Universidad Politécnica de Cartagena

**TRABAJO DE FIN DE GRADO**

**INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN**

etsit escuela técnica superior de ingeniería de telecomunicación

# Punto de Acceso WiFi Inteligente con Antena de tipo Leaky-Wave y Esquema de Salto en Frecuencia


***Guillermo Martínez Pérez***

Dirigido por:
Jose Luis Gómez Tornero

Codirigido por:
Juan Carlos Jacobo Aarnoutse Sánchez


17 de Mayo de 2020

*Dedicado a,*

*mi familia.*

# Índice general

# Resumen

El wifi es un término que proviene de Wireless Fidelity, es decir, fidelidad inalámbrica. Es una tecnología destinada a la comunicación inalámbrica entre dispositivos electrónicos y se basa en el estándar 802.11. [1]

Mayoritariamente, se utiliza para el uso doméstico permitiendo el acceso a internet a los dispositivos móviles como ordenadores portátiles o teléfonos móviles. Sin embargo, su funcionalidad no queda sólo ahí si no que se utiliza con otros muchos fines como, por ejemplo: manejar un dispositivo por control remoto, imprimir documentos en una impresora, etc.

Los dos elementos principales de una red wifi son el *router* o punto de acceso AP y las estaciones que se conectan al AP.

El *router* consta de una o varias antenas que permiten transmitir y recibir información por el aire en forma de señal de radio. Por otro lado, las estaciones o clientes se conectan con el punto de acceso para intercambiar información con él. Por ello, también necesitan una o varias antenas para comunicarse de forma inalámbrica con el *router*.

Estas antenas suelen ser omnidireccionales para garantizar la cobertura en toda el área de la WLAN. El uso de antenas direccionales permite enfocar el haz de la antena a una zona del espacio, mejorando la relación señal a ruido SNR y por tanto permite un aumento en la velocidad de transmisión (*data rate*) [2].

Esta comunicación entre los dispositivos móviles y el *router*, se realiza en varias frecuencias denominadas canales wifi que van desde los 2412 Hz hasta los 2484 Hz en la banda de 2.4 GHz.

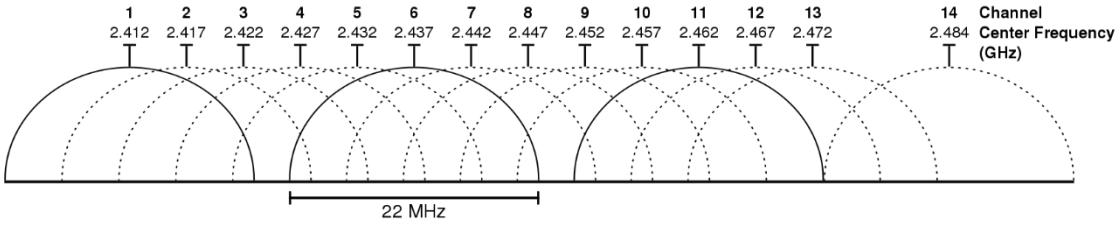



*Figura 1.1: Frecuencias wifi en la banda de 2.4 GHz*

En 2009 se empezó a usar el estándar 802.11n que introdujo la banda de 5 GHz. Esta banda es más estable y fiable que la banda de 2.4 GHz, aunque debido al aumento en la frecuencia de la transmisión, el alcance es menor.

En este proyecto, trabajaremos en la banda de 2.45 GHz y demostraremos que utilizando la técnica de *frequency hopping* junto con una antena direccional denominada leaky-wave, podemos mejorar las prestaciones de las redes WLAN wifi actuales. El formato de comunicación *frequency hopping* consiste en cambiar de forma periódica la frecuencia en la que se están comunicando el punto de acceso y los clientes. Esto permite una mayor seguridad en la comunicación debido a que deberemos conocer la secuencia de los saltos para entender la comunicación que está teniendo lugar.

En nuestro caso, utilizaremos este concepto para que, en función de la frecuencia de comunicación escogida, el haz de la antena apunte en una dirección u otra. Este cambio en la dirección de radiación en función de la frecuencia nos permite elaborar un sistema de localización para detectar el ángulo en el que se posiciona el cliente en relación con la orientación de la antena. Una posible aplicación para este sistema es la integración de los elementos de localización y control en un RPA (*Remoted Piloted Aircraft*) como un multicóptero, más conocido como *drone*.

# Lista de figuras

# Lista de tablas

# Capítulo Primero

## Introducción a las redes wifi

El propósito de este capítulo es contar la historia del WiFi, desde sus comienzos hasta la actualidad y proyectos futuros.

En primer lugar, debemos mencionar a la actriz Hedy Lamar, quien se graduó de ingeniería y junto con el compositor George Antheil, patentó una técnica de modulación de señales en espectro expandido en 1914, que no se utilizó hasta 1960, para fines militares [3]. Aunque este es el origen más extendido, existen patentes anteriores que ya hacían uso de esta técnica como la patentada por Boretjes Willem en [4] o la patentada por Nikola Tesla en [5].

Seguidamente, el físico alemán Rudolph Hertz [6] utilizó ondas electromagnéticas para realizar la primera comunicación inalámbrica y pocos años después, se transmitieron los primeros mensajes completos por el Atlántico.

Estos fueron los precursores de la transmisión de información a través de ondas de radio. Pero los primeros intentos de usar estas ondas con fines informáticos llegaron en el 1971, cuando una red de ordenadores de la Universidad de Hawái llamada ALOHAnet [7], conectó 7 ordenadores con un ordenador central, permitiendo el intercambio de paquetes a través de ondas UHF. Un año después, ALOHA se conectó al continente americano mediante ARPANET, una red de computadoras de EEUU.

En 1985 la FCC americana (Comisión Federal De las Comunicaciones) y más tarde el resto de los países, estableció las características que tenía que tener una red inalámbrica asignándoles las bandas ISM (Industrial, Scientific, Medical). Esta banda incluye numerosas frecuencias entre las que se encuentran la de 2.4 y la de 5 GHz.

En 1997 se aprobó el estándar IEEE 802.11, el cuál fue creado con la misión de establecer unas normas de transmisión de datos a través de las redes inalámbricas WLAN. [8]

A finales de los años 90 se creó WECA (Wireless Ethernet Compatibility Association), una asociación de compañías como Nokia, Lucent o Symbol Technologies cuyos objetivos eran promover la tecnología WiFi y hacer posible la compatibilidad con esta tecnología. WECA, tras garantizar la compatibilidad de los equipos con tecnología inalámbrica, pasó a llamarse Wi-Fi Alliance en 2003, cuyo logo es el que actualmente identifica las zonas wifi. [9]

*Figura 1.2: Logotipo de Wi-Fi Alliance*

Seguidamente y hasta la actualidad, la evolución del protocolo 802.11, se ha basado en mejoras en la velocidad, el alcance y la seguridad.

Estos avances permiten que, en la actualidad, la velocidad que se puede alcanzar mediante WiFi, sea equiparable a la que se alcanza mediante ethernet.

# Capítulo Segundo

## Fundamentos y conceptos generales

### 2.1 El estándar IEEE 802.11

Con el comienzo de las redes inalámbricas, surgieron compañías que utilizaban la tecnología WiFi para satisfacer unas necesidades u otras. Estas compañías necesitaban que sus productos fueran compatibles con los de otras empresas y por ello surgió la necesidad de tener un estándar que seguir.

Esta estandarización fue establecida por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) y ha permitido conseguir una plataforma abierta, con productos de mayores prestaciones y un precio más ajustado.

En el estándar 802.11 se definen varios protocolos, donde cada uno de ellos responde a unas necesidades. Estos protocolos establecen frecuencias de funcionamiento, velocidades de transmisión o número de canales entre otras características. Los canales son establecidos en función del país en el que nos encontremos y cada país tiene una regulación en la que se especifican los canales permitidos en función de qué tipo de uso se le va a dar (civil, militar…).

| Estándar | Año de aprobación | Frecuencia | Ancho del canal | Velocidad de datos |
|---|---|---|---|---|
| 802.11 (original) | 1997 | 2.4 GHz | 20 MHz | 2Mbps |
| 802.11a | 1999 | 5 GHz | 20 MHz | 54 Mbps |
| 802.11b | 1999 | 2.4 GHz | 20 MHz | 11 Mbps |
| 802.11g | 2003 | 2.4/5 GHz | 20 MHz | 54 Mbps |
| 802.11n | 2009 | 2.4/5 GHz | 20, 40 MHz | 450 Mbps |
| 802.11ac wave 1 | 2014 | 5 GHz | 20, 40, 80 MHz | 866.7 Gbps |
| 802.11ac wave 2 | 2015 | 5 GHz | 20, 40, 80, 160 MHz | 1.73 Gbps |
| 802.11ad | 2016 | 60 GHz | 2.16 GHz | 7 Gbps |
| 802.11af | 2014 | 2.4/5 GHz | 6, 8 MHz | 25 Mbps - 425 Mbps (dependiendo del canal) |
| 802.11ah | 2016 | 900 MHz | 1, 2, 4, 8, 16 MHz | 347 Mbps |
| 802.11ax | 2019 | 2.4/5 GHz | 20, 40, 80, 160 MHz | 2.4 Gbps |

*Tabla 2.1: Estándares de WiFi 802.11*

En nuestro caso, utilizaremos el estándar 802.11g para realizar las pruebas necesarias puesto que es el estándar que soporta el *router* del que disponemos.

## 2.2 Funcionamiento de una comunicación WiFi

En este punto se recoge el mecanismo de intercambio de datos entre los dos extremos de la comunicación WiFi.

Por un lado, tenemos tanto puntos de acceso como *routers* WiFi.

Los puntos de acceso se encargan de dar acceso a la red de una forma inalámbrica. No conectan redes lógicas diferentes ni reenvían entre ellas, de eso se encargan los *routers*, que como el mostrado en la figura 2.1, muchos modelos también disponen de una interfaz WiFi.

Este último elemento es el núcleo del sistema y se encarga de administrar el tráfico de información de una red. Puede ser utilizado para proveer de internet a otros elementos de la red además de otras tareas, como por ejemplo controlar la calidad de servicio.

Una vez establecida esta diferencia entre un AP y un *router* con wifi, es importante señalar que en muchas ocasiones también se referencia a un *router* con el término AP dado que en numerosas ocasiones se pueden considerar como un equipo formado por un AP (puesto que da acceso a los equipos inalámbricos) mas un *router* (puesto que realiza las labores de interconexión de redes lógicas de un *router).*


*Figura 2.1: Ejemplo de WAP con antena omnidireccional*


*Figura 2.2: WAP MiMo. Utiliza varias antenas de forma conjunta*

En la Figura 2.1 se muestra un WAP (Wireless Access Point) básico, que contiene una antena omnidireccional inalámbrica. Este equipo irradia de forma esférica y con la misma potencia en todas las direcciones del espacio. Este tipo de punto de acceso es el más básico, puesto que no contiene ningún mecanismo de direccionamiento del haz. Esto significa que no enfocan la potencia de la señal en una dirección del espacio haciendo un desaprovechamiento de gran parte de esta energía.

Por otro lado, tenemos los WAPs que implementan las técnicas MiMo (Multiple input Multiple output), que hacen uso de varias antenas omnidireccionales para que el efecto de la unión de ellas provoque una mayor directividad y ganancia. Un ejemplo de un WAP que utiliza esta técnica se muestra en la Figura 2.2

En el otro extremo de la comunicación tenemos al cliente. Puede ser un teléfono móvil, un ordenador o cualquier otro dispositivo con capacidad de establecer conexión con el *router* (o con el Punto de Acceso). Esta comunicación se realiza a través de ondas de radio, normalmente 2.4 GHz y recientemente 5 GHz. La diferencia entre ambas frecuencias es que 2.4 GHz tiene un alcance mayor y más resistencia a obstáculos, pero al ser una frecuencia menor que 5 GHz, la tasa de datos que se pueden transmitir es menor. En los equipos de mayores prestaciones se combina la transferencia de datos por ambas bandas.

Para que un cliente se conecte al AP, primero debe saber de su existencia. Para ello se utilizan las tramas *beacon*, que consisten en paquetes de datos que el AP transmite de forma periódica en todos los canales para que todos los dispositivos inalámbricos sepan de la presencia de una red WLAN. Estos parámetros se dividen entre una cabecera MAC, que indica la dirección física del AP, un cuerpo y el FCS (*Frame Check Secuence*). Este último especifica una secuencia de detección de errores. Dentro del cuerpo podemos encontrar varios parámetros, pero los más importantes para este trabajo son:

- Timestamp: es una ayuda a la sincronización puesto que especifica el instante de tiempo para que todas las estaciones se sincronicen a ese tiempo.
- Intervalo de Beacon: determina el periodo de tiempo que transcurre entre dos beacon consecutivos.
- Capacidades de seguridad: WEP, WPA, WPA2, etc.
- Canal: indica la frecuencia a la que el AP está configurado.
- SSID: nombre de la red WLAN
- Tasas soportadas: son las velocidades de transmisión que el AP es capaz de gestionar.
- Ancho del canal: 20, 40, 80, 160 MHz
- TIM/DTIM: muy útil para control de potencia. Permite a los dispositivos ahorrar energía si no se está llevando a cabo un intercambio de información.

Para nuestro proyecto, las tramas *beacon* nos servirán para realizar la configuración inicial y para realizar el aviso de cambio de canal por parte del *router* hacia los distintos clientes conectados.

Nuestro *router* consiste en una placa integrada WPJ531 7A03. Esta placa actúa como punto de acceso y contiene una interfaz wifi junto con dos puertos de ethernet y una ranura miniPCI-e. A esta ranura conectaremos un chip inalámbrico con mejores características y que se explica con más detalle en el punto 3. A este *router* se le acoplará una antena LWA (*Leaky Wave Antenna*) que se explica en el punto 2.3.

## 2.3 Leaky-Wave Antenna (LWA)

Las antenas direccionales son costosas y voluminosas ya que, para fabricarlas, se necesitan mecanismos como agrupaciones de antenas o reflectores. A demás, para alimentarlas, se necesita de más cableado y/o de redes de microondas para formar los haces escaneados directivos (BFN, *bean forming network*), lo que supone mayor riesgo de rotura.

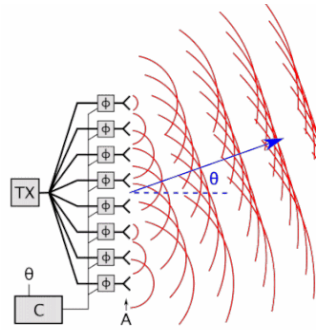



*Figura 2.3: Phased Array Antenna*

*Figura 2.4: Antena parabólica con reflector*

Por otro lado, las antenas direccionales no ven fundamentada su funcionamiento si no somos capaces de controlar hacia dónde se propaga el haz, por lo que es necesario un mecanismo de enfoque y barrido del haz enfocado como las *Phased Array Antenna* (Figura 2.3), que consisten en un conjunto de antenas alimentadas con una fase distinta para cada una de ellas. El resultado es un haz formado por la combinación de las distintas antenas que varía su dirección en función de la configuración de las fases. Otros mecanismos de enfoque son electromecánicos o con reflectores parabólicos (Figura 2.4). Esto hace que los mecanismos de enfoque sean muy costosos, voluminosos y susceptibles a fallos de funcionamiento y roturas.

Las antenas LWA introducen mejoras en el sector de las antenas direccionales. Una de estas mejoras es la facilidad para iluminar grandes aperturas mediante una alimentación única e integrada en la propia antena como se muestra en la figura 2.5. Esto permite reducir considerablemente el tamaño de la antena y eliminar los dispositivos de enfoque necesarios en otro tipo de antenas direccionales.

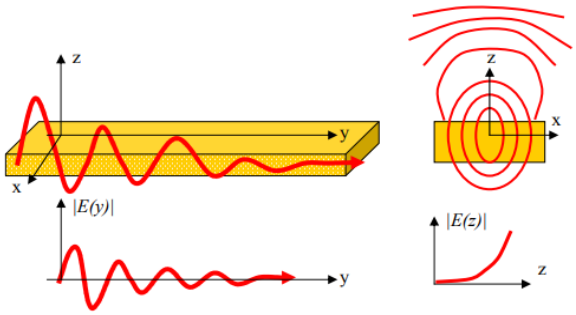



*Figura 2.5: Ondas de Fuga Propagándose y Radiando en una Guía Dieléctrica.*

Las antenas Leaky-Wave Antenna, también conocidas como antenas de fuga son antenas de onda viajera, caracterizadas por la propagación de la onda en una superficie larga en comparación con la longitud de onda. Durante la propagación de la onda sobre la superficie de la antena, se produce el efecto de fuga, el cual se basa en que la mayor parte de las ondas de radio escapan de la antena y se transmiten por el aire. Por ello podemos ver que la intensidad de la onda se reduce conforme se avanza en el plano y.

Estas antenas actúan como un prisma de Newton[1] siendo sensibles a la frecuencia que las recorre, y haremos uso de esta característica para que, en función de la frecuencia de la señal, el ángulo de proyección de la antena sea distinto.

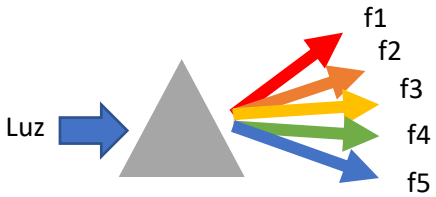



*Figura 2.6: Prisma de Newton*

La primera antena LWA fue propuesta por Hansen Woodyard en 1940, y fueron cuestionadas debido a que parecían contradecir la propia física al tener una constante de propagación transversal compleja del tipo:

[1] El prisma de Newton dispersa la luz blanca que lo atraviesa, formada por la unión de todos los colores, en varias direcciones en función de la longitud de onda del color o lo que es lo mismo en función de su frecuencia.

$$k_z = \beta_z + j\alpha_z \ (m^{-1}) \tag{2.3.1}$$

lo que indica un crecimiento exponencial y una propagación de la onda con crecimiento indefinido conforme la onda de fuga se aleja de la antena, radiando hacia el infinito.

Finalmente se hicieron medidas para comprobar la veracidad de la onda creciente (JN Hines 1953)[10] que demostraron que este hecho sólo ocurría en campo cercano de la antena (cono de luz de una LWA), a partir del cual, la intensidad del campo caía con el cuadrado de la distancia, como era de esperar para que se cumpla la conservación de la energía (condición de potencia de Sommerfield).

Hasta ahora, este tipo de antenas LWA se han usado en aplicaciones RADAR [11],[12],[13], antenas multihaz para aplicaciones de 5G en bandas milimétricas [14],[15], así como en redes inalámbricas para IoT (*Internet of Things*). El objetivo de este TFE es combinar la propiedad de las LWA para enfocar los haces en las distintas frecuencias de la banda de 2.4 GHz junto con el *frequency hopping*. Esta combinación puede permitir localizar en el espacio al cliente que está conectado al AP al analizar la potencia recibida de las tramas *beacon* en función del canal en el que se está realizando la medida de potencia.

Existen numerosos diseños de antenas LWA con multitud de diagramas de radiación. Para nuestro proyecto, nos centraremos en el área de las LWA planares, que son relativamente fáciles de fabricar.

## 2.4 Frequency hopping spread spectrum (FHSS)

Esta técnica de modulación en espectro ensanchado consiste en que la señal se emite en un conjunto de frecuencias saltando de una a otra de forma periódica y rápida. Esto permite que sólo los dispositivos que conozcan la secuencia de salto puedan descifrar la comunicación. Por ende, los dispositivos que no están autorizados y que por lo tanto no conocen la secuencia de salto, escucharán una señal ininteligible.

Esta técnica tiene varias ventajas frente a una comunicación en frecuencia fija:

- FHSS es más resistente a las interferencias ya que cambia de frecuencia constantemente haciendo que la frecuencia que se está acoplando a nuestra comunicación interfiera en un corto periodo de tiempo minimizando así la repercusión en la transferencia de la información.
- Robusto frente a intrusos debido a la secuencia aleatoria de salto.
- Debido a la primera ventaja descrita, las técnicas FHSS pueden compartir las bandas de frecuencia con otras transmisiones convencionales con mínima interferencia.

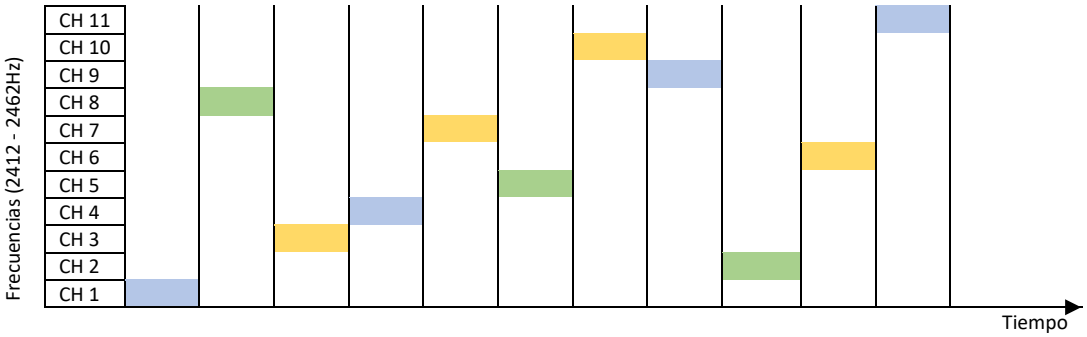



*Figura 2.7. Ejemplo de secuencia de salto de la técnica FHSS.*

En la Figura 2.7 se muestra un ejemplo de secuencia de salto. Esta secuencia se repite de forma periódica y sólo los usuarios autorizados la conocerán pudiendo descifrar la comunicación. En nuestro proyecto, la secuencia de salto no es aleatoria, aumenta linealmente del canal 1 al 11 para una mejor comprensión del funcionamiento. Esto es fácilmente modificable mediante un cambio en el script desarrollado y que se mostrará más adelante.

Una variación de esta técnica es utilizada en Bluetooth y se denomina *Adaptive Frequency-hopping spread spectrum* (AFH). Esto mejora la resistencia a interferencias ya que selecciona la mejor frecuencia de las posibles evitando aquellos canales que sufran interferencias.

Se han propuesto otros sistemas de cambio de frecuencia adaptativo. [16]

## 2.5 RSSI (Received Signal Strength Indicator)

El RSSI es un parámetro utilizado en las comunicaciones WiFi que indica la fuerza de la señal recibida. La escala en la que se mide este parámetro tiene como valor inicial 0 dBm y tiende a valores negativos.

El valor 0 dBm (equivalente a 1mW) es el idílico y es difícil de lograr en la práctica. Nuestra señal variará en potencia desde los 0 dBm hasta aproximadamente -80 dBm, valor que constituye el mínimo aceptable para establecer conexión.

Valores entre -40 y -60 dBm son idóneos para transferencias estables y valores alrededor de -70 dBm suponen una calidad de señal media-baja pudiéndose sufrir problemas en la comunicación por lluvia y viento.

En un principio tomaremos RSSI y dBm como una misma representación de la potencia recibida. Sin embargo, no son exactamente lo mismo, pues el RSSI es un término utilizado para medir la calidad relativa que recibe un cliente, pero no tiene un valor absoluto. En el estándar 802.11 se especifica que este parámetro puede variar entre 0 y 255 aunque cada fabricante especifica su propio rango de valores. Atheros, por su parte, usa una escala entre 0 y 60 mientras que Cisco utiliza otra escala entre 0 y 100.

A priori, este parámetro será el que utilizaremos para medir la calidad del enlace, aunque intervienen más parámetros que son interesantes para entender a fondo el funcionamiento del sistema como la relación señal a ruido (SNR o Signal-to-Noise Ratio), Modulation Coding Scheme (MCS) o la tasa de transmisión y recepción de datos.

En cuanto al ancho del canal, para efectos experimentales, conviene reducirlo al mínimo valor posible, ya que nos va a permitir una mayor limpieza de los datos recogidos y una mayor exactitud en las mediciones.

Existen factores que pueden alterar la señal WiFi:

- Obstrucciones físicas: paredes u objetos que en función del material con el que estén construidos pueden afectar en mayor o en menor medida a la señal.
- Interferencias de otras redes WiFi. Al trabajar en el rango de frecuencias alrededor de 2.45 Ghz, otras redes inalámbricas pueden provocar la caída en la calidad de nuestra señal.
- Aparatos electrónicos que, aunque no estén conectados a la red WiFi, por su propia naturaleza pueden distorsionar la señal. Algunos ejemplos de esos aparatos son el microondas o un monitor para bebés.

Los dos últimos puntos se ven minimizados gracias a la utilización de la técnica FHSS.

# Capítulo Tercero

## Equipamiento necesario

Este capítulo sirve para introducir los elementos que intervienen en el proyecto desarrollado. Estos elementos son los siguientes: una Raspberry Pi 3, una placa integrada WPJ531 7A03 junto con un chipset inalámbrico de Atheros, una antena LWA y un ordenador portátil para controlar los elementos y visualizar los resultados.

### 3.1 Placa integrada inalámbrica WPJ531 7A03

Para ejercer la función de *router*, utilizamos una placa integrada WPJ531 7A03 que se muestra, junto a sus componentes, en la Figura 3.1. A esta placa se le ha instalado una versión del firmware OpenWRT, el cual, al tratarse de un S.O. Linux, nos permite tanto configurar los parámetros necesarios de las interfaces WiFi, como crear scripts (archivos que contienen tareas a realizar automáticamente) para ejecutar la configuración del equipo y realizar el proceso de toma de datos y envío de los mismos a otros equipos.

Las características de la placa se muestran en la tabla 3.1.

| | |
|---|---|
| **Chipset** | CPU: Qualcomm Atheros QCA9531 650MHz |
| **System Memory** | 128MB DDR2 |
| **NOR** | Flash 16MB |
| **Wireless** | Built in 2.4GHz 802.11b/g/n at 26dBm (aggregate)<br>2x U.FL connectors |
| **Expansion** | 1x 9.2mm height Mini PCI-e slot |
| **Interface** | 2x Fast Ethernet Port (Auto MDI-X)<br>1x JTAG 14 Pin Connector1<br>1x Serial Port 4 Pin Connector2 |
| **Power Consumption** | 6.9 Watt (Max) |
| **Supported Operating System** | CompexWRT or OpenWRT/LEDE3 |

*Tabla 3.1: Especificaciones de la placa integrada WPJ531 7A03*

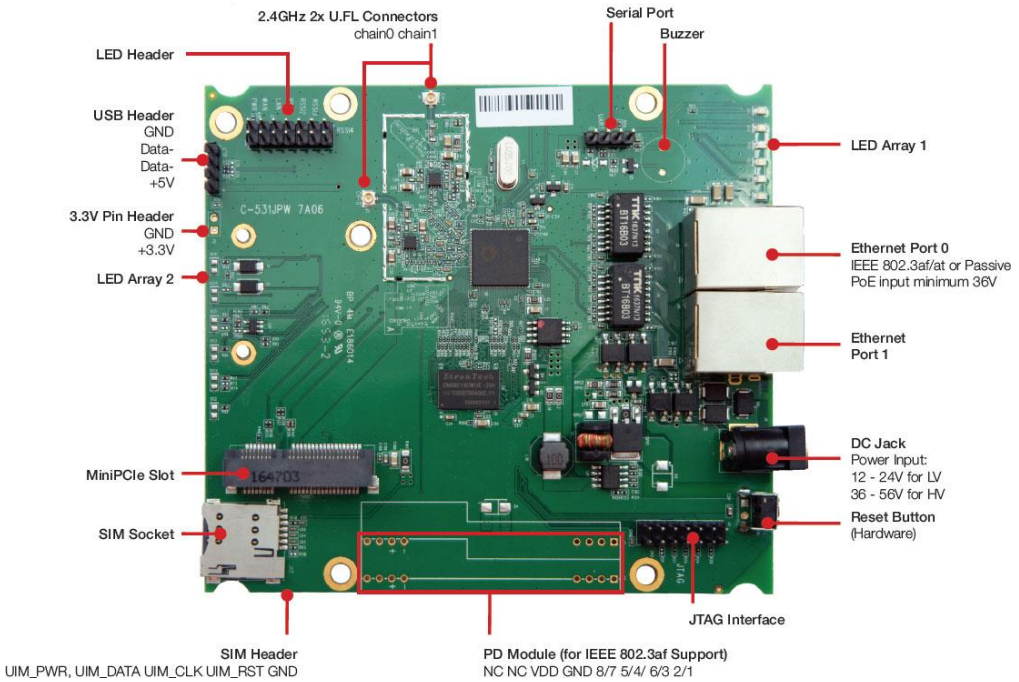



*Figura 3.1: Placa integrada WPJ531 7A03*

## 3.2 Chipset inalámbrico Atheros AR9380

Si leemos las especificaciones de la placa WPJ531, podemos observar la interfaz WiFi que tiene empotrada no es compatible con la banda de 5GHz. Por esta razón se añadió en la interfaz miniPCIe de la placa el chipset Atheros AR9380, que es compatible tanto con la banda de 2.4 GHz como con la de 5 GHz, además de disponer de MiMo 3x3 siendo los conectores para las antenas U-FL. Este chipset se muestra en la Figura 3.2.


*Figura 3.2: Chipset Atheros AR9380.*

Las especificaciones del chip vienen detalladas en la tabla 3.2.

| AR9380 Specifications | |
|---|---|
| **Frequency band** | 2.4/5 GHz |
| **Network Standard** | 802.11a, 802.11b, 802.11g, 802.11n |
| **Modulation Technology** | OFDM with BPSK, QPSK, 16 QAM |
| | 64 QAM; DBPSK, DQPSK, CCK |
| **FEC Coding** | Convolution Code |
| | Low-Density Parity Check (LDPC) |
| **Hardware Encryption** | AES, TKIP, WEP |
| **Quality of Service** | 802.11e |
| **Host Interface** | PCI-Express |
| **Peripheral Interface** | GPIO |
| **Support Data Rates** | IEEE 802.11a 6 -54 Mbps |
| | IEEE 802.11b 1 -11 Mbps |
| | IEEE 802.11g 6 -54 Mbps |
| | IEEE 802.11n 5 -450 Mbps |

*Tabla 3.2: Especificaciones Chipset Atheros AR9380*

Finalmente, uniendo la placa WPJ531 y el chipset AR9380, obtenemos el elemento que actuará de *router* WiFi y que se muestra en la Figura 3.3 a falta de la antena que se introduce en el siguiente punto.

*Figura 3.3: Router inalámbrico*

## 3.3 Antena LWA

Se trata de una Leaky Wave Antenna en tecnología SIW (Substrate Integrated Waveguide) propuesta y desarrollada [17], [18] en la UPCT, y que más recientemente se han aplicado para sistemas radio de localización tanto de sistemas analógicos [19], [20], como usando protocolos comerciales como Zigbee [21], [22], o Bluetooth [23], [24], [25]. Si bien el grupo de la UPCT ha realizado trabajos previos sobre sistemas de radiolocalización usando redes de área local inalámbricas (WLAN) de tipo WiFi [26], [27], este es el primer trabajo en el que se usan las LWAs junto con puntos de acceso (AP, Access Points) para localización de terminales WiFi.

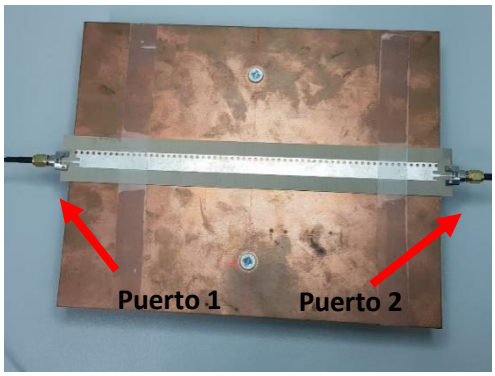



*Figura 3.4: Antena LWA*

La antena presenta una placa de cobre que reduce la emisión hacia atrás y contiene dos tornillos de sujeción muy útiles para unir la antena al sistema de medida de una cámara anecoica pudiendo realizar su caracterización espectral fácilmente. Esta antena contiene dos puertos que se conectan al chipset inalámbrico AR9380 mediante dos de sus tres conectores para antena. Los cables que permiten esta conexión son cables coaxiales SMA macho a SMA macho y un cable coaxial SMA hembra a MHF4 para conectar el anterior cable con el chipset.

*Figura 3.5: Cable coaxial SMA hembra a MHF4*

*Figura 3.6: Cable coaxial SMA Macho a SMA Macho*

Puesto que la placa junto con el chipset Atheros irán embebidos en una carcasa plástica, el cable coaxial SMA hembra a MHF4 sirve de puerta de entrada entre el cable coaxial SMA a SMA que conecta con la antena y la carcasa. De esta forma, el cable mostrado en la Figura 3.5 irá conectado con el chipset Atheros a través del conector MHF4 mientras que el conector SMA irá incrustado en la carcasa externa destinada para aislar los componentes. Por otro lado, el cable de la Figura 3.6 unirá el cable anterior con la antena. Esta conexión se muestra en la figura 3.7.

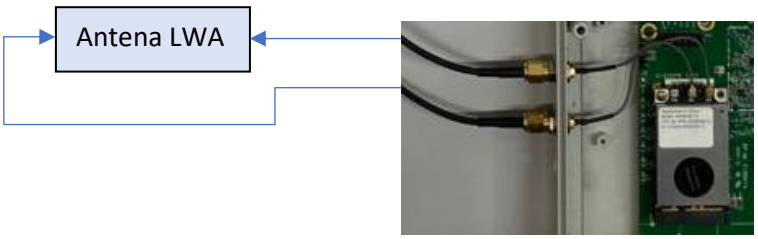



*Figura 3.7: Conexión con la antena*

Como se puede observar en la Figura 3.4, la antena consta de 2 puertos. La existencia de 2 puertos en la antena viene justificada por la necesidad de cubrir la mayor área de cobertura posible.

Debido a la naturaleza de la antena LWA explicada en el punto 2.3, al suministrar la señal por uno de los puertos, la radiación emitida tiene el aspecto mostrado en la Figura 3.8. Por el funcionamiento de la antena, la frecuencia f5 será la de mayor frecuencia y f1 la de menor.

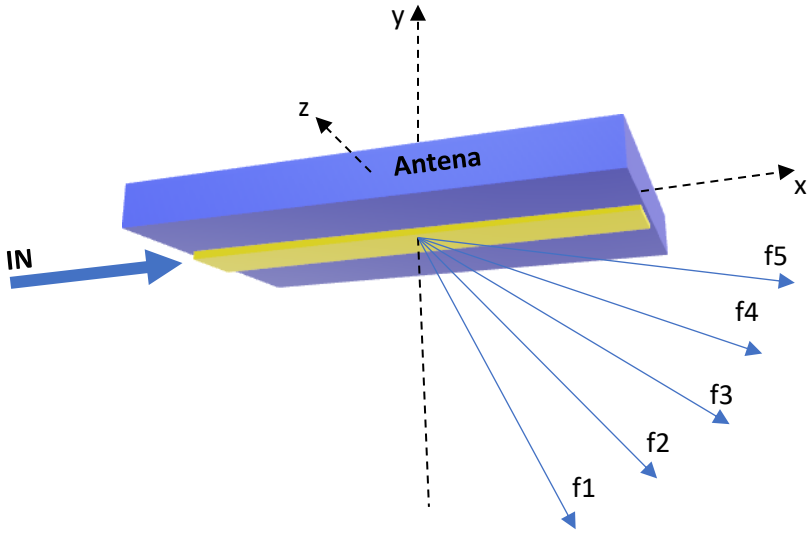



*Figura 3.8: Dirección de radiación al introducir la señal por uno de los puertos.*

Es decir, si introducimos la señal por el puerto de la izquierda, la señal WiFi será proyectada hacia el lugar del espacio de la mitad derecha de la antena. Al ser la antena simétrica, si introducimos la señal por el puerto contrario, la señal wifi se proyectará en la dirección del espacio contraria a la resultante de introducirla por el primer puerto. La Figura 3.9 muestra de forma esquemática este fenómeno.

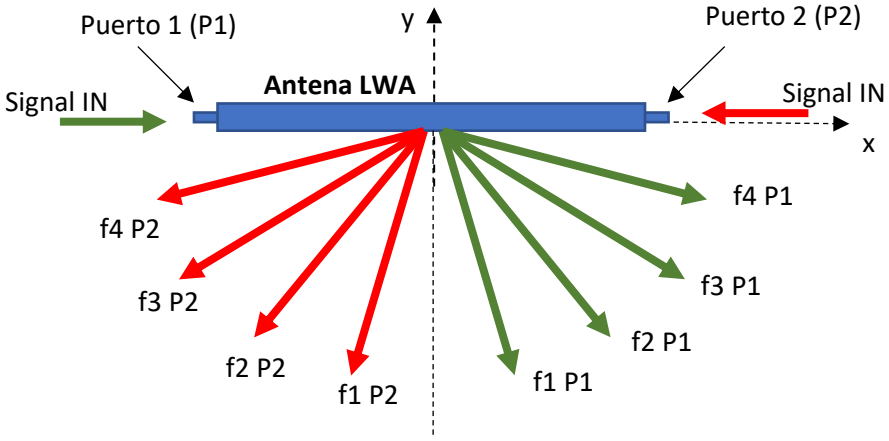



*Figura 3.9: Radiación de Antena LWA al suministrar señal por ambos puertos*

### 3.3.1 Geometría de la antena

A la hora de diseñar una antena es muy importante hacer un cálculo preciso de las diferentes dimensiones de la misma entre otras características. En nuestro proyecto, la antena utilizada, fue diseñada para trabajar en las frecuencias WiFi, por lo que todas las medidas están ajustadas para un funcionamiento óptimo.

La longitud de la antena viene dada por la longitud de onda que se calcula como:

$$\lambda = \frac{v}{f} = v \cdot T \qquad (3.1)$$

Tomando la velocidad como la velocidad de la luz (c = 3e8 m/s) y sabiendo que estamos trabajando en la banda de 2.4 GHz, podemos obtener la longitud de onda del WiFi despejando de 3.1.

$$\lambda = \frac{v}{f} = \frac{3e8}{2.4e9} = 0.125\,m = 12.5\,cm \qquad (3.2)$$

Por lo tanto, la longitud de onda del WiFi es de 12.5cm.

Teniendo en cuenta esta longitud de onda, junto con las propiedades del substrato, las dimensiones de la antena se han de obtener de forma que las propiedades de esta sean las mejores posibles.

Dentro de las antenas LWA, la antena diseñada mediante tecnología planar entra dentro de las denominadas Half-Width Leaky Wave Antenna (HW-LWA) y consiste en una línea microstrip sobre un dieléctrico. La principal propiedad de este grupo de LWA es la existencia de un único borde radiante y una fila de postes PEC (Perfect Electric Conductor). Estos postes permiten que cuando la onda incidente se propague por la línea microstrip, al impactar con ellos, se refleje hacia el borde en su totalidad, es decir, sin que se produzcan pérdidas. Esto permite que la totalidad de la potencia sea radiada.

A la combinación de microstrip y dieléctrico se le suma un plano de masa que reduce considerablemente la radiación hacia direcciones posteriores.

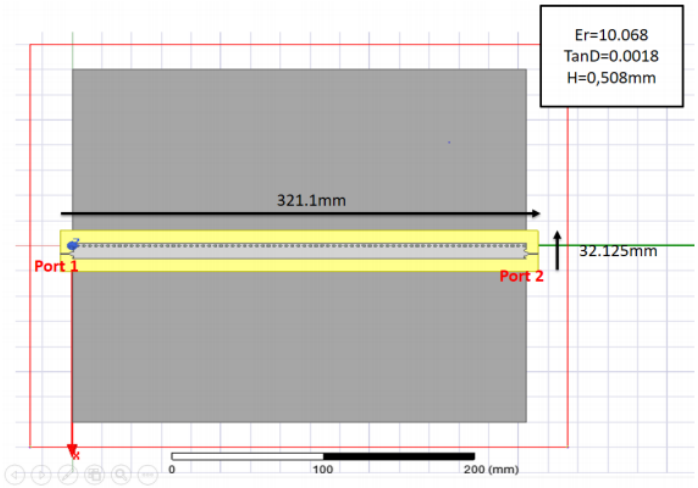



*Figura 3.10: Dimensiones de la antena LWA (a)*

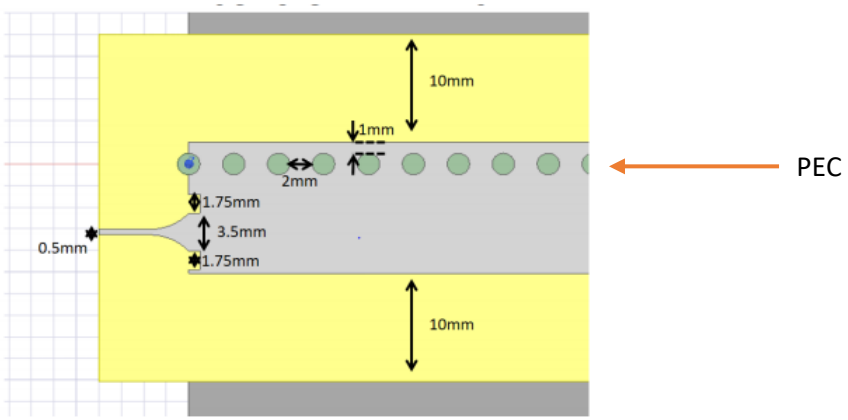



*Figura 3.11: Dimensiones de la antena LWA (b)*

El borde inferior de la figura 3.11, es el responsable del efecto de fuga propio de las antenas LWA. La onda electromagnética, al alcanzar este borde, no encontrará ningún obstáculo que le permita continuar con su propagación al haber un cambio en la permitividad relativa $\varepsilon_r$. Este comportamiento se muestra en la Figura 3.12.

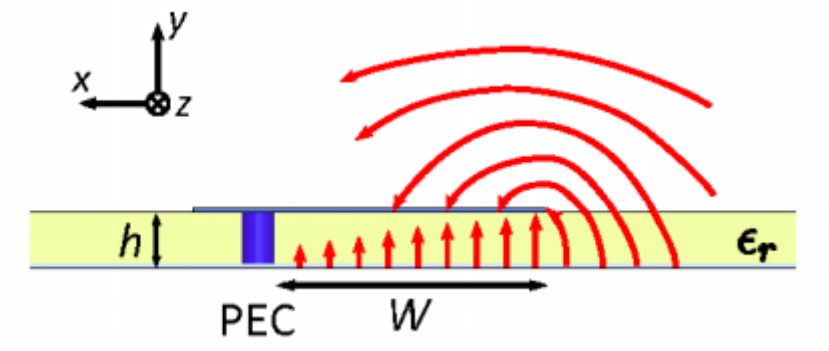



*Figura 3.12: Efecto de Fuga de la antena HW-LWA*

El dieléctrico escogido para la impresión de la antena en tecnología microstrip, es el AD1000. Este substrato comercial posee las propiedades mostradas en la Tabla 3.3

| $\varepsilon_r$ | $\tan_d$ | $H_s$ubs |
|---|---|---|
| 10.068 | 0.0018 | 0.508mm |

*Tabla 3.3: Propiedades del substrato AD1000*

La combinación de los materiales escogidos junto con las dimensiones establecidas permite la obtención de unos parámetros S aceptables, pudiéndose mejorar con un substrato con mejores propiedades. Estos parámetros S, también conocidos como parámetros de dispersión, determinan el buen funcionamiento de la antena.

Por un lado, el parámetro S11 describe la adaptación de la antena y al igual que el resto de los parámetros, se ha obtenido mediante el software ANSYS HFSS en el informe emitido por Alejandro Rafael Gil Martínez en el que se detallan las medidas y parámetros de diseño de la antena LWA. Este parámetro viene representado en la Figura 3.13.

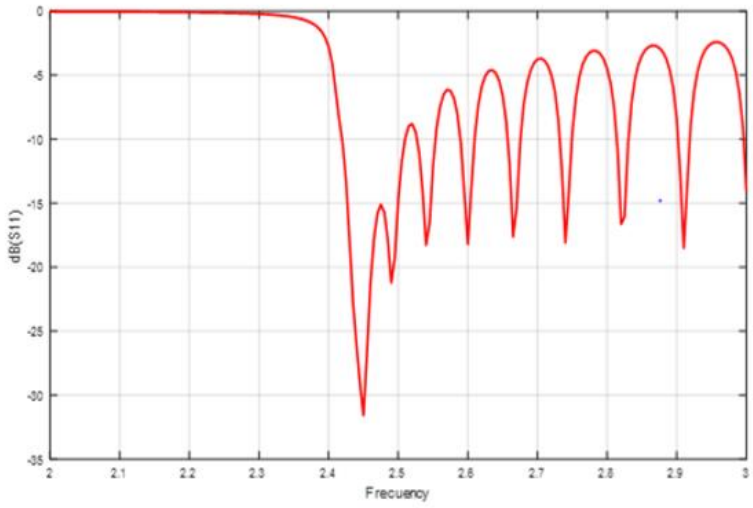



*Figura 3.13: Parámetro S11*

Como podemos observar en la Figura 3.13, la adaptación es buena a las frecuencias WiFi, puesto que presenta un mínimo en ese rango de frecuencias (2.45 GHz).

Por otro lado, al representar el parámetro S21, que determina la transferencia de potencia del puerto 1 al 2, podemos observar en la Figura 3.14 como, a las frecuencias a partir de 2.45GHz, la potencia es aproximadamente 0 dB. Esto indica que respecto a la potencia introducida en el puerto 1, muy poca pasa al puerto 2.

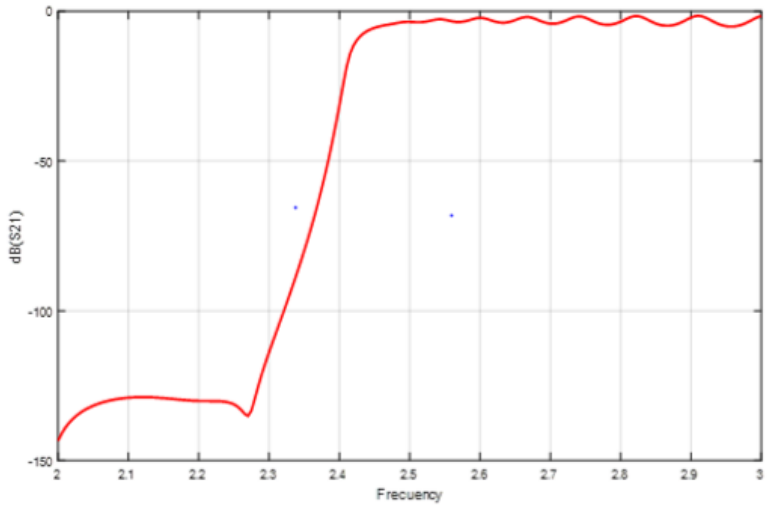



*Figura 3.14: Parámetro S21*

Puesto que la antena que vamos a utilizar en este proyecto será únicamente para la banda de 2.4 GHz de WiFi, conviene centrarnos en esas frecuencias. En la Figura 3.15 se muestran los parámetros S11 y S21 para 13 de los 14 canales existentes. El canal 14 no se muestra ya que es una banda para uso exclusivo militar.

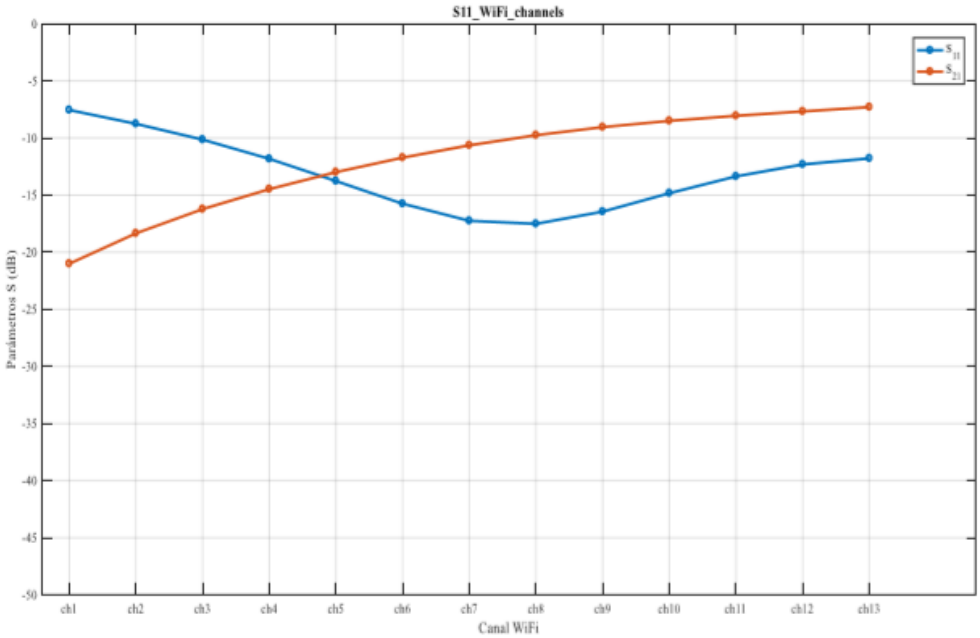



*Figura 3.15: Parámetros S11 y S21 para los canales 1 al 13*

Como se puede apreciar, tenemos una mejor adaptación de la antena en las frecuencias intermedias puesto que el parámetro S11 es menor en esos canales. No obstante, la adaptación es buena en todos los canales al estar por debajo de -10dB.

Por otro lado, el parámetro S21 indica que la transferencia de potencia del puerto 1 al 2 es menor en los canales de menor frecuencia. Esto quiere decir que, en los canales de menor orden,

hay menos parte de la señal de entrada que se transfiere al puerto 2, lo que parece indicar que hay un mayor efecto de fuga para esos canales.

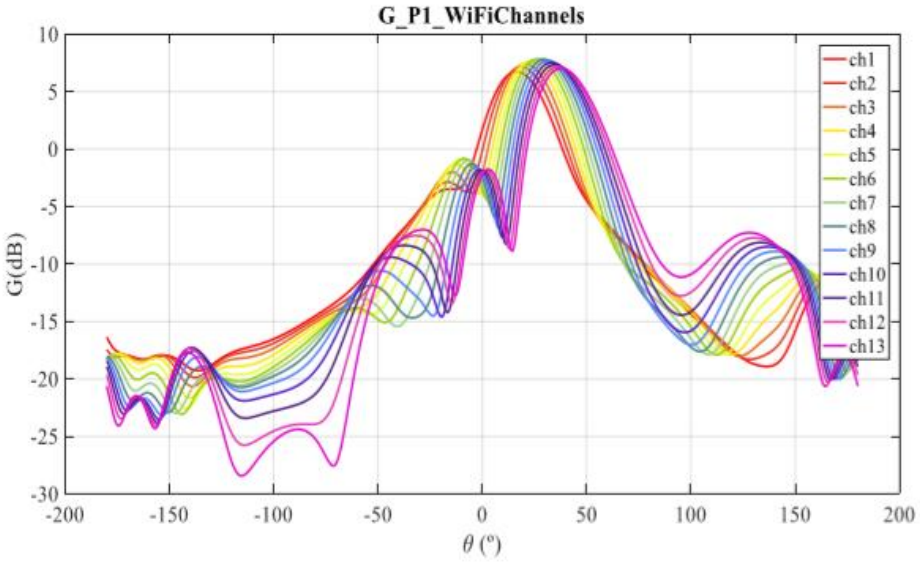



*Figura 3.16: Diagrama de radiación de la LWA en función del ángulo para los 13 canales WiFi al ser alimentada por el puerto 1.*

En la Figura 3.16 se puede apreciar el comportamiento propio de la antena diseñada. El máximo de potencia para cada canal se produce en un ángulo distinto y se aleja de la perpendicular al aumentar la frecuencia. Este efecto también se puede apreciar si representamos únicamente el ángulo en que se alcanza dicho máximo para cada canal. Si realizamos la misma representación, pero esta vez alimentando la antena por el puerto contrario, obtenemos un diagrama de radiación simétrico al mostrado en la Figura 3.16. Este hecho se muestra en la Figura 3.17. Aunque las medidas tomadas son para un substrato con permitividad relativa $\varepsilon_r$ = 10 y 14 canales, nos permiten ver el efecto de cambiar el puerto por el que la antena es alimentada.

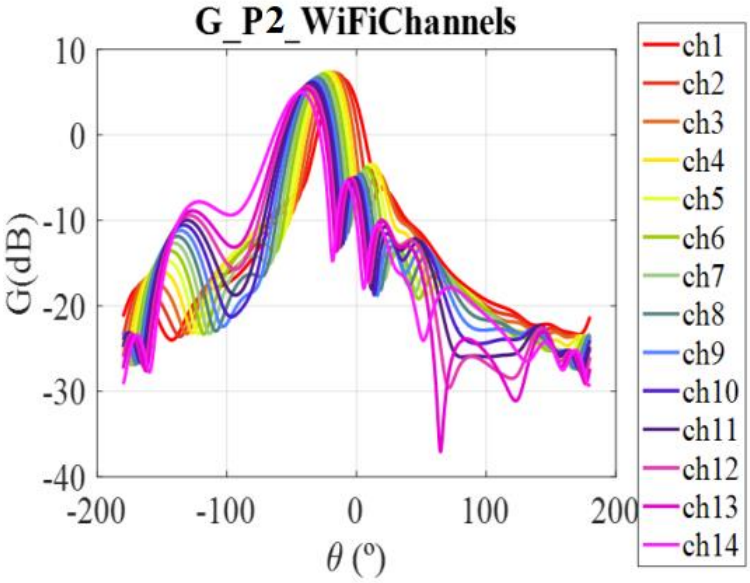



*Figura 3.17: Diagrama de radiación de la LWA para 14 canales WiFi al ser alimentada por el puerto 2.*

En la figura 3.18 se representa el ángulo en el que para cada canal se alcanza el máximo de potencia, es decir, hacia que parte del espacio se enfoca la mayor parte de la energía de la señal, siendo 0º el ángulo que se corresponde con visión directa con el receptor (perpendicular a la antena). Así demostramos la tendencia directamente proporcional entre el ángulo de máxima radiación y el canal de la comunicación, o lo que es lo mismo: la frecuencia. De esta figura también se puede deducir el margen angular de radiación que tiene la antena, que se conoce por Field of View (FoV). Representa los grados de visión que puede alcanzar la antena desde una posición fija variando únicamente el canal de comunicación.

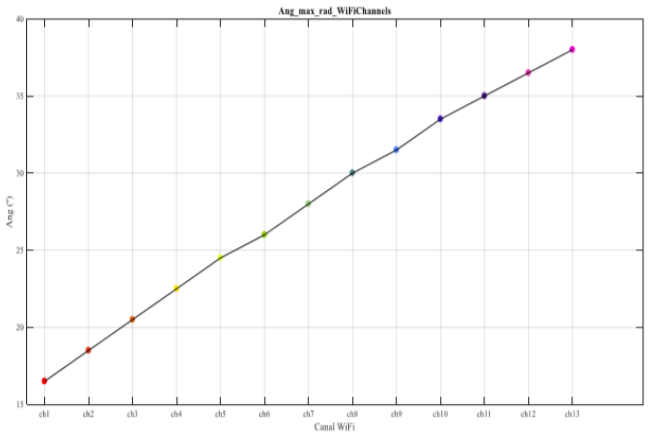



*Figura 3.18: Ángulo de máxima radiación en función del canal*

Si calculamos la diferencia entre el ángulo de radiación para el canal 13 y el correspondiente al canal 1, podemos comprobar que nuestra antena tiene un FoV desde 16º a 38 º al alimentarla por uno de los puertos. Puesto que el comportamiento para el segundo puerto es simétrico, podemos deducir que la antena tiene la capacidad de radiar 44º.

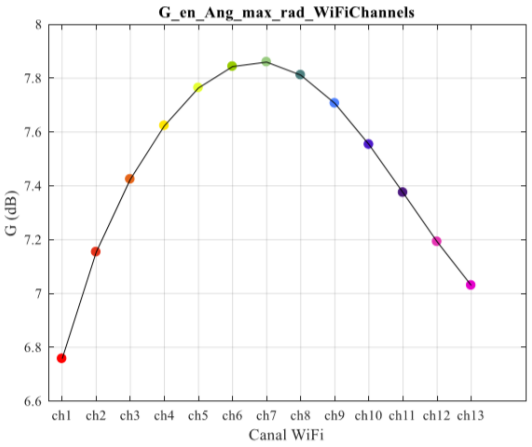



*Figura 3.19: Ganancia de la antena en función del canal*

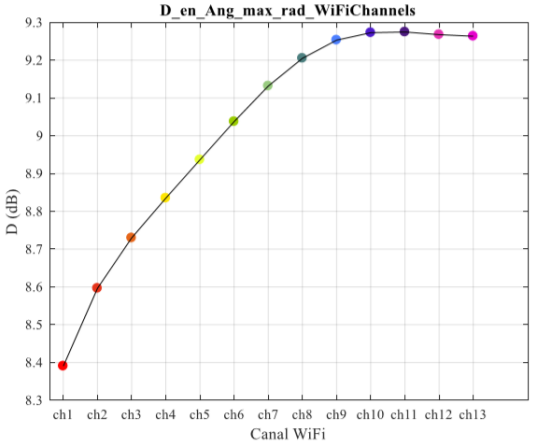



*Figura 3.20: Directividad de la antena en función del canal*

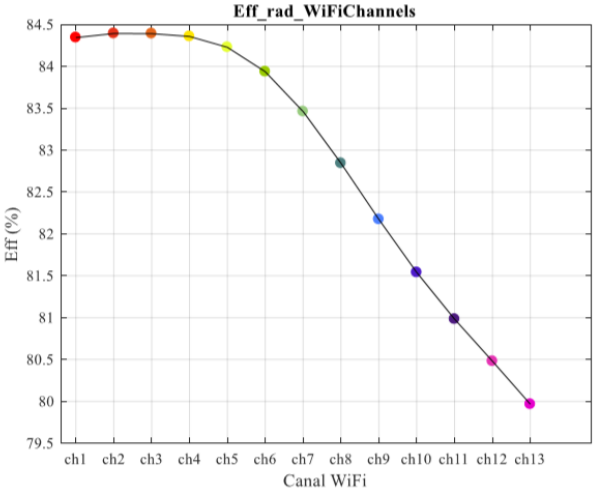



*Figura 3.21: Eficiencia de la antena en función del canal*

Como se puede observar en la Figura 3.19, la ganancia de la antena es mayor en los canales que se corresponden con las frecuencias intermedias.

La directividad representa la capacidad que tiene una antena para enfocar la potencia total irradiada en un único punto del espacio. También se define como la relación entre la potencia radiada por la antena en la dirección de máxima potencia y la intensidad de radiación de una antena omnidireccional o isotrópica que radia con la misma potencia.

Como se aprecia en la Figura 3.20, la directividad es mayor en los canales de mayor frecuencia.

Por otro lado, la eficiencia representa la relación entre la potencia radiada frente a la suministrada a la antena. Esta eficiencia es mayor a canales de menor orden como se puede observar en la Figura 3.21.

Otro parámetro que es conveniente definir es el tipo de polarización de la antena, pues en función de la polarización, las medidas podrían variar significativamente.

La polarización de una antena hace referencia al sentido de las ondas que propaga y puede ser de tipo lineal o circular.

Dentro de la polarización lineal, podemos encontrar polarización horizontal o vertical y una combinación de ambas. Esto lo determina la posición que adopte la antena.

Para que dos antenas se puedan comunicar, deben estar polarizadas de la misma manera. En el caso de que una antena tenga una combinación de las dos polarizaciones lineales como doble polarización o polarización cruzada, puede comunicarse con otras antenas cuya polarización sea exclusivamente horizontal o vertical pero también con combinaciones.

La polarización circular, permite comunicarse con el resto de las polarizaciones.

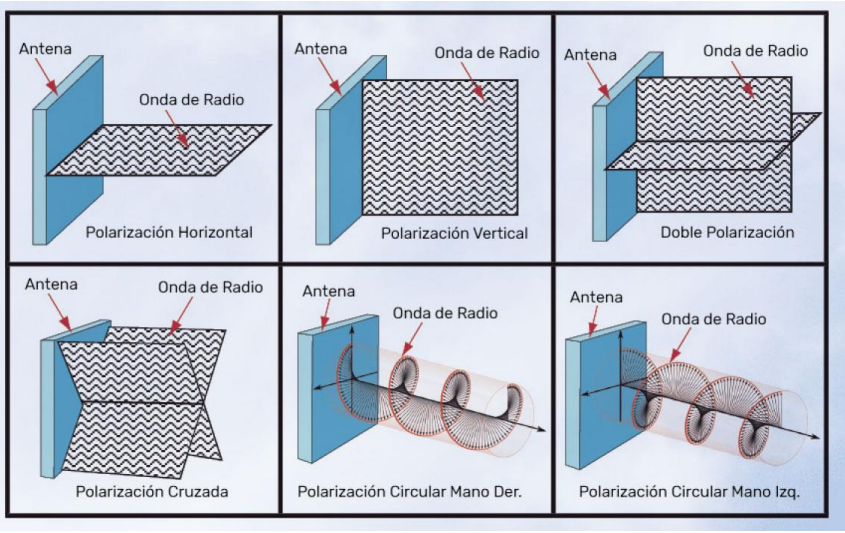



*Figura 3.22: Polarizaciones existentes de antenas*

En el caso de nuestra antena, la polarización es lineal y particularmente será vertical, puesto que la disposición de la antena y la toma de medidas se realizará conforme a la Figura 3.10, es decir, con la antena situada horizontalmente.

## 3.4 Raspberry

El último componente necesario es un equipo que actúe de cliente para así poder medir la potencia que recibimos de él en función del canal en el que se haga la transmisión. Para ello utilizaremos una Raspberry Pi 3 Model B V1.2 junto con un adaptador USB para WiFi. Este

adaptador nos va a permitir controlar la polarización de la onda para así obtener una lectura de potencia más libre de interferencias. A pesar de que este modelo de Raspberry ya cuenta con una interfaz WiFi, no ha sido posible conocer qué tipo de polarización contiene la antena que incorpora, ni su orientación, por lo que se decidió emplear este adaptador USB WiFi compatible con Raspbian, el S.O. que se ha instalado en la Raspberry.

La función de este componente es la de comunicarse de forma continua con el *router* a través de paquetes de datos ICMP, que permiten mantener actualizada la información de la comunicación que el *router* almacena de forma periódica.

*Figura 3.23: Adaptador USB WIFi wn722n*

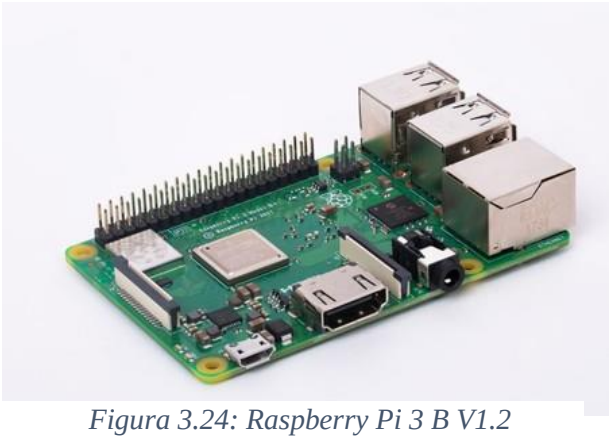


*Figura 3.24: Raspberry Pi 3 B V1.2*

Este componente estará conectado a nuestro AP de forma inalámbrica y enviará de forma continua paquetes ICMP Echo Request (*Internet Control Message Protocol*), es decir, estará mandando continuamente Ping a la dirección IP de nuestro punto de acceso. De esta forma, la información que obtengamos de la potencia medida estará continuamente actualizada.

Las especificaciones de la Raspberry se muestran en la tabla 3.3.

| | |
|---|---|
| **CPU** | Quad Core 1.2GHz Broadcom BCM2837 64bit |
| **RAM memory** | 1GB |
| **Wireless** | BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board |
| **Ethernet** | 100 Base Ethernet |
| **Ports** | 40-pin extended GPIO |
| | 4 USB 2 ports |
| | 4 Pole stereo output and composite video port |
| | Full size HDMI |
| | CSI camera port for connecting a Raspberry Pi camera |
| | DSI display port for connecting a Raspberry Pi touchscreen display |
| | Micro SD port for loading your operating system and storing data |
| | Upgraded switched Micro USB power source up to 2.5A |

*Tabla 3.4: Especificaciones Raspberry Pi 3 B V1.2*

Las especificaciones del adaptador USB WiFi aparecen detalladas en la tabla 3.4.

| **TPLink TL-WN722N High Gain Wireless USB Adapter** | |
|---|---|
| **Interface** | USB 2.0 |
| **Antenna type** | Detachable Omni Directional (RP-SMA) |
| **Antenna Gain** | 4dBi CAPÍTULO 3. EQUIPAMIENTO NECESARIO |
| **Wireless Standards** | IEEE 802.11n, IEEE 802.11g, IEEE 802.11b |
| **Frequency** | 2.400-2.4835GHz |
| **Signal Rate** | 11n: Up to 150Mbps(dynamic)<br>11g: Up to 54Mbps(dynamic)<br>11b: Up to 11Mbps(dynamic) |
| **Reception Sensitivity** | 130M: -68dBm@10% PER<br>108M: -68dBm@10% PER<br>54M: -68dBm@10% PER<br>11M: -85dBm@8% PER<br>6M: -88dBm@10% PER<br>1M: -90dBm@8% PER |
| **Transmision Power** | <20dBm |
| **Wireless Modes** | Ad-Hoc / Infrastructure mode |
| **Wireless Security** | Support 64/128 bit WEP, WPA-PSK/WPA2-PSK |
| **Modulation Technology** | DBPSK, DQPSK, CCK, OFDM, 16-QAM, 64-QAM |

*Tabla 3.5: Especificaciones Adaptador USB WiFi TPLink Archer T2UH*

## 3.5 Ordenador portátil

Este elemento será el encargado de controlar el *router* mostrado en la Figura 3.3, así como tomar las medidas de potencia obtenidas por el *router* y representarlas mediante el software Matlab. También se encargará de controlar la orientación de la mesa posicionadora que se encuentra en el interior de la cámara anecoica.

*Figura 3.25: Ordenador Portátil Lenovo z50-70*

En él instalaremos tres programas. El primero sirve para acceder al *router* vía SSH (Secure Shell), nombre por el que se conoce al protocolo que permite acceder de forma remota a un servidor u otros dispositivos. Este software es Putty. Es un cliente *open source* para el sistema operativo Windows.

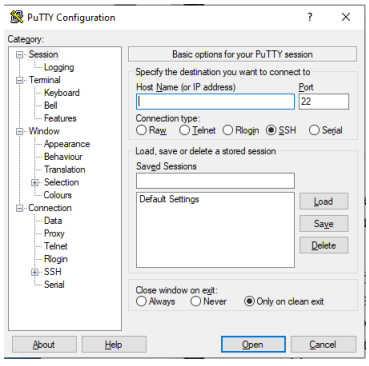



*Figura 3.26: Software Putty*

En la Figura 3.26 se muestra este software. En él debemos especificar la dirección ip de nuestro dispositivo y el puerto para acceder que por defecto para conexiones ssh es el puerto 22.

Por otro lado, necesitaremos el software Matlab. Se trata de un entorno de desarrollo en lenguaje M para cómputo numérico en el que realizaremos los cálculos necesarios para obtener las mediciones de potencia y obtener una representación de la evolución del parámetro RSSI en función del canal de comunicación.

Por último, necesitamos un intérprete de Linux para recibir los archivos que contienen el conjunto de muestras para un ángulo dado junto a los parámetros de la comunicación asociados a ese valor de potencia. Este intérprete lo obtenemos de la tienda de Microsoft que encontramos en Windows. En nuestro caso hemos descargado Ubuntu 18.04 LTS aunque podemos utilizar cualquier otro.

La transmisión de estos ficheros la realizamos a través de *socat* que deberemos instalar tanto en nuestro intérprete de Ubuntu como en el *router* y crearemos un script en cada uno para que cuando finalice de tomar las muestras para un ángulo, envíe por TCP el archivo al ordenador.

Este script es muy sencillo y se muestra en la figura 3.27.

```
#!/bin/bash
max=10
for i in `seq 1 $max`
do
        cat socat_Recv.sh | socat TCP:192.168.55.137:9999 -
        echo "enviado archivo # $i con tamaño "
        sleep 5
done
```

(a)

```
#!/bin/bash
while [ 1 ]
do
        echo "antes de socat"
        Fecha=$(date +%Y_%m_%d_%H_%M_%S)
        socat -u TCP-LISTEN:9999 OPEN:Fecha_$Fecha.dat,create,append,end-close
        echo "Recibido archivo. Guardado como  Fecha_$Fecha"
        sleep 1
done
```

(b)

*Figura 3.27: Script utilizado para enviar (a) y para recibir (b) los archivos de potencia*

## 3.6 Entorno de trabajo

Una vez presentado el hardware que necesitaremos para el desarrollo del proyecto, en la Figura 3.28 se muestra el conjunto de elementos y cómo se conectan entre sí. De esta forma, podemos apreciar de un solo vistazo el entorno con el que trabajaremos.

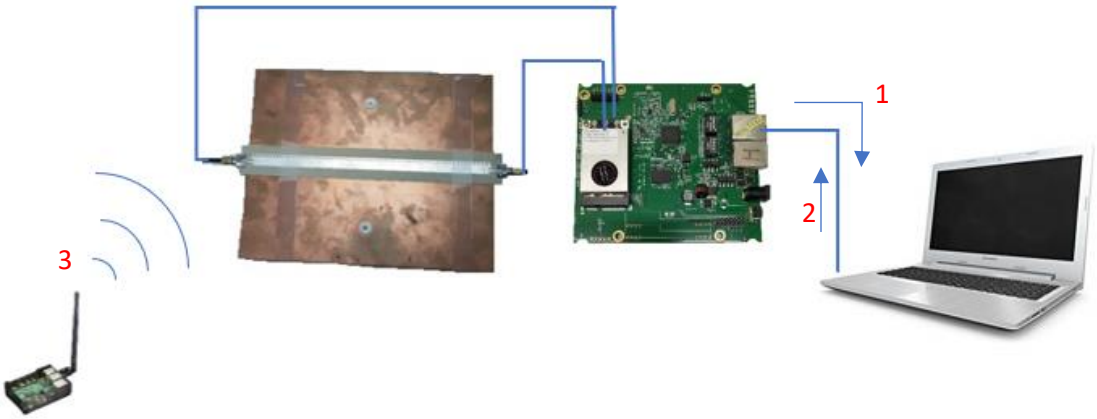



*Figura 3.28: Hardware presente en el sistema*

El punto 1 identifica la transmisión del paquete UDP que contiene una muestra del valor de potencia en un instante junto con la transmisión a través de *socat* del archivo que contiene todos los valores de potencia para los 11 canales en un ángulo específico.

El punto 2 identifica el control del funcionamiento que hace el ordenador del *router* a través de ssh, mediante el cual somos capaces de controla la toma de muestras y la transmisión de los paquetes del *router* al ordenador.

Por último, el punto 3 hace referencia al envío de paquetes ICMP por parte de la Raspberry hacia la antena de forma continua y gracias al que mantendremos la información de la comunicación actualizada a la hora de tomar una muestra. De esta forma cada muestra será única y no tendremos dos valores de potencia que por equivocación interpretemos como dos instantes de tiempo distintos.

# Capítulo Cuarto

## Preparación de los componentes

Este capítulo recoge las configuraciones realizadas para permitir una comunicación entre un AP y un cliente usando la técnica de salto en frecuencia o *frequency hopping*.

### 4.1 Configuración del AP

Como se ha comentado en el punto 3.1, la placa integrada soporta el firmware OpenWrt [28]. Este firmware está basado en Linux y permite modificar todos los ficheros que controlan el funcionamiento del *router* y por lo tanto una completa personalización del dispositivo.

El primer paso será crear una red wifi, que llamaremos OpenWrt. A esta red se conectará nuestra Raspberry y el resto de los clientes, aunque para nuestro experimento únicamente tendremos conectado uno. Para ello, debemos modificar el archivo Wireless [29] que encontramos en el directorio */etc/config*. Este archivo contiene información sobre las interfaces WiFi disponibles. Como se puede apreciar en la Figura 4.1, radio0 y radio1 representan las dos interfaces disponibles en la placa. La primera, la cual se encuentra deshabilitada mediante la opción *disabled = 1*, se corresponde con la interfaz WiFi propia de la placa integrada WPJ531 7A03. La segunda interfaz hace referencia al chipset inalámbrico Atheros AR9380.

Comentando o borrando la línea que hace referencia a la desactivación de la interfaz, la activamos.

Los parámetros que configuran esta interfaz son los siguientes:

- Type: depende del chipset que utilicemos y viene configurado de forma automática al inicio gracias a la autodetección.
- Channel: especifica el canal WiFi en el que se encuentra la red por defecto.
- Hwmode: selecciona el protocolo WiFi.
- Htmode: Ancho de banda a usar
- Mode: Modo AP o monitor
- SSID: nombre de la red que verán los clientes

```
config wifi-device  radio0
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     'platform/qca953x_wmac'
        option htmode   HT20
        # REMOVE THIS LINE TO ENABLE WIFI:
        option disabled '1'

config wifi-iface
        option device   radio0
        option network  lan
        option mode     ap
        option ssid     OpenWrt
        option encryption none

config wifi-device  radio1
        option type     mac80211
        option channel  11
        option hwmode   11g
        option path     'pci0000:00/0000:00:00.0'
        option htmode   HT20

# REMOVE THIS LINE TO ENABLE WIFI:
        #option disabled 1

config wifi-iface
        option device   radio1
        option network  'wifi'
        option mode     ap
        option ssid     OpenWrt
```

*Figura 4.1: Archivo de configuración wireless*

También debemos asignar una ip a la interfaz wlan1 y establecer la ruta que permita comunicación entre los dispositivos de la red. Para ello, mediante los siguientes comandos, asignamos la ip a wlan1 y la añadimos como puerta de enlace para los componentes.

*ifconfig wlan1 192.168.2.1*

*route add default gw 192.168.2.1*

Para aplicar las modificaciones realizadas al archivo *wireless*, OpenWRT proporciona el comando *wifi*, comando que permite recargar la configuración especificada por este archivo. También es válido mediante un *reboot*, ya que OpenWrt carga este archivo al inicio. De esta forma, se pensó en crear un script[2] que hiciera uso de esta instrucción

[2] Script: documento que contiene instrucciones que permiten ejecutar ciertas funciones en un ordenador.

para especificar el canal wifi ya que en este archivo se puede especificar el canal de la comunicación. El problema reside en que, al ejecutar el comando *wifi*, la interfaz inalámbrica se reinicia, provocando la caída momentánea del punto de acceso creado y por lo tanto la pérdida de la conexión por parte de los clientes que estuvieran vinculados a él. El siguiente paso que realiza el cliente cuando pierde el punto de acceso al que estaba conectado es escanear en todos los canales para encontrar el mejor AP con el que establecer conexión, lo cual supone una pérdida de tiempo. De esta forma, el comando *wifi* quedó descartado en un principio como opción para cambiar de canal de forma secuencial.

Tras estudiar con detenimiento el protocolo 802.11, se descubrió que el propio protocolo define un método por el que el AP anuncia el canal al que va a cambiar a continuación (Protocolo 802.11 sección 10.9.8)[30].

Para la utilización de este mecanismo, OpenWrt utiliza un software denominado Hostapd (*Host Access Point Daemon*) que permite la creación de puntos de acceso y la ejecución de comandos para controlar diversas características de este.

En particular, para realizar un cambio en la frecuencia del canal, se utiliza el comando *chan_switch*, el cuál necesita 2 parámetros. Estos parámetros son el número de tramas beacon[3] tras las cuales el AP cambiará de canal y el canal al que cambiará (especificado en Hz). Mediante este comando, el AP lanza un mensaje a todos los dispositivos asociados con él especificando el canal al que cambiará una vez terminen de enviarse las tramas beacon especificadas. De esta forma, antes de cambiar el canal wifi, los clientes ya conocen este cambio y por lo tanto pueden adaptarse antes.

Como ejemplo de uso de este comando, se propone un cambio al canal 6 tras la transmisión de 5 tramas beacon:

*hostapd_cli chan_switch 5 2437*

Este script se ha desarrollado en bash para permitir su ejecución dentro del AP.

## 4.2 Configuración de la Raspberry Pi

Para medir la potencia recibida de forma digital, necesitamos algún elemento que envíe de forma continua pequeños paquetes para que el AP pueda analizarlos y determinar la potencia con la que los recibe. De esta forma, la Raspberry es un elemento muy apropiado para esta labor.

Se trata de un pequeño ordenador de bajo coste que es totalmente configurable en cuanto a su función y a los periféricos que debe gestionar. Para nuestro proyecto, su única función será hacer de cliente para nuestro AP y enviarle información de manera periódica.

El modelo de nuestra Raspberry es Pi 3 Model B+ V1.2. Aunque en este modelo viene incorporado una interfaz wifi, utilizaremos un USB Wifi que contiene una antena con la que podemos controlar la polarización de las ondas. De esta forma obtendremos una mayor limpieza en las medidas.

EL adaptador wifi USB se detecta de forma automática y la Raspberry lo reconoce como una tarjeta de red inalámbrica. Únicamente debemos especificar la dirección IP que tendrá esta interfaz y especificarle la red WiFi a la que irá conectada. Al igual que en la configuración del AP, utilizamos el comando *ifconfig* para especificar la IP de dicha interfaz que deberá estar en el mismo rango de IP que el AP. Una vez establecida la IP, debemos darle los datos de la WLAN.

[3] Trama Beacon: contiene toda la información relacionada con una red WLAN y es transmitida de forma periódica para anunciar la presencia de esta.

Esto lo hacemos a través del archivo de configuración que encontramos en la ruta */etc/wpa_supplicant/wpa_supplicant.conf*. Dentro de este archivo, buscamos la línea network y la modificamos de la siguiente forma:

```
network={
 ssid="OpenWRT"
 key_mgmt=NONE
}
```

Con esto, la Raspberry se conecta automáticamente a nuestra WLAN al reiniciarse.

Otra configuración que no es en sí una configuración, sino más bien una función, consiste en conseguir transmitir de forma continua tramas al AP.

La medida de RSSI que realiza el *router* se basa en un histórico que guarda el estado de la última comunicación entre el AP y el cliente. Por ello, debemos actualizar este dato con al menos el mismo periodo de medida que el script desarrollado en bash, que es de aproximadamente 0.3 segundos. Para asegurarnos de tener las medidas actualizadas, la Raspberry mandará de forma continua cada 0.1 segundos un paquete al AP. Esto lo hacemos mediante la instrucción siguiente:

```
arping -w 100000 192.168.2.1
```

La opción -w es utilizada para especificar el periodo de envío de tramas en microsegundos. Y a continuación especificamos la IP del AP.

En este capítulo se han recogido las configuraciones básicas para permitir la comunicación entre el AP y el cliente. En el siguiente capítulo entraremos en profundidad en los elementos usados para el desarrollo experimental del proyecto. Nos detendremos en cada uno para ver el funcionamiento de cada uno de los elementos y las interacciones entre todos ellos.

# Capítulo Quinto

# Desarrollo del proyecto

## 5.1. Comprobación del funcionamiento del AP

Antes de caracterizar la antena mediante el esquema de FHSS, es conveniente comprobar el buen funcionamiento de la placa WPJ531 y el chipset AR9380. Para ello, se propone un simple experimento en el que simularemos unas condiciones ideales en la comunicación entre la Raspberry y el AP que consistirá en unir de forma directa ambos componentes aunque seguiremos utilizando el protocolo de comunicación WiFi. Estas condiciones ideales consisten en eliminar las posibles interferencias que pueden darse cuando el intercambio de información se realiza por vía inalámbrica.

Para una buena comprobación de los componentes, dividiremos este paso en dos partes. La primera será comprobar cada puerto del AP de forma separada y posteriormente ambos puertos simultáneamente.

La figura 5.1 muestra el experimento realizado.

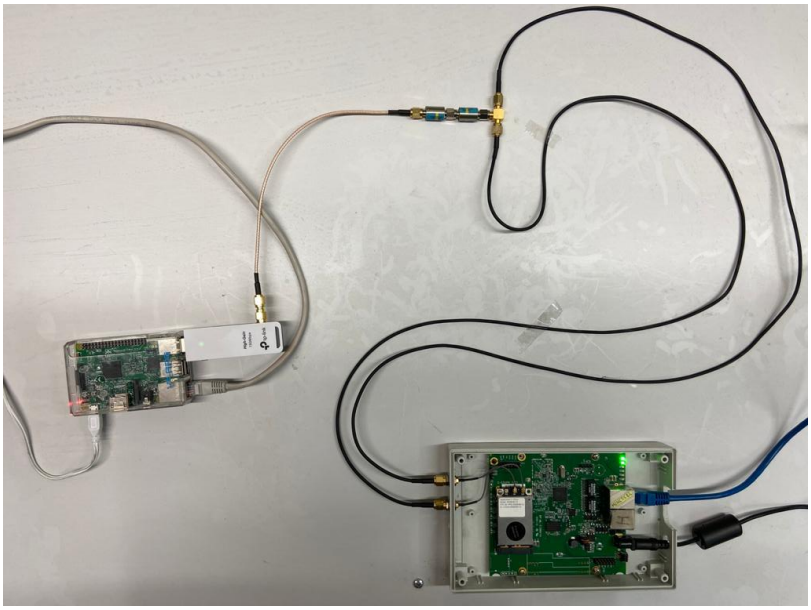


*Figura 5.1: Setup de comprobación de puertos del AP*

Los elementos presentes en el esquema son: una unión en T, dos atenuadores de 20 y 10 dB a los que posteriormente se añadirá otro atenuador de 10dB y un adaptador SMA hembra a SMA hembra invertido, además del *router* y de la Raspberry. Estos elementos irán unidos mediante tres cables coaxiales.

La función principal de los atenuadores es reducir la potencia percibida por los componentes para evitar la saturación por un exceso de potencia. De esta forma, los tres atenuadores producirán una caída de la potencia de 40 dB y puesto que la potencia transmitida de la Raspberry es de 20dBm, si el funcionamiento es correcto, deberíamos percibir una potencia de -20 dBm.

Para comprobarlo, realizamos 200 medidas de RSSI para cada puerto de los dos que usaremos en el chipset Atheros AR9380. Esta medida la haremos para cada uno de los canales wifi identificando de esta forma posibles patrones de comportamiento en función del canal utilizado.

La primera prueba la realizamos eliminando la conexión en T y conectando directamente la Raspberry con el puerto 1 del *router* mediante los atenuadores de 20dB y 10 dB.

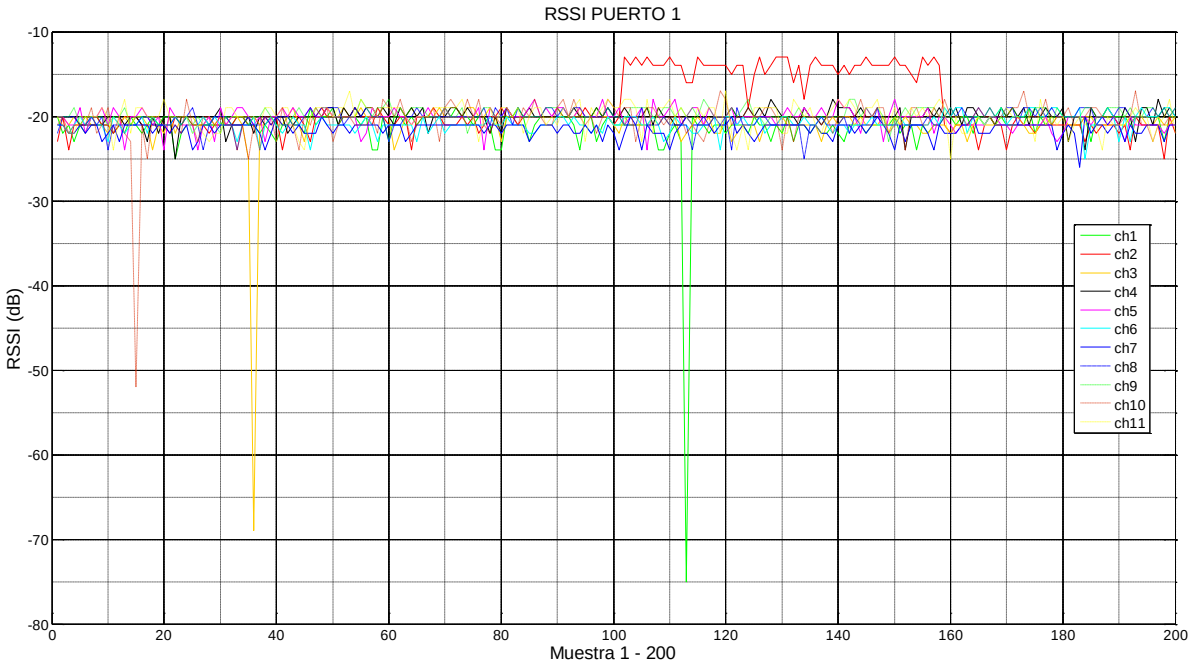



*Figura 5.2: Valores de RSSI para todos los canales en el puerto 1 (Conexión directa)*

Como podemos comprobar en la Figura 5.2, los valores de RSSI oscilan en torno a -20 dBm que es lo esperado. También se pueden apreciar valores de RSSI que aparentemente son erróneos y que necesitan ser procesados para obtener una representación fidedigna de las potencias recibidas. Los canales en los que han ocurrido estos saltos repentinos de potencia son el canal 1, el canal 3 y el canal10. La repetición en varias ocasiones de esta prueba ha mostrado que también se detectan estos valores anómalos en otros canales por lo que parece un comportamiento aleatorio, independiente de la frecuencia a la que se transmite.

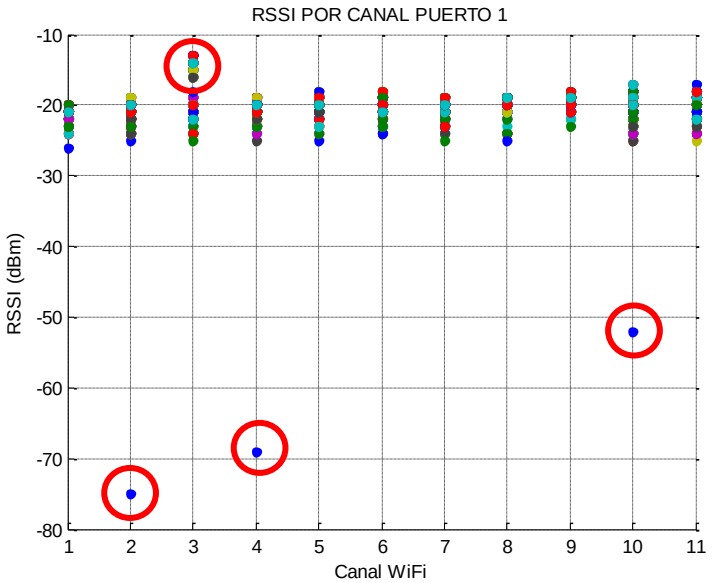



*Figura 5.3: Valores de RSSI en función del canal en el puerto 1 (Conexión directa)*

Por otro lado, aunque el control de potencia está desactivado en los elementos del sistema, podemos observar cómo en ocasiones la potencia para alguno de los canales cambia durante un corto periodo de tiempo. Esto puede provocar una interpretación errónea en el patrón de radiación de la antena puesto que esos valores se salen de la normalidad.

En la Figura 5.3, se observa que la mayoría de las muestras oscilan entorno a valores de potencia esperados, excepto valores como los marcados que se salen de la normalidad. Esto refleja la necesidad de realizar un procesado de los datos de RSSI adquiridos.

Los resultados para el puerto 2 son similares como se puede observar en las figuras 5.4 y 5.5.

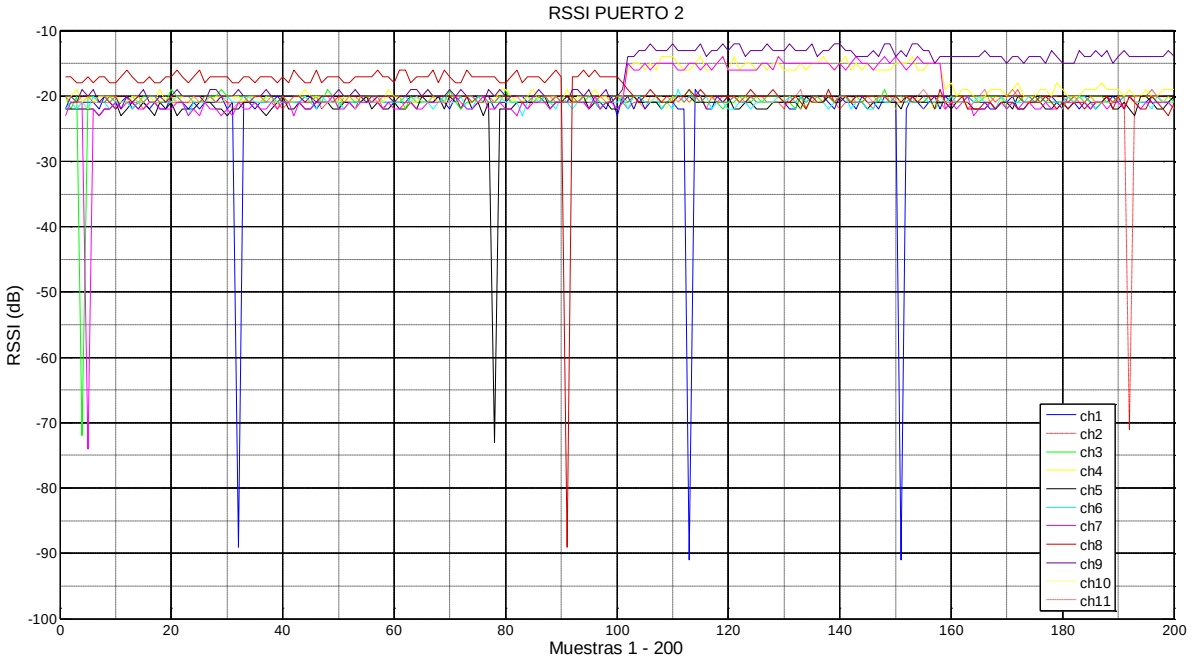



*Figura 5.4: Valores de RSSI para todos los canales en el puerto 2 (Conexión directa)*

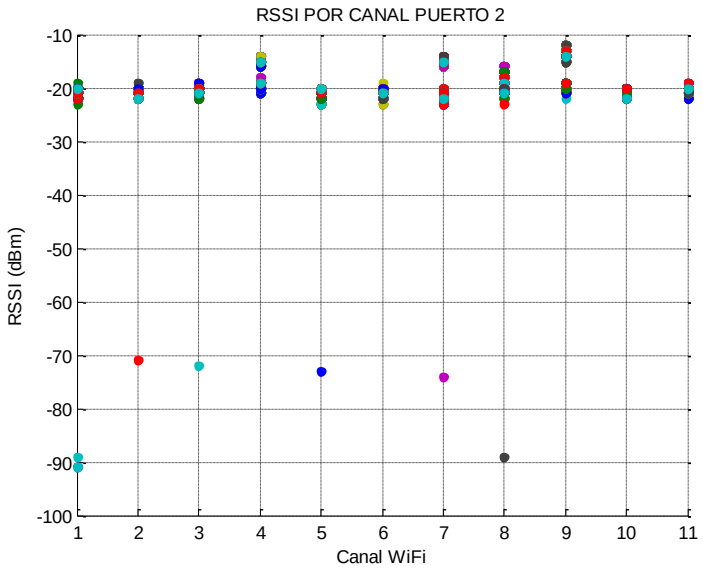



*Figura 5.5: Valores de RSSI en función del canal para el puerto 2 (conexión directa)*

Aunque el comportamiento es similar, podemos apreciar como el número de valores que se salen de la normalidad es mayor que en el puerto 1. Por ello podemos decir que el puerto 1 tiene un mejor funcionamiento que el puerto 2. En este puerto se puede apreciar como en ocasiones aparentemente aleatorias, la potencia de la señal aumenta por encima de la potencia inicialmente configurada durante un periodo de tiempo aun estando todas las medidas de control de potencia desactivadas. No hemos podido explicar el porqué de este funcionamiento extraño, pero no influirá en las medidas finales ya que lo rectificaremos mediante el procesado de los datos.

También apreciamos que los valores anómalos no dependen de la frecuencia ya que, en este caso, estos valores se han observado en los canales 1, 2, 3, 7 y 8.

Por otro lado, podemos comprobar la influencia de recibir señal de ambos puertos de forma simultánea. Para ello, unimos ambos cables coaxiales provenientes del *router* mediante una unión en T, de la cuál obtenemos una salida por coaxial que conectaremos al adaptador USB WiFi, conectado a su vez a la Raspberry. De la misma forma que en ya hemos hecho para cada uno de los puertos, obtenemos la evolución de la potencia en el tiempo.

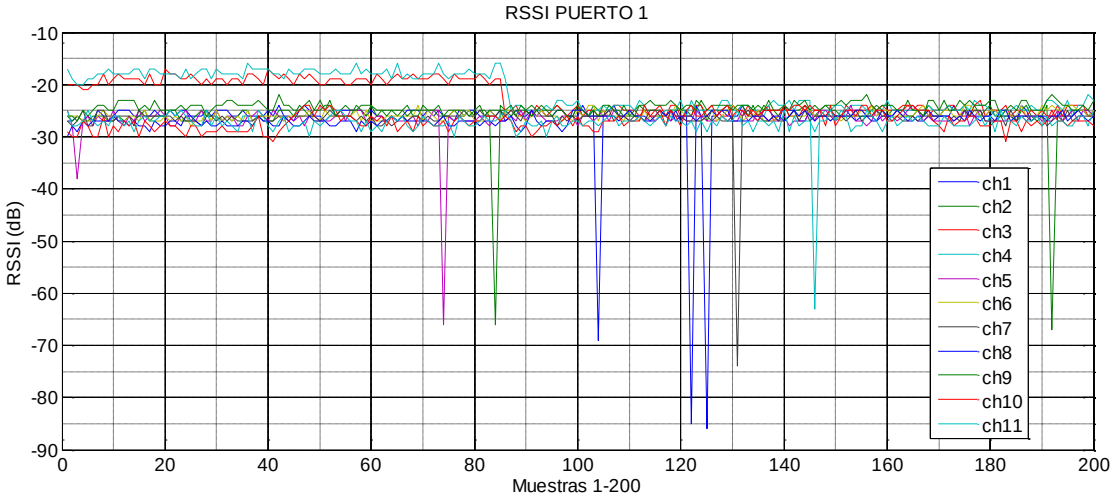



*Figura 5.6: Evolución de RSSI en el tiempo para el puerto 1 mediante la unión en T*

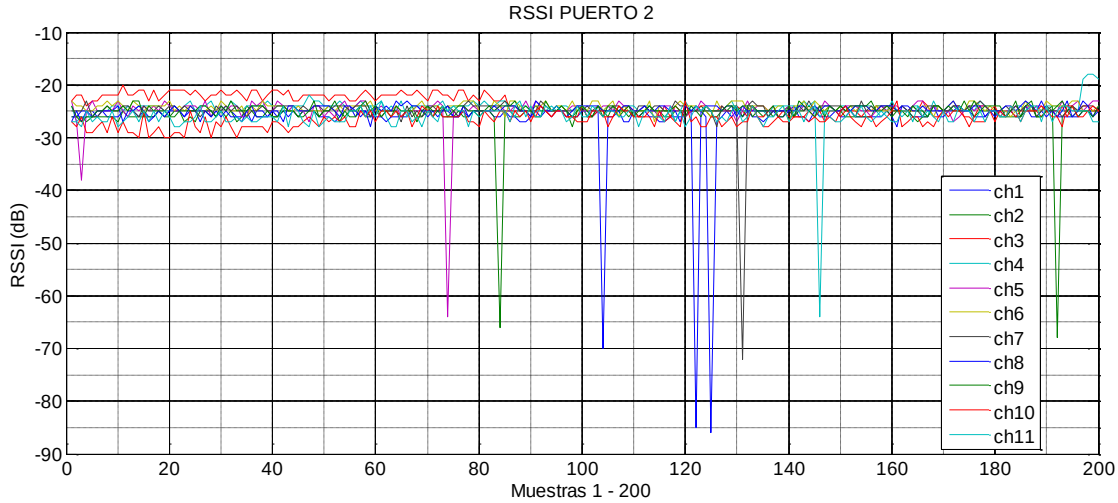



*Figura 5.7: Evolución de RSSI en el tiempo para el puerto 2 mediante la unión en T*

Como podemos ver, la potencia que la Raspberry percibe de cada uno de los puertos es muy similar. Cabe decir que la potencia que se recibe por ambos puertos se mide de forma simultánea

para ambos en un mismo instante. También podemos ver que por ambos puertos tenemos la misma caída de potencia en ciertos valores.

Por otro lado, si comparamos la toma de medidas de forma aislada para cada puerto con la que estamos realizando actualmente, podemos ver que la potencia media ha caído 5 dB. Esto puede deberse a que estamos introduciendo una unión en T además de un adaptador para poder realizar el experimento. También puede deberse a que ambas ondas atraviesan el mismo canal (cable coaxial) produciendo interferencia la una en la otra.

Finalmente, para determinar si el funcionamiento de todos los componentes es correcto, es necesario comparar los resultados arrojados por el experimento con medidas "reales". Por ello es necesario obtener las medidas analógicas que se introducen en el siguiente apartado.

## 5.2. Medidas analógicas

A continuación, se muestran las medidas de la antena tomadas en analógico. Estas medidas sirven para caracterizar la antena dibujando el diagrama de radiación que se obtiene al excitar la antena a diferentes frecuencias.

El proceso de caracterizar la antena se realiza de forma independiente para cada puerto de la antena. Primero alimentamos la antena por el puerto 1 y conectamos al puerto 2 una carga adaptada para reducir las reflexiones y posteriormente hacemos lo mismo para el puerto 2.

Para realizar estas medidas analógicas son necesarios distintos elementos que vienen detallados a continuación.

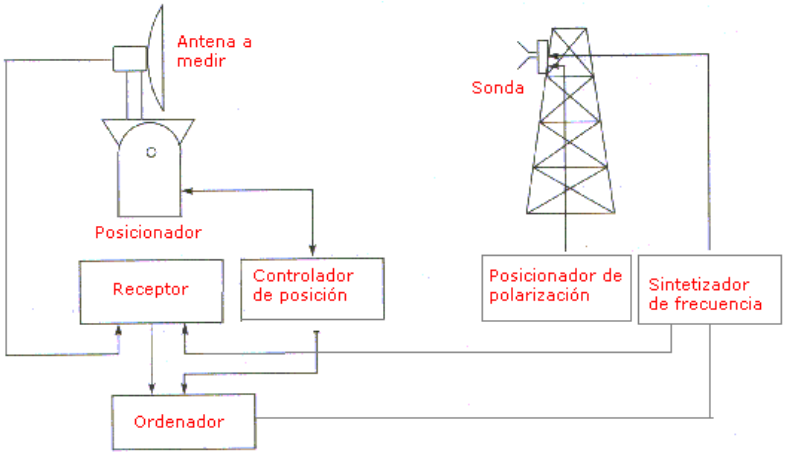



*Figura 5.8: Esquema de la instrumentación necesaria para obtener el diagrama de radiación analógico de una antena*

### 5.2.1. Cámara anecoica

Se trata de un espacio cerrado al exterior que no presenta reflexiones electromagnéticas ni acústicas. Estas reflexiones son nulas puesto que las pareces de la cámara están recubiertas de un material piramidal absorbente. Además, actúa como una jaula de Faraday al blindar las paredes con metal.

Puesto que en nuestro caso se trata de una cámara anecoica específica para radiofrecuencia, el material que forra las paredes lleva impregnado polvo de ferrita que reduce aún más las reflexiones.

Este elemento se encuentra en los laboratorios de la UPCT y simula el comportamiento que tendría la antena que se esté analizando en campo abierto, es decir, en una zona del espacio donde no hay reflexiones.

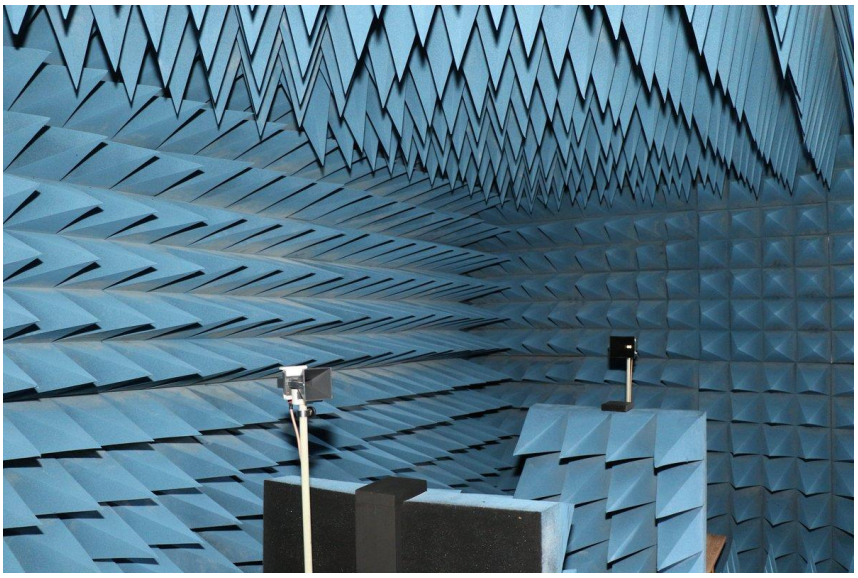


*Figura 5.9: Ejemplo de cámara anecoica*

### 5.2.2. Mesa posicionadora

La mesa giratoria es un elemento clave para facilitar la obtención del diagrama de radiación, puesto que permite automatizar el cambio de ángulo de forma precisa. En la cámara anecoica de la que disponemos, la mesa se trata de una *Compact Table CT 0800-P* de *Innco Systems*. Esta mesa es controlada por un dispositivo CO 2000, también de Innco Systems que se encuentra fuera de la cámara anecoica para poder controlarla sin acceder a ella.

*Figura 5.10: Mesa giratoria CT 0800-P*

La comunicación entre ambos elementos se realiza por fibra óptica y el controlador dispone de una interfaz GPIB que utilizaremos junto con un adaptador GPIB – USB para controlarlo a su vez desde el ordenador.

### 5.2.3. Analizador vectorial de redes

Aunque nos centraremos en las frecuencias de 2.4 GHz, el analizador utilizado opera entre 9kHz y 6GHz. Se trata de un VNA (*Vector Network Analyzer*) ZVL – Network Analyzer de Rohde & Schwarz. Este analizador dispone de un modo de funcionamiento de analizador de espectro, que será el utilizado para el experimento y gracias al que podemos visualizar las ondas de radiofrecuencia.

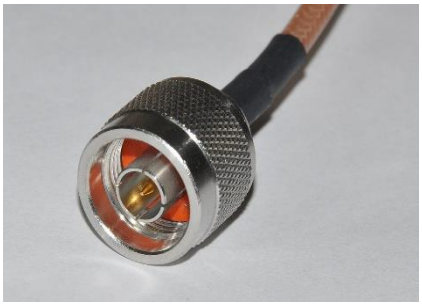


*Figura 5.11: conector tipo N macho*

Este analizador dispone de dos puertos tipo N hembra. Uno de los puertos estará conectado a la LWA mientras que el otro puerto estará conectado a la antena de panel que se introduce

más adelante y que es la encargada de capturar la radiación emitida por la antena LWA.

La conexión de la antena LWA con el VNA se da a través de un cable coaxial SMA macho mostrado en la Figura 3.6 junto con un cable coaxial SMA hembra a tipo N macho (Figura 5.11).

Por otro lado, conectamos la antena de panel al analizador mediante un cable coaxial SMA macho invertido a N macho. De esta forma, al VNA irán conectadas las dos antenas presentes en la cámara anecoica.

En la figura 5.8 aparece el esquema de los elementos necesarios para llevar a cabo esta medida. La función que realiza el sintetizador de frecuencia la realizará en nuestro caso el VNA.

Si bien podemos visualizar los resultados del experimento en la pantalla que lleva integrada el VNA, conviene representarlos en un ordenador para poder procesarlos de forma más sencilla. Para ello necesitamos conectar el VNA al ordenador mediante un cable RJ-45.

Una vez que tenemos claro el esquema de conexiones, el ordenador que va a tomar las medidas a través del VNA, deberá tener instalado el software *Keysight Connection Expert*. Este software, cuando conectemos el controlador de la mesa posicionadora y el VNA, nos detectará automáticamente ambos dispositivos estableciendo una dirección que deberemos utilizar para controlarlos de forma remota.

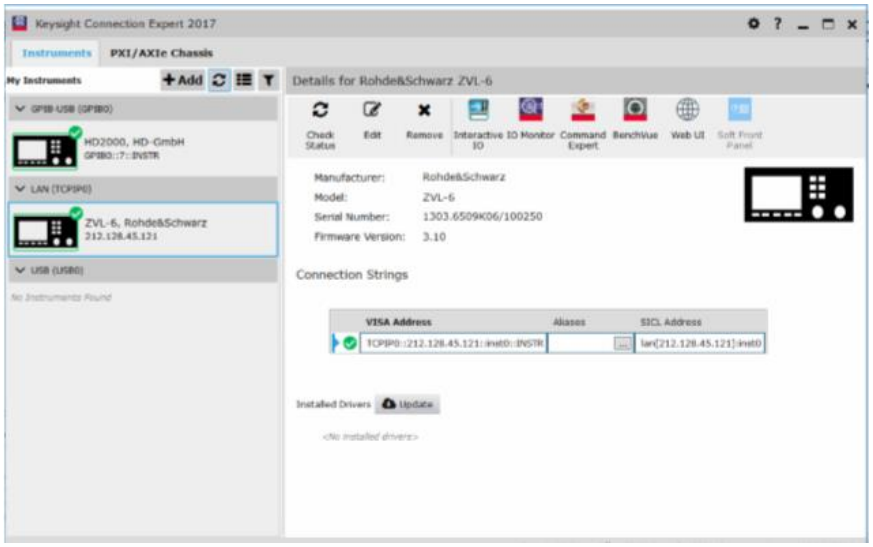



*Figura 5.12: Software Keysight Command Expert*

Posteriormente mediante el Software Matlab ejecutamos un script mostrado en el Anexo A.

### 5.2.4. Antena de panel

La antena de panel es de la empresa Interline y está diseñada en tecnología microstrip. Esta antena hará de receptor capturando las ondas transmitidas procedentes de la antena LWA y enviando los valores de potencia obtenidos al VNA para su procesado y visualización. La hoja de características se muestra en la tabla 5.1.

| | |
|---|---|
| **Frecuencia** | 2.4 – 2.5 GHz |
| **Ganancia** | 14 dBi |
| **VSWR (max)** | 1.6 |
| **Polarización** | Horizontal o Vertical |
| **Anchura de haz Horizontal** | 30º |
| **Anchura de haz vertical** | 30º |
| **Impedancia** | 50Ω |

*Figura 5.13: Antena de panel*

| | |
|---|---|
| **Dimensiones** | 200x196x20mm |
| **Peso** | 0.65kg |

*Tabla 5.1: Características de Antena de Panel*

Dentro de la cámara anecoica tenemos otros componentes que no vamos a detallar ya que no haremos uso de ellos como es el posicionador de polarización que hace girar la antena respecto al eje horizontal.

### 5.2.5. Resultados de las medidas analógicas

Para realizar nuestro experimento, debemos tener un punto de partida para poder comparar los resultados obtenidos digitalmente con los obtenidos de forma analógica. Este punto de partida serán los diagramas de radiación analógicos, los cuales tomaremos como el comportamiento real de la antena LWA.

Estos diagramas de radiación se han obtenido en un amplio rango de frecuencias, pero nos centraremos únicamente en las que se corresponden con los canales WiFi. A continuación, se muestra el diagrama de radiación normalizado para los 11 canales WiFi centrándonos en el lóbulo principal para así apreciar el ángulo de radiación para cada canal.

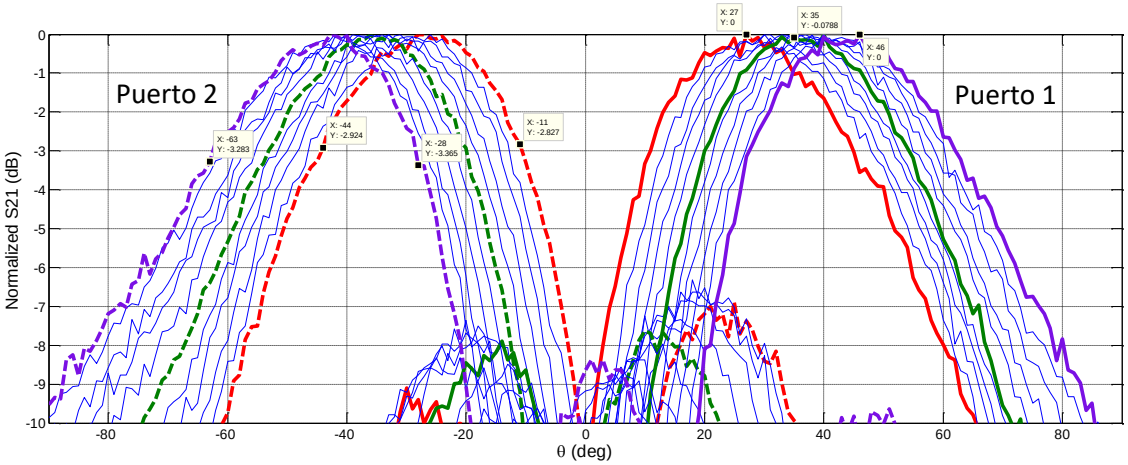



*Figura 5.14: Diagrama de radiación analógico para la LWA en los 11 canales WiFi y los 2 puertos*

Como se puede apreciar en la Figura 5.14, el canal 1 representado en rojo se aproxima más al ángulo 0º que se corresponde con la perpendicular de la antena. El resto de los canales se alejan de la perpendicular conforme aumentamos la frecuencia. También se puede apreciar como el diagrama de radiación para el puerto 1 es simétrico respecto al puerto 2 y el ancho del haz en ambos casos es de aproximadamente 34º.

En el centro de la gráfica en torno a 0º se aprecia una zona de sombra en la que ningún canal genera un haz. Por ello, la potencia recibida en ese punto sería baja aun teniendo visión directa con la antena.

El canal 1 alcanza su máximo de potencia en 27 grados respecto a la horizontal mientras que el canal 11 lo hace a los 46 grados. Esto nos indica que la antena es capaz de radiar 19 grados en por cada puerto, lo que hace un total de 38 grados.

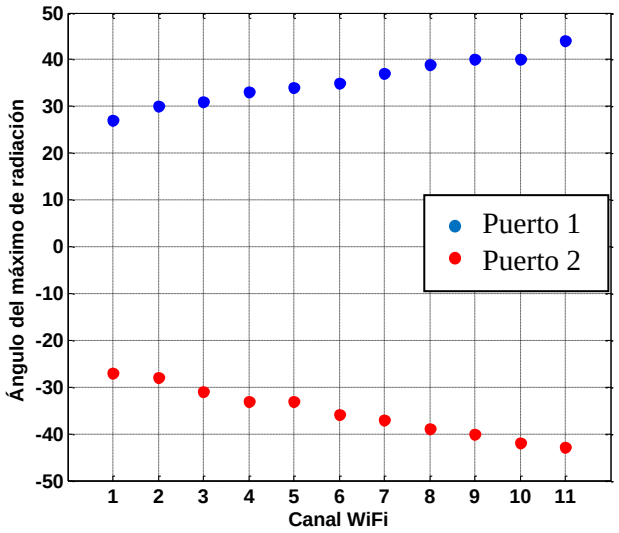



*Figura 5.15: Ángulo de radiación en función del canal.*

La Figura 5.15 se ha obtenido a partir del diagrama de radiación analógico de la antena y en ella se puede apreciar claramente como a medida que aumentamos la frecuencia, pasando con ello a un canal de mayor orden, el ángulo de radiación que se obtiene se va alejando de la perpendicular a la antena.

## 5.3. Medidas digitales

Este experimento consiste en tomar muestras de RSSI para cada canal y cada ángulo y enfrentarlos para así determinar el comportamiento de la antena en un entorno en el que un cliente está conectado al AP.

### 5.3.1. Punto de acceso

Como avanzábamos en el apartado 4.1, para automatizar el cambio de canal utilizaremos el comando que nos proporciona *hostapd* para cambiar de canal WiFi que avisa previamente a todos los clientes asociados del siguiente salto de frecuencia que se va a realizar.

Para obtener la potencia percibida de la Raspberry, el comando *iw dev wlan1 station dump* nos proporciona una valiosa información sobre la comunicación con ese cliente. En nuestro caso, filtraremos los datos que nos proporciona ese comando modificándolo ligeramente para impedir que algún cliente indeseado se cuele en las medidas adquiridas. Aunque esto es poco probable dado que estamos en un entorno controlado y aislado.

El comando resultante de esta modificación es el siguiente: *iw dev wlan1 station get "mac_dir"*. En el campo entrecomillado especificamos la dirección MAC de la interfaz wifi de la Raspberry, que en nuestro caso es d0:37:45:55:6e:a9.

Mediante el comando anterior obtenemos por pantalla el resultado mostrado en la Figura 5.16:

```
root@(none):~# iw dev wlan1 station get d0:37:45:55:6e:a9
Station d0:37:45:55:6e:a9 (on wlan1)
        inactive time:  1140 ms
        rx bytes:       56420
        rx packets:     598
        tx bytes:       49185
        tx packets:     386
        tx retries:     216
        tx failed:      11
        signal:         -34 [-55, -40, -35] dBm
        signal avg:     -48 [-71, -54, -50] dBm
        tx bitrate:     65.0 MBit/s MCS 7
        rx bitrate:     1.0 MBit/s
        expected throughput:    32.42Mbps
        authorized:     yes
        authenticated:  yes
        preamble:       short
        WMM/WME:        yes
        MFP:            no
        TDLS peer:      no
        connected time: 369 seconds
```

*Figura 5.16: Resultado del comando "iw dev wlan1 station dump"*

Para la medida de RSSI únicamente necesitaremos los valores del campo *"signal"*. Este campo contiene 4 valores que se corresponden con un valor medio y los valores de RSSI del primer, segundo y tercer puerto. Puesto que nuestro *router* lleva conectada la antena al puerto 1 y 2 (el tercer puerto disponible es el puerto 0), deberemos adquirir los últimos dos valores, que son los deseados para ser analizados.

Si bien sólo utilizaremos los valores de RSSI para caracterizar la antena digitalmente, conviene guardar el resto de valores para cada muestra del valor de RSSI para que podamos acudir a ellos en caso de cualquier anomalía en el comportamiento. Estos valores podrían indicar algún patrón que provoque ese valor anómalo.

Entre el resto de valores cabe destacar la tasa de transmisión (*tx bitrate*), la tasa de recepción (*rx bitrate*), la tasa de transferencia efectiva (*expected throughput*) y el tiempo conectado (*connected time*).

Cada medida de potencia llevará entonces una serie de datos que son únicos de esa medida y que se deberán guardar para su posterior análisis. Debido a que la capacidad de almacenamiento del *router* es limitada, con la toma de muestras de cada ángulo, el *router* enviará al ordenador por UDP el archivo creado y lo borrará de su memoria. El ordenador por su parte escuchará de forma continua la llegada de ese archivo y a cada ángulo creará una copia asignándole como nombre de fichero el instante de tiempo actual. De esta forma podemos calcular fácilmente el tiempo que tarda en completarse la toma de muestras para cada ángulo.

Por otro lado, cambiar el canal de la comunicación mediante el comando anteriormente mencionado, no es un proceso inmediato. Requiere de una confirmación del cambio por parte del *router* y de una reconexión por parte del cliente. Por ello es necesario asegurarnos de que el cliente está listo para la toma de medidas del nuevo canal, es por eso por lo que implementamos en el código desarrollado varias comprobaciones que nos lo aseguran. Estas comprobaciones son las especificadas a continuación:

- Comprobación del cambio efectivo del canal tras ejecutar la instrucción
- Comprobación de respuesta a ping por parte de la Raspberry
- Comprobación del tiempo conectado (importante ya que es un indicador de que la Raspberry ha cambiado de canal al reiniciarse con dicho cambio)

El *script* completo que se ha desarrollado para la toma de muestras y su envío al ordenador para su procesado se muestra en el Anexo B.1.

### 5.3.2. Cámara anecoica

Una vez observamos el resultado de las medidas analógicas, el experimento que se propone es obtener los mismos diagramas de radiación, pero de forma digital, tomando medidas de RSSI y representando los resultados en función del ángulo. En este punto deberemos montar el sistema como aparece en la Figura 3.28. La antena, junto con el *router*, los situaremos en la mesa posicionadora para barrer los ángulos que indiquemos y la Raspberry la fijaremos al mástil de soporte en el lado contrario de la cámara anecoica. Ambos componentes deben estar alineados tanto vertical como horizontalmente, ya que, si no están alineados, los valores de potencia que recibiremos no podremos caracterizar correctamente la antena.

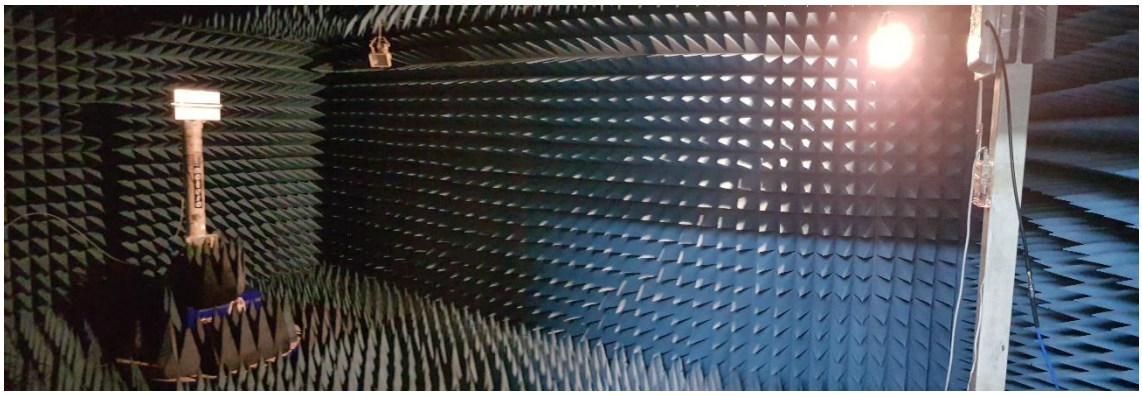


*Figura 5.17: Cámara anecoica y componentes alineados*

Por otro lado, la mesa posicionadora realizará un barrido desde -90º pasando por 0º, que se corresponde con la línea recta entre la antena y la Raspberry, hasta +90º. Este margen de ángulos es el que nos interesa ya que más allá de esos ángulos, la antena comienza a dirigir sus haces hacia regiones opuestas donde la señal percibida por la Rasberry es casi nula. De esta forma mediremos un total de 270 grados.

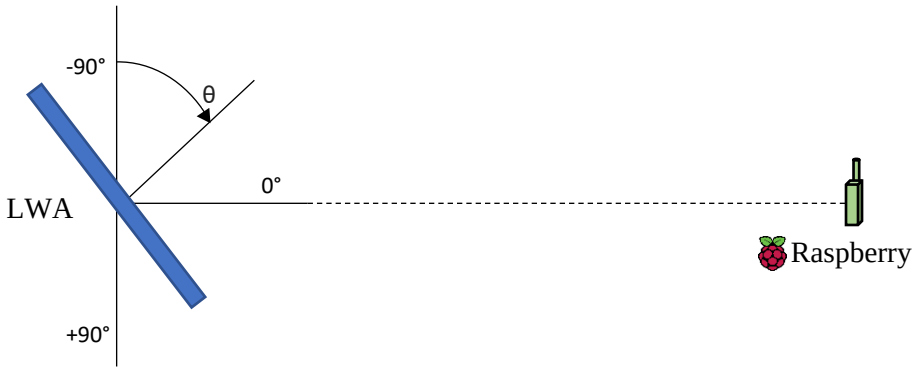



*Figura 5.18: Recorrido de la LWA para la toma de muestras*

Tanto la mesa posicionadora como el *router* los controlamos desde fuera de la cámara anecoica mediante un ordenador a través del cableado destinado para ello. Para establecer conexión con el controlador de la mesa, utilizaremos un cable GPIB-USB como hemos comentado en el apartado 5.2.2. Este cable se muestra en la Figura 5.19, y a través de él podremos mandar instrucciones al controlador, quien, a su vez, ejecutará la instrucción especificada enviando la información a la mesa posicionadora para que actualice su posición mediante el cable de fibra óptica.

*Figura 5.19: Adaptador GPIB - USB*

### 5.3.3. Desarrollo del *script* para la toma de muestras

Para comunicarnos con el controlador de la mesa posicionadora debemos especificar la dirección que nos proporcionará el software *Software Keysight Command Expert*. Esta dirección aparecerá automáticamente si hemos instalado correctamente el software y deberemos incorporarla al *script* elaborado para la toma de muestras.

La dirección debemos introducirla en la instrucción *visa* para establecer este canal de comunicación de la forma siguiente:

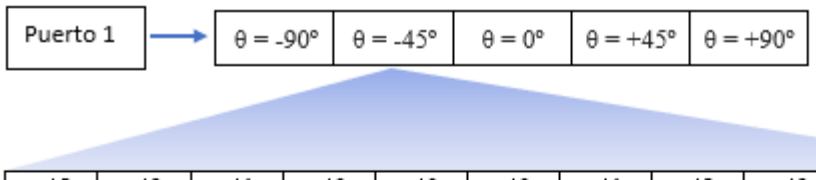
```
visa ('agilent', 'GPIB0::7::INSTR' );
```

Esta instrucción debemos introducirla en el software de Matlab junto a las encargadas de activar la comunicación y la de cambiar de ángulo que son las siguientes:

```
fprintf(mesa,'LD 1 DV');
fprintf(mesa,['LD' num2str(angulo) ' DG NP GO']) ;
```

El funcionamiento del *script* de Matlab desarrollado se centra en el uso de matrices bidimensionales para cada uno de los ángulos. De esta forma tenemos una array de matrices para cada puerto que contendrá en cada posición del array una matriz por cada ángulo. En la Figura 5.20 se muestra un ejemplo para cinco ángulos.

Como se verá más adelante, la media ha sido el método escogido para determinar la medida de RSSI en cada ángulo. Para que este valor sea representativo del total de muestras, se han cogido 100 muestras por canal de forma que los valores más dispersos se compensen con los normales y de esta forma obtener resultados más fiables. Esta toma de muestras es lenta debido a que el cambio de canal no es inmediato y las muestras de potencia consecutivas tienen un retardo de 1ms. Esto provoca que el experimento finalice en 41 horas. A demás recordemos que estamos tomando valores cada 2 ángulos, por lo que si tomáramos todos los ángulos entre -90 y +90 grados, el tiempo de realización del experimento se duplicaría.




| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| -45 | -43 | -41 | -40 | -40 | -40 | -41 | -42 | -43 | -42 | -44 |
| -46 | -42 | -42 | -41 | -40 | -39 | -40 | -41 | -43 | -42 | -43 |
| -44 | -41 | -40 | -41 | -40 | -40 | -41 | -40 | -42 | -41 | -43 |
| -45 | -42 | -42 | -40 | -39 | -41 | -41 | -41 | -43 | -43 | -45 |
| -45 | -42 | -42 | -41 | -41 | -40 | -42 | -42 | -41 | -43 | -45 |
| -46 | -43 | -42 | -41 | -41 | -39 | -39 | -42 | -42 | -42 | -44 |
| -44 | -41 | -41 | -40 | -41 | -39 | -41 | -42 | -43 | -43 | -45 |
| -45 | -42 | -40 | -40 | -42 | -40 | -42 | -41 | -43 | -43 | -44 |
| -44 | -43 | -41 | -43 | -43 | -40 | -41 | -40 | -41 | -44 | -46 |
| -46 | -43 | -40 | -41 | -39 | -39 | -41 | -42 | -43 | -43 | -44 |
| **Ch1** | **Ch2** | **Ch3** | **Ch4** | **Ch5** | **Ch6** | **Ch7** | **Ch8** | **Ch9** | **Ch10** | **Ch11** |

*Figura 5.20: Ejemplo de matriz para el ángulo -45° y puerto 1 con 10 muestras por cada canal.*

De esta forma se irá rellenando la matriz para el ángulo en el que se encuentre en ese momento antes de pasar a la matriz del siguiente ángulo. En caso de que alguna de las muestras llegue de forma errónea o no llegue, se rellenará con el valor de potencia ficticio de -100 dB para que a la hora de representar las potencias identifiquemos rápidamente ese valor y podamos corregirlo.

Por otro lado, en ocasiones una de las tramas UDP enviadas por el *router* no era almacenada en Matlab y, por lo tanto, al transmitir las 100 muestras de RSSI para un canal, en Matlab solo se almacenaban 99 de ellas, quedando así un hueco vacío. Estos huecos los transformaremos, una vez haya terminado de tomar las muestras para todos los ángulos, en la media del resto de valores de ese canal y ese ángulo.

La trama UDP que envía el *router* contiene tanto el canal en el que se está realizando la comunicación como los valores de potencia para ese instante de la forma siguiente:

Channel 2427: Signal -42 [-80 , -45 , -56]

Antes de comprobar el canal al que pertenece la muestra, debemos saber si ha llegado bien o mal. Para ello comprobamos la longitud de la trama UDP que deberá ser de 42 caracteres. En caso de que la longitud de la trama UDP sea menor de 42 caracteres, podemos decir que la trama ha sido recibida de forma errónea. Este hecho se puede dar por 4 formas distintas.

La primera forma consiste en que Matlab en ocasiones, al recibir una trama UDP, interpreta que ha recibido la trama completa cuando llega el primer carácter que en este caso es ‘C’. Por lo general este error sucede al inicio del programa y una vez que ha capturado varias muestras es muy raro que vuelva a suceder. Aun así, lo tendremos en cuenta por si se da en algún otro momento. Puesto que se suele dar al inicio, pondremos la condición de que se hayan obtenido un número de muestras considerable antes de tener que tratar este error. En caso de que falle al inicio, simplemente reiniciaremos el programa.

El segundo fallo de la transmisión y recepción de paquetes UDP se da al recibir el número del canal, pero no las potencias. Lo resolveremos estableciendo un nivel de potencia de -100 dB.

El tercer fallo se da al recibir una trama UDP que contiene la siguiente cadena: "*Channel 2427: 0"*. Esto lo trataremos de la misma forma que el caso de no recibir ninguna potencia.

Por último, se puede dar el caso de que la lectura de potencia por parte del router sea errónea y en este caso lo que Matlab recibe es un paquete que aparentemente está completo pero que alguna de las potencias se interpreta por ejemplo como -9 dBm, valor de potencia imposible dadas las características de los componentes y el resto de las medidas de potencia. Este valor también lo sustituiremos por un valor de potencia ficticio de -100 dB.

Esta gestión de errores la realizamos en Matlab mediante condicionales del tipo *if – else* y una vez que hemos conseguido rellenar todos los huecos de la matriz para un ángulo concreto, enviamos el archivo generado de los datos de respaldo de cada muestra por TCP al ordenador y pasamos al siguiente ángulo, con una pausa entre ángulos para que Matlab haga las operaciones pertinentes.

Gracias a esta gestión de errores podemos tomar las medidas de potencia garantizando que las potencias registradas y guardadas en Matlab, son las detectadas y enviadas por el *router*. Aun así, en WiFi la potencia de la señal es variable y en ocasiones presenta una variación inesperada que debemos tratar. Esta variación de la potencia se puede observar en la Figura 5.21. En ella se representa la evolución de la potencia para un ángulo fijo que está dentro de los ángulos en los que la potencia de la antena se distingue del ruido y un canal cualquiera, en este caso el canal 1. Podemos observar variaciones en los que la potencia recibida es mayor de la esperada. Esto muestra la necesidad de realizar un procesado de la señal recibida que se explica en el siguiente apartado.

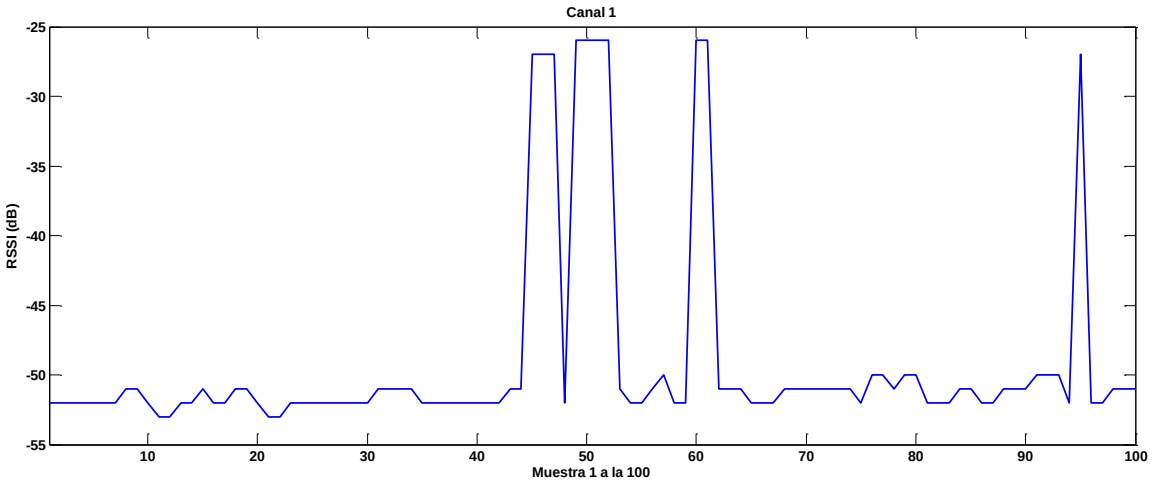



*Figura 5.21: Evolución de potencia del canal 1 a lo largo del tiempo para el ángulo -35 grados introduciendo la señal por el Puerto 1*

### 5.3.4. Procesamiento de los datos de RSSI adquiridos

La figura 5.21, como se comenta en el apartado anterior, representa un ejemplo de la evolución de la potencia de la señal WiFi adquirida y almacenada por Matlab para el ángulo -35 °. Para que la caracterización de la antena sea la mejor posible, la potencia de cada canal y cada ángulo debe ser constante por lo que eliminaremos los valores anormales sustituyéndolos por la media de aquellos valores que están dentro de la normalidad. De esta forma obtenemos el resultado

mostrado en la Figura 5.22, que representa en azul la evolución de la potencia corregida y en rojo la evolución de la potencia original para el mismo ángulo y canal que en la Figura 5.21.

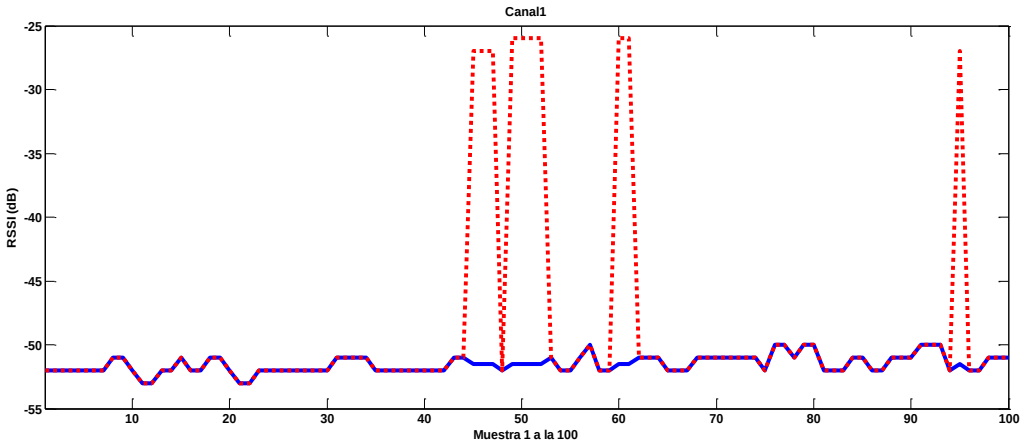



*Figura 5.22: Evolución de la potencia corregida (azul) junto a la no corregida (rojo)*

Como se puede apreciar, los valores anómalos de RSSI han desaparecido y esto lo haremos en todos los canales y en todos los ángulos ya que estas anomalías están presentes en todos ellos.

Una vez hemos obtenido una potencia aproximadamente constante, podemos representar el diagrama de radiación de la LWA. A continuación, se mostrarán los resultados únicamente para el puerto 1 ya que todo el procesamiento realizado para obtener unas lecturas buenas de las medidas de RSSI será el mismo para ambos puertos.

La Figura 5.23 muestra el diagrama de radiación de la LWA para el puerto 1 utilizando como valor real de RSSI en cada ángulo la media del total de 100 muestras por cada canal. Aunque aparecen todos los canales representados, destacaremos los canales 1, 6 y 11 para la representación de estos diagramas.

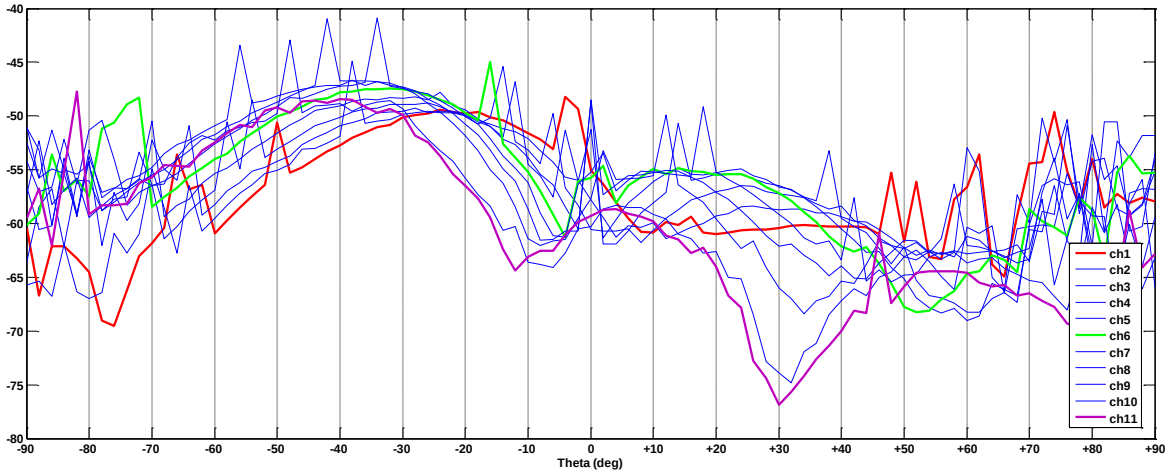



*Figura 5.23: Diagrama de radiación en coordenadas cartesianas para el puerto 1*

En esta gráfica podemos observar una gran cantidad de picos que no representan el comportamiento real de la antena y los cuáles debemos procesar para obtener una tendencia de las curvas real. En un primer momento podemos preguntarnos si la media es el mejor método para la representación de estos diagramas, por lo que también se obtuvieron dichos diagramas

utilizando como valor de RSSI para un ángulo, la moda de todos los valores recogidos para ese ángulo. De esta forma obtenemos el diagrama de la Figura 5.24.

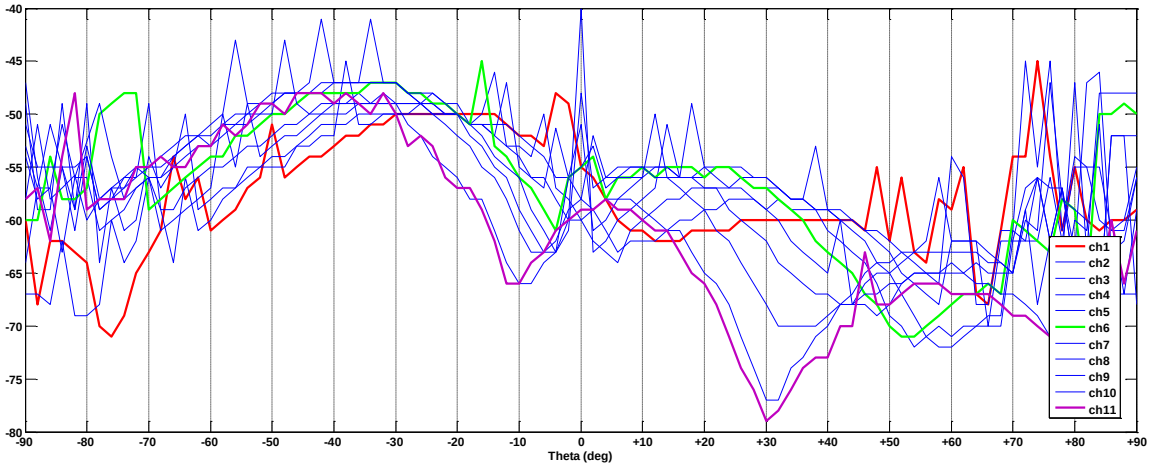



*Figura 5.24: Diagrama de radiación en coordenadas cartesianas para el puerto 1 utilizando la moda*

Como se puede observar en la Figura 5.24, el uso de la moda como método de aproximación no soluciona el problema de los valores anómalos que se observaban también al usar la media. A demás, cada uno de los canales presenta más de un máximo en la región en la que debería existir un único máximo de forma que no podemos apreciar claramente el máximo de radiación para un canal. Por esta razón descartamos la moda como método para representar los diagramas de radiación, aunque todo el proceso realizado mediante la media, también se realizó con la moda para asegurarnos.

El procesado de los diagramas de radiación se basa en comparar el valor de cada pico con los valores que se encuentran a su lado, ya que estos valores "vecinos" sí tienen concordancia con el resto de los valores. Para ello, establecemos un límite de 3dB, que hemos fijado a raíz de la observación detallada de los diagramas, a raíz del cuál si un valor se separa más de 3 dB de los dos valores que están a su lado, este valor será remplazado por la media de los otros dos valores.

Gracias a esta condición eliminamos la gran mayoría de los picos que se observan en la Figura 5.23 y podemos representar el diagrama de radiación normalizado de la antena LWA. Este diagrama se muestra en la figura 5.25. y nos permite ver ya con claridad el comportamiento de la antena LWA descrito en el apartado 2.3.

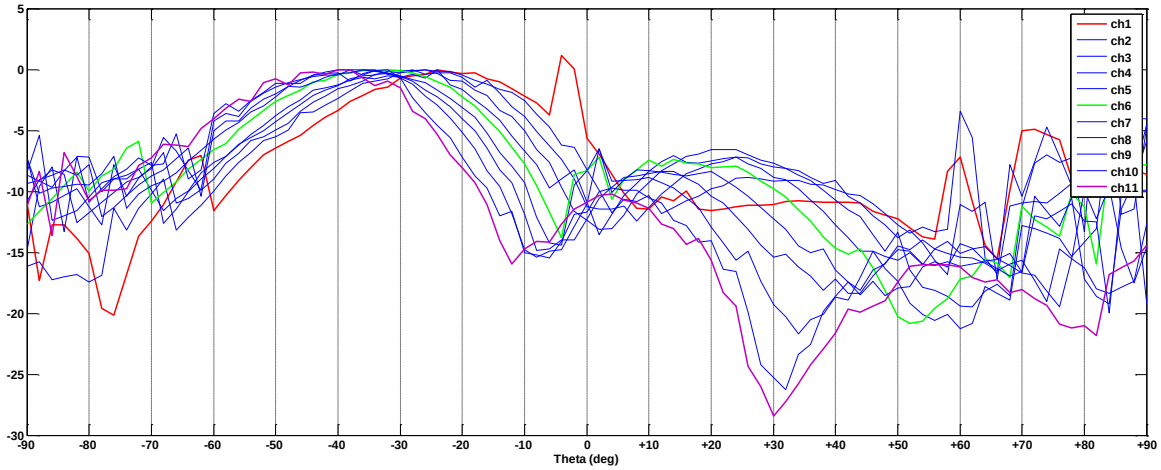



*Figura 5.25: Diagrama de radiación normalizado para el puerto 1*

Aunque hemos aplicado la condición de los "valores vecinos", aún se pueden apreciar picos en los que la potencia en varios ángulos consecutivos es mayor de la esperada como por ejemplo entre -10 y 0 grados para el canal 1. Estos valores los corregiremos de forma manual ya que son valores aleatorios que no ocurren para todos los canales, sino que se dan en raras ocasiones y que afecta a alguna de las frecuencias.

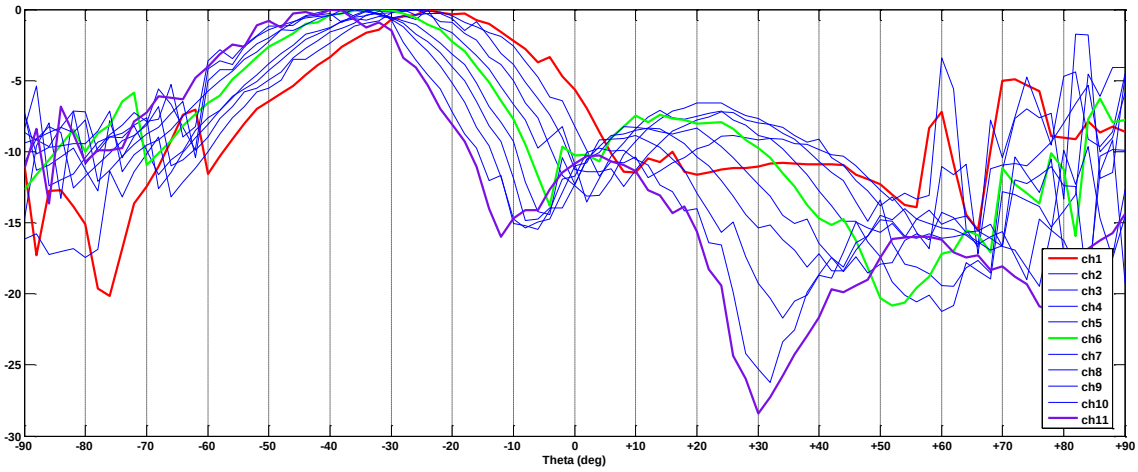



*Figura 5.26: Diagrama de radiación normalizado para el puerto 1 corrigiendo las variaciones aleatorias de potencia*

Como podemos observar en la Figura 5.26, hemos obtenido unos diagramas de radiación muy buenos tras el procesado descrito anteriormente. Gracias a ellos, ya podemos apreciar el desplazamiento del máximo hacia 0 grados que habíamos descrito en el apartado 3.3 y demostrado mediante las medidas analógicas.

Finalmente, realizamos un suavizado de las curvas mediante la función *smooth* de Matlab a la que debemos especificarle un método para suavizar. El método escogido es el filtrado mediante el método de Savitzky Sgolay ya que nos proporcionaba los mejores resultados manteniendo los valores que son interesantes analizar como el ancho de haz y los máximos y mínimos.

Este método consiste en el cálculo de una función polinómica de grado n, utilizando para ello n+1 puntos equidistantes. Utilizando esta función calculamos el nuevo valor en ese ángulo. Un ejemplo de la aproximación realizada se muestra en la Figura 5.27. En amarillo se muestra la curva aproximada a partir de la ventana de valores en rojo.

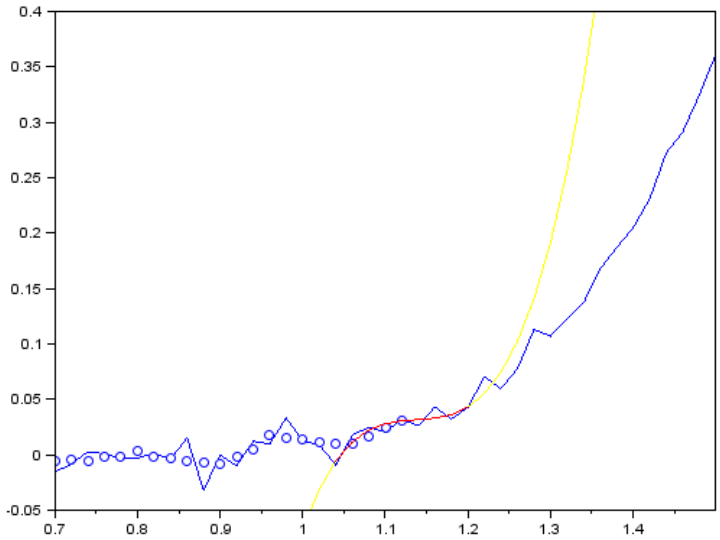



*Figura 5.27: Ejemplo de suavizado de la curva azul mediante la técnica de Savitzky Sgolay*

En resumen, para aproximar la curva, tomamos una ventana de n+1 puntos y el valor central de esa ventana tomará un nuevo valor calculado a partir del polinomio de grado n.

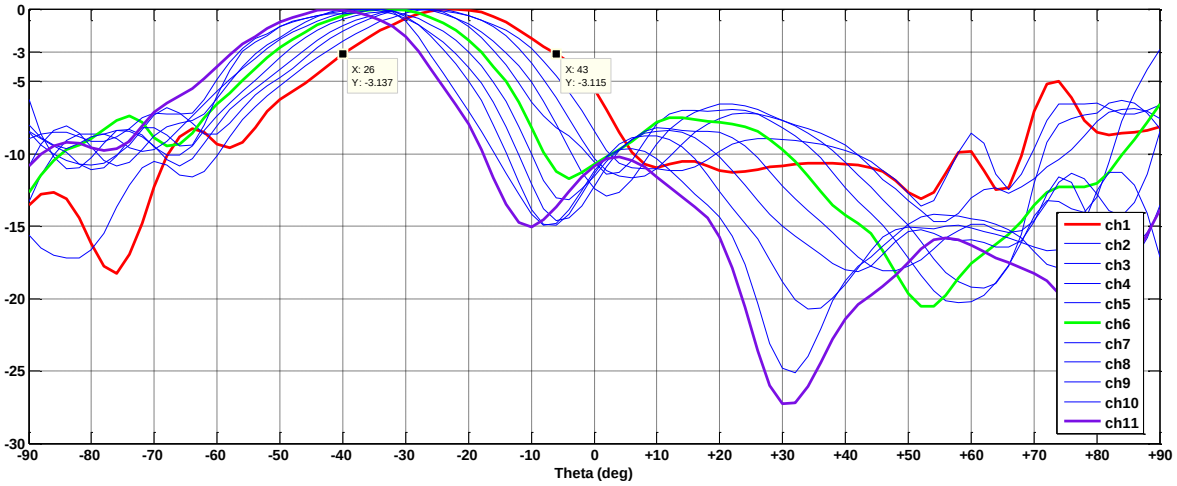



*Figura 5.28: Diagrama de radiación normalizado suavizado para el puerto 1*

Con este método conseguimos una representación del diagrama de radiación ideal para estudiarlo y poder compararlo con los resultados analógicos. Es el mostrado en la Figura 5.28 para el puerto 1 y 5.29 para el puerto 2, que como hemos dicho ha seguido el mismo proceso que el puerto 1.

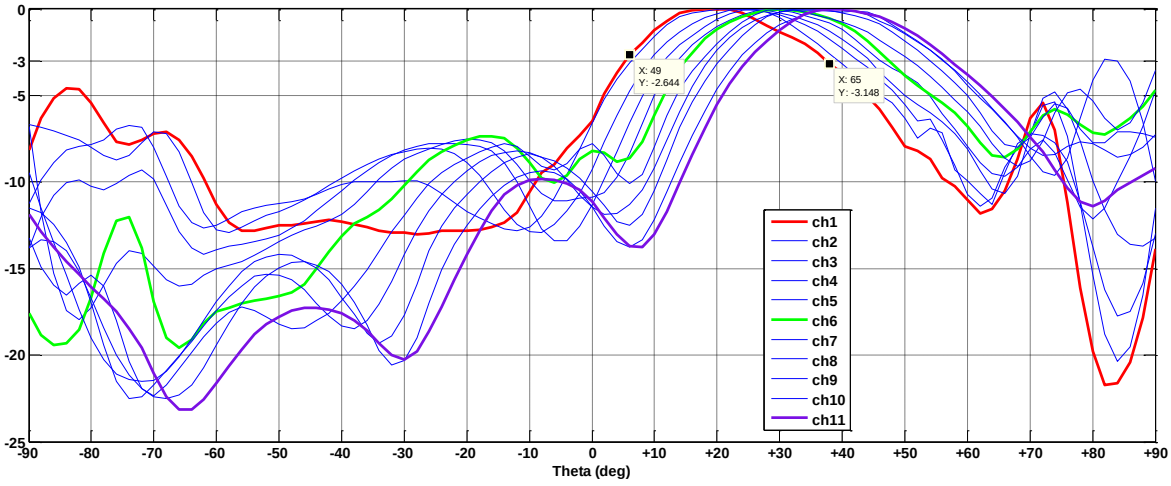



*Figura 5.29: Diagrama de radiación normalizado suavizado para el puerto 2*

En la Figura 5.30 se representan ambos puertos para los 11 canales y como podemos observar, ambos diagramas de radiación se cruzan aproximadamente en 0 grados. Este punto de cruce es el ya adelantado por el diagrama de radiación analógico donde de nuevo, observamos la zona de sombra mencionada anteriormente.

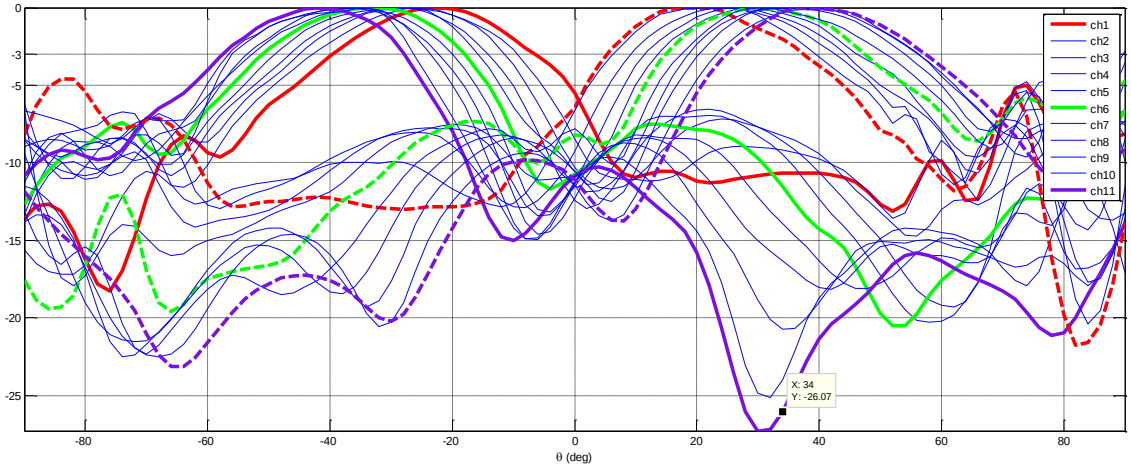



*Figura 5.30: Diagrama de radiación para ambos puertos*

En la Figura 5.37 comparamos estos diagramas obtenidos de forma digital con los diagramas analógicos.

Por otro lado, podemos obtener la posición de los máximos en este diagrama de radiación para determinar si el comportamiento de los componentes es el esperado. Para ello, representamos de forma conjunta los máximos detectados para ambos puertos.

Como se puede observar en la Figura 5.31, en ambos puertos, la tendencia del máximo de potencia es alejarse de 0 grados como ya habíamos adelantado anteriormente. Esto demuestra que el experimento está funcionando correctamente. Si bien la tendencia es ascendente en el caso del puerto 1 y descendente en el puerto 2, algunos puntos se alejan del valor que deberían tener si el comportamiento fuera el óptimo. De esta forma se introduce un pequeño error de cálculo que podría mejorarse al conseguir una separación mayor de los máximos para cada canal. Esto lo conseguiríamos según la teoría con una antena de mayor longitud.

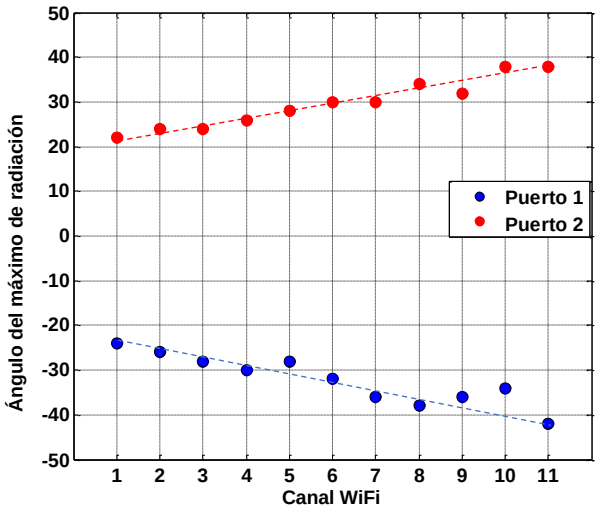



*Figura 5.31: Posición del máximo en función del canal*

Mas adelante compararemos también esta gráfica con la correspondiente a la medida analógica. En concreto, esta comparación se muestra en la Figura 3.38, tanto para el puerto 1 como para el puerto 2. A demás, en la Figura 5.37 (Pag 47), se comparan las respuestas de frecuencia a nivel analógico y a nivel digital WiFi.

### 5.3.5. Vector de RSSI (Steering Vector)

Consiste en fijar un ángulo y con ese ángulo obtener la potencia que se recibe en cada canal. Para ello, tomaremos el ángulo -10 para obtener este parámetro. Este ángulo es un ángulo aleatorio escogido de forma visual mediante el diagrama de radiación digital de la Figura 5.30.

Fijando el ángulo, estamos simulando la presencia de un cliente asociado al AP y obteniendo el vector de RSSI podemos determinar la posición de este cliente en un rango de ángulos. De esta forma, podemos estimar que el cliente está en una posición donde uno de los canales recibe mayor potencia que el resto.

Como se puede observar en la Figura 5.32, el canal 1 correspondiente al puerto 1 es el que tiene mayor potencia que el resto de los canales y si nos fijamos en el diagrama de radiación de la Figura 5.30, efectivamente el canal 1 es quien, en ese lugar, tiene mayor potencia.

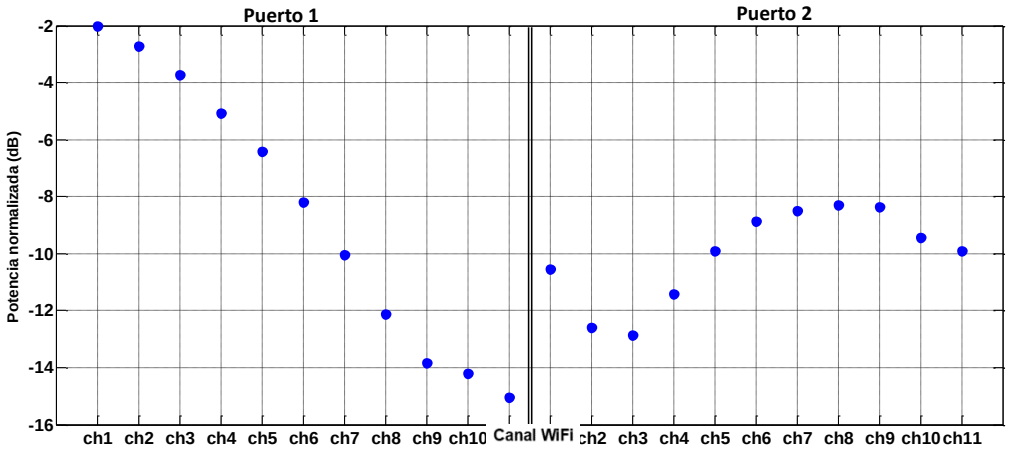



*Figura 5.32: Representación del vector de RSSI para el ángulo -10 grados.*

En el ángulo -10 se ve claramente que el canal 1 tiene mayor potencia que el resto, pero: ¿qué pasaría si escogemos un ángulo en el que no se aprecie de forma clara? Para comprobarlo, cogemos el ángulo +28 grados que se sitúa en el centro de los lóbulos primarios del segundo puerto y volvemos a calcular el vector de RSSI.

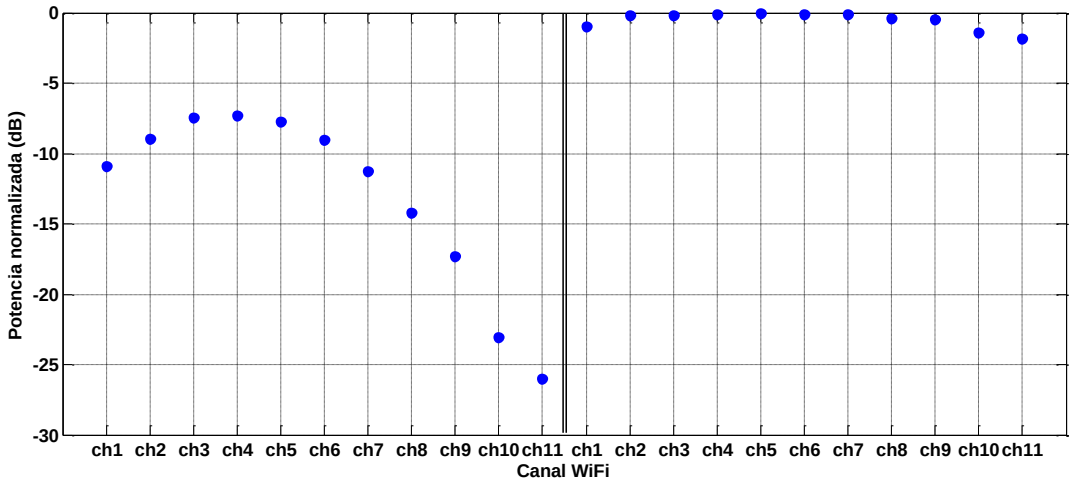



*Figura 5.33: Representación del vector de RSSI para el ángulo +28 grados.*

Como podemos ver en la Figura 5.33, el puerto 2 está detectando mayor potencia que el puerto 1, por lo cual podemos olvidarnos del puerto 1 y nos centramos en el puerto 2 para ver cuál de los canales tiene mayor potencia.

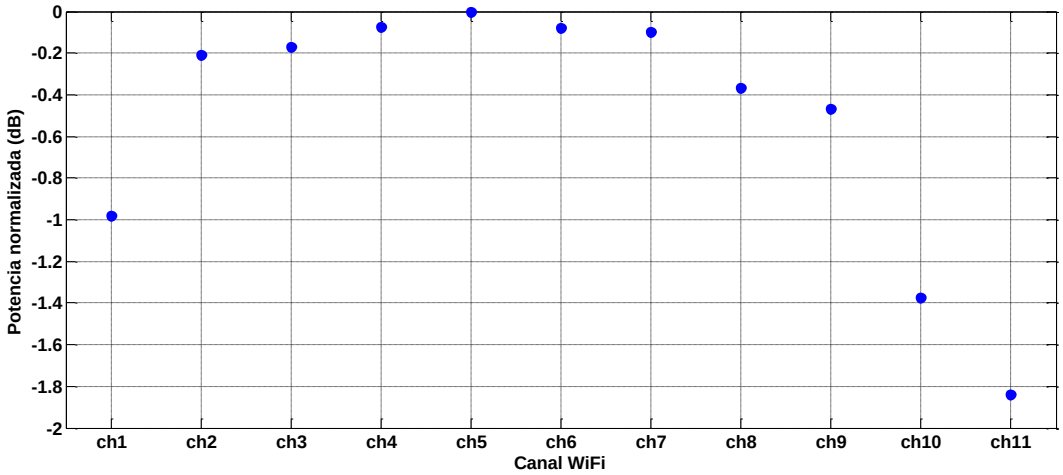



*Figura 5.34: Representación del vector de RSSI para el ángulo +28 grados en el Puerto 2*

Como vemos, el vector de RSSI nos indica que el canal 5 es el que más potencia está recibiendo, reduciéndose la potencia conforme nos alejamos de esa frecuencia. La diferencia entre la potencia detectada por los distintos canales es muy pequeña, lo que podría llevarnos a error. Para aumentar esta diferencia de potencia una de las soluciones que se propone es reducir el ancho de canal o *bandwidth* que nos permite estrechar los canales haciendo cada haz más directivo. De esta forma, hay una menor interferencia entre los canales, lo que permite una mejor diferenciación. Otra solución que ya se ha comentado pasa por ampliar la longitud de la antena LWA, que permite distanciar más los haces de los canales.

Por último, escogemos un ángulo aleatorio con la función de Matlab *randi(91,1)*, la cual nos da un valor entero entre 1 y 91. El valor que nos dé se corresponde con un ángulo par entre -90 y +90.

El objetivo es averiguar este ángulo a partir del vector de RSSI generado por lo que ejecutamos el script que se muestra en la Figura 5.35 y el cuál sirve para calcular y representar este vector.

```
angulo=randi(91,1);
for i= 1:11
    vector_rssi(i)=vector_modas_p1_11_norm(angulo,i);
    vector_rssi(i+11)=vector_modas_p2_11_norm(angulo,i);
end
figure
scatter(1:22,vector_rssi);
grid on
```

*Figura 5.35: Script para calcular y representar el vector de RSSI*

Con este sencillo *script* obtenemos el siguiente gráfico del vector de RSSI:

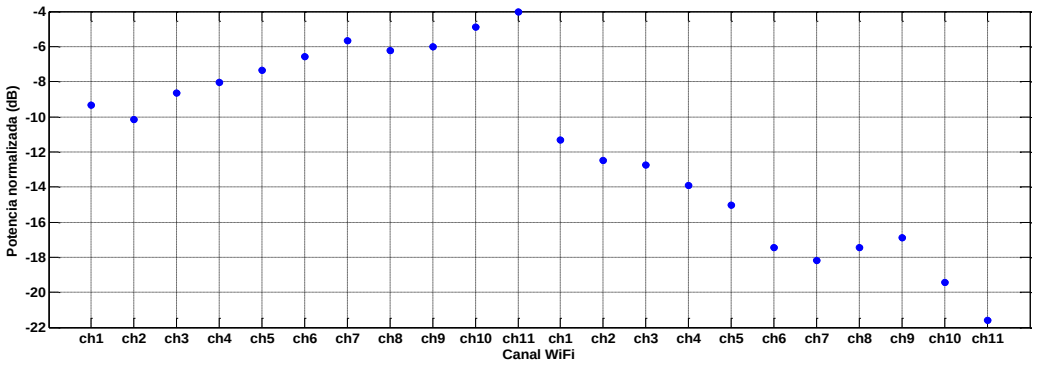



*Figura 5.36: Vector RSSI para un ángulo aleatorio*

Esta figura nos permite determinar que el canal 11 del primer puerto es quien recibe mayor potencia, por lo que observando el diagrama de radiación de la Figura 5.30, podemos decir que el ángulo generado de forma aleatoria está entre -70 y -40 grados. De esta forma, hemos aproximado la posición en función de la potencia. Si comprobamos cuál ha sido el ángulo generado aleatoriamente, veríamos ha sido el -62 grados, confirmando nuestra aproximación.

La aproximación no es muy buena ya que el punto aleatorio ha sido generado en una región donde la diferencia entre canales es muy grande y por lo tanto no podemos aproximarlo con exactitud. Si el punto generado hubiera caído en la región donde se encuentran los máximos absolutos, hubiéramos podido aproximar con mayor exactitud este ángulo aleatorio. Para nuestro ejemplo podríamos comprobar el vector de RSSI generado y compararlo con el diagrama de radiación normalizado, de forma que cuando las potencias coincidan en el diagrama y en el vector, podemos fijar casi con exactitud el ángulo. En el punto 6.2.1 se introduce un posible experimento para detectar la posición de un cliente de manera casi exacta mediante el algoritmo MUSIC.

## 5.4 Medidas analógicas frente a las digitales

Como se ha comentado anteriormente, los diagramas analógicos obtenidos eran el punto de partida a partir de los cuales comprobaríamos si el funcionamiento del sistema de adquisición de medidas de RSSI es el correcto y por tanto poder elaborar un sistema de localización en función de los valores adquiridos mediante este sistema. Para ello, debemos comparar el diagrama analógico obtenido y que se muestra en el apartado 5.2.5, con el diagrama de radiación digital que se ha obtenido mediante el experimento desarrollado a lo largo de ese proyecto. A demás debemos medir ciertos parámetros que nos van a permitir determinar la similitud entre ambos diagramas.

En la Figura 5.37 se muestran los diagramas analógico y digital superpuestos, siendo el analógico el correspondiente a la línea continua mientras que el diagrama digital es el representado mediante la línea discontinua. Como hemos visto anteriormente, los haces que se sitúan en ángulos negativos se corresponden con los generados al introducir la señal por el puerto 1 mientras que los situados en ángulos positivos el resultado de introducir la señal por el puerto 2.

Para poder observar bien la forma de ambos diagramas de radiación, nos centramos en los canales 1, 6 y 11, donde los tonos rojos representan el canal 1 de menor frecuencia (infrarrojo) pasando por el canal 6 en verde, hasta el canal 11 en morado (ultravioleta).

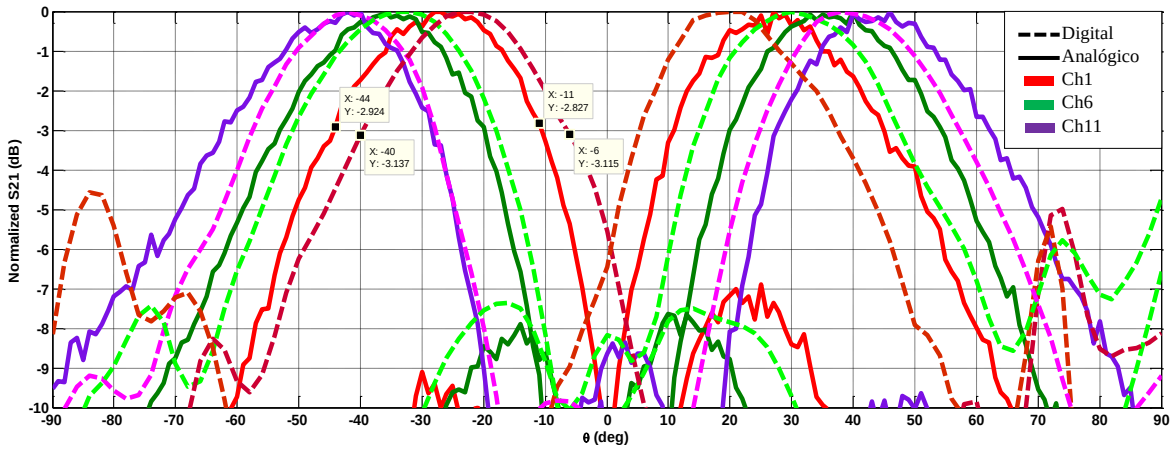



*Figura 5.37: Comparación diagrama de radiación Analógico - Digital*

Uno de los aspectos que podemos ver que diferencian a ambos diagramas es la posición del máximo para cada canal. El diagrama de radiación digital está desplazado hacia cero grados, aunque el canal 2 y 11 del puerto 1 si coinciden los máximos para ambos diagramas.

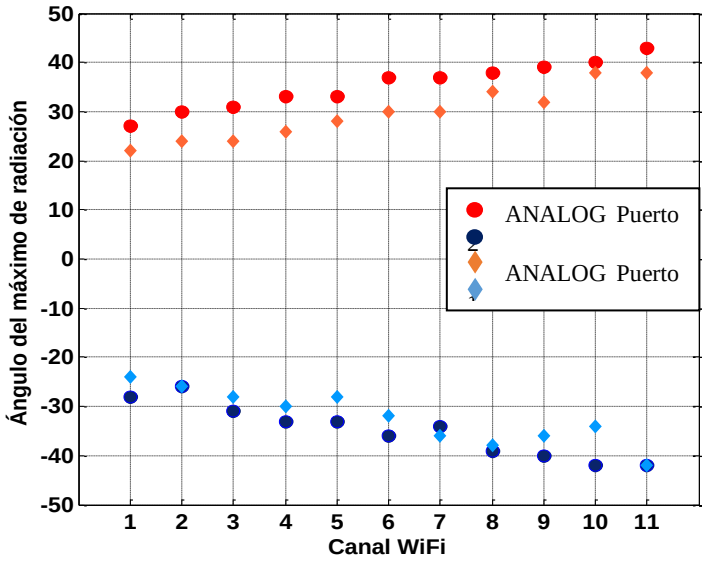



*Figura 5.38: Ángulo máximo de radiación en función del canal*

Este hecho se puede ver de forma clara en la Figura 5.38. La mayoría de los máximos del diagrama digital están desplazados hacia el 0 aunque la tendencia es la misma en ambos diagramas de radiación.

Por otro lado, podemos medir el ancho de haz, que es la separación del haz a -3 dB, es decir, a la mitad de potencia. Para medirlo, usaremos el canal 1 como ejemplo ya que en el resto de los canales el resultado es prácticamente el mismo. En el diagrama digital del puerto 1 (línea roja discontinua), el ancho de haz a -3 dB es de (40 – 6) = 34 grados, mientras que, en el diagrama analógico, este ancho de haz para el mismo puerto y canal es de (44 – 11) = 33 grados.

Podemos ver que el ancho del haz a -3 dB es prácticamente el mismo con un error de 1 grado, que puede deberse a que las medidas en digital, recordemos que las hemos realizado con intervalo de dos grados y por ello perdemos resolución a costa de reducir el tiempo de medida.

El otro parámetro que hemos calculado en los diagramas analógicos es la capacidad de radiar que tiene la antena y esta era de 19 grados por cada puerto, es decir un total de 38 grados. En el caso de los diagramas digitales podemos calcular este parámetro fácilmente mediante la gráfica de la Figura 5.38. Midiendo la diferencia entre los máximos de los canales 1 y 11, podemos obtener la información detallada en la Tabla 5.2.

| | **Analógico** | **Digital** |
|---|---|---|
| **Puerto 1** | 17 º | 18 º |
| **Puerto 2** | 19 º | 18 º |
| **Total** | 36 º | 36 º |

*Tabla 5.2: Comparación de la apertura de la antena Analógico vs Digital*

Como podemos observar en la Tabla 5.2, aunque todos los máximos están desplazados, se mantiene la capacidad que tiene la antena de radiar 36 grados.

# Capítulo Sexto

# Conclusiones y líneas futuras

## 6.1 Conclusiones

El experimento que se ha llevado a cabo consistía en utilizar una antena HW-LWA junto a un *router* que debíamos preparar para que uniendo la capacidad de cambiar de canal y con ello de frecuencia, es decir, aplicando la técnica de *frequency hopping*, aprovecháramos la capacidad de la antena LWA de radiar en un ángulo distinto en función de la frecuencia y así demostrar la aplicabilidad de este sistema a WiFi.

El primer paso se basó en configurar el *router* para que fuera capaz de cambiar de canal sin perder la conexión con el cliente, que en nuestro caso se trata de una Raspberry. Para ello utilizamos el mecanismo de aviso de cambio de canal detallado en el estándar 802.11, y por el cual los clientes conectados al AP son capaces de adaptarse a este cambio. Al mismo tiempo, este *router* debía recopilar información que nos permitiera certificar el funcionamiento esperado de la antena LWA. El parámetro que nos permitió esto es el RSSI, el cual lo obteníamos mediante el comando *station dump* de Linux. Este comando a su vez nos daba otra información acerca de la comunicación valiosa para analizar a posteriori.

De manera simultánea al primer paso, desarrollamos un programa mediante Matlab que se encargaba de recoger la información recopilada por el *router* y procesarla para obtener el diagrama de radiación de la antena LWA. Este diagrama nos permite observar el patrón de radiación de la antena y la potencia recibida para cada ángulo. Debido a las características de la antena, cambiar el canal de la comunicación suponía cambiar el patrón de radiación y con ello el máximo de potencia cambiaba de ángulo.

Finalmente comparamos los diagramas analógicos que describían el comportamiento (que supusimos que era el real) de la antena con los diagramas obtenidos digitalmente mediante el experimento realizado.

A la vista de la comparación realizada en el punto 5.4, podemos decir que los resultados son muy buenos a pesar de que las medidas en bruto presentaban muchas variaciones y valores indeseados. El diagrama de radiación que hemos conseguido nos permite ver con claridad el funcionamiento de la antena LWA y se acerca al diagrama analógico tanto en la forma del patrón de radiación como en los valores que lo caracterizan. La diferencia notable, como se ha comentado anteriormente, es el desplazamiento de los máximos en el diagrama digital hacia la horizontal de la antena.

Teniendo en cuenta estos resultados, podemos elaborar un sistema de localización que detecte la posición de un cliente en función de la potencia recibida y el canal en el que se esté llevando a cabo la comunicación.

## 6.2 Líneas futuras

En este proyecto se ha demostrado la capacidad de utilizar el *frequency hopping* junto a una antena LWA para implementar un sistema de localización del cliente en base a las potencias registradas. Este sistema de localización no ha resultado ser muy preciso debido a las características del patrón de radiación de la antena que se han descrito. Para mejorar estos resultados se ha mencionado la posibilidad de reducir el ancho de banda de los canales a la vez

que aumentar la longitud de la antena. Por lo tanto, este sería un posible punto de partida para mejorar los diagramas adquiridos.

Recordemos que hemos utilizado el mínimo ancho de banda que los componentes nos permitían y este era de 20 MHz, pudiéndose reducir con otros componentes o con un código de país que permita reducir este ancho.

Como se ha comentado a lo largo del proyecto, la toma de las muestras de RSSI es un proceso muy largo, en concreto, para nuestro proyecto hemos tomado 100 muestras por cada canal lo cuál ha llevado un total de 41 horas. De esta forma, la elección de unos componentes que reduzcan el ancho de canal también iría condicionada por reducir el tiempo de cambio de canal ya que este cambio tarda aproximadamente 7 segundos en hacerse efectivo. Por otro lado, el procesamiento de los datos que se ha requerido para obtener unos valores de RSSI estables no hubiera sido necesario en caso de que dichos valores hubieran sido de por sí válidos.

Esta estabilidad necesaria para conseguir buenos resultados ha sido el principal motivo por el que se tomaron 100 muestras, de forma que en caso de obtener componentes que alcancen esta estabilidad, el número de muestras podría reducirse y con ello el tiempo del experimento sería menor.

Aún así, el experimento realizado nos sirve para verificar que podemos implementar un sistema de localización utilizando el sistema de comunicación WiFi. En el siguiente punto se introduce un posible experimento por el que utilizando un algoritmo de localización denominado MUSIC, podemos detectar la posición exacta de un cliente.

Por otro lado, este sistema de localización usando el cambio de ángulo de radiación en función del canal utilizado, podría ser utilizado para personalizar el canal por el que un cliente se comunica con el AP. Para ello sería necesario analizar más parámetros de la comunicación a parte del RSSI como son la tasa de bits, el MCS o el *throughput*, que definen la calidad de la comunicación. Para extender aún más esta posibilidad, la utilización de un AP MU-MIMO (*Multiple User – Multiple Input Multiple Output*) que nos permita el acceso de varios clientes de forma simultánea al AP, podría permitirá su vez personalizar el canal por el que cada usuario se comunica con el punto de acceso.

### 6.2.1. Algoritmo MUSIC

El algoritmo MUSIC consiste en la estimación del ángulo de llegada (AoA)[31] o también conocida como estimación espectral y consiste en definir una función que como resultado proporcione el ángulo de llegada en función de los máximos y ángulos. Existen varios métodos de aplicar esta estimación del ángulo de llegada como la estimación *Barlett AOA*, *CAPON AOA*, y en el que nos vamos a centrar *MUSIC AOA*.

Este algoritmo, cuyas siglas hacen referencia a Clasificación Múltiple de Señales (*MUltiple Signal Classification*), permite dar como resultado estimaciones del número de señales de llegada, sus ángulos y los puntos clave de la forma de la onda. Para ello hace uso de la correlación del ruido de cada canal y genera una matriz de correlación que en caso de que las señales recibidas estén altamente correladas, supone un punto débil de este algoritmo haciendo necesario el uso de otro algoritmo. A partir de esta información, también es capaz de detectar las señales consideradas interferentes y su posición.

Este algoritmo, puesto que se basa en estimar el ángulo de donde proviene la señal captada, también se ha utilizado en la detección de actividad cerebral en pacientes con algún tipo de lesión cerebral.

Debido a la capacidad de detectar el ángulo de llegada de la señal recibida, este algoritmo ha sido probado en bluetooth con éxito[32] obteniendo una estimación del ángulo exacto haciendo uso de la misma antena que se ha usado para este proyecto. Un ejemplo de la precisión de este algoritmo se muestra en la Figura 6.1. En esta figura se muestra la estimación del ángulo de llegada y en este caso el ángulo estimado en función de valores de RSSI obtenidos es +15º.

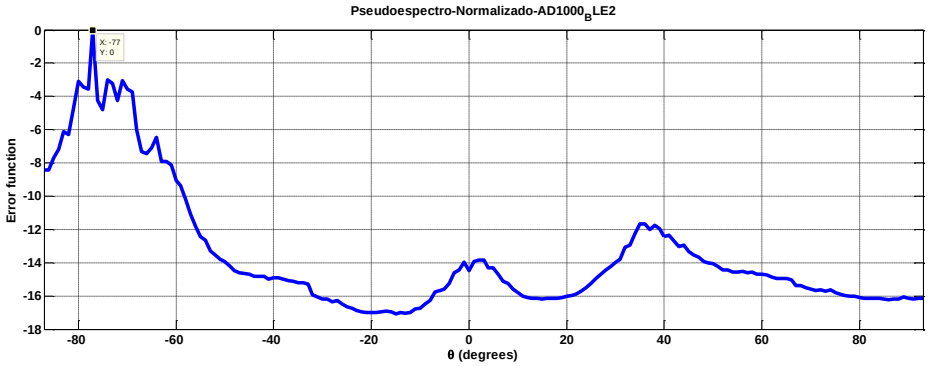



*Figura 6.1: Estimación del ángulo de llegada mediante el algoritmo MUSIC*

Este ángulo se corresponde con la región donde los haces de la antena tienen mayor presencia, pero la precisión no es tan buena en otras regiones, aunque igualmente se detecta el ángulo fácilmente. La Figura 6.2 representa el ángulo detectado -77º.

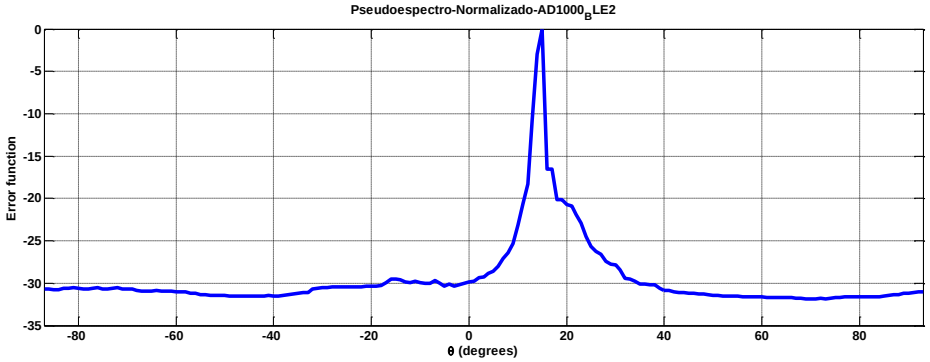



*Figura 6.2: Estimación del ángulo de llegada mediante el algoritmo MUSIC*

Como podemos ver, el ángulo se estima con buena precisión en bluetooth, por lo que aplicar este algoritmo a WiFi no debería tener resultados distintos y se podría detectar el ángulo de llegada de una señal en función de las potencias registradas.

# Anexo A

# Toma de medidas analógicas

El siguiente *script* ha sido el utilizado en las caracterización analógica de la antena LWA. Este *script* nos permite comunicarnos con la mesa posicionadora para variar el ángulo de la misma, a la vez que especificar tanto al VNA como a la antena de panel, la frecuencia a la que se va a realizar la medida. De esta forma

```
g=visa('agilent', 'TCPIP0::212.128.45.121::inst0::INSTR')
h = visa ('agilent', 'GPIB0::7::INSTR' )
set(g, 'Timeout', 60);
set(g, 'InputBufferSize',1e7)
fopen(g)
fopen(h)

fprintf(h,'LD 1 DV')
fprintf(h,'LD 000 DG NP GO')
ocupado='1';
while str2num(ocupado)==1
    fprintf(h,'BU');
    ocupado=fscanf(h);
end;
fprintf(g,'Calc1:format comp')
step=1;
for angulo_mesa_giratoria=0:step:359
    fprintf(h,'LD 1 DV')
    fprintf(h,['LD ' num2str(angulo_mesa_giratoria) ' DG NP GO'])
    ocupado='1';
    while str2num(ocupado)==1
          fprintf(h,'BU');
          ocupado=fscanf(h);
    end;
    fprintf(g,'form:dexp: asc')
    fprintf(g,'form:dexp:form comp')
    fprintf(g,'form:dexp:sour fdat')
    fprintf(g,'func "xfr:pow:s21"')
    pause(25e-3)
    freq=str2num(query(g,'trac:stim? Ch1data; *WAI'));
    pause(25e-3)
    curva21=str2num(query(g,'TRAC? CH1DATA; *WAI'));
    c_real21=curva21(1:2:length(curva21));
    c_imag21=curva21(2:2:length(curva21));
    s21(:,angulo_mesa_giratoria+1)=c_real21+j*c_imag21;
end;
fclose(g)
fclose(h)
```

```
delete(g)
delete(h)
clear c_imag21 c_real21 curva21
s21_245=s21(2,:);
s21_240=s21(1,:);
s21_250=s21(3,:);
save(nombreArchivo);
```

# Anexo B

# Toma de medidas digitales

## B.1. Captura de medidas de RSSI y comunicación con el ordenador

El script que se detalla a continuación es el desarrollado en OpenWrt e introducido en el *router* para que realice las funciones que se especifican en él. Entre estas funciones están la de cambio de canal, la de medir el RSSI en un instante y la de envío de estos valores al ordenador para su procesado. A demás de estas instrucciones, el script cuenta con funciones que nos permiten asegurarnos de que los datos que se envían se corresponden con los esperados. Este script está preparado para que se repita todo el proceso de cambio de canal y de envío de muestras para 91 valores de ángulo.

En resumen, este script es el que se ejecutará a la vez que el desarrollado en Matlab para la toma de muestras. Trabajarán de forma conjunta para obtener las medidas de RSSI y almacenarlas para su procesado.

```
#!/bin/bash

rm lecturapotencias.txt

t_before='10000'

start_time=$(date +%s)

for angulo in `seq 1 91`

do

  sleep 2

for i in [2462 2457 2452 2447 2442 2437 2432 2427 2422 2417 2412  2412
         2417 2422 2427 2432 2437 2442 2447 2452 2457 2462]

  do

echo '-----------------------------------------------------'

echo "Changing channel to ch: $i"

hostapd_cli -i wlan1 chan_switch 1 $i  # JK: para cambiar de canal:
                                         chan_switch. usage:
                                         <cs_count> <freq>
                                         [sec_channel_offset=]
                                         [center_freq1=]

                                         # [center_freq2=]
                                         [bandwidth=] [blocktx]
                                         [ht|vht]

num="$i"  #JK ¿Es necesaria esta variable? Con $i deberia servir,
supongo.
```

```
  #COMPROBACION DEL CANAL
  while :
  do
  chan=$(iw dev | grep 'channel' |awk '{print $11;}')
  if [ $num = $chan ]; then
      echo "Changed Channel to ch $chan"
      break
  fi
  done

  #COMPROBACION DE TIEMPO CONECTADO

while :
do
t_connected=$(iw dev wlan1 station get d0:37:45:55:6e:a9 | grep
'connected time'|awk '{print $3;}')
    if [ $t_connected -lt $t_before ]; then
         echo "Time ok"
         break
    fi
done
for j in `seq 1 10`  #JK: para hacer las 10 medidas.
do
   #COMPROBACION DEL DISPOSITIVO CONECTADO
  while :
  do
      check_ping=`ping -c 1 192.168.2.101`
      check=$? #devuelve 0 si el ping ha llegado y 1 si no llega
      zero="0"
      if [ $check = $zero ]; then
      echo "Device connected"
```

```
      break
      fi
  done


#MEDIDA DE POTENCIA Y ENVIO DE DATOS
  chan=$(iw dev | grep 'channel' |awk '{print $11;}')
  echo "Canal Actual: $chan"
  if [ $j = 1 ]; then
      finish_time=$(date +%s)
      echo "Time duration: $((finish_time-start_time)) secs."
  fi

  if [ $j = 10 ]; then
      start_time=$(date +%s)
  fi

data=$(iw dev wlan1 station get d0:37:45:55:6e:a9)

pot=$(echo $data | awk '{print $27, $28, $29, $30, $31;}')
echo $pot
txbitrate=$(echo $data | awk '{print $40,$41,$42,$43;}')
echo $txbitrate
rxbitrate=$(echo $data | awk '{print $44,$45,$46,$47;}')
echo $rxbitrate
expected=$(echo $data | awk '{print $48,$49,$50;}')
bad="authorized: yes authenticated:"

if [ "$expected" = "$bad" ]; then
  expected="expected throughput: NaN"
  echo $expected
  else
```

```
  echo $expected
fi
echo


#medidas auxiliares
echo "Canal: $chan Muestra: $j $data" >> lecturapotencias.txt
echo >> lecturapotencias.txt


  paq_udp="Channel: ${chan} ${pot}"
  echo $paq_udp | stdbuf -o 0 socat - udp-sendto:192.168.55.101:1883
  #sleep 1
  done


  sleep 3
  lim1="2412"
  lim2="2462"
  if [ $i = $lim1 ] || [ $i = $lim2 ]; then
    t_before='100000'
    else
    t_before=$(iw dev wlan1 station get d0:37:45:55:6e:a9 | grep
'connected time'|awk '{print $3;}')
  fi
  done # for channel
  echo
  done # for angulo
```

## B.2. Obtención de medidas de RSSI

El *script* que se muestra a continuación ha sido el utilizado para el proceso de toma de muestras de RSSI, desde la comunicación con la mesa posicionadora para controlar la orientación de la antenna LWA, pasando por la recepción de los valores de RSSI enviados por el router a través de UDP, hasta el almacenamiento de estos valores en matrices identificadas para cada ángulo y cada canal de forma inequívoca. En este *script*, también se lleva a cabo una parte del procesamiento

que consiste en la corrección de los paquetes UDP que han llegado de forma errónea y aquellos valores en los que el paquete UDP no ha sido recibido por el ordenador.

El contenido del script se ha desarrollado en el apartado 5.3.3 y es el siguiente:

```
%Función para tomar medidas de la RSSI

close all;
clear all;
clc;

n=100; %Número de muestras por canal
ang=91; %Número de angulos para tomar medidas

port=1883; %Puerto para las potencias
timeout=10000;
ip_add='192.168.55.82'; %dirección ip de donde recibe los UDP (router)
data=udp(ip_add,'LocalPort',port,'Timeout',timeout); %definicion del paquete
udp

posp1=1;%valores para seleccionar la posicion en el dibujo
posp2=2;
VECTOR_P1= zeros([n,22,ang]);   %vectores que guardan una matriz de medidas
por cada angulo
VECTOR_P2= zeros([n,22,ang]);
VECTOR_P1_MEDIOS_11=zeros([ang,11]);
VECTOR_P2_MEDIOS_11=zeros([ang,11]);
VECTOR_P1_MEDIOS_11(:,:)=-100;
VECTOR_P2_MEDIOS_11(:,:)=-100;

VECTOR_P1_raw= zeros([n,22,ang]);   %medidas sin procesar
VECTOR_P2_raw= zeros([n,22,ang]);
figure('Name','CARACTERIZACIÓN ANTENA','NumberTitle','off');
set(gcf, 'Position',  [400, 200, 1000, 600])

legend('ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','ch9','ch10','ch11')
title('Potencia en dBm para 11 canales en función del ángulo')
xlim([1 ang-1]);
XTick = 1:2.5:ang-1;
set(gca,'xtick',XTick)
set(gca,'xticklabel',({'-90','-85','-80','-75','-70','-65','-60','-55','-
50','-45','-40','-35','-30','-25','-20','-15','-10','-
5','0','+5','+15','+20','+25','+30','+35','+40','+45','+50','+55','+60','+65'
,'+70','+75','+80','+85','+90'}))
xlabel('Ángulo Theta')
ylabel('Potencia (dBm)')
grid on
%%variables de simulacion
%Configuración de mesa
step=2;
angulo_inicio=90;
angulo_giro= 270;
correctang=angulo_inicio-1; %para rellenar bien las matrices
a=1;b=1;c=1;d=1;e=1;f=1;g=1;h=1;k=1;l=1;m=1;
retardo=1;

mesa = visa ('agilent', 'GPIB0::7::INSTR' );
```

```
% % Abrimos comunicacion
open(mesa);

fprintf(mesa,'LD 1 DV');    % seleccionamos la mesa giratoria
fprintf(mesa,['LD' num2str(angulo_inicio) ' DG NP GO']) ;
pause(retardo);

%Bucle de giro

theta=1;
 for theta2=angulo_inicio:step:angulo_giro

     disp(['Cambio de grados ',num2str(theta2)]);
     fprintf(mesa,'LD 1 DV');   % Seleccionamos la mesa giratoria
     fprintf(mesa,['LD ' num2str(theta2) ' DG NP GO']);

     pause(retardo);

   for pasada=1:2
       a=1;b=1;c=1;d=1;e=1;f=1;g=1;h=1;k=1;l=1;m=1;

   for i=1:11*n
       if (a>n && pasada==1)
           break;
       end

       if(m>n && pasada ==2)
           break;
       end

      VECTOR_P1(:,:,theta)
       canales=['    ch1' '   ch2' '   ch3' '   ch4' '   ch5' '   ch6' '
      ch7' '   ch8' '   ch9' '  ch10' '  ch11' '   ch11' '   ch10' '   ch9'
      '   ch8' '   ch7' '   ch6' '   ch5' '   ch4' '   ch3' '   ch2' '
      ch1']
       disp(theta2)

       %%%%%%%%%%%%%%LEEMOS TRAMAS UDP%%%%%%%%%%%%%%%%%%
        fopen(data);
        Potencias=fread(data);
        char(Potencias')
        fclose(data);
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

       tic
       long=length(Potencias');
    if (long<42 || char(Potencias(23))==0)                %%comprobacion en
caso de desconexión o potencia 0

        if((length(Potencias)==1) && (i>50) && (char(Potencias)=='C'))
%comprobación de seguridad en caso de que la lectura de matlab de la trama
UDP falle.
            continue;
        end
```

```
        if str2double(char(Potencias(10:13)'))==2412                %%El
programa pone potencia -100 si ha habido desconexión o la potencia recibida
es 0dBm
           if pasada==1
            VECTOR_P1(a,1,theta)=-100;
            VECTOR_P2(a,1,theta)=-100;
            a=a+1;
           elseif pasada==2
            VECTOR_P1(a,22,theta)=-100;
            VECTOR_P2(a,22,theta)=-100;
            a=a+1;
           end
        elseif str2double(char(Potencias(10:13)'))==2417
           if pasada==1
            VECTOR_P1(b,2,theta)=-100;
            VECTOR_P2(b,2,theta)=-100;
            b=b+1;
           elseif pasada==2
            VECTOR_P1(b,21,theta)=-100;
            VECTOR_P2(b,21,theta)=-100;
            b=b+1;
           end
        elseif str2double(char(Potencias(10:13)'))==2422
           if pasada==1
            VECTOR_P1(c,3,theta)=-100;
            VECTOR_P2(c,3,theta)=-100;
            c=c+1;
           elseif pasada==2
            VECTOR_P1(c,20,theta)=-100;
            VECTOR_P2(c,20,theta)=-100;
            c=c+1
           end
        elseif str2double(char(Potencias(10:13)'))==2427
           if pasada==1
            VECTOR_P1(d,4,theta)=-100;
            VECTOR_P2(d,4,theta)=-100;
            d=d+1;
           elseif pasada==2
            VECTOR_P1(d,19,theta)=-100;
            VECTOR_P2(d,19,theta)=-100;
            d=d+1
            end
        elseif str2double(char(Potencias(10:13)'))==2432
           if pasada==1
            VECTOR_P1(e,5,theta)=-100;
            VECTOR_P2(e,5,theta)=-100;
            e=e+1;
           elseif pasada==2
            VECTOR_P1(e,18,theta)=-100;
            VECTOR_P2(e,18,theta)=-100;
            e=e+1;
            end
        elseif str2double(char(Potencias(10:13)'))==2437
           if pasada==1
            VECTOR_P1(f,6,theta)=-100;
            VECTOR_P2(f,6,theta)=-100;
            f=f+1;
           elseif pasada==2
            VECTOR_P1(f,17,theta)=-100;
```

```
        VECTOR_P2(f,17,theta)=-100;
        f=f+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2442
        if pasada==1
        VECTOR_P1(g,7,theta)=-100;
        VECTOR_P2(g,7,theta)=-100;
        g=g+1;
        elseif pasada==2
        VECTOR_P1(g,16,theta)=-100;
        VECTOR_P2(g,16,theta)=-100;
        g=g+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2447
        if pasada==1
        VECTOR_P1(h,8,theta)=-100;
        VECTOR_P2(h,8,theta)=-100;
        h=h+1;
        elseif pasada==2
        VECTOR_P1(h,15,theta)=-100;
        VECTOR_P2(h,15,theta)=-100;
        h=h+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2452
        if pasada==1
        VECTOR_P1(k,9,theta)=-100;
        VECTOR_P2(k,9,theta)=-100;
        k=k+1;
        elseif pasada==2
        VECTOR_P1(k,14,theta)=-100;
        VECTOR_P2(k,14,theta)=-100;
        k=k+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2457
        if pasada==1
        VECTOR_P1(l,10,theta)=-100;
        VECTOR_P2(l,10,theta)=-100;
        l=l+1;
        elseif pasada==2
        VECTOR_P1(l,13,theta)=-100;
        VECTOR_P2(l,13,theta)=-100;
        l=l+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2462
        if pasada==1
        VECTOR_P1(m,11,theta)=-100;
        VECTOR_P2(m,11,theta)=-100;
        m=m+1;
        elseif pasada==2
        VECTOR_P1(m,12,theta)=-100;
        VECTOR_P2(m,12,theta)=-100;
        m=m+1;
        end

    end

elseif str2double(char(Potencias(10:13)'))==2412
    if pasada==1
    VECTOR_P1(a,1,theta)=str2double(char(Potencias(33:35)'));
```

```
    VECTOR_P2(a,1,theta)=str2double(char(Potencias(38:40)'));
    a=a+1;
    elseif pasada==2
    VECTOR_P1(a,22,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(a,22,theta)=str2double(char(Potencias(38:40)'));
    a=a+1;
    end
elseif str2double(char(Potencias(10:13)'))==2417
    if pasada==1
    VECTOR_P1(b,2,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(b,2,theta)=str2double(char(Potencias(38:40)'));
    b=b+1;
    elseif pasada==2
    VECTOR_P1(b,21,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(b,21,theta)=str2double(char(Potencias(38:40)'));
    b=b+1;
    end
elseif str2double(char(Potencias(10:13)'))==2422
    if pasada==1
    VECTOR_P1(c,3,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(c,3,theta)=str2double(char(Potencias(38:40)'));
    c=c+1;
    elseif pasada==2
    VECTOR_P1(c,20,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(c,20,theta)=str2double(char(Potencias(38:40)'));
    c=c+1;
    end
elseif str2double(char(Potencias(10:13)'))==2427
    if pasada==1
    VECTOR_P1(d,4,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(d,4,theta)=str2double(char(Potencias(38:40)'));
    d=d+1;
    elseif pasada==2
    VECTOR_P1(d,19,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(d,19,theta)=str2double(char(Potencias(38:40)'));
    d=d+1;
    end
elseif str2double(char(Potencias(10:13)'))==2432
    if pasada==1
    VECTOR_P1(e,5,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(e,5,theta)=str2double(char(Potencias(38:40)'));
    e=e+1;
    elseif pasada==2
    VECTOR_P1(e,18,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(e,18,theta)=str2double(char(Potencias(38:40)'));
    e=e+1;
    end
elseif str2double(char(Potencias(10:13)'))==2437
    if pasada==1
    VECTOR_P1(f,6,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(f,6,theta)=str2double(char(Potencias(38:40)'));
    f=f+1;
    elseif pasada==2
    VECTOR_P1(f,17,theta)=str2double(char(Potencias(33:35)'));
    VECTOR_P2(f,17,theta)=str2double(char(Potencias(38:40)'));
    f=f+1;
    end
elseif str2double(char(Potencias(10:13)'))==2442
    if pasada==1
```

```
        VECTOR_P1(g,7,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(g,7,theta)=str2double(char(Potencias(38:40)'));
        g=g+1;
        elseif pasada==2
        VECTOR_P1(g,16,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(g,16,theta)=str2double(char(Potencias(38:40)'));
        g=g+1
        end
    elseif str2double(char(Potencias(10:13)'))==2447
       if pasada==1
        VECTOR_P1(h,8,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(h,8,theta)=str2double(char(Potencias(38:40)'));
        h=h+1;
       elseif pasada==2
        VECTOR_P1(h,15,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(h,15,theta)=str2double(char(Potencias(38:40)'));
        h=h+1;
       end
    elseif str2double(char(Potencias(10:13)'))==2452
        if pasada==1
        VECTOR_P1(k,9,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(k,9,theta)=str2double(char(Potencias(38:40)'));
        k=k+1;
        elseif pasada==2
        VECTOR_P1(k,14,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(k,14,theta)=str2double(char(Potencias(38:40)'));
        k=k+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2457
        if pasada==1
        VECTOR_P1(l,10,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(l,10,theta)=str2double(char(Potencias(38:40)'));
        l=l+1
        elseif pasada==2
        VECTOR_P1(l,13,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(l,13,theta)=str2double(char(Potencias(38:40)'));
        l=l+1;
        end
    elseif str2double(char(Potencias(10:13)'))==2462
        if pasada==1
        VECTOR_P1(m,11,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(m,11,theta)=str2double(char(Potencias(38:40)'));
        m=m+1;
        elseif pasada==2
        VECTOR_P1(m,12,theta)=str2double(char(Potencias(33:35)'));
        VECTOR_P2(m,12,theta)=str2double(char(Potencias(38:40)'));
        m=m+1;
        end
%     elseif str2double(char(Potencias(10:13)'))==2467
%         ch12_p1(p)=str2double(char(Potencias(70:72)'));
%         ch12_p2(p)=str2double(char(Potencias(75:77)'));
%         p=p+1
%     elseif str2double(char(Potencias(10:13)'))==2472
%         ch13_p1(o)=str2double(char(Potencias(70:72)'));
%         ch13_p2(o)=str2double(char(Potencias(75:77)'));
%         o=o+1;
    end

   end %fin del bucle de rellenar muestra
```

```
   % plot(i,y);
   end
   a=1;b=1;c=1;d=1;e=1;f=1;g=1;h=1;k=1;l=1;m=1;p=1;o=1;

    for j=1:22
        for i=1:n
            if(VECTOR_P1(i,j,theta)==0)

VECTOR_P1(i,j,theta)=mean(VECTOR_P1(find((VECTOR_P1(:,j,theta))<0),j,theta));
            end
             if(VECTOR_P2(i,j,theta)==0)

VECTOR_P2(i,j,theta)=mean(VECTOR_P2(find((VECTOR_P2(:,j,theta))<0),j,theta));
            end
        end
    end



   toc
   %copia de las medidas originales
   VECTOR_P1_raw(:,:,theta)=VECTOR_P1(:,:,theta);
   VECTOR_P2_raw(:,:,theta)=VECTOR_P2(:,:,theta);
   %%%%%%%%%%%%%%%% llamamos para procesar los datos %%%%%%%%%%%%%%%%%
    processdata;
    %%representacion;
    posp1=posp1+2;
    posp2=posp2+2;

  theta=theta+1;


end %while

% wait(0);

save('medidas_rssi_09032020');

fprintf(mesa,'LD 1 DV');            % seleccionamos la mesa giratoria
fprintf(mesa,['LD' num2str(angulo_inicio) ' DG NP GO']) ;

fclose(mesa);
```

## B.3. Procesamiento de los valores obtenidos

En el código que se muestra en este apartado los valores de RSSI del archivo generado a la hora de tomar las muestras de RSSI se cargan en memoria junto a las variables creadas en ese mismo *script*. Partimos de los valores de RSSI brutos almacenados en la variable "VECTOR_P1_raw" y "VECTOR_P2_raw" que se corresponden al primer y segundo puerto para procesarlos, corrigiendo aquellos valores anormales para finalmente representar el diagrama de radiación digital.

El contenido de este *script* se ha explicado detalladamente en el apartado 5.3.4 y es el siguiente:

```
load('medidas_rssi_09032020.mat')
vector_modas_p1=zeros(91,22);
vector_modas_p2=zeros(91,22);
vector_modas_p1_11=zeros(91,11);
vector_modas_p2_11=zeros(91,11);
vector_p1_raw_aux=VECTOR_P1_raw;
vector_p2_raw_aux=VECTOR_P2_raw;
vector_modas_p1_11_norm=zeros(91,11);
vector_modas_p2_11_norm=zeros(91,11);
lim=3; %limite para eliminar picos que se separen de los valores a su lado

%eliminamos los valores imposibles (por encima de 35db)
for column=1:22
    for theta=1:91
        vector_p1_raw_aux(find(VECTOR_P1_raw(:,column,theta)>-
40),column,theta)=mean(VECTOR_P1_raw(find(VECTOR_P1_raw(:,column,theta)<-
40),column,theta));
        vector_p2_raw_aux(find(VECTOR_P2_raw(:,column,theta)>-
40),column,theta)=mean(VECTOR_P2_raw(find(VECTOR_P2_raw(:,column,theta)<-
40),column,theta));

    end

end

 figure
 for u=1:91
 plot(1:100,vector_p2_raw_aux(:,:,20))
 hold on
 end


for i=1:22
    for j=1:91
        vector_modas_p1(j,i)=mean(vector_p1_raw_aux(:,i,j));
        vector_modas_p2(j,i)=mean(vector_p2_raw_aux(:,i,j));
    end
end

%SOLO LAS MODAS
figure
for i=1:11
%plot(1:91,vector_modas_p1(:,i))
plot(1:91,vector_modas_p2(:,i))
hold on

end
legend('ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','ch9','ch10','ch11')
%MODAS QUITANDO LOS PICOS
for i=1:22
    for j=2:90
        if ((vector_modas_p1(j,i)>vector_modas_p1(j+1,i)+lim) &&
(vector_modas_p1(j,i)>vector_modas_p1(j-1,i)+lim));  %si el valor esta 5db
por encima de los que tiene a los lados
```

```
        vector_modas_p1(j,i)=(vector_modas_p1(j-1,i) +
vector_modas_p1(j+1,i))/2; %cambia ese valor por la media de los que tiene a
los lados
        end
        if ((vector_modas_p2(j,i)>vector_modas_p2(j+1,i)+lim) &&
(vector_modas_p2(j,i)>vector_modas_p2(j-1,i)+lim));  %si el valor esta 5db
por encima de los que tiene a los lados
        vector_modas_p2(j,i)=(vector_modas_p2(j-1,i) +
vector_modas_p2(j+1,i))/2; %cambia ese valor por la media de los que tiene a
los lados
        end
    end

    %eliminar picos hasta el angulo 15
     for j=1:15
         if(vector_modas_p1(j,i)>-53)
             vector_modas_p1(j,i)=vector_modas_p1(j,i)-5;
         end
     end
%      for j=74:91
%          if(vector_modas_p2(j,i)>-53)
%              vector_modas_p2(j,i)=vector_modas_p2(j,i)-5;
%          end
%      end


         %vector_modas_p2(76,i)=vector_modas_p2(76,i)-3
end

figure
for i=1:11
plot(1:91,vector_modas_p2(:,i),'r')
hold on

end
hold off
legend('ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','ch9','ch10','ch11')

%correcciones de valores concretos
%puerto 1
vector_modas_p1(44,1)=vector_modas_p1(44,1)-4.5;
vector_modas_p1(45,1)=vector_modas_p1(45,1)-4.7;
vector_modas_p1(45,7)=vector_modas_p1(45,7)-1;
vector_modas_p1(46,7)=vector_modas_p1(46,7)-4;
vector_modas_p1(47,7)=vector_modas_p1(47,7)-4;
vector_modas_p1(45,6)=vector_modas_p1(45,6)-1;
vector_modas_p1(46,6)=vector_modas_p1(46,6)-2;
vector_modas_p1(47,6)=vector_modas_p1(47,6)-3;
vector_modas_p1(47,9)=vector_modas_p1(49,6)-1;
%puerto 2
vector_modas_p2(45,1)=vector_modas_p2(45,1)-5;
vector_modas_p2(44,1)=vector_modas_p2(44,1)-3;
vector_modas_p2(46,1)=vector_modas_p2(46,1)-1;
vector_modas_p2(47,1)=vector_modas_p2(47,1)-2;

   for i=1:11
       vector_modas_p1(:,i)=smooth(vector_modas_p1(:,i),8,'sgolay');
      vector_modas_p2(:,i)=smooth(vector_modas_p2(:,i),8,'sgolay');
```

```
   end
% plot(1:91,vector_modas_p1(:,1:11))
%NORMALIZAR
for theta=1:91
    for column=1:11
             vector_modas_p1_11_norm(theta,column)= -
max(vector_modas_p1(20:35,column))+vector_modas_p1(theta,column);
             vector_modas_p2_11_norm(theta,column)= -
max(vector_modas_p2(54:68,column))+vector_modas_p2(theta,column);
    end
end
figure
for i=1:11
plot(1:91,vector_modas_p1_11_norm(:,i),'r')
hold on

end
legend('ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','ch9','ch10','ch11')

%correccion de valores normalizados
%puerto 1
vector_modas_p1_11_norm(18,10)=vector_modas_p1_11_norm(18,10)-1;
vector_modas_p1_11_norm(18,9)=vector_modas_p1_11_norm(18,9)-1;
vector_modas_p1_11_norm(18,8)=vector_modas_p1_11_norm(18,8)-0.5;
vector_modas_p1_11_norm(19,10)=vector_modas_p1_11_norm(19,10)-0.5;
vector_modas_p1_11_norm(19,9)=vector_modas_p1_11_norm(19,9)-0.5;
vector_modas_p1_11_norm(19,8)=vector_modas_p1_11_norm(19,8)-0.2;
%puerto 2
vector_modas_p2_11_norm(50,1)=vector_modas_p2_11_norm(50,1)-0.5;
vector_modas_p2_11_norm(51,1)=vector_modas_p2_11_norm(51,1)-0.9;
vector_modas_p2_11_norm(52,1)=vector_modas_p2_11_norm(52,1)-0.4;
vector_modas_p2_11_norm(53,1)=vector_modas_p2_11_norm(53,1)+0.7;
vector_modas_p2_11_norm(56,1)=vector_modas_p2_11_norm(56,1)+0.1;
vector_modas_p2_11_norm(52,1)=vector_modas_p2_11_norm(52,1)+0.2;
vector_modas_p2_11_norm(76,2)=vector_modas_p2_11_norm(76,2)-9.2;
vector_modas_p2_11_norm(77,2)=vector_modas_p2_11_norm(77,2)-9.7;
vector_modas_p2_11_norm(76,3)=vector_modas_p2_11_norm(76,3)-8;
vector_modas_p2_11_norm(77,3)=vector_modas_p2_11_norm(77,3)-6;
vector_modas_p2_11_norm(78,3)=vector_modas_p2_11_norm(78,3)-5;
vector_modas_p2_11_norm(75,1)=vector_modas_p2_11_norm(75,1)-4.1;
vector_modas_p2_11_norm(76,1)=vector_modas_p2_11_norm(76,1)-4.2;
vector_modas_p2_11_norm(44,1)=vector_modas_p2_11_norm(44,1)-3;
vector_modas_p2_11_norm(45,1)=vector_modas_p2_11_norm(45,1)-3;


 for i=1:11

vector_modas_p1_11_norm(:,i)=smooth(vector_modas_p1_11_norm(:,i),8,'sgolay');

vector_modas_p2_11_norm(:,i)=smooth(vector_modas_p2_11_norm(:,i),8,'sgolay');
 end

for theta=1:91
    for column=1:11
             vector_modas_p1_11_norm(theta,column)= -
max(vector_modas_p1_11_norm(20:35,column))+vector_modas_p1_11_norm(theta,colu
mn);
```

```
            vector_modas_p2_11_norm(theta,column)= -
max(vector_modas_p2_11_norm(54:68,column))+vector_modas_p2_11_norm(theta,colu
mn);
    end
end

vector_modas_p2_11_norm(49:53,1)=vector_modas_p2_11_norm(49:53,1)+0.6;
vector_modas_p2_11_norm(54:56,1)=vector_modas_p2_11_norm(54:56,1)+0.15;
vector_modas_p2_11_norm(54,1)=vector_modas_p2_11_norm(54,1)+0.1;
vector_modas_p2_11_norm(54:55,2)=vector_modas_p2_11_norm(54:55,2)-0.1;
figure
ejex=-90:2:90;
plot(ejex,vector_modas_p1_11_norm(:,1),'r','LineWidth',2)
hold on
plot(ejex,vector_modas_p1_11_norm(:,2),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,3),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,4),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,5),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,6),'g','LineWidth',2)
hold on
plot(ejex,vector_modas_p1_11_norm(:,7),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,8),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,9),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,10),'b')
hold on
plot(ejex,vector_modas_p1_11_norm(:,11),'y','LineWidth',2)
hold on
plot(ejex,vector_modas_p2_11_norm(:,1),'r--','LineWidth',2)
hold on
plot(ejex,vector_modas_p2_11_norm(:,2),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,3),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,4),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,5),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,6),'g--','LineWidth',2)
hold on
plot(ejex,vector_modas_p2_11_norm(:,7),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,8),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,9),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,10),'b')
hold on
plot(ejex,vector_modas_p2_11_norm(:,11),'y--','LineWidth',2)
hold  off

legend('ch1','ch2','ch3','ch4','ch5','ch6','ch7','ch8','ch9','ch10','ch11')
 figure
```

```
 angulos=-90:2:90;
 angulos_max1=zeros(1,11);
 for i=1:11

angulos_max1(i)=find(vector_modas_p1_11_norm(:,i)==max(vector_modas_p1_11_nor
m(:,i)))
    angulos_max1(i)=angulos(angulos_max1(i))
 end
 scatter(1:11,angulos_max1,65,'filled');
% grid on
% grid minor
 hold on
%
figure
angulos_max2=zeros(1,11);
for i=1:11

angulos_max2(i)=find(vector_modas_p2_11_norm(:,i)==max(vector_modas_p2_11_nor
m(:,i)));
   angulos_max2(i)=angulos(angulos_max2(i));
end
scatter(1:11,angulos_max2,65,'filled','r');
% grid on
% grid minor
```

# Anexo C

## Algoritmo MUSIC empleado en bluetooth

El código que se presenta a continuación se corresponde con el algoritmo MUSIC utilizado para averiguar el ángulo de llegada de una señal. Este algoritmo ha sido aplicado a bluetooth por lo que realizar la prueba con WiFi está dentro de las líneas futuras que este proyecto permite seguir.

```
%% Definicion de variables globales
clear all
close all
clc

%PARAMETROS A DEFINIR-----------------------------------------------
antenna_name='AD1000_BLE2';
dataraw = importdata('Digital_BLE2_AD1000.txt'); %DATOS DE ENTRADA
step=1; %SALTOS ANGULARES
angulo_inicio=90; %angulos de mesa giratoria
angulo_giro= 270;
offset_angular=3; %OFFSET ANGULAR (donde corta el haz principal, mirar
a ojo)
mac_beacon1='3';
mac_beacon2='4';
ang_detectar=0;
%-------------------------------------------------------------------

eje_ang=(-90:step:90)+offset_angular; %eje angular donde se
representan los diagramas de radiación
mkdir(antenna_name);
ruta_guardado=strcat('./', antenna_name);
ruta_guardado_rssi=strcat(ruta_guardado,'/', 'espectro de
calibracion_db.mat');
ruta_guardado_rssi_norm=strcat(ruta_guardado,'/', 'maximos_RSSI.mat');
ruta_guardado_mf=strcat(ruta_guardado,'/', 'fm.mat');


ang_detectar_real=ang_detectar+180-offset_angular;


%Eleccion de funciones monopulso (manualmente)
%Seleccion de funciones monopulso combinacion canal_puerto
puerto(1,2), canal(37,38,39)
%[1(ch37_1),2(ch38_1),3(ch39_1),4(ch37_2),5(ch38_2),6(ch39_2)]

%Combinaciones momnopulso correspondientes a antena AD1000 2BLE
ind_fm_1=[4 1];
ind_fm_2=[2 1];
ind_fm_3=[3 1];
ind_fm_4=[3 2];
ind_fm_5=[4 5];
ind_fm_6=[4 6];
ind_fm_7=[5 6];


%% Obtención de diagramas de radiacion
```

```
for i=1:length(dataraw.textdata(:,2))
    MAC_aux1=strjoin(dataraw.textdata(i,2));
    MAC_aux(i,1)=MAC_aux1(1);
end

k=1;
for angulo_mesa_giratoria=angulo_inicio:step:angulo_giro

    indices37_1 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon1); %filas de la matriz
de datos donde están los datos de cada ángulo/canal/beacon
    indices38_1 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon1);
    indices39_1 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon1);
    indices37_2 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon2);
    indices38_2 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon2);
    indices39_2 = find(dataraw.data(:,4)==angulo_mesa_giratoria &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon2);


    mean_RSSI37_1(k) = mean(dataraw.data(indices37_1,2)); %media de
RSSI de todos los valores anteriormente encontrados
    mean_RSSI38_1(k) = mean(dataraw.data(indices38_1,2));
    mean_RSSI39_1(k) = mean(dataraw.data(indices39_1,2));
    mean_RSSI37_2(k) = mean(dataraw.data(indices37_2,2));
    mean_RSSI38_2(k) = mean(dataraw.data(indices38_2,2));
    mean_RSSI39_2(k) = mean(dataraw.data(indices39_2,2));
     k=k+1;
end

% represtanción de diagramas

figure
plot(eje_ang,mean_RSSI37_1,'r', 'LineWidth', 2, 'LineStyle', '-'),hold
on
plot(eje_ang,mean_RSSI38_1,'g', 'LineWidth', 2, 'LineStyle', '-')
plot(eje_ang,mean_RSSI39_1,'b', 'LineWidth', 2, 'LineStyle', '-')
plot(eje_ang,mean_RSSI37_2, 'r', 'LineWidth', 2, 'LineStyle', '-
.'),hold on
plot(eje_ang,mean_RSSI38_2,'g', 'LineWidth', 2, 'LineStyle', '-.')
plot(eje_ang,mean_RSSI39_2,'b', 'LineWidth', 2, 'LineStyle', '-.')
xlabel('\theta (°)'), ylabel('RSSI(dBm)')
legend('ch37 P1','ch38 P1','ch39 P1','ch37 P2','ch38 P2','ch39 P2')
grid on
gca.FontSize=20;
gca.FontName='Times New Roman';
xlim([eje_ang(1) eje_ang(end)])
title(strcat('Diagrama ',antenna_name))


%normalización de diagramas

maximos_RSSIs=[max(mean_RSSI37_1) max(mean_RSSI38_1)
max(mean_RSSI39_1) max(mean_RSSI37_2) max(mean_RSSI38_2)
max(mean_RSSI39_2)];
```

```
maximo_RSSIs=max(maximos_RSSIs); %Valores de normalizacion en dBs

figure
plot(eje_ang,mean_RSSI37_1-maximos_RSSIs(1), 'r', 'LineWidth', 2,
'LineStyle', '-'),hold on
plot(eje_ang,mean_RSSI38_1-maximos_RSSIs(2),'g', 'LineWidth', 2,
'LineStyle', '-')
plot(eje_ang,mean_RSSI39_1-maximos_RSSIs(3),'b', 'LineWidth', 2,
'LineStyle', '-')
plot(eje_ang,mean_RSSI37_2-maximos_RSSIs(4), 'r', 'LineWidth', 2,
'LineStyle', '-.'),hold on
plot(eje_ang,mean_RSSI38_2-maximos_RSSIs(5),'g', 'LineWidth', 2,
'LineStyle', '-.')
plot(eje_ang,mean_RSSI39_2-maximos_RSSIs(6),'b', 'LineWidth', 2,
'LineStyle', '-.')
xlabel('\theta (°)'), ylabel('RSSI(dBm)')
legend('ch37 P1','ch38 P1','ch39 P1','ch37 P2','ch38 P2','ch39 P2')
xlim([eje_ang(1) eje_ang(end)])
grid on
gca.FontSize=20;
gca.FontName='Times New Roman';
title(strcat('Diagrama ',antenna_name ,' Normalizado'))

%ESPECTRO DE CALIBRACION DESDE -90:90 grados (-90° columna1, -89°
columna2.....90° columna 181)
%rssi_values=[beacon1_ch37; beacon1_ch38; beacon1_ch39; beacon1_ch37;
%beacon2_ch38; beacon2_ch39
espectro_calibracion_db=[mean_RSSI37_1; mean_RSSI38_1; mean_RSSI39_1;
mean_RSSI37_2; mean_RSSI38_2; mean_RSSI39_2]; %matriz 6x181 con todos
los valores de RSSI medios

save(ruta_guardado_rssi, 'espectro_calibracion_db');
save(ruta_guardado_rssi_norm, 'maximos_RSSIs');

%% Calculo de funciones monopulso

clc


espectro_calibracion_db=load(ruta_guardado_rssi);
maximos_RSSIs=load(ruta_guardado_rssi_norm);

espectro_calibracion_db=espectro_calibracion_db.espectro_calibracion_d
b;
maximos_RSSIs=maximos_RSSIs.maximos_RSSIs;

%espectro calibracion (dB) --> espectro de calibracion normalizado en
%lineal

for i=1:length(maximos_RSSIs)
    espectro_calibracion_lineal(i,:) =
(10.^((espectro_calibracion_db(i,:)-maximos_RSSIs(i))/10)).^2;
end


fm_1=(espectro_calibracion_lineal(ind_fm_1(1),:)-
espectro_calibracion_lineal(ind_fm_1(2),:))./(espectro_calibracion_lin
eal(ind_fm_1(1),:)+espectro_calibracion_lineal(ind_fm_1(2),:));
```

```
fm_2=(espectro_calibracion_lineal(ind_fm_2(1),:)-
espectro_calibracion_lineal(ind_fm_2(2),:))./(espectro_calibracion_lin
eal(ind_fm_2(1),:)+espectro_calibracion_lineal(ind_fm_2(2),:));
fm_3=(espectro_calibracion_lineal(ind_fm_3(1),:)-
espectro_calibracion_lineal(ind_fm_3(2),:))./(espectro_calibracion_lin
eal(ind_fm_3(1),:)+espectro_calibracion_lineal(ind_fm_3(2),:));
fm_4=(espectro_calibracion_lineal(ind_fm_4(1),:)-
espectro_calibracion_lineal(ind_fm_4(2),:))./(espectro_calibracion_lin
eal(ind_fm_4(1),:)+espectro_calibracion_lineal(ind_fm_4(2),:));
fm_5=(espectro_calibracion_lineal(ind_fm_5(1),:)-
espectro_calibracion_lineal(ind_fm_5(2),:))./(espectro_calibracion_lin
eal(ind_fm_5(1),:)+espectro_calibracion_lineal(ind_fm_5(2),:));
fm_6=(espectro_calibracion_lineal(ind_fm_6(1),:)-
espectro_calibracion_lineal(ind_fm_6(2),:))./(espectro_calibracion_lin
eal(ind_fm_6(1),:)+espectro_calibracion_lineal(ind_fm_6(2),:));
fm_7=(espectro_calibracion_lineal(ind_fm_7(1),:)-
espectro_calibracion_lineal(ind_fm_7(2),:))./(espectro_calibracion_lin
eal(ind_fm_7(1),:)+espectro_calibracion_lineal(ind_fm_7(2),:));

%matriz de N funciones monopulso caracterizadas de -90°:90° de
dimensiones Nx181
fm=[fm_1; fm_2; fm_3; fm_4; fm_5; fm_6; fm_7];
save(ruta_guardado_mf, 'fm');

%% Representación de funciones monopulso


fm=load(ruta_guardado_mf);
fm=fm.fm;

%elegir manualmente FoV de la función monopulso para pintarla en otro
%estilo de linea

figure
plot(eje_ang(76:97),fm(1,76:97),  'r', 'LineWidth', 2, 'LineStyle', '-
'),hold on
plot(eje_ang(88:100),fm(2,88:100),  'g', 'LineWidth', 2, 'LineStyle',
'-')
plot(eje_ang(104:128),fm(3,104:128),  'b', 'LineWidth', 2,
'LineStyle', '-')
plot(eje_ang(104:128),fm(4,104:128),  'y', 'LineWidth', 2,
'LineStyle', '-')
plot(eje_ang(73:88),fm(5,73:88),  'c', 'LineWidth', 2, 'LineStyle', '-
')
plot(eje_ang(64:73),fm(6,64:73),  'm', 'LineWidth', 2, 'LineStyle', '-
')
plot(eje_ang(48:72),fm(7,48:72),  'k', 'LineWidth', 2, 'LineStyle', '-
')

plot(eje_ang,fm(1,:),  'r', 'LineWidth', 1, 'LineStyle', '--'),hold on
plot(eje_ang,fm(2,:),  'g', 'LineWidth', 1, 'LineStyle', '--')
plot(eje_ang,fm(3,:),  'b', 'LineWidth', 1, 'LineStyle', '--')
plot(eje_ang,fm(4,:),  'y', 'LineWidth', 1, 'LineStyle', '--')
plot(eje_ang,fm(5,:),  'c', 'LineWidth', 1, 'LineStyle', '--')
plot(eje_ang,fm(6,:),  'm', 'LineWidth', 1, 'LineStyle', '--')
plot(eje_ang,fm(7,:),  'k', 'LineWidth', 1, 'LineStyle', '--')

grid on
xlabel('\theta (degrees)'), ylabel('Monopulso')
title(strcat('Funciones monopulso',' ',antenna_name))
```

```
legend('monopulso1','monopulso2','monopulso3','monopulso4','monopulso5
','monopulso6','monopulso7')
xlim([eje_ang(1) eje_ang(end)])
gca.FontSize=20;
gca.FontName='Times New Roman';

%% Estimacion del angulo por funciones monopulso con Valores MEDIOS
(testbench)

%Eleccion de valores medios (0) o valores instantaneos aleatorios (1)
datos_entrada=0;
%-------------------------------------------------------------------

maximos_RSSIs=load(ruta_guardado_rssi_norm);
maximos_RSSIs=maximos_RSSIs.maximos_RSSIs;

fm=load(ruta_guardado_mf);
fm=fm.fm;

if(datos_entrada==0)
    espectro_calibracion_db=load(ruta_guardado_rssi);

espectro_calibracion_db=espectro_calibracion_db.espectro_calibracion_d
b;
else
    for i=1:length(dataraw.textdata(:,2))
        MAC_aux1=strjoin(dataraw.textdata(i,2));
        MAC_aux(i,1)=MAC_aux1(1);
    end
end



k=1;
figure
for h=-90+offset_angular:90+offset_angular
%monopulse values

    ang_detectar_real=h+180-offset_angular;

    if (datos_entrada==0)
        rssi_MV=[
            espectro_calibracion_db(1,(ang_detectar_real-89))-
maximos_RSSIs(1)
            espectro_calibracion_db(2,(ang_detectar_real-89))-
maximos_RSSIs(2)
            espectro_calibracion_db(3,(ang_detectar_real-89))-
maximos_RSSIs(3)
            espectro_calibracion_db(4,(ang_detectar_real-89))-
maximos_RSSIs(4)
            espectro_calibracion_db(5,(ang_detectar_real-89))-
maximos_RSSIs(5)
            espectro_calibracion_db(6,(ang_detectar_real-89))-
maximos_RSSIs(6)
        ];
    else
        %tomamos un valor de rssi en cada puerto canal en el angulo
que se quiere
        %estimar
```

```
        %indices donde se encuentran los valores de rssi para el
angulo a estimar
        indices37_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon1); %filas de la matriz
de datos donde están los datos de cada ángulo/canal/beacon
        indices38_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon1);
        indices39_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon1);
        indices37_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon2);
        indices38_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon2);
        indices39_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon2);

        %dimension minima de los arrays anteriores
        min_index=min([size(indices37_1,1) size(indices38_1,1)
size(indices39_1,1) size(indices37_2,1) size(indices38_2,1)
size(indices39_2,1)]);

        %arrays con los valores de rssi obtenidos para el ángulo a
estimar
        for i=1:min_index

            rssis37_1_dod(i)=dataraw.data(indices37_1(i), 2);
            rssis38_1_dod(i)=dataraw.data(indices38_1(i), 2);
            rssis39_1_dod(i)=dataraw.data(indices39_1(i), 2);
            rssis37_2_dod(i)=dataraw.data(indices37_2(i), 2);
            rssis38_2_dod(i)=dataraw.data(indices38_2(i), 2);
            rssis39_2_dod(i)=dataraw.data(indices39_2(i), 2);

        end

        %monopulse values
        rssi_MV=[
            rssis37_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(1)
            rssis38_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(2)
            rssis39_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(3)
            rssis37_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(4)
            rssis38_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(5)
            rssis39_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(6)
        ];



    end


    %pasar valores a lineal para poder obtener monopulsos
    rssi_inst_lineal= (10.^(rssi_MV/10)).^2;

    MV=[
```

```
        (rssi_inst_lineal(ind_fm_1(1))-
rssi_inst_lineal(ind_fm_1(2)))/(rssi_inst_lineal(ind_fm_1(1))+rssi_ins
t_lineal(ind_fm_1(2)))
        (rssi_inst_lineal(ind_fm_2(1))-
rssi_inst_lineal(ind_fm_2(2)))/(rssi_inst_lineal(ind_fm_2(1))+rssi_ins
t_lineal(ind_fm_2(2)))
        (rssi_inst_lineal(ind_fm_3(1))-
rssi_inst_lineal(ind_fm_3(2)))/(rssi_inst_lineal(ind_fm_3(1))+rssi_ins
t_lineal(ind_fm_3(2)))
        (rssi_inst_lineal(ind_fm_4(1))-
rssi_inst_lineal(ind_fm_4(2)))/(rssi_inst_lineal(ind_fm_4(1))+rssi_ins
t_lineal(ind_fm_4(2)))
        (rssi_inst_lineal(ind_fm_5(1))-
rssi_inst_lineal(ind_fm_5(2)))/(rssi_inst_lineal(ind_fm_5(1))+rssi_ins
t_lineal(ind_fm_5(2)))
        (rssi_inst_lineal(ind_fm_6(1))-
rssi_inst_lineal(ind_fm_6(2)))/(rssi_inst_lineal(ind_fm_6(1))+rssi_ins
t_lineal(ind_fm_6(2)))
        (rssi_inst_lineal(ind_fm_7(1))-
rssi_inst_lineal(ind_fm_7(2)))/(rssi_inst_lineal(ind_fm_7(1))+rssi_ins
t_lineal(ind_fm_7(2)))
    ];

    MV_aux(:,k)=MV;
    %Calculo de la funcion de error dimensiones Nx181
    for j=1:length(fm(:,1))
        error(j,:)=abs(fm(j,:)-MV(j));
    end


    %Calculo del pseudoespectro
    N=length(fm(:,1));
    PS=sqrt (1/N*sum( error(1:N,:).^2 , 1));
    PS= 1./(PS);


    %Eliminar valores infinitos de la funcion PS (valor infinito donde
se
    %encuentra la estimación máxima del angulo)
    PS(isinf(PS))=nan;
    PS(isnan(PS))=max(PS)*2;

    %Normalizacion
    PS_norm=PS/max(PS);

    %Pasamos a escala logaritmica
    PS_norm_db=10*log10(PS_norm);

    %angulo estimado como el máximo del pseudoespectro
    angulo_estimado=find(PS_norm_db==max(PS_norm_db),1,'first')+89-
180+offset_angular;

    error_estimacion(k)=h-angulo_estimado;

    k=k+1;


    plot(eje_ang, PS_norm_db, 'b', 'LineWidth', 2, 'LineStyle', '-')
    grid on
```

```
    xlabel('\theta (degrees)'), ylabel('Error function')
    title(strcat('Pseudoespectro-Normalizado-',' ',antenna_name))
    xlim([eje_ang(1) eje_ang(end)])
    gca.FontSize=20;
    gca.FontName='Times New Roman';
    pause(0.1)
end

%Calculo del error cuadratico medio
rmse=sum(error_estimacion.^2)/length(error_estimacion)

%Calculo del error medio absoluto
mae=sum(abs(error_estimacion))/length(error_estimacion)

plot(eje_ang,zeros(1,181), 'r', 'LineWidth', 4, 'LineStyle', '-'),
hold on
plot(eje_ang, error_estimacion, 'b', 'LineWidth', 1, 'LineStyle', '.')
grid on
xlabel('\theta (degrees)'), ylabel('1/Error function(dB)')
title(strcat('Error-estimacion-Monopulso',' ',antenna_name))
xlim([eje_ang(1) eje_ang(end)])
ylim([min(error_estimacion)-2 max(error_estimacion)+2])
gca.FontSize=20;
gca.FontName='Times New Roman';

%% Estimacion del angulo por distancia euclidea entre valores
monopulso instantenos MEDIOS frente a valores monopulso calibrados
(testbench)

%Eleccion de valores medios (0) o valores instantaneos aleatorios (1)
datos_entrada=0;
%-----------------------------------------------------------------------


maximos_RSSIs=load(ruta_guardado_rssi_norm);
maximos_RSSIs=maximos_RSSIs.maximos_RSSIs;

fm=load(ruta_guardado_mf);
fm=fm.fm;


if(datos_entrada==0)
    espectro_calibracion_db=load(ruta_guardado_rssi);

espectro_calibracion_db=espectro_calibracion_db.espectro_calibracion_d
b;
else
    for i=1:length(dataraw.textdata(:,2))
        MAC_aux1=strjoin(dataraw.textdata(i,2));
        MAC_aux(i,1)=MAC_aux1(1);
    end
end



k=1;
figure
for h=-90+offset_angular:90+offset_angular
%monopulse values
```

```
    ang_detectar_real=h+180-offset_angular;

    if (datos_entrada==0)
        rssi_MV=[
            espectro_calibracion_db(1,(ang_detectar_real-89))-
maximos_RSSIs(1)
            espectro_calibracion_db(2,(ang_detectar_real-89))-
maximos_RSSIs(2)
            espectro_calibracion_db(3,(ang_detectar_real-89))-
maximos_RSSIs(3)
            espectro_calibracion_db(4,(ang_detectar_real-89))-
maximos_RSSIs(4)
            espectro_calibracion_db(5,(ang_detectar_real-89))-
maximos_RSSIs(5)
            espectro_calibracion_db(6,(ang_detectar_real-89))-
maximos_RSSIs(6)
        ];
    else
        %tomamos un valor de rssi en cada puerto canal en el angulo
que se quiere
        %estimar
        %indices donde se encuentran los valores de rssi para el
angulo a estimar
        indices37_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon1); %filas de la matriz
de datos donde están los datos de cada ángulo/canal/beacon
        indices38_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon1);
        indices39_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon1);
        indices37_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon2);
        indices38_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon2);
        indices39_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon2);

        %dimension minima de los arrays anteriores
        min_index=min([size(indices37_1,1) size(indices38_1,1)
size(indices39_1,1) size(indices37_2,1) size(indices38_2,1)
size(indices39_2,1)]);

        %arrays con los valores de rssi obtenidos para el ángulo a
estimar
        for i=1:min_index

            rssis37_1_dod(i)=dataraw.data(indices37_1(i), 2);
            rssis38_1_dod(i)=dataraw.data(indices38_1(i), 2);
            rssis39_1_dod(i)=dataraw.data(indices39_1(i), 2);
            rssis37_2_dod(i)=dataraw.data(indices37_2(i), 2);
            rssis38_2_dod(i)=dataraw.data(indices38_2(i), 2);
            rssis39_2_dod(i)=dataraw.data(indices39_2(i), 2);

        end

        %monopulse values
        rssi_MV=[
            rssis37_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(1)
```

```
            rssis38_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(2)
            rssis39_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(3)
            rssis37_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(4)
            rssis38_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(5)
            rssis39_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(6)
        ];



    end


    %pasar valores a lineal para poder obtener monopulsos
    rssi_inst_lineal= (10.^(rssi_MV/10)).^2;

    MV=[
        (rssi_inst_lineal(ind_fm_1(1))-
rssi_inst_lineal(ind_fm_1(2)))/(rssi_inst_lineal(ind_fm_1(1))+rssi_ins
t_lineal(ind_fm_1(2)))
        (rssi_inst_lineal(ind_fm_2(1))-
rssi_inst_lineal(ind_fm_2(2)))/(rssi_inst_lineal(ind_fm_2(1))+rssi_ins
t_lineal(ind_fm_2(2)))
        (rssi_inst_lineal(ind_fm_3(1))-
rssi_inst_lineal(ind_fm_3(2)))/(rssi_inst_lineal(ind_fm_3(1))+rssi_ins
t_lineal(ind_fm_3(2)))
        (rssi_inst_lineal(ind_fm_4(1))-
rssi_inst_lineal(ind_fm_4(2)))/(rssi_inst_lineal(ind_fm_4(1))+rssi_ins
t_lineal(ind_fm_4(2)))
        (rssi_inst_lineal(ind_fm_5(1))-
rssi_inst_lineal(ind_fm_5(2)))/(rssi_inst_lineal(ind_fm_5(1))+rssi_ins
t_lineal(ind_fm_5(2)))
        (rssi_inst_lineal(ind_fm_6(1))-
rssi_inst_lineal(ind_fm_6(2)))/(rssi_inst_lineal(ind_fm_6(1))+rssi_ins
t_lineal(ind_fm_6(2)))
        (rssi_inst_lineal(ind_fm_7(1))-
rssi_inst_lineal(ind_fm_7(2)))/(rssi_inst_lineal(ind_fm_7(1))+rssi_ins
t_lineal(ind_fm_7(2)))
    ];

    for i=1:length(eje_ang)
        dist(i)=pdist2(MV', fm(:,i)','euclidean');
    end

    dist=1./dist;
    dist(isinf(dist))=nan;
    dist(isnan(dist))=max(dist)*2;
    dist_norm=dist/max(dist);
    %angulo estimado como el máximo del pseudoespectro
    angulo_estimado=find(dist==max(dist),1,'first')+89-
180+offset_angular;

    error_estimacion(k)=h-angulo_estimado;

    k=k+1;
```

```
    plot(eje_ang, dist_norm, 'b', 'LineWidth', 2, 'LineStyle', '-')
    grid on
    xlabel('\theta (degrees)'), ylabel('1/DistEucl')
    title(strcat('1/DistEuc-',' ',antenna_name))
    xlim([eje_ang(1) eje_ang(end)])
    gca.FontSize=20;
    gca.FontName='Times New Roman';
    pause(0.1)
end

%Calculo del error cuadratico medio
rmse=sum(error_estimacion.^2)/length(error_estimacion)

%Calculo del error medio absoluto
mae=sum(abs(error_estimacion))/length(error_estimacion)


plot(eje_ang,zeros(1,181), 'r', 'LineWidth', 4, 'LineStyle', '-'),
hold on
plot(eje_ang, error_estimacion, 'b', 'LineWidth', 1, 'LineStyle', '.')
grid on
xlabel('\theta (degrees)'), ylabel('1/Error function(dB)')
title(strcat('Error-estimacion-Monopulso',' ',antenna_name))
xlim([eje_ang(1) eje_ang(end)])
ylim([min(error_estimacion)-2 max(error_estimacion)+2])
gca.FontSize=20;
gca.FontName='Times New Roman';

%% Modificacion de huella espectral por distancia euclidea entre
vectores RSSI instaneos y datos de calibracion medios

%Eleccion de valores medios (0) o valores instantaneos aleatorios (1)
datos_entrada=0;
%----------------------------------------------------------------------

maximos_RSSIs=load(ruta_guardado_rssi_norm); %Cargo valores de
normalización generados en la sección (2)
maximos_RSSIs=maximos_RSSIs.maximos_RSSIs;


espectro_calibracion_db=load(ruta_guardado_rssi); %Cargo el espectro
de calibración generado en la sección (2)
espectro_calibracion_db=espectro_calibracion_db.espectro_calibracion_d
b;

%Si genero valores instantaneos con combinaciones aletorios
%(datos_entrada=1) cargo los datos de la prueba en cámara aneocoica
if(datos_entrada==1)
    for i=1:length(dataraw.textdata(:,2))
        MAC_aux1=strjoin(dataraw.textdata(i,2));
        MAC_aux(i,1)=MAC_aux1(1);
    end
end


%matriz del espectro calibrado normalizado
for i=1:6
```

```
   espectro_calibracion_norm_db(i,:)=espectro_calibracion_db(i,:)-
maximos_RSSIs(i);
end

k=1;
figure
for h=-90+offset_angular:90+offset_angular
%monopulse values

    ang_detectar_real=h+180-offset_angular;

    if (datos_entrada==0) %Genero un vector de RSSI intantantaneas del
vector de calibración (valores medios, error 0)
        rssi_inst=[
            espectro_calibracion_db(1,(ang_detectar_real-89))-
maximos_RSSIs(1)
            espectro_calibracion_db(2,(ang_detectar_real-89))-
maximos_RSSIs(2)
            espectro_calibracion_db(3,(ang_detectar_real-89))-
maximos_RSSIs(3)
            espectro_calibracion_db(4,(ang_detectar_real-89))-
maximos_RSSIs(4)
            espectro_calibracion_db(5,(ang_detectar_real-89))-
maximos_RSSIs(5)
            espectro_calibracion_db(6,(ang_detectar_real-89))-
maximos_RSSIs(6)
        ];
    else
        %tomamos un valor de rssi en cada puerto canal en el angulo
que se quiere
        %estimar
        %indices donde se encuentran los valores de rssi para el
angulo a estimar
        indices37_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon1); %filas de la matriz
de datos donde están los datos de cada ángulo/canal/beacon
        indices38_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon1);
        indices39_1 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon1);
        indices37_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==37 & MAC_aux(:)==mac_beacon2);
        indices38_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==38 & MAC_aux(:)==mac_beacon2);
        indices39_2 = find(dataraw.data(:,4)==ang_detectar_real &
dataraw.data(:,1)==39 & MAC_aux(:)==mac_beacon2);

        %dimension minima de los arrays anteriores
        min_index=min([size(indices37_1,1) size(indices38_1,1)
size(indices39_1,1) size(indices37_2,1) size(indices38_2,1)
size(indices39_2,1)]);

        %arrays con los valores de rssi obtenidos para el ángulo a
estimar
        for i=1:min_index

            rssis37_1_dod(i)=dataraw.data(indices37_1(i), 2);
            rssis38_1_dod(i)=dataraw.data(indices38_1(i), 2);
            rssis39_1_dod(i)=dataraw.data(indices39_1(i), 2);
            rssis37_2_dod(i)=dataraw.data(indices37_2(i), 2);
```

```
            rssis38_2_dod(i)=dataraw.data(indices38_2(i), 2);
            rssis39_2_dod(i)=dataraw.data(indices39_2(i), 2);

        end

        %monopulse values (vector de RSSI instanteo sobre el que se va
a
        %estimar el ángulo) GUILLE AQUI EMPIEZA LA ESTIMACIÓN, LO DE
ANTES
        %SON PERIFERNALIAS
        rssi_inst=[
            rssis37_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(1)
            rssis38_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(2)
            rssis39_1_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(3)
            rssis37_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(4)
            rssis38_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(5)
            rssis39_2_dod(round(rand*(min_index-1))+1)-
maximos_RSSIs(6)
        ];



    end

    %Correlacion del vector de RSSI instantaneo con el espectro
calibrado
    %normalizado

%     for h=1:length(eje_ang)
%         correlacion_aux=corrcoef(rssi_inst,
espectro_calibracion_norm_db(:,h));
%         correlacion(h)=correlacion_aux(1,2);
%     end
%
%     %para una representacion mas clara
%     corr_representacion=1./(1-correlacion);
%     corr_representacion(isinf(corr_representacion))=nan;
%
corr_representacion(isnan(corr_representacion))=max(corr_representacio
n)*2;
%
corr_representacion=10*log10(corr_representacion/max(corr_representaci
on));
%
%
%     %buscamos donde se encuentra el maximo de la correlacion
%
angulo_estimado=find(correlacion==max(correlacion),1,'first')+89-
180+offset_angular;
%

%Tornero lo hace con Correlación (lo que está arriba implementado),
pero el
%salas me pidió que lo hiciese con distancia euclidea para ver si
cambiaba
```

```
%(es lo mismo pero cambiando corr por pdis2)

    for i=1:length(eje_ang)
        dist(i)=pdist2(rssi_inst',espectro_calibracion_norm_db(:,i)');
    end

    %Aqui lo que se hace es hacer la inversa y sustituir los infinitos
    %(1/0) por un valor alto para que se pueda representas (una
zorreria)
    dist=1./dist;
    dist(isinf(dist))=nan;
    dist(isnan(dist))=max(dist)*2;
    dist_norm=dist/max(dist);

    %angulo estimado como el máximo del pseudoespectro
    angulo_estimado=find(dist==max(dist),1,'first')+89-
180+offset_angular;

    error_estimacion(k)=h-angulo_estimado;

    k=k+1;


    plot(eje_ang, dist_norm, 'b', 'LineWidth', 2, 'LineStyle', '-')
    grid on
    xlabel('\theta (degrees)'), ylabel('1/DistEucl')
    title(strcat('1/DistEuc-',' ',antenna_name))
    xlim([eje_ang(1) eje_ang(end)])
    gca.FontSize=20;
    gca.FontName='Times New Roman';
    pause(0.1)
end

%Calculo del error cuadratico medio
rmse=sum(error_estimacion.^2)/length(error_estimacion)

%Calculo del error medio absoluto
mae=sum(abs(error_estimacion))/length(error_estimacion)


plot(eje_ang,zeros(1,181), 'r', 'LineWidth', 4, 'LineStyle', '-'),
hold on
plot(eje_ang, error_estimacion, 'b', 'LineWidth', 1, 'LineStyle', '.')
grid on
xlabel('\theta (degrees)'), ylabel('Error cometido (°)')
title(strcat('Error-estimacion-Huella-Espectral-',' ',antenna_name))
xlim([eje_ang(1) eje_ang(end)])
ylim([min(error_estimacion)-2 max(error_estimacion)+2])
gca.FontSize=20;
gca.FontName='Times New Roman';
```

# Bibliografía


[1] A. Paul and S. Rho, “Introduction: intelligent vehicular communications,” *Intell. Veh. Networks Commun.*, 2017.

[2] “IEEE_802.11n Constellations and Modulation teaching notes.” [Online]. Available: http://ecee.colorado.edu/~liue/teaching/comm_standards/2015S_IEEE_802.11n/Webpages/constellation.html.

[3] L. Morrón, “Hedy Lamarr, la inventora,” *Los Mundos Brana*, 2015.

[4] B. Willem, “Method of maintaining secrecy in the transmission of wireless telegraphic messages,” *ACM SIGGRAPH Comput. Graph.*, 1932.

[5] N. TESLA, “SYSTEM OF SIGNALING,” 1903.

[6] R. Aguilar, “El WiFi cumple 20 años,” 2019. [Online]. Available: https://www.genbeta.com/a-fondo/wifi-cumple-20-anos-arma-para-luchar-nazis-a-ser-usado-13-000-millones-dispositivos.

[7] “ALOHAnet.” [Online]. Available: https://en.wikipedia.org/wiki/ALOHAnet.

[8] J. J. Yunquera Torres, “DISEÑO DE UNA RED WI-FI PARA LA E.S.I.”

[9] J. A. Pascual, “La historia del WiFi: así empezó todo en los años ochenta,” 2018.

[10] J. L. Gómez Tornero, “ANÁLISIS DE MODOS DE FUGA EN ESTRUCTURAS PLANARES APANTALLADAS LATERALMENTE Y DISEÑO DE NUEVAS ANTENAS ‘LEAKY-WAVE’ EN TECNOLOGÍA HÍBRIDA IMPRESA-APANTALLADA.”

[11] S. I. Matsuzawa, K. Sato, Y. Inoe, and T. Nomura, “Steerable composite right/left-handed leaky wave antenna for automotive radar applications,” *Proc. 36th Eur. Microw. Conf. EuMC 2006*, vol. 0, no. September, pp. 1155–1158, 2006.

[12] C. Rusch, J. Schafer, H. Gulan, P. Pahl, and T. Zwick, “Holographic mmW-antennas with TE0 and TM0 surface wave launchers for frequency-scanning FMCW-radars,” *IEEE Trans. Antennas Propag.*, vol. 63, no. 4, pp. 1603–1613, 2015.

[13] D. A. Schneider, M. Rosch, A. Tessmann, and T. Zwick, “A Low-Loss W-Band Frequency-Scanning Antenna for Wideband Multichannel Radar Applications,” *IEEE Antennas Wirel. Propag. Lett.*, vol. 18, no. 4, pp. 806–810, 2019.

[14] M. Ettorre, R. Sauleau, and L. Le Coq, “Multi-beam multi-layer leaky-wave SIW pillbox antenna for millimeter-wave applications,” *IEEE Trans. Antennas Propag.*, vol. 59, no. 4, pp. 1093–1100, 2011.

[15] Y. J. Cheng, W. Hong, K. Wu, and Y. Fan, “Millimeter-wave substrate integrated waveguide long slot leaky-wave antennas and two-dimensional multibeam applications,” *IEEE Trans. Antennas Propag.*, vol. 59, no. 1, pp. 40–47, 2011.

[16] P. Popovski, H. Yomo, and R. Prasad, “Strategies for adaptive frequency hopping in the unlicensed bands,” *IEEE Wirel. Commun.*, vol. 13, no. 6, pp. 60–67, 2006.

[17] A. J. Martínez-Ros, J. L. Gómez-Tornero, and G. Goussetis, "Planar Leaky-Wave Antenna with Flexible Control of the Complex Propagation Constant," *IEEE Trans. Antennas Propag.*, vol. 60, no. 3, pp. 1625–1630, 2012.

[18] A. J. Martínez-Ros, J. L. Gómez-Tornero, and F. Quesada-Pereira, "Efficient Analysis and Design of Novel SIW Leaky-Wave Antenna," *IEEE Antennas Wirel. Propagat. Lett.*, vol. 12, pp. 496–499, 2013.

[19] M. Poveda-García, D. Cañete-Rebenaque, and J. L. Gómez-Tornero, "Frequency-Scanned Monopulse Pattern Synthesis Using Leaky-Wave Antennas for Enhanced Power-Based Direction-of-Arrival Estimation," *IEEE Trans. Antennas Propag.*, vol. 67, no. 11, pp. 7071–7086, 2019.

[20] A. Gil-Martínez, M. Poveda-García, and J. L. Gómez-Tornero, "Direct Synthesis of Frequency-Scanned Monopulse Half-Width Microstrip Leaky-Wave Antennas," in *14TH EUROPEAN CONFERENCE ON ANTENNAS AND PROPAGATION (EuCAP 2020)*, 2020.

[21] M. Poveda-García, J. Oliva-Sánchez, R. Sanchez-Iborra, D. Cañete-Rebenaque, and J. L. Gómez-Tornero, "Dynamic Wireless Power Transfer for Cost-Effective Wireless Sensor Networks using Frequency-Scanned Beaming," *IEEE Access J. Spec. Sect. Wirelessly Powered Networks*, vol. 7, pp. 8081–8094, 2019.

[22] J. L. Gomez-Tornero and M. Poveda-Garcia, "Beaming Efficiency of 1-D Frequency-Scanned Based Radiative WPT System for Wireless Sensor Networks," in *2019 IEEE Wireless Power Transfer Conference (WPTC)*, 2019, pp. 338–342.

[23] M. Poveda-García, A. Gomez-Alcaraz, A. Gil-Martinez, A. S. Canete-Rebenaque, D. Martinez-Sala, and J. L. Gómez-Tornero, "Frequency-Scanned Active Monopulse Radar based on Bluetooth Low Energy Devices using an Array of Two Planar Leaky-Wave Antennas," in *International Conference on Electromagnetics in Advanced Applications (ICEAA 2019)*, 2019.

[24] M. Poveda-García, A. Gómez-Alcaraz, D. Cañete-Rebenaque, A. S. Martínez-Sala, and J. L. Gómez-Tornero, "RSSI-Based Direction-of-Departure Estimation in Bluetooth Low Energy Using an Array of Frequency-Steered Leaky-Wave Antennas," *IEEE Access J. Spec. Sect. Emerg. Trends, Issues Challenges Array Signal Process. Its Appl. Smart City*, vol. 8, pp. 9380–9394, 2020.

[25] M. Poveda-García, A. Gil-Martínez, and J. L. Gómez-Tornero, "Frequency-Scanned Focused Leaky-Wave Antennas for Direction-of-Arrival Detection in Proximity BLE Sensing Applications," in *14TH EUROPEAN CONFERENCE ON ANTENNAS AND PROPAGATION (EuCAP 2020)*, 2020.

[26] J. L. Gómez-Tornero, D. Cañete-Rebenaque, J. A. López-Pastor, and A. Martínez-Salas, "Hybrid Analog-Digital Processing System for Amplitude-Monopulse RSSI-based MiMo WiFi Direction-of-Arrival Estimation," *IEEE J. Sel. Top. Signal Process. Spec. Issue Hybrid Analog - Digit. Signal Process. Hardware-Efficient Large Scale Antenna Arrays*, vol. 12, pp. 529–540, 2018.

[27] J. A. López-Pastor, A. Gómez-Alcaraz, D. Cañete-Rebenaque, A. S. Martínez-Sala, and J. L. Gómez-Tornero, "Near-Field Monopulse DoA Estimation for Angle-Sensitive Proximity WiFi Readers," *IEEE Access J. Spec. Sect. Emerg.*

*Trends, Issues Challenges Array Signal Process. Its Appl. Smart City*, vol. 7, pp. 88450–88460, 2019.

[28] OpenWrt, “OpenWrt - About OpenWrt.” [Online]. Available: https://openwrt.org/releases/19.07/notes-19.07.2.

[29] Openwrt, “Wireless configuration Openwrt.” [Online]. Available: %0A.

[30] I. Standard, “INTERNATIONAL STANDARD ISO / IEC / IEEE 802.11,” vol. 2012, 2012.

[31] A. F. García, C. Gómez, T. Sánchez, A. D. Redondo, L. Betancur, and R. C. Hincapié, “Algoritmos de Radiolocalización basados en ToA, TDoA y AoA,” *Ing. y Región*, 2015.

[32] A. G. Alcaraz, “Diseno de un beacon avanzado para estimación del ángulo de salida en sistemas BLE, basado en un array de antenas leaky-wave monopulso escaneadas en frecuencia,” 2019.