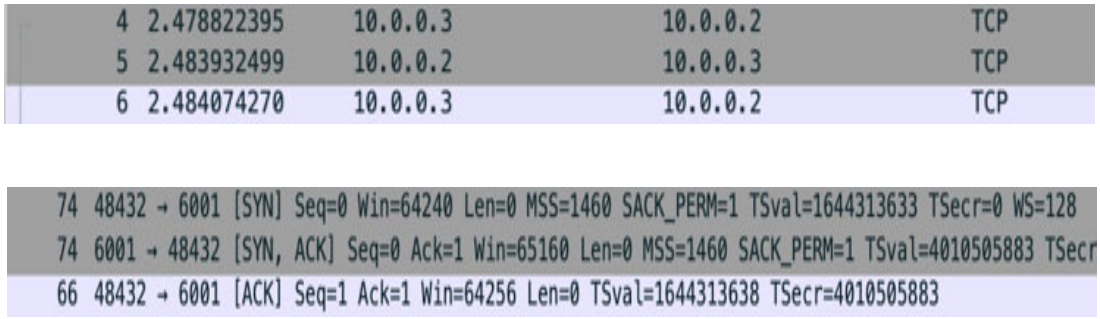

Figura 5.9. Intercambio de paquetes para establecer la conexión.

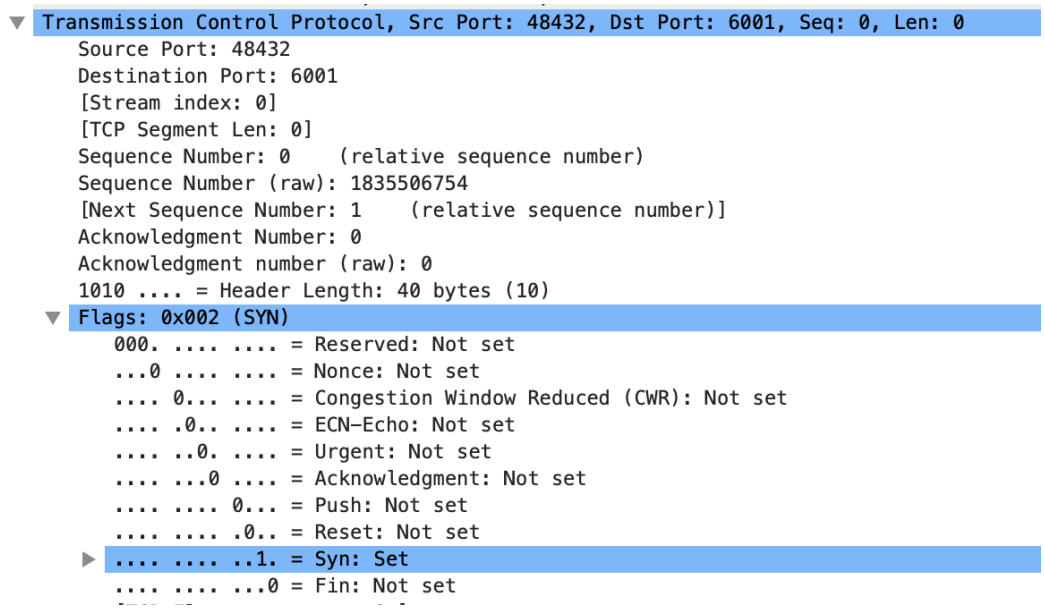

Figura 5.10. Primer paquete de establecimiento de conexión.

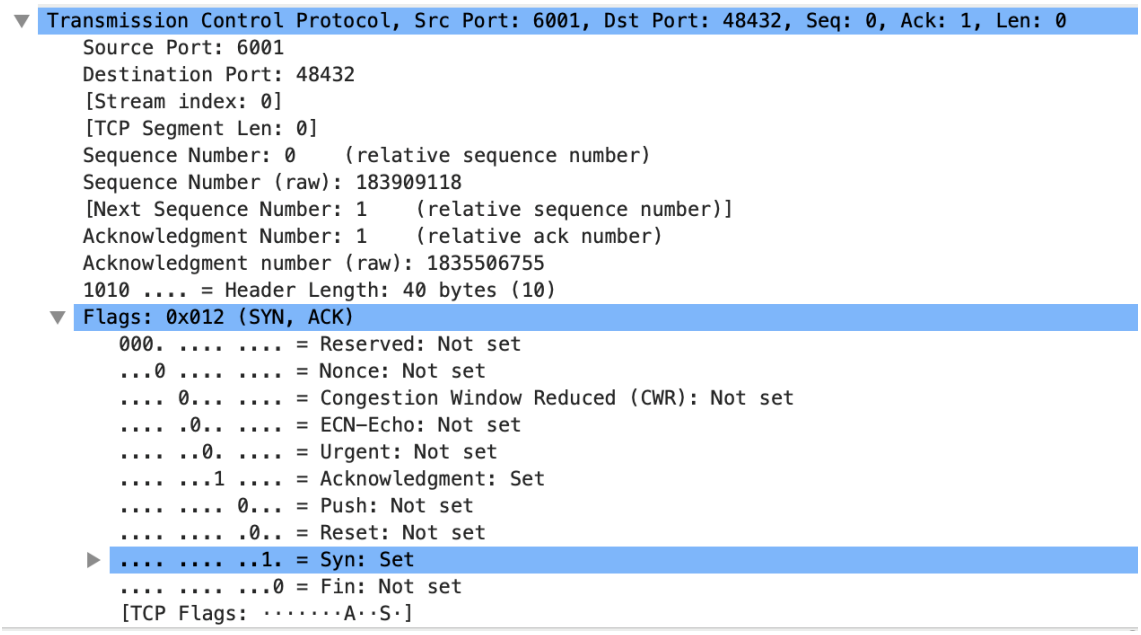

Figura 5.11. Respuesta del servidor a la petición de conexión.

El servidor responde al cliente con un mensaje de tipo *Synchronize-Acknowledgment* (*SYN-ACK*), informando al cliente que se ha recibido la petición de establecimiento de conexión correctamente e indicándole que puede iniciar la transmisión de datos. Analizando nuevamente el contenido de este segundo paquete, como se muestra en la Figura 5.11, esta vez se activa tanto el indicador *SYN* como el *ACK*.

Finalmente, el cliente envía un último paquete indicando que recibió correctamente el paquete y se finaliza el proceso de establecimiento de conexión. A partir de este paquete da comienzo la transmisión de datos entre ambos nodos.

5.1.2 Transmisión de datos

Tras establecer la conexión, se inicia el envío de datos por parte del cliente como se muestra en la Figura 5.12. En este caso, para el envío del archivo únicamente hacen falta dos paquetes. Analizando el contenido del primer paquete se puede comprobar que se envía el nombre del archivo y su tamaño, como se muestra en la Figura 5.13.

En el segundo paquete, se envía el contenido del archivo JSON, es decir, los tiempos que se registraron durante la ruta. El contenido del paquete se muestra en la Figura 5.14. Además, en la Figura 5.15 se puede ver como el indicador *Fin* está activado, indicando el cliente al servidor que se ha finalizado la transmisión de datos y se va a iniciar el proceso de desconexión.

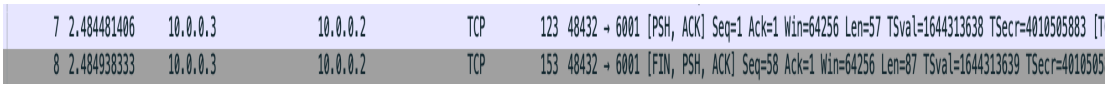

Figura 5.12. Envío de datos.

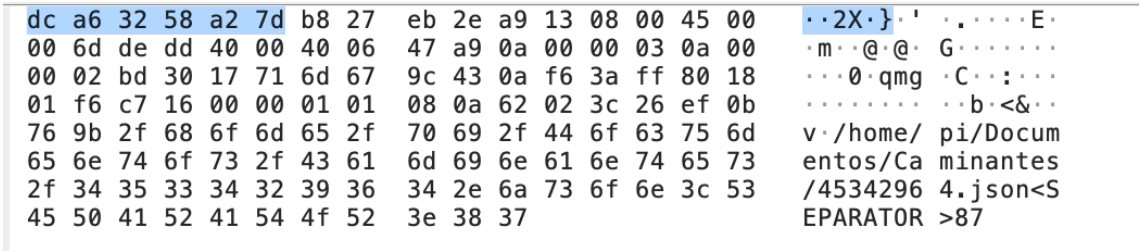

Figura 5.13. Contenido del primer paquete.

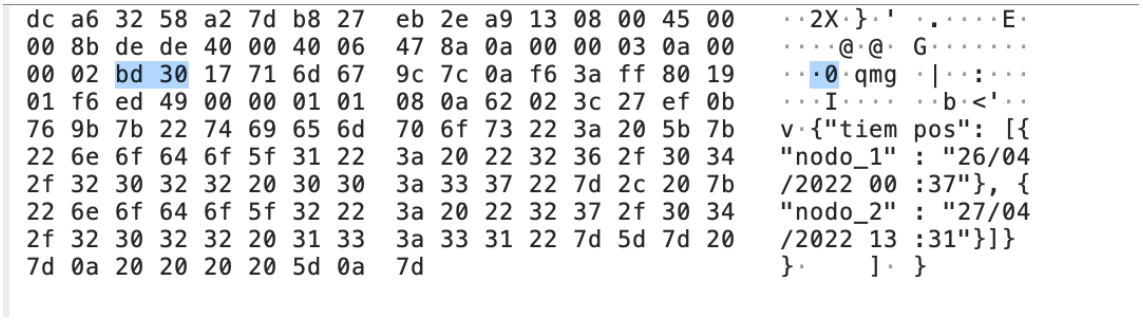

Figura 5.14. Contenido del segundo paquete.

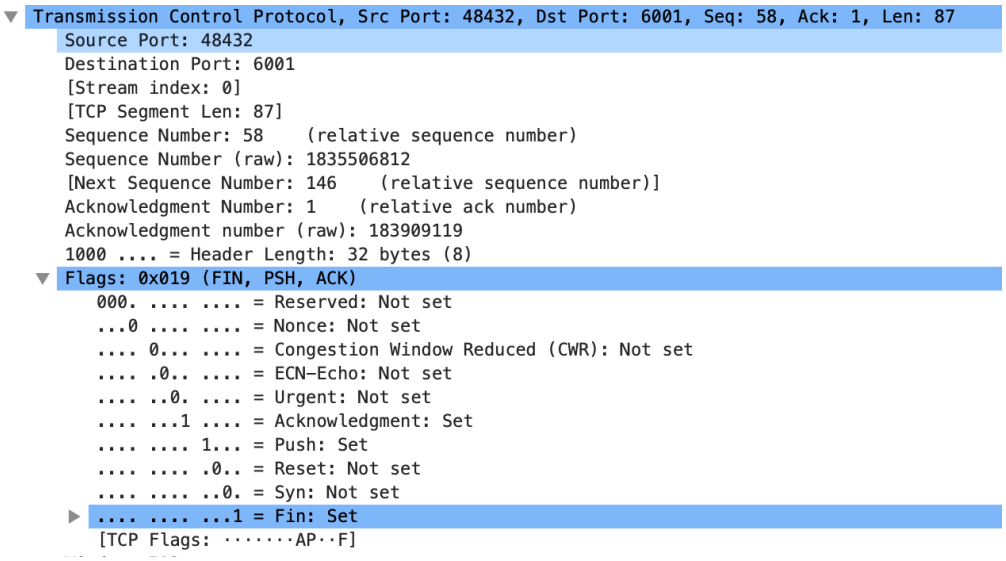

Figura 5.15. Indicadores del segundo paquete.

5.1.3 Finalización de la conexión

El proceso de desconexión se inició con el último paquete que le envía el cliente al servidor durante la transmisión de datos. En dicho paquete el cliente le indica al servidor que la transmisión de datos ha terminado y quiere finalizar la conexión mediante el indicador *FIN* del último paquete de datos, como se muestra en la Figura 5.15.

El servidor le contesta con un paquete de tipo *ACK* y otro paquete de tipo *FIN,* como se muestra en la Figura 5.16. Finalmente, el cliente, que fue quien inició la petición de finalización de conexión, envía un paquete de tipo ACK, cerrando así el *socket* abierto.

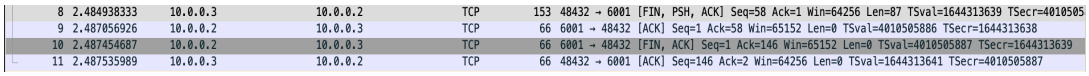

Figura 5.16. Intercambio de paquetes para finalizar la comunicación.

3.5 Prueba de la plataforma en un caso real

Para terminar la parte experimental del proyecto, se lleva a cabo un experimento donde se pone a prueba todo el funcionamiento del sistema desarrollado. Para ello, se va a desplegar la red Ad-Hoc desarrollada en dos zonas diferentes. En primer lugar, la red se instalada en una zona interior con acceso a la red eléctrica, donde se usan tres RPi, dos que actúan como detectores de personas y un nodo final que actúa como servidor.

La segunda parte de este experimento consiste en instalar la red inalámbrica en una ruta de senderismo y comprobar el funcionamiento en un caso real de uso del sistema desarrollado. Para esta parte del experimento únicamente se usarán dos RPi ya que solo de disponen de dos baterías. La red inalámbrica estará formada por dos nodos y uno de ellos actuará también como servidor.

Para poder llevar a cabo el experimento se va a utilizar una aplicación móvil ya desarrollada llamada *Beacon Simulator* [59]. Esta aplicación se descarga a través de la *Play Store y s*e instala en un teléfono móvil. Esta puede tanto escanear como transmitir *Beacons*, sin embargo, en nuestro caso se usa como transmisor de Beacons actuando como el dispositivo transmisor que deben usar las personas para poder ser registradas.

Como se muestra en la Figura 5.17, la aplicación te permite seleccionar el tipo de trama Beacon que se quiere utilizar, en nuestro caso se emplea el formato de trama iBeacon desarrollado por Apple. Además, se pueden configurar los diferentes campos, siendo el más importante el UUID, cuyos 8 primeros números deben ser el DNI del usuario, en este caso 45342964. En la Figura 5.18 se muestra la configuración completa de la trama que se ha usado para el desarrollo de las dos pruebas que constituyen este experimento.

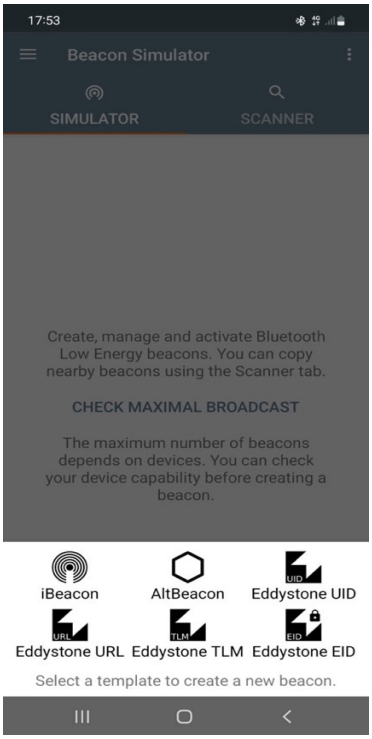

Figura 5.17. Selección del tipo de trama Beacon.

Figura 5.18. Configuración de los paquetes Beacons.

A continuación, se profundiza en el despliegue de la red inalámbrica en ambas zonas de prueba. Mostrando en detalle el funcionamiento y los resultados obtenidos.

5.1.4 Despliegue en zona interior

Para empezar con la prueba en una zona interior se despliegan las tres RPi a lo largo de una nave industrial ubicada en el polígono industrial de San Isidro, ubicación apta para la prueba ya que existen pocas señales de otros dispositivos que puedan interferir en la prueba. Los equipos se ubican en las tomas de corriente disponibles como se muestra en la Figura 5.19.

Una vez ubicadas las RPi se conectan a la red eléctrica. Cuando los equipos ya se han iniciado completamente de accede a ellos de forma remota vía *ssh* nuevamente. En cada uno de ellos se ejecuta un programa *python* diferente ya que desempeñan tareas distintas. El nodo 1 ejecuta el programa explicado en el apartado 4.3 y 4.4, de esta forma es capaz tanto de detectar voz humana como de registrar a una persona. Además, actúa como cliente, enviando el archivo generado al nodo 2. En la figura 5.20 se muestra el código completo del programa y se inicia su ejecución con la orden *sudo python audio.py*.

Figura 5.19. Localización de los nodos.

En el nodo 2 se ejecuta el programa python de la Figura 4.13 que permite ejecutar dos programas de forma simultanea. En este caso, ejecuta el programa servidor, explicado en el apartado 4.2.2, para recibir los archivos enviados por el nodo 1, y el programa de la Figura 5.20, para detectar y registrar personas en dichos archivos, enviándolos a su vez nodo 3. Es necesario modificar la dirección y puerto destino en el código de la Figura 5.20 para que el envío se haga al nodo 3. Se inicia el programa mediante la orden *sudo python ambos.py*.

Sin embargo, en el nodo 3 únicamente se ejecuta el programa explicado en el apartado 4.2.2, mediante la orden *sudo python3 servidor4.py*, ya que solo recibe los archivos enviados y por el nodo 2 y los aloja en el servidor para que sean accesible para el usuario.

Una vez los equipos están colocados y ejecutando el programa Python comentado, ya se puede comenzar la prueba de funcionamiento. Para ello, en primer lugar, se tiene que activar la transmisión de *Beacon* a través de la aplicación comentada anteriormente como se muestra en la Figura 5.21.

A continuación, nos ubicamos en el primer nodo de la red y simulamos una conversación para que la red acústica sea capaz de captar la voz humana. Una vez esta sea captada, la RPi comienza el proceso de escaneo de *Beacons* y genera un archivo llamado, en este caso, *45342964.json.* El nombre del archivo corresponde al DNI del usuario que se utilizó como UUID para crear las tramas Beacons a través de la aplicación móvil. En este archivo se tiene que registrar la hora a la que se nos detectó en ese nodo y tiene que ser enviado al siguiente.

Se tiene que seguir avanzando a lo largo de la ruta hasta alcanzar el segundo nodo. Este segundo nodo tiene que repetir el proceso comentado y generar una segunda entrada al archivo registrando la hora a la que la persona fue detectada en ese nodo. Finalmente, tiene que enviar el archivo al nodo final para que sea almacenado en el servidor.

Una vez el archivo este almacenado en el servidor, el usuario ya podría tener acceso a sus datos y comprobar los tiempos que fueron registrados en el archivo. Para ello, cuando se finalice la ruta y alcance el nodo final, se tiene que acceder a la red WiFi generada por ese último nodo. Como se muestra en la Figura 5.22 la red recibe el nombre de *Caminantes.*

```
while True:
    audio = pyaudio.PyAudio()



    stream = audio.open(format = form_1,rate = samp_rate,channels = chans,
                        input = True, frames_per_buffer=chunk)

    frames = []


    for ii in range(0,int((samp_rate/chunk)*record_secs)):
        data = stream.read(chunk)
        frames.append(data)


    print("Grabacion terminada")


    stream.stop_stream()
    stream.close()
    audio.terminate()


    wavefile = wave.open(wav_output_filename,'wb')
    wavefile.setnchannels(chans)
    wavefile.setsampwidth(audio.get_sample_size(form_1))
    wavefile.setframerate(samp_rate)
    wavefile.writeframes(b''.join(frames))
    wavefile.close()


    archivo = '/home/pi/Documentos/SDL_Pi_iBeaconScanner-main/valor.wav'
    fsonido, sonido = waves.read(archivo)
    sonido = sonido / (2.**15)

    from scipy.fftpack import fft
    n = len(sonido)
    AudioFreq = fft(sonido)

    MagFreq = np.abs(AudioFreq)
    muestraTop = obtenerMayor(MagFreq)
    print(muestraTop)

    if muestraTop > umbral:
        print("Se ha detectado sonido")
        #remove("/home/pi/Descargas/valor.wav")
        dev_id = 0
        try:
            sock = bluez.hci_open_dev(dev_id)
            print ("ble thread started")

        except:
            print ("error accessing bluetooth device...")
            sys.exit(1)

        blescan.hci_le_set_scan_parameters(sock)
        blescan.hci_enable_le_scan(sock)
        rondaRegistro = 0
        registrados = []

        while True:
            returnedList = blescan.parse_events(sock, 10)
            print ("----------")
            rango = slice(24,32)
            for beacon in returnedList:
                subeacon = beacon[rango]
                respuesta = subeacon in registrados
                if respuesta == True:
                    print("Ya está registrado")
                else:
                    rutaDestino = "/home/pi/Documentos/Caminantes/"+subeacon+".json"
                    time = datetime.now()
                    fecha = time.strftime('%d/%m/%Y %H:%M')
                    data = {}
                    data['tiempos'] = []
                    data['tiempos'].append({
                        'nodo_1': fecha})
                    with open(rutaDestino,'w') as file:
                        json.dump(data,file, indent=4)
                    print("Archivo creado")
                    registrados.append(subeacon)
                    #enviar_archivo(rutaDestino)
            #time.sleep(10)


            rondaRegistro +=1
            if rondaRegistro == 5:
                break


        #registrados.clear()
        remove("/home/pi/Documentos/SDL_Pi_iBeaconScanner-main/valor.wav")


    else:
        print("No se ha detectado sonido")
        remove("/home/pi/Documentos/SDL_Pi_iBeaconScanner-main/valor.wav")
```

Figura 5.20. Código completo del programa.

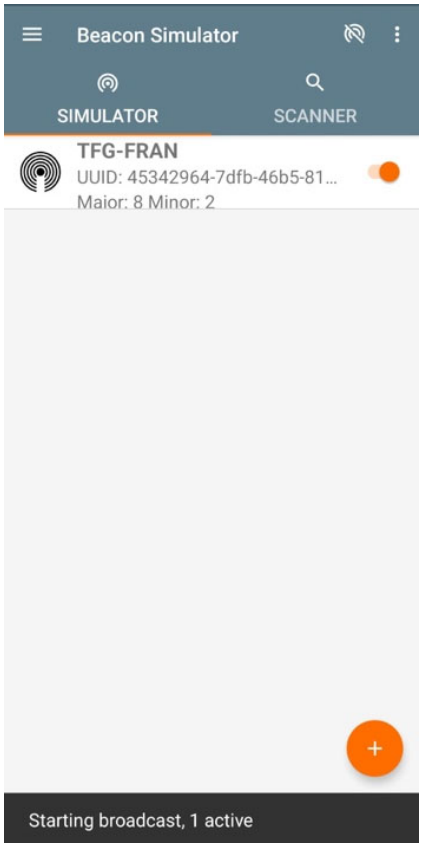

Figura 5.21. Transmisión activa de Beacons.

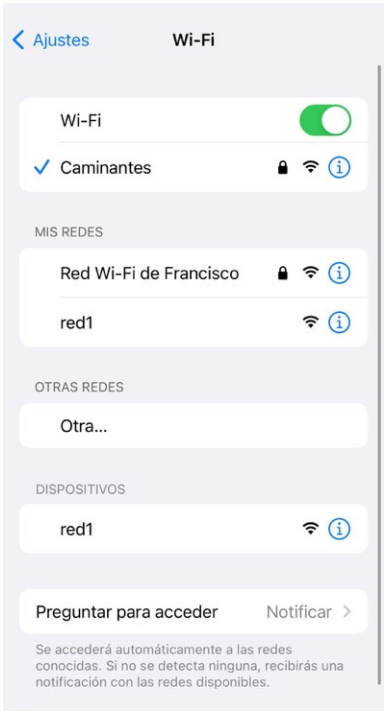

Figura 5.22. Acceso a la red WiFi del nodo final.

Cuando se haya creado la conexión con el PA generado por la RPi, se puede a acceder al archivo alojado en el servidor y comprobar así que todo el proceso se ha ejecutado correctamente. Para ello se accede a través del navegador del teléfono móvil a la siguiente dirección http://192.168.4.1/45342964.json. La dirección *192.168.4.1* corresponde con la dirección asignada a la *wlan1* del nodo final, la cual está generando la red WiFi. De esta forma se visualiza el archivo generado, como se muestra en la Figura 5.23, a través del navegador de nuestro móvil.

Se puede comprobar como el proceso de detección y registro se ejecutó correctamente en cada uno de los nodos de las red. En el archivo se puede ver una primera entrada donde el nodo 1 registró la hora a la que fuimos detectados, y una segunda entrada correspondiente al proceso de detección del nodo 2. Además, se comprueba que tanto el servidor como el punto de acceso funcionan correctamente ya que nos hemos podido conectar a él y tener acceso a nuestro archivo.

Figura 5.23. Archivo de datos.

5.1.5 Despliegue en zona exterior

La segunda parte de este último experimento consiste en el despliegue del sistema en una ruta de senderismo donde poder simular un caso de uso real del proyecto desarrollado. En este caso en concreto nos hemos desplazado nuevamente hasta la ruta circular de los Tilos, ubicada en el municipio de Moya que se muestra en la Figura 5.1.

Las RPi se desplieguen en un tramo de dicha de ruta, pudiéndose ver la localización exacta de los equipos en la Figura 5.24. En este caso únicamente se han utilizado dos equipos debido a que solo se disponen de dos baterías. La conexión con este tipo de baterías portátiles es muy sencilla y se muestra en la figura 5.25. Solo se requiere de un cable micro USB conectado a la entrada de alimentación de la RPi.

Una vez las baterías ya estén conectadas a los equipos, se accede a ambas RPi mediante *ssh* para iniciar los diferentes programas desarrollados. El nodo 1 realiza grabaciones del medio en busca de personas que puedan ser registradas. Para ello se ejecuta el mismo programa python de la Figura 5.20 mediante la orden *sudo python audio.py*. El nodo 2 se encarga de iniciar el servidor y el PA. Además, recibe los archivos generados por el nodo 1 ejecutando el programa servidor explicado en el apartado 4.2.2. Se inicia la ejecución del programa a través de la orden *sudo python3 servidor4.py*.

Figura 5.24. Ubicación de los nodos en la ruta.

Figura 5.25. Conexión de la Raspberry Pi.

Cuando el nodo 1 detecta a un caminante, este inicia el proceso comentado anteriormente de escaneo de *Beacons*. Para esta prueba se emplea la misma configuración de trama *Beacons* con nuestro DNI como UUID, usando la aplicación *Beacons Simulator* como transmisor de Beacons. En este caso, una vez se genere el archivo de datos se envía directamente al servidor debido a que solo se cuenta con un único nodo capaz de escanear el medio. Como resultado se obtiene un archivo de datos con una única marca de tiempo.

Se sigue avanzando a lo largo de la ruta hasta alcanzar el segundo nodo. Luego, nos tenemos que conectar a la red WiFi generada por este. En la Figura 5.26 se muestra como se crea y se puede acceder a la red correctamente. Una vez el teléfono móvil tenga conexión, se accede a la dirección *http://192.168.4.1/45342964.json* a través del navegador web para visualizar el archivo de datos generado. Como se muestra en la Figura 5.27 la prueba ha salido correctamente ya que se tiene acceso al archivo donde se puede ver la hora registrada a la que nos detectó el nodo 1.

5.1.6 Comparación de los resultados

En este apartado se comparan los resultados obtenidos en la prueba en exterior e interior teniendo en cuenta que las dos han salido correctamente. Se pueden destacar dos diferencias principales, la primera y más importante, es que en la prueba en exteriores únicamente se usaba

un dispositivo capa de detectar y registrar persona. Esto implica que el archivo generado solo dispone de una entrada de tiempo.

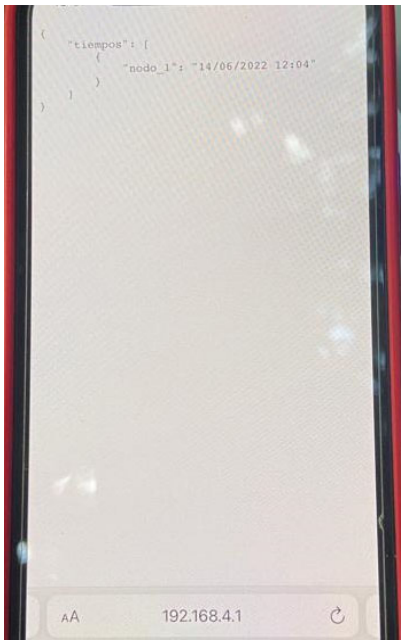

Figura 5.26. Creación de la red correctamente.

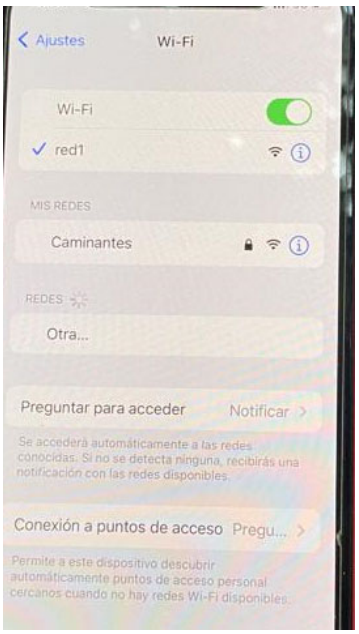

Figura 5.27. Acceso al archivo de datos generado.

La segunda diferencia es que en la zona donde se hizo la prueba en interior se detectaron dispositivos *Bluetooth* del entorno que no pertenecían a nuestro dispositivo transmisor. Debido a esto se generaron archivos de datos que no correspondían a ningún usuario y fueron alojados en el servidor. Sin embargo, este error se solucionó una vez se instaló la red en la ruta de Los Tilos de Moya. En esta zona las conexiones son prácticamente inexistentes, de esta forma únicamente se detectaba el dispositivo transmisor.

Se puede concluir que el sistema funciona correctamente y que su funcionamiento ideal sería en rutas de senderismo donde no se detecten señales Bluetooth no deseadas que puedan generar archivos que no correspondan a personas.

6 Conclusiones y posibles ampliaciones

En este capítulo se muestran las conclusiones conseguidas tras la configuración y el montaje de la plataforma, así como las de los experimentos realizados. Finalmente, se presentan una serie de posibles ampliaciones que se podrían llevar a cabo en el TFG.

6.1 Conclusiones

El senderismo es una actividad que se encuentra en pleno desarrollo actualmente. Cada vez son más las personas que optan por este tipo de actividades, tanto como práctica deportiva como simplemente actividad de ocio. Las condiciones geográficas y climática de las islas canarias convierten al archipiélago canario en un lugar ideal para el desarrollo de este tipo de actividades. Y esta es una forma de impulsar la economía canaria a través de una captación mayor de turismo rural, dejando así de sustentarla únicamente del turismo de playa, que siempre ha caracterizado a las islas.

El gran avance de las comunicaciones inalámbricas puede ayudar a continuar y fomentar el desarrollo de este sector. De esta idea surgió este proyecto en el cual se ha desarrollado una red inalámbrica Ad-Hoc para su instalación en rutas de senderismo y conseguir así que el usuario pueda disfrutar de una experiencia mucho más enriquecedora. Siendo ideal para todas aquellas personas que viven el senderismo como un tipo de deporte ya que el sistema desarrollado permite registrar información sobre la actividad realizada.

El registro de información acerca del usuario y la actividad que está realizando se consiguió gracias al uso de Beacons BLE. El uso de esta tecnología está muy extendido y le brinda al desarrollador una gran cantidad de posibilidades con las que trabajar. En nuestro caso nos permitió desarrollar un sistema capaz de identificar y registrar personas que se encontraran dentro del área de cobertura de la red.

Pensando en el ahorro del nivel de baterías de los equipos se diseñó una red acústica, la cual se encargaba de detectar el paso de personas cerca de alguno de los nodos de la red. De esta forma se consigue que el proceso de escaneo de *Beacons* solo se inicie cuando es necesario evitando así un alto consumo por parte de los nodos de la red.

Tras el montaje y configuración del sistema y tras los experimentos realizados, llevando a cabo una prueba real de su funcionamiento en Los Tilos de Moya entre otras pruebas, se puede concluir que se cumplen todos los objetivos planteados en la propuesta de TFG. El uso de este sistema es viable en todas aquellas rutas de senderismo en las que se quiera brindar al usuario una experiencia mucho más completa y convertir las zonas rurales de las islas en un reclamo para ese sector de la población amante de las caminatas de montaña.

En cuanto a los inconvenientes encontrados, únicamente hay que señalar que a través de los experimentos realizados se pudo determinar que el uso óptimo del sistema tiene lugar en zonas donde la existencia de otro tipo de señales procedente de dispositivos Bluetooth sea baja. Esto se debe a que estas señales también son detectadas por los equipos de la red acústica y pueden llegar a generar archivos que no correspondan a ninguna persona.

Finalmente, hay que añadir que con el desarrollo de este TFG he podido poner en práctica gran parte de los conocimientos aprendidos durante el grado como la creación de una red inalámbrica, el tratamiento de una señal digital, la configuración y manejo de ordenadores de placa reducida y el uso de los conocimientos sobre la programación para el desarrollo de los diferentes programas, entre otras muchas más cosas. Además, de poner en práctica mis habilidades de ingeniero para buscar, interpretar e implementar información en cosas desconocidas y poder analizar pruebas con determinación. Adquiriendo gracias al desarrollo de este proyecto un gran conocimiento sobre tecnologías nunca antes estudiadas que me aportan nuevas herramientas que me ayudarán en proyectos futuros.

6.2 Posibles ampliaciones

Con este proyecto se ha conseguido desarrollar un sistema de detección y registro de personas para uso en rutas de senderismo. Sin embargo, existen muchas funcionalidades que pueden ser añadidas dotando al sistema de una mayor cantidad de características y ampliando las posibilidades y escenarios de uso. A continuación, se detallan algunas de estas posibles ampliaciones que mejorarían el proyecto actual:

- *Configurar una red móvil Ad-Hoc*. Esta opción está pensada como una forma de poder gestionar y controlar por parte de los monitores de una excursión a todas las personas que formen el grupo. En este caso los monitores se encargarían de distribuir los equipos e ir recogiéndolos una vez se pase cerca de ellos. Los nodos se encargarían de detectar a los participantes y avisar al monitor si alguno no ha alcanzado el punto de control por algún motivo. En este caso sí deberían trabajar con algún protocolo de encaminamiento que calcule las nuevas rutas con los nodos que se van añadiendo a la red a lo largo de la ruta.

- *Tarjeta SIM en uno de los nodos*. Si uno de los nodos de la red Ad-Hoc estuviera equipada con una tarjeta SIM se podría tener acceso a Internet y aumentar así las prestaciones de del sistema. Se podría encaminar todo el tráfico generado en los diferentes nodos al nodo principal y subirlo a la red de forma instantánea. Además, se podría alertar a los servicios

de emergencia a través de nuestra red de posibles accidentes sufridos por las personas durante la actividad. Resolviendo así un grave problema actual debido a que son zonas donde la cobertura móvil es prácticamente inexistente.

- *Sensores de temperatura*. Se instalarían sensores de temperatura en los nodos de la red y se desarrollaría un programa capaz de detectar si la temperatura se eleva de manera repentina. Esto podría ser indicador de que un incendio está comenzando. Gracias al punto comentado anteriormente se podría avisar a los servicios de emergencia y poder actuar lo antes posible para evitar un crecimiento del incendio con todo lo que ello supone. Esta sería una gran ayuda a un problema que ha afecta en gran medida a las islas en los últimos 20 años [60].

Glosario

ACK	*Acknowledgement*
AODV	*Ad-hoc on Demand Distance Vector*
ARM	*Acorn RISC Machine*
BLE	*Bluetooth Low Energy*
BSS	*Basic Service Set*
CCK	*Complementary Code Keying*
CPU	*Central Processing Unit*
CSMA/CA	*Carrier-sense multiple access with collision avoidance*
CTS	*Clear to Send*
DBPSK	*Differential Binary Phase Keying*
DCF	*Distributed Coordination Function*
DHCP	*Dynamic Host Configuration Protocol*
DIFS	*DCF Interframe Space*
DNI	*Documento Nacional de Identidad*
DNS	*Domain Name System*
DQPSK	*Differential Quadrature Phase Shift Keying*
DSSS	*Direct Sequence Spread Spectrum*
FHSS	*Frequency Hopping Spread Spectrum*
GPIO	*General Purpose Input/output*
HDMI	*High-Definition Multimedia Interface*
HTTP	*Hypertext Transfer Protocol*
IBSS	*Independent Basic Service Set*
ID	*Identification*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
IP	*Internet Protocol*
JSON	*Javascript Object Notation*
LAMP	*Linux Apache MySQL PHP*
LLC	*Logical Link Control*
LXDE	*Lightweight X11 Desktop Environment*
MAC	*Media Access Control*
NACK	*Negative Acknewledgement*
OFDM	*Orthogonal Frequency Division Multiplexing*
OFDMA	*Orthogonal Frequency Division Multiplexing Access*
OLSR	*Optimized Link State Routing*
OS	*Operating System*
OSI	*Open System Interconnection*
PCF	*Point Coordination Function*
PLCP	*Physical Layer Convergence Procedure*

PMD	*Physical Medium Dependent*
QAM	*Quadrature Amplitude Modulation*
RAM	*Random Access Memory*
RPi	*Raspberry Pi*
RSSI	*Received Signal Strength Indicator*
RTS	*Ready To Send*
RU	*Resource Units*
SD	*Secure Digital*
SIM	*Subscriber Identity Module*
SoC	*System on Chip*
SYN	*Synchronize*
SYN-ACK	*Synchronize-Acknowledgment*
TFG	*Trabajo de Fin de Grado*
TCP	*Transmission Control Protocol*
UDP	*User Datagram Protocol*
USB	*Universal Serial Bus*
UUID	*Universally Unique IDentifier*
WEP	*Wired Equivalent Privacy*
WiFi	*Wireless Fidelity*
WLAN	*Wireless Local Area Network*
WMAN	*Wireless Metropolitan Area Network*
WPA/PSK	*WiFi Protected Access/Pre Shared Key*
WPAN	*Wireless Personal Area Network*
WPA2	*WiFi Protected Access 2*
WWAN	*Wireless Wide Area Network*

Referencias

[1] S. López, “Senderismo y el auge del turismo de naturaleza”, *El Guadarramista, 2021* [En línea]. Disponible en: https://elguadarramista.com/2021/02/16/senderismo-y-el-auge-del-turismo-de-naturaleza/. [Accedido: Feb. 06, 2022]

[2] R. Hernández Martín y A. Santana Talavera, “Destinos turísticos maduros ante el cambio: reflexiones desde canarias”. *Instituto Universitario de Ciencias Políticas y Sociales*, Univ. de la Laguna, 2010 [En línea]. Disponible en: https://riull.ull.es/xmlui/handle/915/762. [Accedido: Feb. 06, 2022]

[3] J. González Hernández, “Expansión, crisis y reajuste del sector turístico canario”, tesis doctoral, Univ. de la Laguna, 2021 [En línea]. Disponible en: https://riull.ull.es/xmlui/handle/915/24357. [Accedido: Feb. 06, 2022]

[4] F. J. Redondo Albajara, “Nuevas tecnologías aplicables a las empresas”, trabajo fin de grado, Univ. de Cantabria, 2017 [En línea]. Disponible en: https://repositorio.unican.es/xmlui/handle/10902/11282. [Accedido: Feb. 06, 2022].

[5] D. Gascón, “Redes de Sensores Inalámbricos, la tecnología invisible”, *Tecnología y Sociedad, 53-55, 2010* [En línea]. Disponible en: https://www.coit.es/sites/default/files/archivobit/pdf/bit-180-181-tecnologiaysociedad-redes-de-sensores-inalambricos.pdf. [Accedido: Feb. 06, 2022]

[6] “Python.org”. [Online]. Available: https://www.python.org/about/. [Accesed Jun. 24, 2022]

[7] “Python – Raspberry Pi Documentation” [Online]. Available: https://www.raspberrypi.org/documentation/usage/python/. [Accesed: Jun. 24, 2022]

[8] Debian, “Información sobre Debian bullseye” [En Línea]. Disponible en: https://www.debian.org/releases/bullseye/. [Accedido: Jun. 24, 2022]

[9] J.Salazar, “Redes Inalámbricas”, *TechPedia* [En Línea]. Disponible en: https://upcommons.upc.edu/bitstream/handle/2117/100918/LM01_R_ES.pdf. [Accedido: Jun. 24, 2022]

[10] “Informe de banda ancha en Canarias 2020”, Observatorio Canario de las Telecomunicaciones y de la Sociedad de la Información, 2021 [En línea]. Disponible en: https://www.octsi.es/images/documentos/2022/informe_banda_ancha_canarias_2020_edicion_2021.pdf. [Accedido: Jun. 24, 2022]

[11] “Redes Inalámbricas” [En Línea]. Disponible en: https://conceptoabc.com/redes-inalambricas/. [Accedido: Jun. 24, 2022]

[12] “IEEE 802.11, The Working Group Setting the Standards for Wireless LANs” [Online]. Available: http://www.ieee802.org/11/. [Accesed: Jun. 24, 2022]

[13] E. Karapistoli, F. Pavlidou, I. Gragopoulos and I. Tsetsinas, "An overview of the IEEE 802.15.4a Standard," in IEEE Communications Magazine, vol. 48, no. 1, pp. 47-53, January 2010, doi: 10.1109/MCOM.2010.5394030.

[14] “Red fija de estaciones remotas de vigilancia y control de la contaminación acústica” [En línea]. Disponible en: https://bit.ly/3yimTWH. [Accedido: Jun. 24, 2022]

[15] J. F. Alenza García, “La nueva estrategia contra la contaminación acústica y el ruido ambiental,” Revista Jurídica de Navarra, julio-diciembre, no. 36, pp. 65-120, 2003 [En linea]. Disponible en: https://academica-e.unavarra.es/handle/2454/27057. [Accedido: Feb. 22, 2022]

[16] Universidad Internacional de Valencia, “Tipos de redes inalambricas más comunes”, 2017 [En Línea]. Disponible en: https://www.universidadviu.com/es/actualidad/nuestros-expertos/tipos-de-redes-inalambricas-mas-comunes. [Accedido: Jun. 24, 2022]

[17] L. J. Perez Chaves y S. Y. Rodríguez Amortegui, “Desarrollo de un prototipo de red mesh basada en dispositivos Raspberry Pi para la comunicación en situaciones de desastres”, trabajo fin de grado, Univ. Antonio Nariño, 2020 [En línea]. Disponible en: http://repositorio.uan.edu.co/bitstream/123456789/1988/1/2020JulianaPerezYulianaRodriguez.pdf. [Accedido: Jun. 24, 2022]

[18] A. Recalde Baraibar y M. Rodríguez Alija, “Redes inalámbricas” [En línea]. Disponible en:

https://www.tlm.unavarra.es/~daniel/docencia/rba/rba06_07/trabajos/resumenes/gr01-RedesInalambricas.pdf. [Accedido: Jun. 24, 2022]

[19] J. P. Hernández y D. Márquez, “Redes móviles Ad-Hoc”, trabajo fin de grado, Univ. Nacional de Rosario, 2006. [En línea] Disponible en: https://www.dsi.fceia.unr.edu.ar/downloads/distribuidos/material/monografias/RedesMovilesAdHoc.pdf. [Accedido: Jun. 24, 2022]

[20] J. M. García Campos, “Evaluación de protocolos de encaminamiento para redes vehiculares VANET”, trabajo fin de grado, Univ. de Sevilla, 2014 [En línea]. Disponible en: https://bit.ly/3A9oGyM. [Accedido: Jun. 24, 2022]

[21] J. J. Vinagre Díaz, “Teoría del encaminamiento en redes Ad-Hoc inalámbricas”, tesis doctoral, Univ. Carlos III de Madrid [En línea]. Disponible en: https://e-archivo.uc3m.es/bitstream/handle/10016/2462/Tesis_JJ_Vinagre_Diaz.pdf?sequence=1&isAllowed=y. [Accedido Jun. 24, 2022]

[22] F. J. Hidalgo Pastor, “Estudio de viabilidad de la utilización de redes inalámbricas Ad-Hoc en edificos departamentales”, trabajo fin de máster, Univ. Politécnica de Valencia, 2008 [En línea]. Disponible en: https://riunet.upv.es/handle/10251/13183. [Accedido: Jun. 24, 2022]

[23] L. Coya Rey, T. O. Ledesma Quiñones y W. Baluja García, “Protocolos de enrutamiento aplicables a redes MANET”, Telemática, vol. 13, No. 3, pp. 59-74, 2014 [En Línea]. Disponible en: https://revistatelematica.cujae.edu.cu. [Accedido: Jun. 24, 2022]

[24] E. Aragón Monroy, “Comparación del desempeño de los protocolos de enrutamiento AODV y DSR sobre una red MANET experimental”, Univ. Militar Nueva Granada, Bogotá 2012 [En línea]. Disponible en: https://bit.ly/3I1iM4x. [Accedido: Jun. 24, 2022]

[25] G. R. Piermattei Monney y B. E. Garcia Ferrari, “Análisis de los protocolos de ruteo OLSR y AODV en redes Ad-Hoc IBSS”, trabajo fin de cátedra, Univ. Nacional de Río Cuarto, 2011 [En línea]. Disponible en: https://41jaiio.sadio.org.ar/sites/default/files/40_EST_2012.pdf. [Accedido: Jun. 24, 2022]

[26] “¿Qué es Raspberry Pi?”. [En línea]. Disponible en: https://raspberrypi.cl/que-es-raspberry/. [Accedido: Jun. 24, 2022]

[27] “Comparativa y análisis: Raspberry Pi vs competencia”. [En línea]. Disponible en: https://www.comohacer.eu/comparativa-y-analisis-raspberry-pi-vs-competencia/?amp. [Accedido: Jun. 24, 2022]

[28] “Mini USB Microphone”. [Online]. Available: https://www.adafruit.com/product/3367. [Accesed: Jun. 24, 2022]

[29] “Archer T2U Plus”. [En línea]. Disponible en: https://www.tp-link.com/es/home-networking/adapter/archer-t2u-plus/#specifications. [Accedido: Jun. 24, 2022]

[30] “TP-Link - Descargas”. [En línea]. Disponible en: https://www.tp-link.com/es/support/download/. [Accedido: Jun. 24, 2022]

[31] R. Solé, “¿Cuál es el mejor sistema operativo para la Raspberry Pi?”, *Profesional Review, 2021*. [En línea]. Disponible en: https://www.profesionalreview.com/2021/09/10/sistema-operativo-raspberry-pi/. [Accedido: Jun. 24, 2022]

[32] “Raspbian”. [Online]. Available: https://www.raspbian.org/FrontPage. [Accesed: Jun. 24, 2022]

[33] “Raspberry Pi OS” [Online]. Available: https://www.raspberrypi.com/software/. [Accesed: Jun. 24, 2022]

[34] “El lenguaje de programación Python” [En Línea]. Disponible en: https://czayas.gitbooks.io/paradigmas-de-la-programacion-con-python/content/a1-python.html. [Accedido: Jun. 24, 2022]

[35] P. Catania, “Principales lenguajes de programación en Raspberry Pi”, *Adsl Zone, 2021*. [En línea]. Disponible en: https://www.adslzone.net/noticias/productos/top-lenguajes-programacion-raspberry-pi-desarrollo/. [Accedido: Jun. 24, 2022].

[36] “¿Qué es Apache y para qué sirve? – Servidor Web”. [En Línea]. Disponible en: https://dinahosting.com/ayuda/que-es-apache-y-para-que-sirve/. [Accedido: Jun. 24, 2022]

[37] “Apache HTTP Serve Project”. [En Línea]. Disponible en: https://bit.ly/3u2sj5O. [Accedido: Jun. 24, 2022]

[38] 802.11-1997 IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements-Part 11: Wireless LAN Mediu.

[39] J. Mauricio Chimento, “Redes inalámbricas”, Univ. Nacional de Rosario [En línea].

Disponible en: https://bit.ly/3nkmQUh. [Accedido: Jun. 24, 2022]

[40] “Control de acceso al medio”. [En Línea]. Disponible en: https://guimi.net/monograficos/G-Redes_de_comunicaciones/G-RCnode31.html. [Accedido: Jun. 24, 2022]

[41] “What Is OFDMA?” [Online]. Available: https://www.cisco.com/c/en/us/products/wireless/what-is-ofdma.html. [Accesed: Jun. 24, 2022]

[42] “El estándar Bluetooth IEEE 802.15.1”. [En Línea]. Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lem/archundia_p_fm/capitulo3.pdf. [Accedido: Jun. 24, 2022]

[43] StarTech, “What is a Bluetooth class and what is a Bluetooth profile?”. [Online]. Available: https://www.startech.com/en-us/faq/bluetooth-adapters-classes-and-profiles. [Accesed: Jun. 24, 2022]

[44] Digital Guide IONOS, “¿Qué es Bluetooth?”, 2020 [En Línea]. Disponible en: https://www.ionos.es/digitalguide/servidores/know-how/que-es-bluetooth/. [Accedido: Jun. 24, 2022]

[45] D. Ros López, “Diseño y desarrollo de una aplicación web de monitorización de dispositivos Bluetooth Low Energy”, trabajo fin de grado, Univ. Politécnica de Cartagena, 2021 [En línea]. Disponible en: https://repositorio.upct.es/bitstream/handle/10317/10081/tfg-ros-dis.pdf?sequence=1. [Accedido: Jun. 24, 2022]

[46] Moko Smart, “Protocolo Eddystone y especificaciones”. [En Línea]. Disponible en: https://www.mokosmart.com/es/eddystone-protocol-and-specifications/. [Accedido: Jun. 24, 2022].

[47] J. Molina Machado, “Diseño y desarrollo de Readers Bluetooth Low Energy para IoT”, trabajo fin de grado, Univ. Politécnica de Cartagena, 2018 [En línea]. Disponible en: https://repositorio.upct.es/handle/10317/7448. [Accedido: Jun. 24, 2022]

[48] Unipython, “Programación de redes en python: sockets”. [En Línea]. Disponible en: https://unipython.com/programacion-de-redes-en-python-sockets/. [Accedido: Jun. 24, 2022]

[49] S. De Luz, “Qué es un socket TCP o UDP y qué diferencias hay con los puertos”,

Redes Zone, 2022. [En línea]. Disponible en: https://www.redeszone.net/tutoriales/configuracion-puertos/que-es-socket-tcp-udp-diferencias-puertos/. [Accedido: Jun. 24, 2022]

[50] S. De Luz, "¿Qué protocolo es mejor?: TCP vs UDP, descubre cuándo utilizar cada uno", *Redes Zone, 2022*. [En línea]. Disponible en: https://www.redeszone.net/tutoriales/internet/tcp-udp-caracteristicas-uso-diferencias/. [Accedido: Jun. 24, 2022]

[51] "Algoritmo de Checksum de Internet". [En Línea]. Disponible en: http://www.arcesio.net/checksum/checksuminternet.html. (último acceso Jun. 24, 2022).

[52] T. Monreal y P. Ibáñez, "Programación en red sobre TCP/IP Interface sockets" [En Línea]. Disponible en: http://webdiis.unizar.es/asignaturas/labcom/documentacion-sockets.pdf. [Accedido: Jun. 24, 2022]

[53] "How to install pyaudio" [Online]. Available: https://forums.raspberrypi.com/viewtopic.php?t=25173. [Accesed: Jun. 24, 2022]

[54] Programador Clic, "Instale y configure numpy y scipy". [En Línea]. Disponible en: https://programmerclick.com/article/82221061450/. [Accedido: Jun. 24, 2022]

[55] "waves" [Online]. Available: https://libraries.io/pypi/waves. [Accesed: Jun. 24, 2022]

[56] L. Llamas, "Como montar un servidor web Apache en Raspberry Pi", *Ingeniería, informática y diseño, 2019* [En línea]. Disponible en: https://www.luisllamas.es/como-montar-un-servidor-web-apache-en-raspberry-pi/. [Accedido: Jun. 24, 2022]

[57] VidaTecno, "Cómo convertir tu Raspberry Pi en un punto de acceso inalámbrico" [En Línea]. Disponible en: https://vidatecno.net/como-convertir-tu-raspberry-pi-en-un-punto-de-acceso-inalambrico/. [Accedido: Jun. 24, 2022].

[58] "Protocolo SSH" [En Línea]. Disponible en: https://bit.ly/3yofqFT. [Accedido: Jun. 24, 2022]

[59] "Beacon Simulator". [En Línea]. Disponible en: https://play.google.com/store/apps/details?id=net.alea.beaconsimulator&hl=es&gl=US. [Accedido: Jun. 24, 2022]

[60] N. Torres, “14 grandes incendios en Canarias desde el año 2000”, *Diario de Avisos, 2019*. [En línea]. Disponible en: https://diariodeavisos.elespanol.com/2019/08/14-grandes-incendios-en-canarias-desde-el-ano-2000/. [Accedido: Jun. 24, 2022]

Pliego de condiciones

En este episodio se detallan los aspectos relacionados con la información de licencias, los derechos de autoría y las responsabilidades. Así como las condiciones bajo las que se ha desarrollado este TFG y las que se establecen al usuario para poder usar la plataforma que se ofrece.

PL1. Condiciones Hardware

Durante el desarrollo de este TFG se han usado los dispositivos hardware recogidos en la Tabla PL.1.

Equipo	Modelo	Fabricante
Alimentador y cable micro USB genérico	5V @ 2A	Genérico
Mini micrófonos USB	MI-305	Adafruit
Ordenador portátil	MacBook Pro 2015	Apple
Raspberry Pi	3 Model B+ y 4	Fundación Raspberry Pi
Tarjetas MicroSD	UHS-1 Clase 10	Sandisk

Tabla PL.1. Condiciones del Hardware.

PL2. Condiciones Software

En la Tabla PL.2 se exponen las herramientas software utilizadas, especificando su versión.

Aplicación	Versión
MacOS Catalina	10.15.6
Microsoft Office	16.42
Python	3.9.2
Raspbian	Debian Bullseye
Visual Studio Code	1.67.1
Wireshark	3.4.10

Tabla PL.2. Condiciones del Software.

PL3. Condiciones de licencia

El código desarrollado en este trabajo es propiedad de la Universidad de Las Palmas de Gran Canaria y todos los que pretendan hacer uso de él deben aceptar todas las cláusulas establecidas en esta licencia. El uso de las plataformas se podría hacer bajo autorización del autor, tutores del TFG, y la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria.

PL4. Derechos de autor

Tanto el código como la plataforma e información que se adjunta están protegidos por las leyes de propiedad intelectual que les sean aplicables, así como las disposiciones de los tratados internacionales. Por tanto, el código se considera un producto protegido por derechos de autor. Esto no es óbice para que una persona pueda copiar o utilizar el código bajo la autorización del autor, tutores del TFG, y la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de Las Palmas de Gran Canaria.

PL5. Restricciones

No se permite el uso de ingeniería inversa. Sí se permite la transferencia del código a un tercero siempre que no se conserve ninguna copia, incluyendo actualizaciones o material escrito adicional.

PL6. Garantía

El autor del TFG lo presenta "AS IS" (tal cual), sin garantía implícita de ningún tipo. No se responsabiliza de los daños que pudieran causar a equipos o personas por el uso del código o la documentación. El autor no asegura, garantiza, o realiza ninguna declaración sobre el uso y resultados derivados de la utilización del código y/o del resto de la información proporcionada.

El código y la plataforma elaborada no están exentos de errores y no está diseñado para entornos de riesgo que requieran de un funcionamiento a prueba de fallos. El autor rechaza expresamente cualquier garantía explícita e implícita de adecuación del código para actividades de riesgo.

PL7. Limitación de responsabilidad

En ningún caso el autor ni los tutores, ni la Escuela de Ingeniería de Telecomunicación y Electrónica de la Universidad de las Palmas de Gran Canaria serían responsables de los perjuicios directos, indirectas incidentales o consiguientes, gastos, lucro cesante, pérdida

de ahorros, interrupción de negocios, pérdida de información comercial o de negocio, o cualquier otra pérdida que resulte del uso o de la incapacidad de usar el código o la documentación. El usuario conoce y acepta este riesgo, así como el resto de las cláusulas y restricciones. El autor no reconoce otra garantía que no haya sido indicada anteriormente.

PL8. Otras consideraciones

En el supuesto de que cualquier disposición de esta licencia sea declarada total o parcialmente inválida, las cláusulas afectadas serían modificadas convenientemente de manera que sean ejecutables una vez modificadas, permaneciendo el resto de este contrato en vigencia. Este contrato se rige por las leyes de España. El usuario acepta la jurisdicción exclusiva de los tribunales españoles con relación a las disputas derivadas de la presente licencia.

Presupuesto

Este capítulo contiene el presupuesto que recoge los gastos generados en la realización del presente TFG.

P.1 Componentes del presupuesto

El presupuesto calculado se divide en las siguientes partes:

- Recursos materiales.
- Trabajo tarifado por tiempo empleado.
- Material Fungible.
- Redacción de la documentación.
- Derechos de visado del *Colegio Oficial de Ingenieros Técnicos de Telecomunicación* (*COITT*).
- Gastos de tramitación y envío.
- Aplicación de impuestos y coste total.

P.2 Recursos Materiales

Para la correcta ejecución y desarrollo del TFG han sido necesarios diversos recursos hardware y herramientas software, las cuales pueden llevar asociadas un coste en sus licencias. Entre estos recursos, se podrían destacar el paquete de Microsoft Office para la redacción de esta memoria, las Raspberry Pi 3 y 4 usadas como nodos en la red Ad-Hoc y sus respectivos accesorios, el ordenador portátil y los micrófonos usados.

Así cabe destacar que el software utilizado no conlleva un pago de licencia, a excepción del Microsoft Office.

Con el fin de establecer un cálculo de la amortización, se presupone el sistema de amortización como lineal, de tal forma que se asume que el inmovilizado material se desprecia de forma constante a lo largo de su vida útil.

Así, para llevar a cabo el cálculo de la cuota de amortización anual, se calcula usando la Ecuación P.1.

$$Cuota\ anual = \frac{Valor\ adquisicion - Valor\ residual}{Número\ de\ años\ de\ vida\ útil}$$

Ecuación P.1

De esta manera, la amortización total aparece reflejada en la Tabla P.1, así como los diferentes valores y cuotas de los diferentes recursos empleados en este TFG.

Recurso	Unidades	Valor de adquisición (€)	Valor residual (€)	Vida útil (años)	Cuota anual (€)	Uso (meses)	Cuota aplicable (€)
MacBook Pro Intel® Core™ i7 8 GB RAM, 256 GB SSD	1	1600	400	6	98	4	200
RPi 3 Model B+	2	38	11,4	5	5,32	4	5,32
RPi 4	1	38	11,4	5	5,32	4	5,32
Alimentador y cable micro USB Genéricos	3	9	2,7	5	1,30	4	1,30
MicroSD SanDisk Ultra 16GB, UHS-1	3	9	2,7	2	3,15	4	3
Mini micrófono USB MI-305	2	4	1	5	0.8	4	0.6
Batería externa TP-Link 10400mAh	2	22	6	5	3,5	4	3,2
Licencia Paquete Office Windows (2017)	1	150	0	5	30	4	10
Adaptador WiFi TP-Link Archer T2U Plus	1	20	5	1	3,5	4	15
Teléfono iphone 12 mini 128GB iOS 15	1	500	350	2	166	4	75
MATERIALES AMORTIZABLES	**TOTAL**						318.74

Tabla P.1. Amortización total.

El valor residual puede ser definido como el valor teórico supuesto que tendría el elemento después de su vida útil. En el caso de un terminal móvil y portátil se deprecian en una tasa del 30 % con respecto a su valor. Según *Artículo 34*, *LISR*.

El coste de los materiales amortizables es de un total de: trescientos dieciocho euros con setenta y cuatro céntimos (318.74€).

P.3 Trabajo tarificado por tiempo empleado

Este concepto contabiliza los gastos que corresponden a la mano de obra, según el salario correspondiente a la hora de trabajo de un Ingeniero Técnico de Telecomunicación.

Según la tabla retributiva de personal contratado en proyectos de investigación elaborada por la ULPGC en el año 2016, este salario, con una dedicación de 20 horas semanales, asciende a 896,31€ mensuales. Lo que se aproxima a una retribución de 11,20 €/h. Este TFG tal como comprende el Proyecto Docente ha conllevado 300 horas, por lo que se calcula el coste total por tiempo empleado en:

$$11{,}20^{*}\ 300 = 3360{,}00\ €$$

Por lo tanto, el trabajo tarifado por tiempo empleado asciende a la cantidad de tres mil trescientos sesenta euros (3.360,00€).

P.4 Redacción del trabajo

Se ha utilizado la Ecuación P.2 para determinar el coste asociado a la redacción de la memoria del presente TFG.

$$R = 0{,}07\ X\ P\ X\ C_n$$

Ecuación P.2

Donde:

- R son los honorarios por la redacción del trabajo.
- P es el presupuesto.

- C_n es el coeficiente de ponderación en función del presupuesto.

El valor del presupuesto P se calcula sumando los costes del trabajo tarifado por tiempo empleado y de la amortización del inmovilizado material, tanto hardware como software. El resultado de los costes se muestra en la Tabla P.2.

Como el coeficiente de ponderación Cn para presupuestos menores de 30.050,00€ viene definido por el COITT con un valor de 1.00, el coste derivado de la redacción del TFG es de:

$$R = 0{,}07 \times 3.678{,}74€ \times 1 = 257{,}52\ €$$

Ascendiendo de esta forma el coste de la redacción del trabajo a doscientos cincuenta y siete euros con cincuenta y dos céntimos (257,52€).

Concepto	**Coste (€)**
Trabajo tarifado por tiempo empleado	3.360,00
Amortización del material	318,74
Total	3.678,74

Tabla P.2. Valor del presupuesto.

P.5 Material Fungible

Los costes asociados a la de edición de documentos, así como los gastos del material de oficina se recogen en la Tabla P.3.

Los gastos asociados al material fungible ascienden a un total de: sesenta y tres con ochenta euros.

Descripción de gastos	**Coste (€)**
Impresión	60
Encuadernación	3,80
Total	63,80

Tabla P.3. Material fungible.

P.6 Derechos de visado del COITT

El COITT establece que, para proyectos técnicos de carácter general, los derechos de visado para 2016 se calculan en base a:

$$V = 0{,}006 \; X \; P_1 \; X \; C_1 + 0{,}003 \; X \; P_2 \; X \; C_2$$

Donde:

- V es el coste de visado del trabajo.
- P_1 es el presupuesto del proyecto.
- C_1 es el coeficiente reductor en función del presupuesto.
- P_2 es el presupuesto de ejecución material correspondiente a la obra civil.
- C_2 es el coeficiente reductor en función a P2.

El valor del presupuesto P_1 se halla sumando los costes de las secciones correspondientes al trabajo tarifado por tiempo empleado, a la amortización del inmovilizado material y a la redacción del documento. Esta suma se muestra en la Tabla P.4. Al igual que en el caso anterior, el coeficiente C_1 para proyectos de presupuesto inferior a 30.050,00€ es de 1,00, asimismo el valor de P_2 es de 0,00€ ya que no se realiza ninguna obra.

De esta forma, aplicando a la P.3 los datos de la P.4 y el coeficiente especificado se obtiene:

$$V = 0{,}006 \text{ x } 3.936{,}26 \text{ € x } 1 = 23{,}61 \text{ €}$$

Los costes por derechos de visado del presupuesto ascienden a veintitrés euros con sesenta y dos céntimos (23,62 €).

P.7 Gastos de tramitación y envío

Los gastos de tramitación y envió están estipulados en seis euros (6,00 €) por cada documento visado telemático.

Concepto	Coste (€)
Trabajo tarifado por tiempo empleado	3.360,00
Amortización del material	318,74
Redacción del trabajo	257,52
Total	3.936,26

Tabla P.4. Presupuesto.

P.8 Aplicación de impuestos y coste total

El presupuesto total del presente TFG está gravado por el Impuesto General Indirecto Canario (*IGIC*), que está establecido en la actualidad en un siete por ciento (7 %). El coste total de este TFG se encuentra desglosado en la Tabla P.5.

Descripción	Subtotal (€)
Amortización de materiales	318,74
Trabajo tarifado por tiempo empleado	3.360,00
Costes de material fungible	63,68
Redacción del trabajo	257,52
Derechos de visado del COIT	23,61
Gastos de tramitación y envío	6,00
Suma (€)	4.029,55
IGIC (7%)	282,07
TOTAL	4.311,62

Tabla P.5. Coste total.

El presupuesto total del proyecto Implementación de un sistema de detección acústica empleando los nodos de una red inalámbrica en zonas rurales asciende a: cuatro mil trescientos once euros con sesenta y dos céntimos (4.311,62€).

Fdo.: D. Francisco Carlos Felipe Rodríguez

En Las Palmas de Gran Canaria a 7 de julio de 2022.