Escuela Politécnica Superior de Linares

Grado en Ingeniería de Tecnologías de Telecomunicación

UNIVERSIDAD DE JAÉN
Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CLASIFICADOR DE MODULACIÓN BASADO EN APRENDIZAJE AUTOMÁTICO

Alumno: **Raúl Maldonado Cerezo**

Tutor: Prof. D. Pedro Jesús Reche López
Depto.: Ingeniería de Telecomunicación

Septiembre, 2022

Universidad de Jaén

Escuela Politécnica Superior de Linares

Trabajo Fin de Grado

CLASIFICADOR DE MODULACIÓN BASADO EN APRENDIZAJE AUTOMÁTICO

Alumno: Raúl Maldonado Cerezo

Tutor: Prof. D. Pedro Jesús Reche López

Depto.: Ingeniería de Telecomunicación

Septiembre, 2022

Firma del autor:

MALDONADO CEREZO RAUL - 77036298Y

Firmado digitalmente por MALDONADO CEREZO RAUL - 77036298Y
Fecha: 2022.09.05 12:41:23 +02'00'

Firma del tutor:

RECHE LOPEZ PEDRO JESUS - 26226792F

Firmado digitalmente por RECHE LOPEZ PEDRO JESUS - 26226792F
Fecha: 2022.09.05 13:54:52 +02'00'

AGRADECIMIENTOS

A pesar de haber realizado un bachillerato cuya modalidad (Humanidades y Ciencias Sociales) no se correspondía en absoluto al camino recomendado para acceder a esta carrera, finalizar la misma en cuatro años ha sido fruto de un gran esfuerzo y perseverancia, no obstante, esto no hubiese sido posible sin la ayuda conjunta o aislada de determinadas personas a las cuales van dirigidos los agradecimientos.

En primer lugar, me gustaría agradecer a todos los profesores que he tenido a lo largo de estos cuatro años de carrera por haber fomentado de una manera u otra el desarrollo de mi curiosidad, haciendo posible que determinados temas que a simple vista pareciesen aburridos o inservibles resultarán ser de lo más interesantes y útiles además de prácticos.

A mi tutor, Pedro Jesús Reche López, me gustaría agradecerle la disponibilidad y dedicación a la hora de explicarme cualquier cosa relacionada con el trabajo, así como sus buenos consejos a la hora de redactar la memoria del mismo.

También me gustaría agradecer a la Universidad de Jaén todos los acuerdos con diferentes instituciones, revistas y organizaciones científicas, ya que, me han sido de gran utilidad a la hora de consultar artículos y libros, los cuales de otra forma hubiese resultado de gran dificultad el acceso a los mismos.

Por último, me gustaría agradecer a mi familia y a mis compañeros el tiempo dedicado en mí.

Cada una de las personas que se sientan identificadas con estos agradecimientos ha de saber que, en un momento u otro del transcurso recorrido, me ha dado fuerzas para continuar adelante.

RESUMEN

La transmisión de cualquier señal inalámbrica, (desde las señales emitidas por un sensor hasta las señales emitidas por estaciones base) requieren una modulación de la misma antes de ser transmitida.

La modulación de la señal por parte del bloque transmisor, requiere una demodulación por parte del bloque receptor para recuperar la señal original. Para que el bloque receptor pueda demodular correctamente la señal recibida, debe saber a priori el tipo de modulación usada por el transmisor.

Aquí es donde entra en juego la clasificación automática de modulaciones (*Automatic Modulation Classification - AMC*), cuyo objetivo es identificar el tipo de modulación que está empleando una señal entrante al receptor en los sistemas de comunicación inalámbricos, donde el determinar el tipo de modulación de la señal detectada es el paso intermedio entre la detección de la señal y la demodulación.

En el presente proyecto, se han tenido en cuenta modulaciones analógicas, concretamente las siguientes modulaciones:

- Modulación de Amplitud (*Amplitude Modulation - AM*)
- Modulación de Doble Banda Lateral con Portadora Suprimida (*Double Side Band Suprimied Carrier - DSB-SC*)
- Modulación de Banda Lateral Inferior (*Lower Side Band - LSB*)
- Modulación de Banda Lateral Superior (*Upper Side Band - USB*)
- Modulación de Frecuencia (*Frecuency Modulation - FM*)

El clasificador automático de modulaciones ha sido desarrollado mediante algoritmos supervisados, específicamente basado en redes neuronales artificiales (*Artificial Neural Network - ANN*), mediante la extracción de características clave de las muestras en fase y cuadratura de la envolvente compleja correspondientes a las modulaciones anteriores.

Las fases de entrenamiento, validación y testeo han sido realizadas mediante la introducción a la red neuronal de las características clave comentadas anteriormente para cada tipo de modulación a clasificar, siendo estas características calculadas en una fase de preprocesado, además, se añaden a dichas componentes en fase y cuadratura ruido blanco Gaussiano (*Additive White Gaussian Noise - AWGN*) en su equivalente paso bajo para simular el efecto del ruido en el canal de transmisión, ya que, estas tres fases, se han realizado con señales que no van sobre el '*aire*', para, posteriormente mediante el uso de dos Radios

Definidos por Software (*Software Defined Radio - SDR*) bladeRF x40, transmitir y recibir señales moduladas para su clasificación.

Se han obtenido unos resultados de clasificación exitosa del 94.8% para señales que no van sobre el aire frente a un 75.74% de clasificaciones exitosas para señales transmitidas y recibidas mediante SDR.

Palabras clave: clasificación automática de modulación, red neuronal artificial, características clave, componentes en fase y cuadratura, radio definida por software, modulación de amplitud, modulación de frecuencia, modulación de doble banda lateral con portadora suprimida, modulación de banda lateral inferior, modulación de banda lateral superior.

LISTA DE ACRÓNIMOS

ADC	*Analog-to-Digital Converter*
AI	*Artificial Intelligence*
AM	*Amplitude Modulation*
AMC	*Automatic Modulation Classification*
ANN	*Artificial Neural Network*
AWGN	*Additive White Gaussian Noise*
BB	*Base Band*
BW	*Band Width*
DAC	*Digital-to-Analog Converter*
DFT	*Discrete Fourier Transform*
DP	*Deep Learning*
DSB-SC	*Double Side Band – Suprimied Carrier*
FC	*Frecuency Carrier*
FFT	*Fast Fourier Transform*
FT	*Fourier Transform*
GD	*Gradient Descent*
HL	*Hidden Layer*
HT	*Hilbert Transform*
IQ	*In-phase and Quadrature*
LNA	*Low-Noise Amplifier*
LPF	*Low Pass Filter*
LR	*Learning Rate*
LSB	*Lower Side Band*
ML	*Machine Learning*
MSE	*Mean Squared Error*
PA	*Power Amplifier*
PLL	*Phase-Locked Loop*
RF	*Radiofrecuency*
SDR	*Software Defined Radio*
SNR	*Signal-to-Noise Ratio*
USB	*Upper Side Band*

ÍNDICE

LISTA DE FIGURAS 5

LISTA DE TABLAS 9

1 INTRODUCCIÓN Y OBJETIVOS 11

1.1 Motivación 11

1.2 Estado del arte 12

1.3 Objetivos 14

1.4 Estructura de la memoria 14

2 PLANIFICACIÓN Y ESTIMACIÓN DE COSTES 17

2.1 Fases en el desarrollo del proyecto 17

2.1.1 Fase 1: Establecimiento de los objetivos 17

2.1.2 Fase 2: Toma de contacto con los dispositivos SDR y Software para su desarrollo 17

2.1.3 Fase 3: Estudio de Modulaciones 17

2.1.4 Fase 4: Estudio de Algoritmos de aprendizaje supervisado 17

2.1.5 Fase 5: Creación, entrenamiento, validación, test y evaluación de rendimiento del algoritmo seleccionado 17

2.1.6 Fase 6: Documentación 18

2.1.7 Fase 7: Redacción de la memoria 18

2.2 Diagrama de Gantt y tiempo estimado 18

2.3 Recursos 19

2.4 Estimación de costes 20

2.5 Presupuesto total 21

3 MATERIALES Y MÉTODOS 23

3.1 Sistema de Radiocomunicaciones Clásico 23

3.2 Radios Definidas por Software 24

3.2.1 Plataformas SDR comerciales y elección 26

3.2.2 Introducción al bladeRF 29
3.2.2 Modulador en fase y cuadratura 33
3.2.3 Demodulador en fase y cuadratura 35
3.2.4 ¿Por qué debemos trabajar con señales complejas IQ en el ordenador? ……...36
3.2.5 Software elegido para el desarrollo del SDR..... 39
3.3 Técnicas de Modulaciones Analógicas..... 41
3.3.1 Modulaciones lineales 41
3.3.2 Modulaciones angulares 50
3.4 Aprendizaje Automático 54
3.4.1 Algoritmos de Aprendizaje Supervisado 56
3.5 Redes Neuronales Artificiales..... 58
3.5.1 Introducción e Inspiración Biológica de la Red Neuronal Artificial.... 58
3.5.2 Características y aplicaciones 60
3.5.3 Funcionamiento de las ANN 62
3.5.4 Software para el desarrollo de la red neuronal artificial y elección 66
3.6 Herramientas Software..... 67
3.6.1 Matlab 67
3.6.2 GNU radio 68
3.7 Tipo de Red Neuronal implementada y características..... 69
3.8 Algoritmos para el entrenamiento empleados..... 71
3.8.1 Descenso del Gradiente 71
3.8.2 Momentum y Ratio de Aprendizaje Adaptativo..... 77
3.8.3 BackPropagation..... 81
3.9 Desarrollo e implementación del AMC 87
3.9.1 Características clave para la clasificación de señales analógicas 87
3.9.2 Generación del Training Data Set..... 92

3.9.3 Entrenamiento y aprendizaje de la red neuronal, validación y testeo... 97

4 RESULTADOS Y DISCUSIÓN..... 101

4.1 Resultados en la clasificación de señales que no van sobre el aire..... 101

4.2 Testeo con SDR 103

4.2.1 Proceso de recepción y clasificación de señales que van sobre el aire103

4.2.2 Valores de los bloques SDR usados en transmisión y recepción 107

4.2.3 Problemas con la transmisión y recepción de señales mediante SDR 107

4.2.4 ¿A qué se debe la aparición de esa señal no deseada desplazada a -f Hz?, ¿Soluciones? ……112

4.2.5 Resultados de la clasificación de señales moduladas transmitidas y recibidas mediante SDRs 119

4.3 Conclusiones..... 123

4.3.1 Conclusión científica 123

4.3.2 Conclusión académica 123

4.4 Líneas futuras..... 124

5 ANEXOS 125

5.1 Anexo I: Instalación de librerías, controladores y soportes para el bladeRF ……..125

5.2 Anexo II: Esquemas implementados en GNU radio..... 132

5.2.1 Esquemas implementados en transmisión 132

5.2.2 Esquemas implementados en recepción 137

5.3 Anexo III: Scripts y Funciones desarrolladas en Matlab 140

5.3.1 Diagrama de flujo del procedimiento completo llevado a cabo para la clasificación de señales que no van sobre el aire 141

5.3.2 Diagrama de flujo del procedimiento completo llevado a cabo para la clasificación de señales recibidas por el SDR..... 142

5.3.3 Descripción de los Scripts y funciones desarrolladas en Matlab..... 142

6 BIBLIOGRAFÍA..... 151

LISTA DE FIGURAS

Figura 1.1 Sistema de Comunicación con Modulación Adaptativa [1] 11
Figura 2.1 Diagrama de Gantt 18
Figura 3.1 Dispositivo SDR transceptor [9] 24
Figura 3.2 Arquitectura SDR [10] 25
Figura 3.3 Diagrama de bloques de plataforma SDR genérica [10] 26
Figura 3.4 Dispositivo RTL-SDR 27
Figura 3.5 Dispositivo HackRF One 27
Figura 3.6 Dispositivo bladeRF 28
Figura 3.7 Dispositivo bladeRF x40 30
Figura 3.8 Arquitectura interna del bladeRF x40 [13] 30
Figura 3.9 Arquitectura interna del LMS6002D [13] 33
Figura 3.10 Modulador IQ [10] 33
Figura 3.11 Demodulador IQ [10] 35
Figura 3.12 Proceso de muestreo de una señal continua en el tiempo [16] 37
Figura 3.13 Esquemático completo del procesado de la señal banda base 41
Figura 3.14 Representación temporal de la Señal Moduladora, Portadora y Modulada en AM cuando m≤1 43
Figura 3.15 Representación espectral de la Señal Moduladora, Portadora y Modulada en AM cuando m≤1 43
Figura 3.16 Representación temporal de la Señal Moduladora, Portadora y Modulada en AM cuando m≥1 44
Figura 3.17 Representación espectral de la Señal Moduladora, Portadora y Modulada en AM cuando m≥1 45
Figura 3.18 Representación temporal de la Señal Moduladora, Portadora y Modulada en DSB-SC 46
Figura 3.19 Representación espectral de la Señal Moduladora, Portadora y Modulada en DSB-SC 47
Figura 3.20 Espectro en frecuencia de la envolvente compleja VY(f) de la señal LSB 48
Figura 3.21 Espectro en frecuencia de la envolvente compleja VY(f) de la señal USB 49

Figura 3.22 Forma de onda temporal de la señal modulada en FM con un valor de β elevado 52
Figura 3.23 Espectro FM con valor de β elevado 52
Figura 3.24 Forma de onda temporal de la señal modulada en FM con un valor de β pequeño 53
Figura 3.25 Espectro FM con valor de β pequeño 53
Figura 3.26 Esquemático de Funcionamiento del Aprendizaje Supervisado 55
Figura 3.27 Esquemático de Funcionamiento del Aprendizaje No Supervisado .. 56
Figura 3.28 Proceso de Desarrollo del Clasificador Supervisado a Implementar .. 57
Figura 3.29 Modelo fisiológico de una neurona [29] 59
Figura 3.30 Neurona Artificial o Perceptrón Simple 60
Figura 3.31 Modelo de Regresión Lineal Simple 63
Figura 3.32 Icono de Matlab 68
Figura 3.33 Interfaz de GNU radio y ejemplo de simulación 69
Figura 3.34 Red Neuronal Artificial FeedForward 69
Figura 3.35 FeedForward en una capa 70
Figura 3.36 Características de la red neuronal a implementar 70
Figura 3.37 Tipos de Funciones de Coste 72
Figura 3.38 Punto de inflexión (Pendiente Nula) 72
Figura 3.39 Derivada parcial del error con cada parámetro 74
Figura 3.40 ± Vector Gradiente 74
Figura 3.41 Convergencia del error cuando el valor del LR es elevado 76
Figura 3.42 Convergencia del error cuando es usado un valor de LR pequeño 76
Figura 3.43 Convergencia del error cuando es usado un valor óptimo del LR 77
Figura 3.44 Convergencia del error cuando es usado momentum para optimizar el descenso del gradiente, usando un valor óptimo del LR 79
Figura 3.45 Convergencia del error cuando es usado momentum y LR adaptativa como algoritmos optimizadores del descenso del gradiente 80
Figura 3.46 Efecto de modificar un parámetro en las conexiones de la ANN 82
Figura 3.47 Neurona de la última capa 83
Figura 3.48 Ejemplo de generación de señal AM variando el parámetro m 94
Figura 3.49 Ejemplo de almacenamiento de valores de las características 95
Figura 3.50 Conjunto de entradas usado para el entrenamiento de la red neuronal 96
Figura 3.51 Targets de la red neuronal 96

Figura 3.52 Data Set de entrenamiento de la red neuronal 97
Figura 3.53 División del Training Data Set 99
Figura 3.54 Entrenamiento de la ANN 99
Figura 3.55 Proceso de entrenamiento, validación y testeo de la ANN 100
Figura 4.1 Matriz de Confusión obtenida para señales que no viajan sobre el aire 102
Figura 4.2 Proceso de clasificación de señales provenientes del aire 105
Figura 4.3 Sistema físico (TX-RX) 106
Figura 4.4 Problema 1 en la transmisión y recepción de señales con los SDRs 108
Figura 4.5 Problema 2 en la transmisión y recepción señales con los SDRs (TX-RX) 109
Figura 4.6 Problema 3 en la transmisión y recepción señales con los SDRs (TX-RX) 110
Figura 4.7 Resultado de la clasificación de señal AM transmitida y recibida mediante SDR 111
Figura 4.8 Proceso de calibración manual del SDR receptor 116
Figura 4.9 Proceso de calibración_1 116
Figura 4.10 Proceso de calibración_2 117
Figura 4.11 Efecto en las variaciones de temperatura de los SDRs sobre el espectro de señal recibido 118
Figura 4.12 Resultado de clasificación de señal DSB-SC mezclada con un tono no deseado en sus componentes en fase y Cuadratura 119
Figura 4.13 Resultado de clasificación de señal USB transmitida y recibida mediante SDR 121
Figura 4.14 Resultado de clasificación de señal FM transmitida y recibida mediante SDR 122
Figura 5.1 BladeRF Setup Wizard_1 125
Figura 5.2 BladeRF Setup Wizard_2 126
Figura 5.3 BladeRF Setup Wizard_3 126
Figura 5.4 BladeRF Setup Wizard_4 127
Figura 5.5 BladeRF Setup Wizard_5_6 127
Figura 5.6 BladeRF Setup Wizard_7 128
Figura 5.7 BladeRF Setup Wizard_8 128
Figura 5.8 BladeRF Setup Wizard_9 129

Figura 5.9 BladeRF Setup Problem_1 129
Figura 5.10 Testeo Básico del SDR 130
Figura 5.11 BladeRF Setup Solution2Problem_2_1 131
Figura 5.12 BladeRF Setup Solution2Problem_2_2 131
Figura 5.13 Esquema transmisor AM 132
Figura 5.14 Propiedades del bloque SDR transmisor 134
Figura 5.15 Esquema transmisor DSB-SC 135
Figura 5.16 Esquema transmisor LSB 135
Figura 5.17 Esquema transmisor USB 136
Figura 5.18 Esquema transmisor FM 136
Figura 5.19 Esquema receptor AM 137
Figura 5.20 Propiedades del bloque SDR receptor 137
Figura 5.21 Esquema receptor DSB-SC 138
Figura 5.22 Esquema receptor LSB 139
Figura 5.23 Esquema receptor USB 139
Figura 5.24 Esquema receptor FM 140
Figura 5.25 Diagrama de flujo sobre el procedimiento llevado a cabo sobre señales que no van sobre el aire 141
Figura 5.26 Diagrama de flujo sobre el procedimiento llevado a cabo sobre señales que van sobre el aire 142

LISTA DE TABLAS

Tabla 2.1 Estimación total para cada fase del proyecto 18
Tabla 2.2 Coste total de recursos humanos 20
Tabla 2.3 Coste total de recursos hardware 20
Tabla 2.4 Coste total de recursos software 21
Tabla 2.5 Presupuesto total del TFG 21
Tabla 3.1 Comparación de Plataformas SDR 28
Tabla 3.2 Especificaciones bladeRF x40 29
Tabla 3.3 Rangos de funcionamiento del transceptor LMS6002D 32
Tabla 3.4 Comparación de los Softwares usados 40
Tabla 3.5 Funciones de Activación 65
Tabla 3.6 Diferencias entre el Software para el desarrollo de la red neuronal 66
Tabla 3.7 Parámetros a variar en la generación de modulaciones 93
Tabla 3.8 Valores por defecto para los parámetros 95
Tabla 3.9 Asignación de las salidas para las diferentes entradas a la red neuronal. 96
Tabla 3.10 Ejemplo del Training Data Set 97
Tabla 3.11 Valores de los parámetros usados para el entrenamiento 97
Tabla 4.1 Valores usados en los bloques SDR 107
Tabla 4.2 Tasa total de aciertos y fallos para señales transmitidas y recibidas mediante SDR 122

1 INTRODUCCIÓN Y OBJETIVOS

A continuación, se presentarán las motivaciones que han dado lugar a la realización del proyecto, así como la situación actual en el ámbito investigado.

Por último, se comentará tanto el objetivo principal del proyecto como los distintos sub objetivos que han debido de realizarse para la consecución del trabajo, además, se presentará una breve estructuración de la memoria.

1.1 Motivación

El gran avance de la tecnología, ha permitido el desarrollo y la implementación de sistemas de comunicaciones inalámbricas con modulación adaptativa (Figura 1.1), logrando una gestión eficiente del espectro, es decir, las señales de radio pueden codificarse mediante diferentes formatos de modulación predefinidos, dependiendo entre otras cosas de las condiciones del canal de propagación.

La identificación automática de los tipos de modulación de las señales que se reciben, permite al receptor una correcta demodulación, por tanto, la identificación de la modulación puede llegar a ser una prioridad en comunicaciones basadas en radios definidas por software, donde gran parte del hardware tradicional es implementado mediante software.

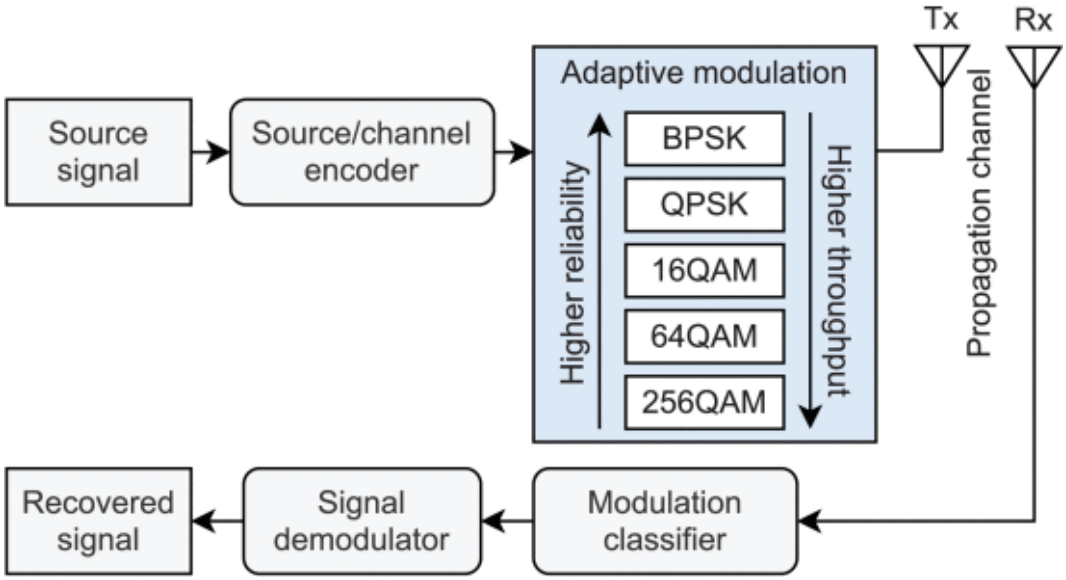


Figura 1.1 Sistema de Comunicación con Modulación Adaptativa [1]

El AMC no solo tiene aplicación en ámbitos civiles con esa adaptación del enlace como se ha visto, sino que también tiene aplicación en ámbitos militares, donde no hay ninguna información disponible sobre la señal en una determinada banda de frecuencias,

generando la necesidad de que el receptor identifique completamente a ciegas la modulación de la señal recibida.

1.2 Estado del arte

A lo largo de estos últimos años se ha experimentado un crecimiento exponencial de la Inteligencia Artificial (*Artificial Intelligence - AI*) y de las diferentes tecnologías que la conforman, este crecimiento en gran parte viene ligado al gran desarrollo y evolución de la tecnología en general, traduciéndose esto al ámbito investigado en una enorme variedad de nuevas técnicas que permiten la clasificación de modulación de una forma mucho más eficiente y exitosa. Y es que, gracias a la Inteligencia Artificial se pueden llegar a desarrollar de forma eficiente soluciones a problemas complejos.

La gran variedad de tecnologías asociadas a la Inteligencia Artificial, ha permitido que a lo largo del tiempo se realicen múltiples investigaciones en el área del reconocimiento automático de modulación.

A continuación, se presentarán una serie de investigaciones en dicha área, estas irán referenciadas de forma ascendente en función del año de publicación, permitiendo observar la evolución en la eficiencia de los clasificadores de modulación conforme han pasado los años, viendo diferentes técnicas de clasificación y observando diferentes resultados en el uso de diferentes algoritmos, para, finalmente, concluir en el gran desarrollo de la tecnología que se ha vivido y se sigue viviendo permitiendo obtener mejores y más eficientes resultados de clasificación, posibilitando la clasificación incluso de modulaciones digitales con constelaciones densas, también, mediante dichas comparaciones se obtendrá el algoritmo que permite obtener mejores resultados en la clasificación.

Azzouz y Nandi, fueron de los primeros en proponer algoritmos de reconocimiento de modulaciones, publicando las investigaciones realizadas en el 1995, específicamente, desarrollaron dos clasificadores de modulación, uno de ellos para reconocer modulaciones digitales [2] y otro clasificador para modulaciones analógicas [3], ambos clasificadores se basaban en árboles de decisión como algoritmo de aprendizaje supervisado, donde las señales a clasificar eran simuladas por ordenador, limitadas en banda y corrompidas por ruido Gaussiano limitado en banda.

Los resultados de la clasificación fueron totalmente exitosos consiguiendo para ambos tipos de modulaciones tanto analógicas como digitales una correcta clasificación en el 90% de los casos con una relación señal a ruido (*Signal-to-Noise Ratio - SNR*) de 10 dB.

Se observa que en el tipo de modulaciones digitales a clasificar no han incluido modulaciones con constelaciones densas, es decir, con muchos puntos, esto es se debe a que el rendimiento del clasificador se ve afectado por el número de muestras de las que se extraen las características.

Estudios posteriores publicados en 1997 por los mismos autores, demuestran que las redes neuronales como algoritmos supervisados ofrecen incluso mejores resultados que los árboles de decisión, obteniendo tasas de éxito del 96% [4] para las mismas modulaciones y mismo SNR usados en su artículo de 1995. A diferencia del artículo de 1995 donde se desarrollaban dos clasificadores independientes entre sí para la clasificación de dos tipos de modulación, este estudio en cuestión muestra como mediante un único bloque del algoritmo ANN formado por tres sub redes neuronales dependientes entre sí, se consigue la clasificación de ambos tipos de modulación.

El entrenamiento de la red neuronal requirió de 250.000 épocas[1] hasta ajustar sus parámetros de la forma más óptima, lo que podría traducirse en minutos e incluso horas de entrenamiento.

Artículos más recientes publicados en 2016, siguen implementado redes neuronales como algoritmo clasificador, aunque, con más capas ocultas, concretamente en el siguiente artículo [5] se desarrolla una red neuronal profunda capaz de distinguir 7 tipos de modulaciones digitales, entre las que se encuentran modulaciones con constelaciones más densas que las observadas en los artículos anteriores como 64QAM.

Además, se observa un número máximo de épocas de entrenamiento de simplemente 15.000, de las cuales aproximadamente unas 5.000 han sido usadas para el entrenamiento de la red neuronal o lo que es lo mismo, el entrenamiento ha sido realizado en cuestión de pocos minutos e incluso segundos.

En este caso, las señales moduladas generadas en simulación, no solo han sido limitadas en banda a las cuales se ha añadido ruido blanco Gaussiano, sino que también se ha añadido otro efecto del canal conocido como Rician Fading, lo que ha permitido una correcta clasificación de la modulación de aproximadamente el 99,99% de veces para diferentes valores de SNR y diferentes valores del Rician Fading.

Por último, en investigaciones más cercanas a la actualidad, publicadas entre 2019 y 2020 [6], [7] en el que se clasifican modulaciones digitales, se observan nuevos enfoques en la clasificación de modulación, basados en su mayoría, en métodos de clasificación de

[1] Ciclo completo de los datos de entrenamiento a través de la red neuronal.

modulación basada en imágenes, en estos nuevos acercamientos se usan redes neuronales convolucionales (*Convolutional Neuronal Networks - CNN*) como algoritmo para la clasificación de las imágenes, evitando de esta manera la extracción de características de las señales moduladas como se había hecho hasta ahora con los demás algoritmos vistos en los diferentes artículos previos. Aunque este método basado en imágenes de constelación, no puede distinguir entre mismas modulaciones con diferentes puntos, es decir, se imposibilita la distinción entre modulaciones como 16QAM y 64QAM, obteniendo rendimientos del 83,3%.

No obstante, en [7], se hace un nuevo acercamiento, realizando una mezcla entre clasificadoras basados en imágenes y clasificadores basados en extracción de características, obteniendo un clasificador basado en imágenes de características, obteniendo resultados de clasificación del 99%.

1.3 Objetivos

El objetivo principal del trabajo fin de grado es el desarrollo e implementación de un clasificador automático de modulaciones mediante redes neuronales. Para la consecución de dicho objetivo principal, ha sido necesaria la realización de una serie de objetivos específicos previos, los cuales se muestran a continuación:

- Estudio de la tecnología SDR.
- Estudio y evaluación de las herramientas software que permiten programar los SDRs.
- Elección de las modulaciones a clasificar.
- Estudio y análisis de las modulaciones a clasificar.
- Estudio y elección del algoritmo de aprendizaje supervisado a emplear.
- Estudio, análisis y elección de las características clave a calcular.
- Generación de conjuntos de entrenamiento, validación y test.
- Evaluación del rendimiento del AMC implementado.

1.4 Estructura de la memoria

En este sub apartado, se detallará la estructura de esta memoria, compuesta por un total de 6 capítulos, dos de ellos incluyen anexos y bibliografía que tienen como finalidad facilitar al lector de un conocimiento adicional, así como el poder investigar por cuenta ajena, si se desea, algunos procedimientos seguidos en la elaboración del trabajo, ambos capítulos

no serán comentados. A continuación, se describe de forma resumida lo que se podrá contemplar en cada capítulo.

- **Capítulo 1**: Introducción y objetivos. En este capítulo se sitúa al lector en un contexto adecuado sobre el trabajo realizado. Tras una breve introducción, se definen una serie de motivaciones que han dado lugar a la realización de este trabajo, posteriormente, se ha realizado un análisis de la situación actual en el ámbito investigado, para, finalmente concluir detallando una serie de objetivos a alcanzar para la consecución del proyecto.
- **Capítulo 2**: Planificación y estimación de costes. En este capítulo se detalla la planificación del proyecto mediante una serie de fases temporales que involucran en su totalidad o en su conjunto el comienzo, desarrollo y fin del proyecto. Posteriormente se realiza una estimación del coste total vinculado al desarrollo del proyecto.
- **Capítulo 3**: Materiales y métodos. El capítulo 4 muestra las diferentes herramientas que se han ido seleccionando para la elaboración del proyecto, así como el ¿Por qué han sido seleccionadas? Para concluir detallando el proceso de elaboración y desarrollo del AMC.
- **Capítulo 4**: Resultados y discusión. Bajo este capítulo se ofrecerán los resultados obtenidos con el AMC desarrollado, se comentarán una serie de problemas y soluciones, finalmente se sacan una serie de conclusiones y se recogerán una serie de posibles mejoras para futuras versiones del AMC desarrollado.

2 PLANIFICACIÓN Y ESTIMACIÓN DE COSTES

2.1 Fases en el desarrollo del proyecto

El proyecto ha sido desarrollado en las siguientes fases:

2.1.1 Fase 1: Establecimiento de los objetivos

Como fase inicial del proyecto, se parte de la premisa '¿Qué se pretende conseguir?', en base a esta premisa se hace un establecimiento general de objetivos y, por tanto, se definen las siguientes fases.

2.1.2 Fase 2: Toma de contacto con los dispositivos SDR y Software para su desarrollo

Adquiridos los dos dispositivos SDR de los cuales se han hecho uso en el proyecto, funcionando uno como transmisor y otro como receptor, la fase 2 se ha basado en adquirir todo el conocimiento posible acerca del funcionamiento de los mismos, así como el instalar todos los componentes necesarios para un correcto funcionamiento del hardware con el software en cuestión.

Posteriormente se ha investigado sobre los diferentes Softwares que permiten su desarrollo.

2.1.3 Fase 3: Estudio de Modulaciones

Conocido el funcionamiento de los SDRs en cuestión, en esta tercera fase se han seleccionado y analizado en base a diferentes criterios, las diferentes modulaciones a generar para una posterior clasificación.

2.1.4 Fase 4: Estudio de Algoritmos de aprendizaje supervisado

Esta fase ha estado orientada a la búsqueda de información y a la elección del algoritmo, teniendo en cuenta que el objetivo principal del proyecto se trata de clasificación, en base a este objetivo principal y mediante comparaciones de las probabilidades de acierto entre los diferentes algoritmos, se ha seleccionado el algoritmo de aprendizaje supervisado.

2.1.5 Fase 5: Creación, entrenamiento, validación, test y evaluación de rendimiento del algoritmo seleccionado

Seleccionado un algoritmo que permita desarrollar de forma eficiente el AMC, se ha realizado una base de datos con un número de muestras lo suficientemente amplia y sin llegar a sobresaturar, que permita entrenar, validar y testear el algoritmo elegido.

Como cierre de la fase 5, se hace una evaluación del rendimiento del AMC desarrollado.

2.1.6 *Fase 6: Documentación*

La fase de documentación se ha ido realizando de forma paralela con cada una de las fases anteriores a excepción de la fase 1, esta fase se ha fundamentado en la selección y descarte de información de manera que se agilice la siguiente fase.

2.1.7 *Fase 7: Redacción de la memoria*

El cierre de todas y cada una de las fases anteriores, da paso a la redacción de la memoria del Trabajo Fin de Grado (TFG), última fase del proyecto.

2.2 Diagrama de Gantt y tiempo estimado

Identificadas todas las fases, se procede a realizar un diagrama de Gantt (Figura 2.1) en el cual se representa de forma gráfica, el tiempo de cada una de las fases.

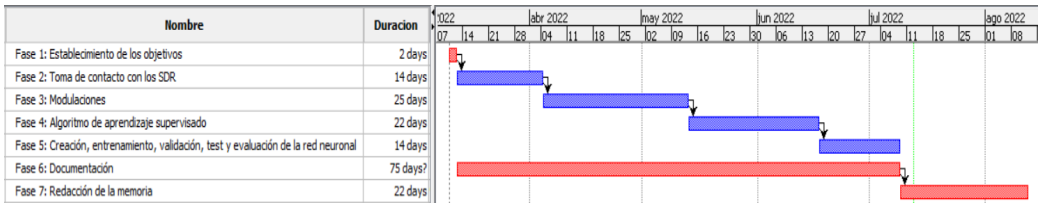


Figura 2.1 Diagrama de Gantt

De forma complementaria al diagrama de Gantt, se ha confeccionado una tabla (Tabla 2.1), en la que se asigna para cada fase o tarea mencionada, el tiempo dedicado en horas.

FASE	**Tiempo estimado (horas)**
Establecimiento de los objetivos	11
Toma de contacto con los SDR	63
Modulaciones	137
Algoritmo de aprendizaje supervisado	110
Creación, entrenamiento, validación, test y evaluación de la red neuronal	77
Documentación	(Tiempo ya incluido en cada una de las fases anteriores a excepción de la fase 1 en la cual no se ha realizado documentación)
Redacción de la memoria	110
	TOTAL: 497 horas

Tabla 2.1 Estimación total para cada fase del proyecto

2.3 Recursos

A continuación, se detallarán los diferentes recursos usados a lo largo del desarrollo del TFG.

- **Humanos**
  - Raúl Maldonado Cerezo, alumno del Grado en Ingeniería de Tecnologías de Telecomunicación, especialidad de Sistemas de Telecomunicación, como autor de este TFG.
  - Pedro Jesús Reche López, Profesor Titular del Área de Teoría de la Señal y Comunicaciones de la Universidad de Jaén (UJA), como supervisor de este trabajo.
- **Hardware**
  - Ordenador portátil personal ASUS K52J, con procesador Intel Core i3 y 4 Gigabytes de memoria RAM. Este ordenador ha sido usado como soporte para el funcionamiento del canal de transmisión del SDR número 1.
  - Ordenador portátil personal LENOVO LEGION Y720, con procesador Intel Core i7 y 16 Gigabytes de memoria RAM. Este ordenador ha sido usado tanto para dar soporte al funcionamiento del canal de recepción del SDR número 2 como para todo lo que involucra al desarrollo del AMC.
  - Dos Radios Definidos por Software bladeRF x40, para la transmisión y recepción de señales inalámbricas.
  - Línea de acceso a Internet.
- **Software**
  - Sistema operativo Windows 7 de 64 bits.
  - Sistema operativo Windows 10 de 64 bits.
  - MATLAB.
  - GNU radio.
  - bladeRF-CLI.

2.4 Estimación de costes

Descritos todos los recursos y calculadas todas las horas invertidas en la elaboración del proyecto, se procederá a realizar una estimación del coste total.

- **Humanos**

En la siguiente tabla (Tabla 2.2) se presentan los costes derivados de los recursos humanos.

	Coste (€ / hora)	**Cantidad (horas)**	**Total (€)**
Tutorías	40	25	1000
Trabajo personal	15	497	7455
			TOTAL: 8455 €

Tabla 2.2 Coste total de recursos humanos

- **Hardware**

En cuanto a los costes derivados del hardware usado en el proyecto se muestra en la siguiente tabla (Tabla 2.3).

Dispositivo	**Coste (€)**
Ordenador portátil personal ASUS K52J	475.05
Ordenador portátil personal LENOVO LEGION Y720	772
(2x) BladeRF x40	(2x) 420
Línea de acceso a Internet	180
	TOTAL: 2267.05 €

Tabla 2.3 Coste total de recursos hardware

➢ **Software**

Por último, se presentan los costes asociados al software usado en la siguiente tabla (Tabla 2.4).

Software	**Coste (€)**
Licencia Windows 7	24.95
Licencia Windows 10	28
GNU radio	0
Matlab (Licencia de estudiante) Toolboxes: • Audio Toolbox • Deep Learning Toolbox • Signal Processing Toolbox	35 • 20 • 20 • 20
BladeRF-CLI	0
	TOTAL: 112.95 €

Tabla 2.4 Coste total de recursos software

2.5 Presupuesto total

Obtenidos todos los costes asociados a la realización del trabajo, se obtiene el presupuesto total como la suma de todos y cada uno de los costes mostrados anteriormente.

El coste estimado del TFG asciende a un total de 10775 euros (Tabla 2.5).

Recurso	**Coste (€)**
Humano	8455
Hardware	2267.05
Software	112.95
	TOTAL: 10835 €

Tabla 2.5 Presupuesto total del TFG

3 MATERIALES Y MÉTODOS

3.1 Sistema de Radiocomunicaciones Clásico

Este apartado, se comenzará con una breve explicación general de un sistema de radiocomunicaciones, para, a partir de esta explicación, introducir la tecnología SDR.

Un sistema de radiocomunicaciones, es un sistema que puede comunicar información a través de ondas electromagnéticas en el espectro de radiofrecuencia (*Radio Frecuency - RF*) desde 3 kHz hasta 300 GHz.

Típicamente un sistema de radiocomunicaciones se compone de los bloques transmisor y receptor.

El bloque transmisor modifica la señal para una transmisión eficiente, generalmente este bloque se compone de los siguientes sub-bloques: modulador, codificador, conversor digital a analógico (*Digital-to-Analogic Converter - DAC*), filtros, amplificadores de ganancia, mezcladores y amplificadores de potencia (*Power Amplifier - PA*).

El bloque receptor recibe la señal de RF y reconstruye la señal original, de forma general, el bloque receptor se compone de: Amplificador de bajo ruido (*Low Noise Amplifier - LNA*), mezclador, amplificador de ganancia, filtros, conversor analógico a digital (*Analogic-to-Digital Converter - ADC*), demodulador y decodificador.

Tradicionalmente, todos los componentes descritos anteriormente y, que son parte de un sistema de radiocomunicaciones, solían implementarse mediante circuitos electrónicos.

3.2 Radios Definidas por Software

Gracias a la evolución en la capacidad de procesamiento de los procesadores digitales, muchas de estas tareas de procesamiento de señal que se solían hacer mediante circuitos electrónicos, se pueden llevar a cabo mediante software, obteniendo como resultado la tecnología de las radios definidas por software, basada en un procesamiento de señales realizada normalmente en ordenadores a través de programas.

Donde, dependiendo del fabricante o modelo del SDR en cuestión, nos podremos encontrar una gran variedad de formatos, por ejemplo, podemos encontrar SDRs que solo posean la capacidad de recibir o transmitir.

Además, dependiendo del modelo de SDR, se tendrá un mayor o menor grado de libertad a la hora de procesar las señales mediante software, algo denominado como '*los niveles de un SDR*' [8].

El dispositivo SDR se puede definir por tanto como un sistema de RF, donde gran parte de los componentes que tradicionalmente se implementaban mediante circuitos electrónicos, ahora son implementados mediante software, que, recibe la información y la transmite al ordenador al que está conectado para su procesamiento.

De esta forma bastaría con un dispositivo SDR, un ordenador con puerto USB y una o varias antenas para tener un sistema de radiocomunicaciones completo.

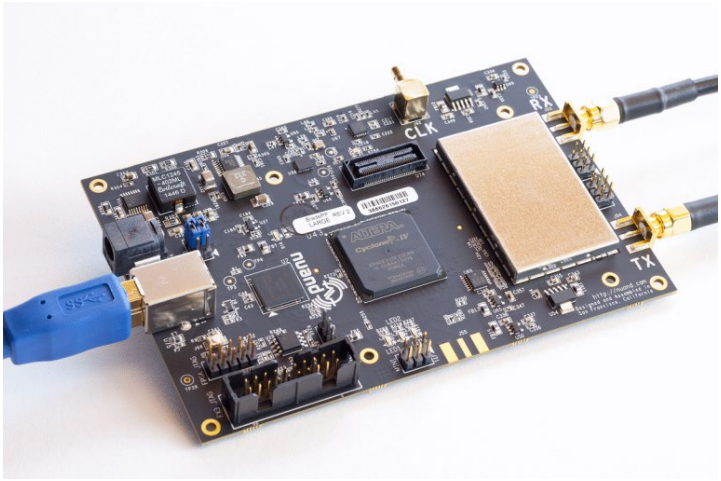


Figura 3.1 Dispositivo SDR transceptor [9]

Explicada de forma general la tecnología SDR, a continuación, se detallará el funcionamiento de un SDR genérico en base a su arquitectura (Figura 3.2).

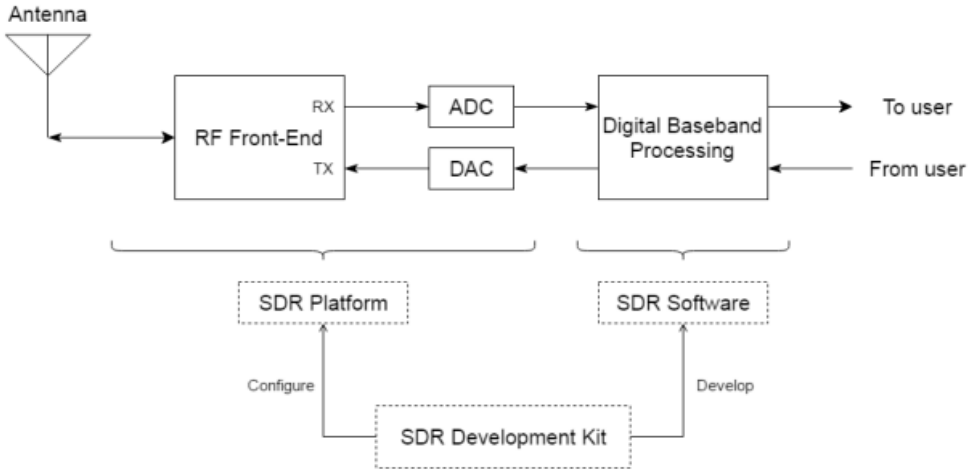


Figura 3.2 Arquitectura SDR [10]

- **Plataforma SDR**

  El bloque de plataforma SDR, hace referencia al hardware en sí, esto dependerá del modelo o del fabricante del SDR, este bloque sirve como puente entre la señal de RF analógica y la señal digital en banda base (*Base Band - BB*).

  La principal función de este bloque es, por un lado, en recepción, realizar la conversión de la señal de RF analógica a la señal de banda base digital y, por otro lado, en transmisión, realizar la conversión de la señal banda base digital a señal RF analógica.

  La función del sub-bloque RF Front-End, es transmitir y/o recibir las señales de radiofrecuencia para adaptarlas, esta adaptación, en transmisión consiste entre otras cosas en una modulación (subida de frecuencia) de la señal, y en recepción consiste en una demodulación.

  En cuanto a los sub-bloques ADC/DAC, estos se encargarán respectivamente de digitalizar una señal recibida para su posterior procesado en un ordenador (bloque ADC) y, de convertir en analógica una señal digital (bloque DAC) recibida a través del ordenador para su posterior transmisión a través del aire.

- **Software SDR**

  El bloque de software SDR, como bien su nombre indica, se trata del Software en cuestión que permitirá realizar el procesado de las señales que se transmiten y/o reciben a través del SDR, la compatibilidad con los diferentes tipos de software depende del hardware.

Posteriormente, en otros subapartados, se comentarán las plataformas SDR y los softwares SDR existentes y los seleccionados para la realización del TFG.

3.2.1 Plataformas SDR comerciales y elección

En la última década, se ha experimentado un gran desarrollo y avance de la tecnología, en el mundo de la radio definida por Software, esto se traduce en un gran incremento de empresas y organizaciones dispuestas a desarrollar y crear nuevos productos.

La figura 3.3 ilustra el diagrama de bloques típico de cualquier plataforma SDR.

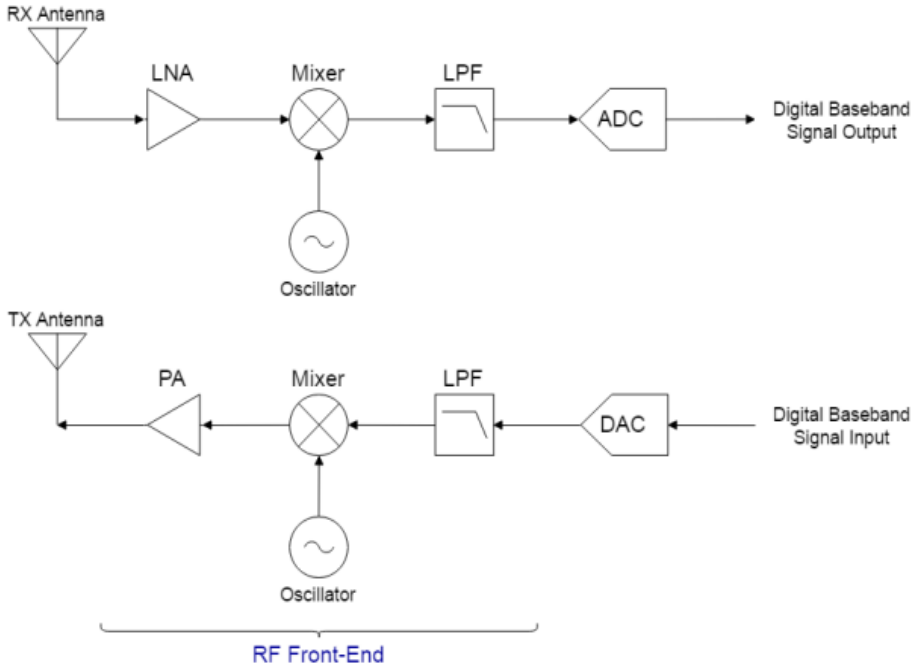


Figura 3.3 Diagrama de bloques de plataforma SDR genérica [10]

A continuación, se detallan los dispositivos más populares en el mercado actual:

- ***RTL-SDR (R820T2 RTL2832U)***. [11] Receptor de señales de RF diseñado en 2010. Este receptor trabaja en la banda de frecuencias de los 500 kHz hasta los 1.7GHz, admitiendo frecuencias de muestreo de 3.2 MHz con un ancho de banda instantáneo de 2.4 MHz.

  Es compatible con un gran número de herramientas Software como SDRSharp, Gqrx, SdrDx, entre otros.

  El rango de precios es reducido, variando entre 12 € y 79 €.

Figura 3.4 Dispositivo RTL-SDR

- ***HackRF One***. [12] Plataforma SDR que puede operar en el rango de RF desde 1 MHz hasta 6 GHz, con capacidad tanto para transmitir como para recibir, es decir, es un transceptor, con un ancho de banda instantáneo máximo de 20 MHz y modo de funcionamiento semidúplex.
  Es compatible con las principales herramientas Software de configuración de SDR como GNU radio, Matlab y SDRSharp.
  Los precios del mismo varían entre 300 € y 600 €.

Figura 3.5 Dispositivo HackRF One

- ***Nuand bladeRF***. [9] Es un dispositivo SDR transceptor de bajo coste, capaz de operar en el rango de frecuencias entre 300 MHz a 3.8 GHz, con un ancho de banda máximo disponible de 28 MHz y un modo de funcionamiento full dúplex.
  Compatible con GNU radio, Matlab, SDRangel, SDRSharp.
  Los precios varían entre 240 € y 1600 €.

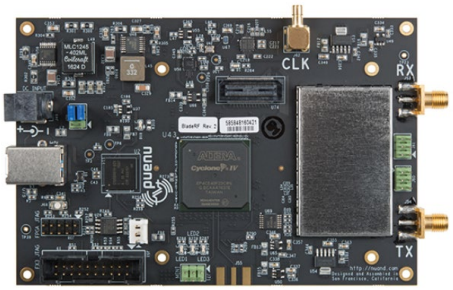


Figura 3.6 Dispositivo bladeRF

Se procede a realizar una tabla (Tabla 3.1) a modo de comparación entre las diferentes plataformas de SDR expuestas a excepción del SDR tipo RTL-SDR que queda descartado al no tener opción de transmitir.

Diseñador	Nuand	Great Scott
Tipo	bladeRF	HackRF
Rango RF (Hz)	(0.3 ~ 3.8) G	(1 M ~ 6 G)
Rango de Ancho de Banda de RF	(1.5 ~ 28) M	(3 ~ 20) M
Resolución ADC (bits)	12	8
Resolución DAC (bits)	12	8
Frecuencia de muestreo ADC (MHz)	40	20
Frecuencia de muestreo DAC (MHz)	40	20
Modo Dúplex	Full	Semi
Soporte para GNU radio	✓	✓
Soporte para Matlab	✓	✓
Precio (€)	420	300

Tabla 3.1 Comparación de Plataformas SDR

A continuación, se elige el bladeRF, concretamente el modelo bladeRF x40 como la plataforma SDR usada en el proyecto por los siguientes motivos:

- Los anchos de banda de las señales que se generarán en el proyecto están entre (10 ~ 60) kHz. El ancho de banda mínimo del bladeRF es de 1.5 MHz, suficiente para el ancho de banda requerido por las señales generadas en el proyecto.
- El bladeRF soporta una comunicación full dúplex, se podrá por tanto transmitir y recibir señales de forma simultánea.

- La plataforma de bladeRF es de código abierto, bien documentada y con gran soporte en cuanto a tutoriales y ejemplos.
- La EPSL cuenta con dispositivos bladeRF los cuales pueden ser prestados a sus alumnos.

3.2.2 Introducción al bladeRF

BladeRF es una plataforma SDR de alto rendimiento, orientada principalmente tanto a estudiantes y aficionados como a profesionales, y es que, el ser de código abierto permite indagar y experimentar con el mundo de la RF añadiendo la posibilidad de que el dispositivo sea totalmente reprogramable.

El dispositivo cuenta con canales de transmisión y recepción que son completamente independientes, lo que permite alcanzar una comunicación full-dúplex. Además, esta plataforma cuenta con un puerto USB 3.0, lo que habilita altas velocidades en las transferencias de datos entre el bladeRF y el ordenador.

A continuación, se muestra una tabla (Tabla 3.2) con todas las especificaciones del dispositivo bladeRF x40 [13] usado en el proyecto.

Tipo	bladeRF
Modelo	x40
Rango RF (Hz)	(0.3 ~ 3.8) G
Rango de Ancho de Banda de RF	(1.5 ~ 28) M
Modo dúplex	Full
Resolución ADC/DAC (bits)	12
Frecuencia de muestreo máxima ADC/DAC (MHz)	40
Potencia de Transmisión típica (dBm)	6
Interfaz	USB 3.0 (5 Gbps) Compatible con USB 2.0
Microcontrolador	Cypress FX3
FPGA	Altera Cyclone IV
Fuente abierta	HDL + firmware + esquemática
Soporte de Software	MATLAB / GNU radio
Precio (€)	420

Tabla 3.2 Especificaciones bladeRF x40

Seguidamente, se muestra una figura (Figura 3.7) del dispositivo bladeRF x40 junto con su conector USB 3.0.

Figura 3.7 Dispositivo bladeRF x40

3.2.2.1 Arquitectura interna del bladeRF

En este sub apartado se comentará de forma general y breve la arquitectura interna del bladeRF x40 en base a la figura mostrada a continuación (Figura 3.8), para, posteriormente centrarnos en un componente fundamental.

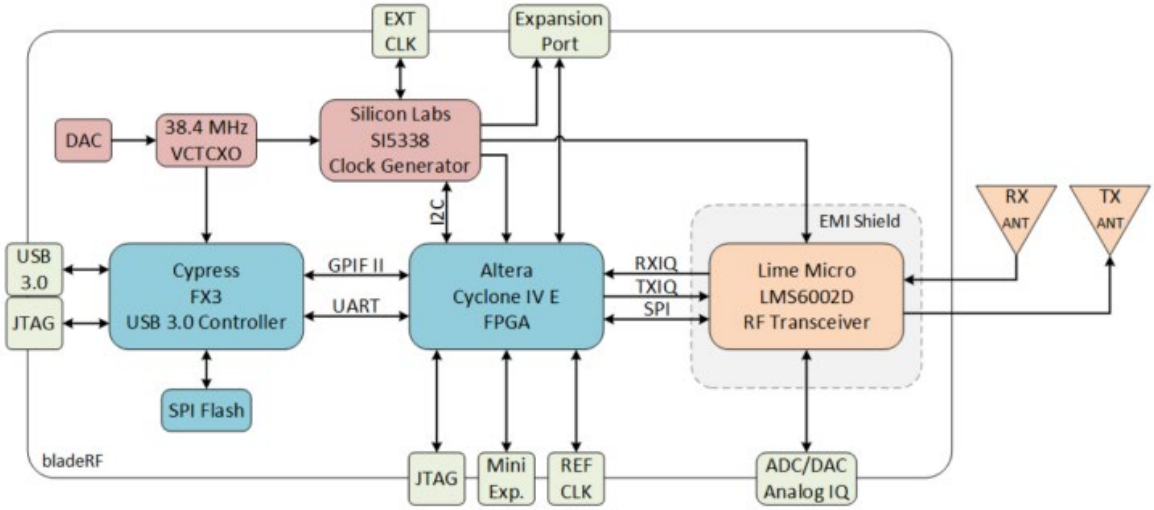


Figura 3.8 Arquitectura interna del bladeRF x40 [13]

El puerto USB 3.0 es la interfaz entre el ordenador personal y la plataforma bladeRF, este puerto, además es compatible con puertos USB 2.0 aunque con algunas limitaciones en cuanto a las velocidades de transferencia de datos, esta compatibilidad ha sido de gran ayuda ya que, uno de los ordenadores usados en el desarrollo del proyecto disponía de este tipo de puertos.

Además, este puerto USB es capaz de transferir potencia al bladeRF, es decir, de forma simultánea hay una transferencia de potencia y de datos.

El bladeRF está equipado con un Cypress FX3 USB 3.0 Controller, el cual se encargará de controlar las comunicaciones de datos entre el puerto USB 3.0 y el GPIF II (General Programmable Interface Generation II).

La interfaz GPIF II está directamente conectada con la FPGA (Field Programmable Gate Arrays) del tipo Altera Cyclone IV. La FPGA no solo proporciona la interfaz entre el controlador Cypress FX3 USB 3.0 y el transceptor LMS6002D, sino que realiza un almacenamiento y procesamiento de señales.

Por último, se comenta el oscilador de cristal controlado por tensión con compensación de temperatura (VCTCXO), el ser compensado por temperatura, podría acarrear problemas cuando son usados más de 1 bladeRF, dependiendo del tipo de aplicación en cuestión para los cuales los bladeRF sean usados [14].

3.2.2.2 *Transceptor LMS6002D y arquitectura interna*

El componente electrónico fundamental y del cual se debe saber cómo funciona para la realización del proyecto es el microsistema LMS6002D, este es el circuito integrado que contiene gran parte de los elementos contados en el punto 3.1, y, es el encargado de la conversión entre la señal banda base digital a la señal RF analógica y viceversa.

Este componente es elemental, ya que, a través de Software se podrá programar su funcionamiento, por ejemplo, podremos ajustar ganancias de transmisión y/o recepción o también ajustar frecuencias de muestreo, frecuencias de portadora e incluso anchos de banda entre otras cosas.

Por ello, a priori se debe saber cuáles son los rangos mínimos y máximos dentro de los cuales pueden trabajar los elementos que componen dicho microsistema para, de esta forma evitar introducir valores erróneos a la hora de transmitir o recibir señales.

Para ello se hace uso del Software bladeRF-CLI que la desarrolladora Nuand pone a disposición del usuario.

Con la información de salida ofrecida por el comando '*print*', se establece la siguiente tabla (Tabla 3.3):

Tipo	LMS6002D
Rango RF (Hz)	(0.3 ~ 3.8) G
Saltos en los anchos de banda del canal (MHz)	1.5, 1.75, 2.5, 2.75, 3, 3.84, 5, 5.5, 6, 7, 8.75, 10, 12, 14, 20, 28
Ganancia del canal de transmisión (dB)	(0, 60)
Ganancia del canal de recepción (dB)	(0, 60)

Tasa de muestreo del canal de transmisión	(0.08, 40) MHz
Tasa de muestreo del canal de recepción	(0.08, 40) MHz

Tabla 3.3 Rangos de funcionamiento del transceptor LMS6002D

Conocidos los rangos de funcionamiento de los elementos que conforman el transceptor LMS6002D, se pasará a detallar de forma general la arquitectura y funcionamiento interno de dicho componente.

- **Arquitectura interna del transceptor LMS6002D**

El transceptor LMS6002D cuya arquitectura interna se encuentra en la Figura 3.9 posee dos canales independientes por los cuales transcurren las señales, uno para transmitir, otro para recibir. Cada canal tiene su propio bucle de enganche de fase (*Phase Locked Loop – PLL)* mostrados en la figura como TXPLL y RXPLL, esto quiere decir que, aunque ambos PLLs compartan la misma fuente de reloj externa (*Phase Locked Loop Clock – PLLCLK*), ambos pueden generar diferentes frecuencias de portadora (*Frecuency Carrier - FC*), lo que significa que de manera simultánea podemos estar transmitiendo en una frecuencia distinta a la que recibimos, dentro del rango entre (0.3 - 3.8) GHz.

Cada uno de estos PLLs, genera un par de sinusoides desfasadas 90 grados la una con la otra, para, posteriormente ser introducidas en un oscilador junto con las componentes en fase y en cuadratura (*In-Phase* and *Quadrature - IQ*) de la señal digital banda base generada por el usuario en el caso de transmisión, o bien junto con las componentes IQ de la señal de RF analógica en recepción. [10]

Y es que la multiplicación en el tiempo de la señal banda base digital y/o de la señal RF analógica con esas sinusoides desfasadas entre sí, está generando una modulación y/o demodulación en fase y cuadratura. Esto es fundamental saberlo para el desarrollo del proyecto y para la posterior generación de las modulaciones.

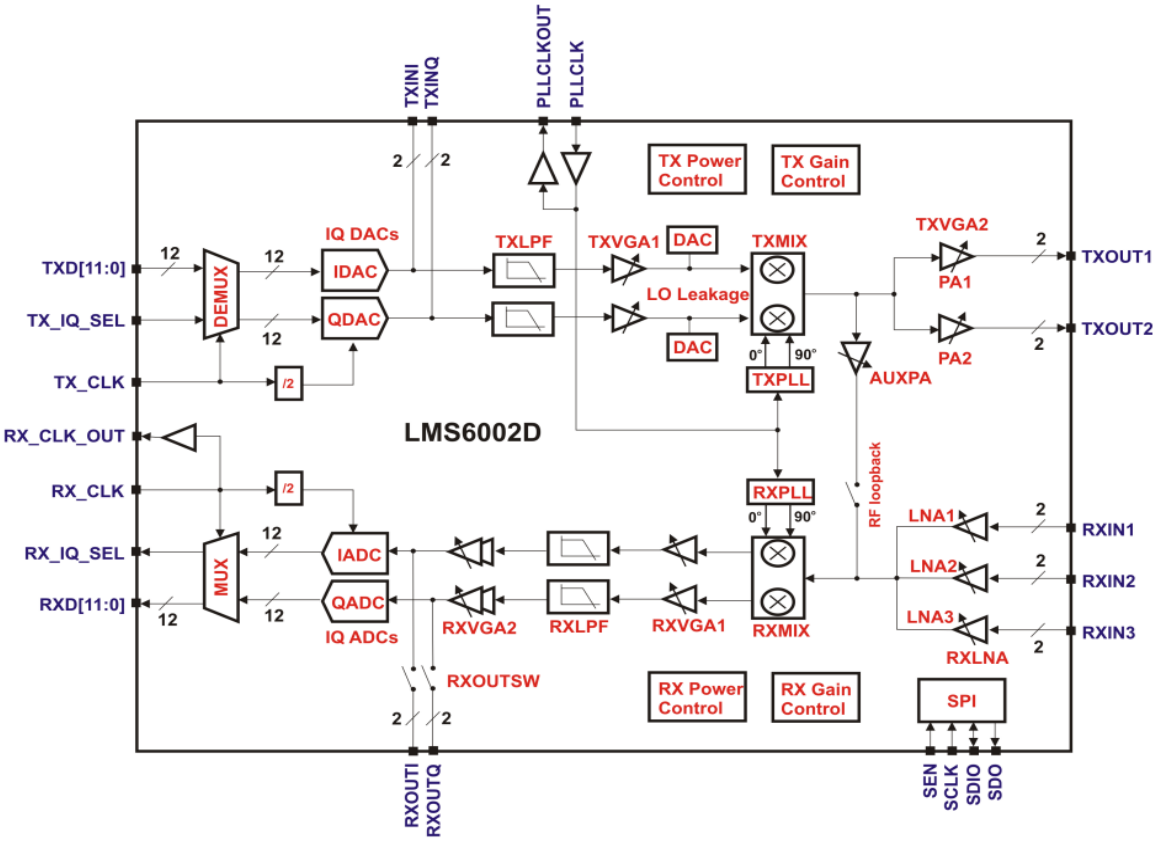


Figura 3.9 Arquitectura interna del LMS6002D [13]

3.2.2 Modulador en fase y cuadratura

A continuación, se presenta una figura que muestra más en detalle la modulación de las componentes IQ por parte del SDR.

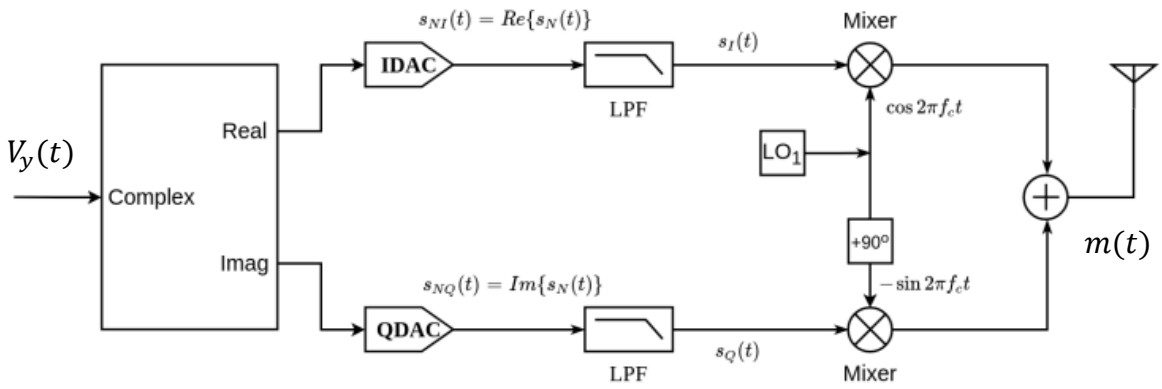


Figura 3.10 Modulador IQ [10]

En primer lugar, el SDR, espera recibir las muestras complejas en fase y cuadratura de la señal digital banda base generada por el usuario.

A continuación, se analiza de forma matemática la manera en la cual el SDR realiza la modulación de información, para, establecer la forma en la que el SDR debe recibir las señales digitales por parte del usuario.

La expresión de una modulación de forma genérica se presenta como:

$$m(t) = a(t) \cdot \cos\left(2\pi f_c \cdot t + \phi(t)\right) = a(t) \cdot \cos\left(\omega_c \cdot t + \phi(t)\right) \quad (1)$$

Donde:

- a(t) es la envolvente real
- f_c es la frecuencia de portadora
- $\omega_c = 2\pi f_c$ es la frecuencia angular de la portadora
- t es el tiempo en el que existe la señal
- $\phi(t)$ es la componente no lineal de la fase instantánea de la señal

Aplicando la siguiente identidad trigonométrica a la expresión (ec.1):

$$\cos(a + b) = \cos a \cos b - \sin a \sin b \quad (2)$$

Obtenemos la siguiente expresión:

$$m(t) = a(t) \cdot \cos(\phi(t)) \cdot \cos(2\pi f_c \cdot t) - a(t) \cdot \sin\big(\phi(t)\big) \cdot \sin\left(2\pi f_c \cdot t\right) \quad (3)$$

Donde:

$$I(t) = a(t) \cdot \cos(\phi(t)) \quad (4)$$

$$Q(t) = a(t) \cdot \sin\big(\phi(t)\big) \quad (5)$$

Siendo (ec.4) la componente en fase y, (ec.5) la componente en cuadratura de la señal modulada o señal paso banda, siendo estas señales paso bajo, trabajando el usuario con la suma compleja de las mismas (ec.14) o equivalente paso bajo en el ordenador mediante el

software correspondiente. Para posteriormente el SDR tomar la parte real e imaginaria de cada muestra digital.

Sustituyendo las expresiones (ec.4) y (ec.5) en (ec.3) queda finalmente:

$$m(t) = I(t) \cdot \cos(2\pi f_c \cdot t\,) - Q(t) \cdot \sin\,(2\pi f_c \cdot t) \qquad (6)$$

Representando mediante la (ec.6) la forma exacta en la que el dispositivo SDR recibe las señales complejas paso bajo generadas por el usuario, extrayendo la parte real e imaginaria de cada muestra, obteniendo la componente en fase y cuadratura, para posteriormente mezclarlas con las señales generadas por los PLLs y, por tanto, realizar la modulación.

En 3.2.4 se explicará la gran importancia de trabajar con las señales complejas paso bajo, así como el análisis matemático para obtener la envolvente compleja a partir de la señal paso banda.

3.2.3 *Demodulador en fase y cuadratura*

De forma análoga a la modulación en IQ tenemos la demodulación en IQ (Figura 3.11) realizada por el SDR, el concepto es similar al concepto explicado en el apartado anterior.

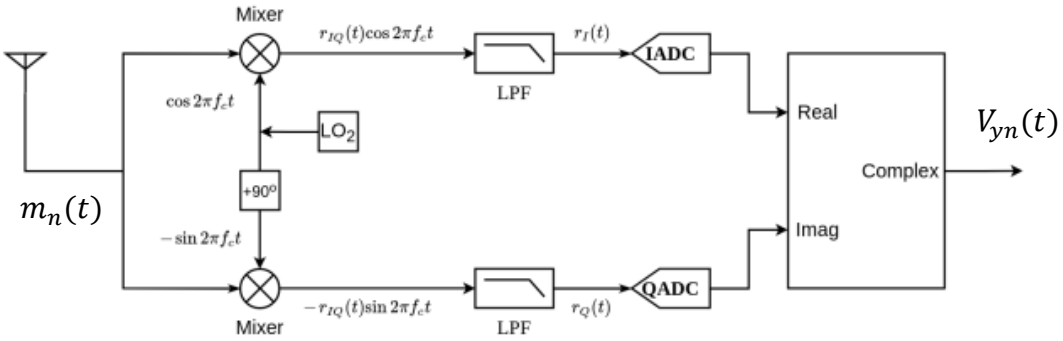


Figura 3.11 Demodulador IQ [10]

Normalmente, la señal de RF recibida es débil, por lo que primeramente es amplificada por un amplificador de bajo ruido, este primer amplificador, el cual podremos ajustar su ganancia, intentando maximizarla, tendrá una mínima figura de ruido, ya que, como se explica a continuación con la fórmula de Friis (ec.7), el primer amplificador por el cual transcurre la señal recibida es el predominante en el factor de ruido total del sistema.

$$F_{system} = F_{LNA} + \frac{F_1 - 1}{G_{LNA}} + \frac{F_2 - 1}{G_{LNA} G_1} + \frac{F_3 - 1}{G_{LNA}\, G_1 G_2} + \cdots + \frac{F_n - 1}{G_{LNA}\, G_1 G_2 \ldots G_{n-1}} \qquad (7)$$

Una vez amplificada la señal de RF es mezclada con las señales generadas por los PLLs para realizar la conversión de señal de RF en sus componentes IQ a señales IQ banda base.

Obtenida la señal en banda base, es decir centrada en 0 Hz, es filtrada mediante un filtro paso bajo para limitar el ancho de banda de la misma para posteriormente digitalizarla y, obtener las muestras complejas en fase y cuadratura que, nuevamente, al igual que se dijo anteriormente, serán las señales con las que trabajaremos en el ordenador.

3.2.4 ¿Por qué debemos trabajar con señales complejas IQ en el ordenador?

La necesidad de trabajar con señales complejas, aparece en la etapa de generación y recepción de señales moduladas, es decir, en la etapa de simulación.

La simulación requiere una señal muestreada a una frecuencia de muestreo dada, debido a que un simulador solo puede realizar cálculos con un número finito de muestras.

El teorema de Nyquist [15] establece la siguiente condición para poder reconstruir la señal original a partir de la muestreada:

$$f_s \geq 2 \cdot F_{máx} \qquad (8)$$

Donde;

- f_s es la frecuencia de muestreo dada
- $F_{máx}$ es la frecuencia máxima de la señal a transmitir

A continuación, (Figura 3.12) se muestra el proceso de muestreo de una señal analógica o continua en el tiempo.

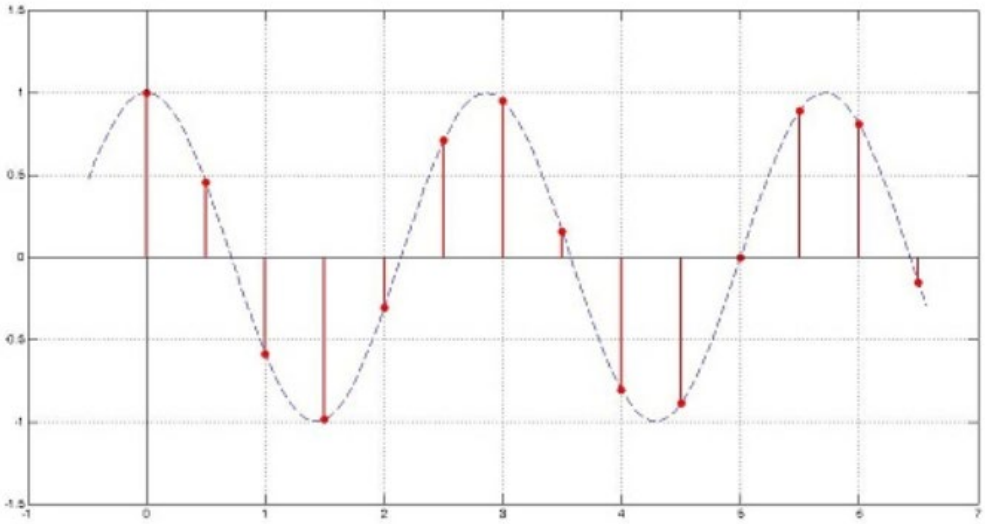


Figura 3.12 Proceso de muestreo de una señal continua en el tiempo [16]

Donde la separación entre muestra y muestra viene dada por el periodo de muestreo, aumentando la frecuencia de muestreo, la distancia entre muestras se reducirá, aumentando el número de muestras y, por tanto, aumentando el parecido entre la señal original y la señal muestreada.

El período de muestreo viene dado por la siguiente expresión:

$$T_s = \frac{1}{f_s} \qquad (9)$$

Esto nos lleva a la siguiente conclusión, y es que, como sabemos, los ordenadores trabajan dentro del mundo digital, es decir, cualquier tipo de señal se debe de digitalizar para poder trabajar con ella dentro del ordenador, pues tienen una memoria finita, una parte de la digitalización de una señal analógica es el muestreo de la misma, y ese muestreo debe cumplir la condición descrita anteriormente por la ec.8.

A partir de esta conclusión, se abren dos caminos:

- Señales Banda Base

  Cuando trabajamos con señales con una frecuencia máxima en torno a 0Hz, no hay ningún problema, pues la frecuencia de muestrear dicha señal no será muy elevada.

- Señales Paso Banda

  La generación de señales moduladas, requieren frecuencias de portadora muy elevadas, esto implica también que la frecuencia de muestreo será muy elevada, por lo que el muestreo directo de la señal no va a ser posible con hardware convencional como un ordenador, por tanto, con el ordenador se ha de trabajar con el equivalente paso bajo, de la señal paso banda, es decir, cada señal paso banda deberá estar representada por su señal compleja banda base equivalente, de esta forma la frecuencia máxima de la misma no será muy elevada pues estará en torno a 0 Hz, que, al igual que la frecuencia de muestreo, no será muy elevada.

3.2.4.1 Envolvente compleja, señal banda base equivalente o señal compleja

Los tres términos descriptivos de este subapartado son equivalentes entre sí [17].

El espectro de las señales paso banda está ubicado generalmente en torno a la f_c, ya que, cualquier señar modulada es una señal paso banda.

Un teorema [18], [19], [20], establece que cualquier señal paso banda que tenga un ancho de banda limitado, puede representarse mediante un equivalente paso bajo que tenga el mismo ancho de banda. Esta señal equivalente será la usada en simuladores ya que permite reducir la frecuencia de muestreo en comparación con el muestreo directo de la señal paso banda.

Teniendo claro cuál es el problema de trabajar con señales paso banda en sistemas digitales convencionales como ordenadores, se procederá a detallar matemáticamente la solución.

Partiendo nuevamente de la expresión de una modulación genérica (ec.1), obtenemos su representación compleja reemplazando la función '*coseno'* por una función exponencial.

$$\tilde{m}(t) = a(t) \cdot e^{j(2\pi fc \cdot t + \phi(t))} = a(t) \cdot e^{j\phi(t)} e^{j(2\pi f_c \cdot t)} = V_y(t) \cdot e^{j(2\pi f_c \cdot t)} \qquad (10)$$

Donde la envolvente compleja o señal banda base equivalente sería:

$$V_y(t) = \tilde{m}(t) \cdot e^{-j(2\pi f_c \cdot t)} = a(t) \cdot e^{j\phi(t)} \qquad (11)$$

La multiplicación de una señal por $e^{-j(2\pi f_c \cdot t)}$ corresponde a un desplazamiento de frecuencia del espectro.

Aplicando el teorema de DeMoivre a la expresión (ec.11)

$$e^{jn\theta} = \cos n\theta + j \sin n\theta = (\cos \theta + j \sin \theta)^n \quad (12)$$

Obtenemos finalmente:

$$V_y(t) = a(t) \cdot \cos(\phi(t)) + ja(t) \cdot \sin(\phi(t) \quad (13)$$

Pudiendo ser reescrita de forma más compacta como la suma compleja de las componentes en fase y cuadratura:

$$V_y(t) = I(t) + jQ(t) \quad (14)$$

De esta forma se ha obtenido la señal paso bajo equivalente de la señal paso banda, que será la señal con la que podamos trabajar en el ordenador, ya que está contenida en torno a 0 Hz, siendo el hardware SDR quien se encargue de subirla en frecuencia realizando la modulación en fase y cuadratura comentada anteriormente.

3.2.5 *Software elegido para el desarrollo del SDR*

Cualquier software para el desarrollo del SDR, implementa algoritmos de procesamiento de señales digitales como era de esperar, donde la mayoría de estos softwares, o bien nos permiten desarrollar el SDR mediante código, como Matlab o bien mediante bloques ya predefinidos como GNU radio.

Explicado cómo funcionan de forma general la mayoría de softwares que permiten el desarrollo del SDR, se pasará directamente a detallar los softwares elegidos así del ¿Por qué? Se han elegido, comentando ventajas y desventajas de los mismos, para finalizar mostrando una tabla con las ventajas y desventajas que aportan cada uno respecto del otro.

Los softwares elegidos para desarrollar los SDRs han sido por un lado GNU radio, y por otro lado Matlab, en apartados posteriores se comentarán en detalle el funcionamiento de los mismos.

- **GNU radio**

  GNU radio, ha sido utilizado para desarrollar el canal de transmisión del SDR número 1 y para desarrollar el canal de recepción del SDR número 2, aunque el desarrollo del canal de recepción del SDR con GNU radio, ha sido desarrollado únicamente con la finalidad de comprobar que la recepción es acorde a lo que se debería recibir.

  Y es que GNU radio, nos va a permitir de una forma muy sencilla tanto realizar un procesamiento simple de la señal de entrada al SDR como la configuración del mismo, así como la visualización en tiempo real de lo que se transmite y recibe. Todo esto de una forma muy rápida y amigable.

  Como desventaja, añadiría la poca posibilidad que permite GNU radio a la hora de realizar procesados más complejos de las señales tales como calcular características de las mismas etc. De ahí el uso de Matlab.

- **Matlab**

  Matlab, ha sido utilizado tanto para la configuración del canal de recepción del SDR número 2, como para el procesado de las señales recibidas del SDR.

A continuación, se muestra una tabla (Tabla 3.4) en la cual se recoge la comparación de ambos softwares.

SOFTWARE	GNU radio	Matlab
Fuente abierta	✓✓✓	×
Gratis	✓✓✓	×
Facilidad de uso	✓✓✓	✓✓
Soporte para SDR	✓✓✓	✓✓
Tutoriales y ejemplos	✓✓✓	✓✓
Procesamiento de señales	✓	✓✓✓

Tabla 3.4 Comparación de los Softwares usados

Como última información adicional en este subapartado, se hace hincapié en que la gran mayoría de softwares disponibles para desarrollar un SDR, están ambientados únicamente en la parte de recepción, sin posibilidad alguna de desarrollar el transmisor.

3.3 Técnicas de Modulaciones Analógicas

Las modulaciones analógicas, concretamente cinco de ellas, son las modulaciones seleccionadas para la clasificación, en los siguientes subapartados serán analizadas tanto de forma teórica-gráfica como matemática, para, finalmente obtener las componentes en fase y cuadratura correspondientes a cada modulación.

Ahora se comentará el procedimiento genérico y válido para todas las modulaciones que nos permitirá modular las señales usando un tipo de modulación u otra mediante el SDR.

Recordando la expresión (ec.6), expresión que representaba la forma exacta en la cual el SDR recibe y modula una señal banda base recibida del usuario, pues, partiendo de esta misma expresión, se podrá realizar cualquier tipo de modulación que se desee, para ello, simplemente se deberán de escoger adecuadamente las componentes IQ correspondientes a la modulación deseada en cuestión, es decir, cada modulación tiene unas componentes IQ distintas, dependiendo del tipo de componentes IQ escogidas se modulará de una forma u otra, puesto que el procedimiento realizado por el SDR para convertir la señal a paso banda es idéntico para cada modulación.

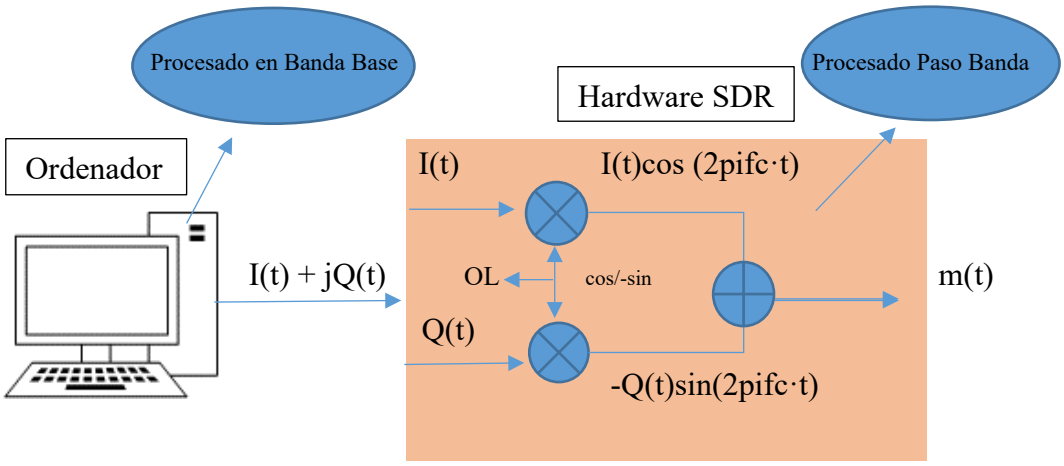


Figura 3.13 Esquemático completo del procesado de la señal banda base

3.3.1 Modulaciones lineales

Las modulaciones lineales serán aquellas en las que dependiendo de cómo varíe la señal de información o señal moduladora (normalmente expresada como '*x(t)*') la amplitud de la portadora (normalmente expresada como '*c(t)*') variará. Este será el concepto de partida para dichas modulaciones.

Para la explicación de dichas modulaciones, se partirá de la expresión genérica de una señal modulada (ec.1) para posteriormente particularizar a cada modulación en cuestión.

Se supondrá una señal determinística como moduladora, en este caso un tono con frecuencia 1 Hz, y una señal portadora con frecuencia f_c 25 Hz para la representación de figuras explicativas.

3.3.1.1 Modulación de Amplitud

Partiendo de (ec.1), igualando '$\phi(t) = 0$', y, sustituyendo $a(t)$ por $\cdot (1 + m \cdot x(t))$. obtenemos:

$$m_{AM}(t) = A_c \cdot \big(1 + m \cdot x(t)\big) \cdot \cos(2\pi f_c \cdot t) \qquad (15)$$

Siendo esta la expresión general de una señal AM.

Donde:

- A_c es la amplitud de la portadora
- m es el índice de modulación
- x(t) es la señal de información o moduladora

Obteniendo fácilmente las componentes en fase y cuadratura y por ende el equivalente paso bajo para la señal AM.

$$I(t)_{AM} = A_c \cdot \big(1 + m \cdot x(t)\big) \qquad (16)$$

$$Q(t)_{AM} = 0 \qquad (17)$$

$$V_y(t)_{AM} = A_c \cdot \big(1 + m \cdot x(t)\big) + j0 \qquad (18)$$

Aplicando la transformada de Fourier (FT) a la expresión (ec.15) se obtiene:

$$M(f) = \frac{A_c}{2}[\delta(f - f_c) + \delta(f + f_c)] + \frac{A_c \cdot m}{2}[X(f - f_c) + X(f + f_c)] \qquad (19)$$

Donde se comprueba que simplemente por el hecho de tener esa suma en la expresión temporal (ec.16), estamos invirtiendo energía en dos términos, siendo un término la portadora, esto se detallará al final de este subapartado.

Analizando AM, podemos observar dos tipos de comportamientos que, dependerán de cómo se comporte el índice de modulación 'm'.

- Caso m ≤ 1 $(I(t)_{AM}\ siempre\ es\ positivo)$

  En el caso dispuesto, podremos recuperar fácilmente la señal moduladora '*x(t)*' a partir de la envolvente '$V_y(t)$' de la señal modulada, esto implica un

beneficio muy grande, y es que el proceso de recuperación/demodulación es muy sencillo, bastando con usar receptores no coherentes que extraigan el valor absoluto de la amplitud de la señal recibida, esto es lo que hizo que AM se eligiese como técnica de radiodifusión, y es que el receptor era muy sencillo de implementar, implicando esto hardware sencillo y por tanto coste reducido.

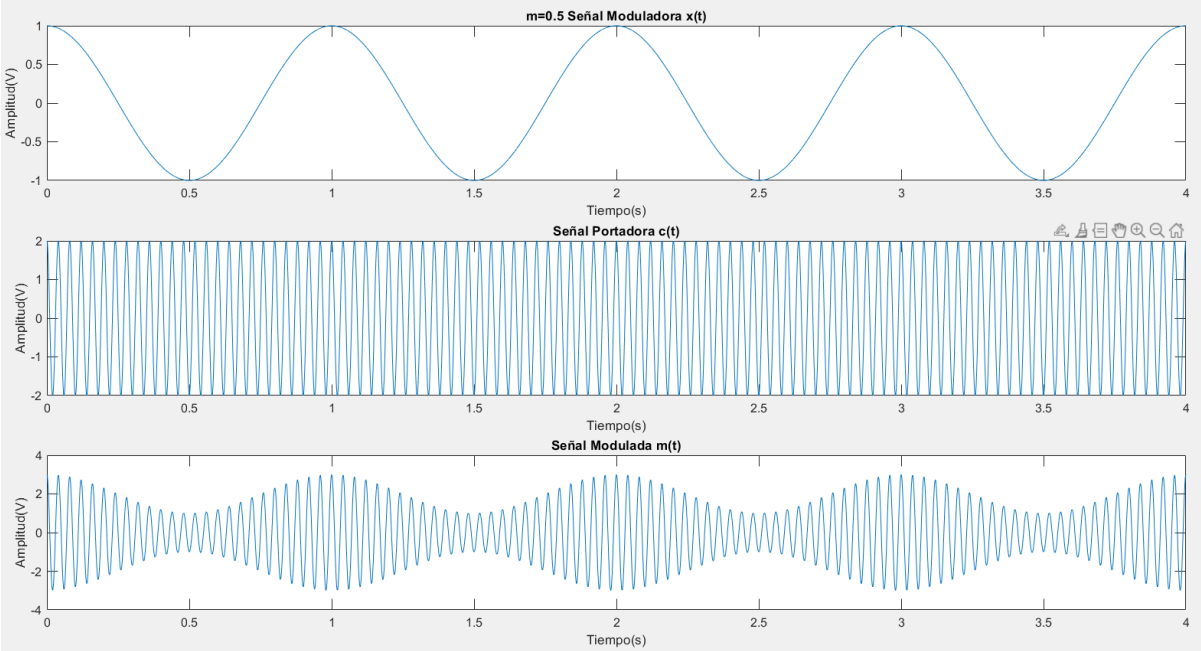


Figura 3.14 Representación temporal de la Señal Moduladora, Portadora y Modulada en AM cuando m≤1

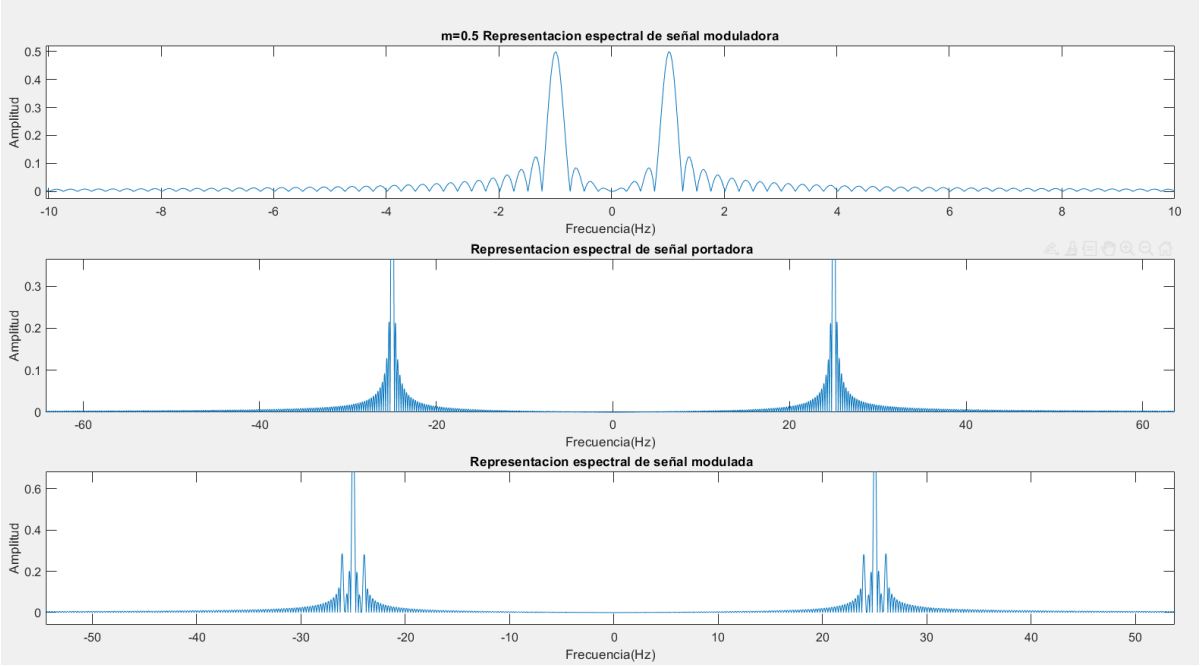


Figura 3.15 Representación espectral de la Señal Moduladora, Portadora y Modulada en AM cuando m≤1

Como se puede observar, el espectro de la señal modulada en AM se sitúa entre:

$$f_c - f_m, \qquad f_c, \qquad f_c + f_m \tag{20}$$

- Caso m ≥ 1 (caso de sobre modulación)

  Se podrían observar en este caso, cambios de fase de la señal modulada, y es que $I(t)_{AM}$ puede tener valores negativos dependiendo de la señal moduladora, haciendo mucho más complejo la recuperación de la señal original a partir de la envolvente, y es que, al aparecer cambios de fase, habrá zonas de la envolvente que ya no coincidan con la señal original.

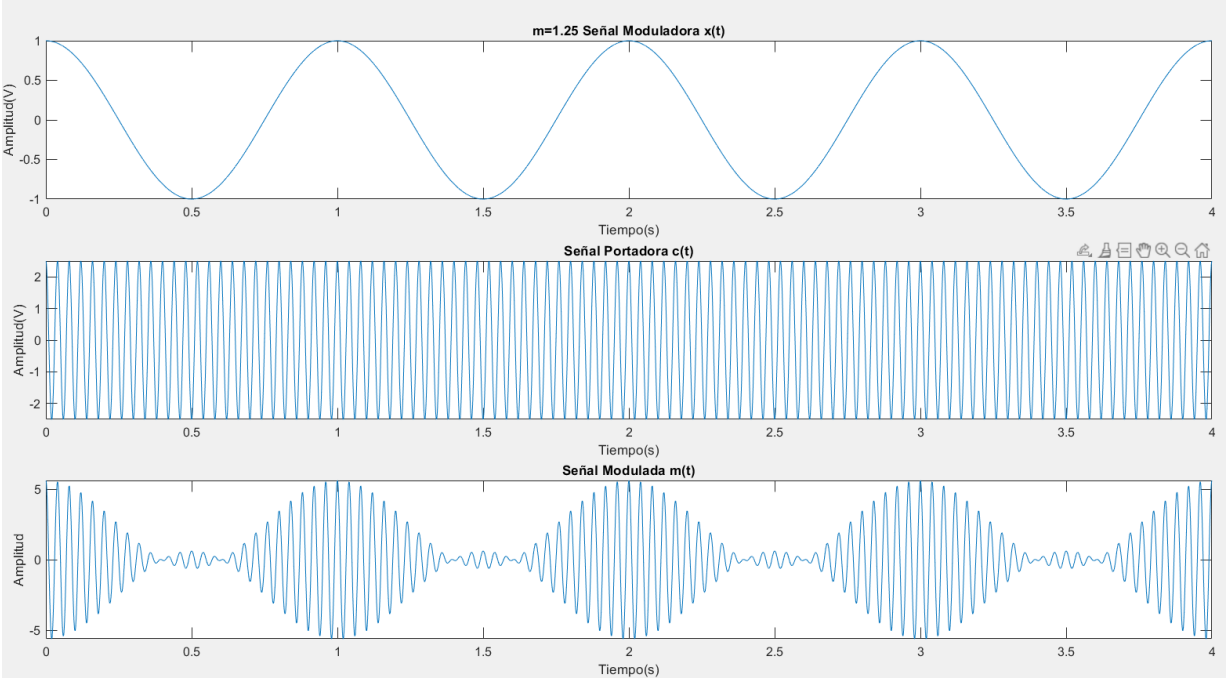


Figura 3.16 Representación temporal de la Señal Moduladora, Portadora y Modulada en AM cuando m≥1

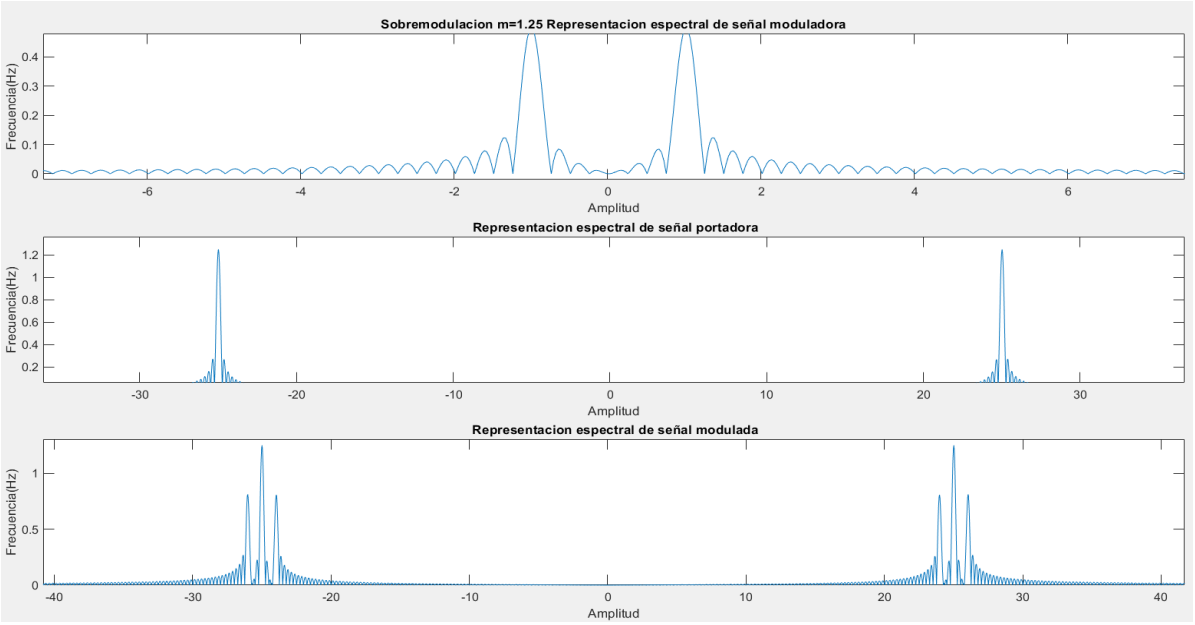


Figura 3.17 Representación espectral de la Señal Moduladora, Portadora y Modulada en AM cuando m≥1

Observando y analizando las figuras representadas, podemos observar una serie de defectos que presenta la modulación de amplitud.

El primero de ellos es que una señal AM requiere el doble de ancho de banda comparado con la señal moduladora.

El segundo defecto está relacionado con la potencia que se transmite, y es que, no solo estamos invirtiendo potencia en las bandas laterales que es lo que interesa, sino que también se invierte potencia en la portadora, y eso es potencia desperdiciada, además si utilizamos un índice de modulación '*m*' inferior o igual a 1, estamos invirtiendo más potencia en la portadora que incluso en las bandas laterales, podríamos invertir más potencia en las bandas laterales aumentando el valor de '*m*' a costa de que nos acerquemos a la sobre modulación, cosa que no interesa.

En base principalmente a estos defectos, surgen posteriormente otro tipo de modulaciones lineales intentando hacer más eficaz la modulación AM.

Cabe destacar, que, en el proyecto no se han generado modulaciones AM con un índice de modulación superior a 1 para su clasificación.

3.3.1.2 Modulación en Doble Banda Lateral con Portadora Suprimida

En la modulación de DBS-SC, como su nombre indica, no se transmite la portadora, es decir, no se invierte potencia para su transmisión, no malgastando esa potencia comentada anteriormente.

Esto se debe a la expresión en cuestión de este tipo de modulación, siendo la misma:

$$m_{DSB-SC}(t) = A_c \cdot x(t) \cdot \cos(2\pi f_c \cdot t) \quad (21)$$

Obteniendo fácilmente las componentes en fase y cuadratura y por ende el equivalente paso bajo para la señal DSB-SC.

$$I(t)_{DSB-SC} = A_c \cdot x(t) \quad (22)$$

$$Q(t)_{DSB-SC} = 0 \quad (23)$$

$$V_y(t)_{DSB-SC} = A_c \cdot x(t) + j0 \quad (24)$$

A diferencia de $I(t)_{AM}$ cuando m≤1, $I(t)_{DSB-SC}$ puede tomar valores negativos, afectando esto a la fase de la señal, debiendo usar receptores más complejos (receptores coherentes).
A continuación, se muestran las formas de onda temporales de la señal moduladora, portadora y señal modulada en DSB-SC (Figura 3.18).

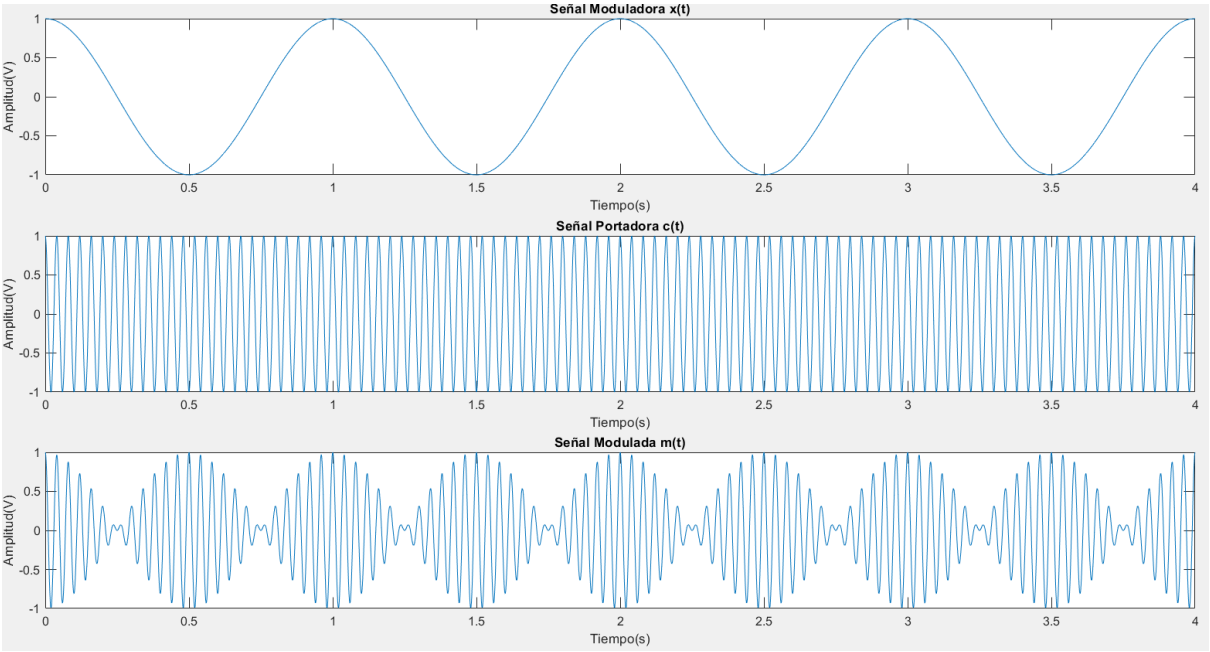


Figura 3.18 Representación temporal de la Señal Moduladora, Portadora y Modulada en DSB-SC

Por último, se muestra el espectro en frecuencia (Figura 3.19).

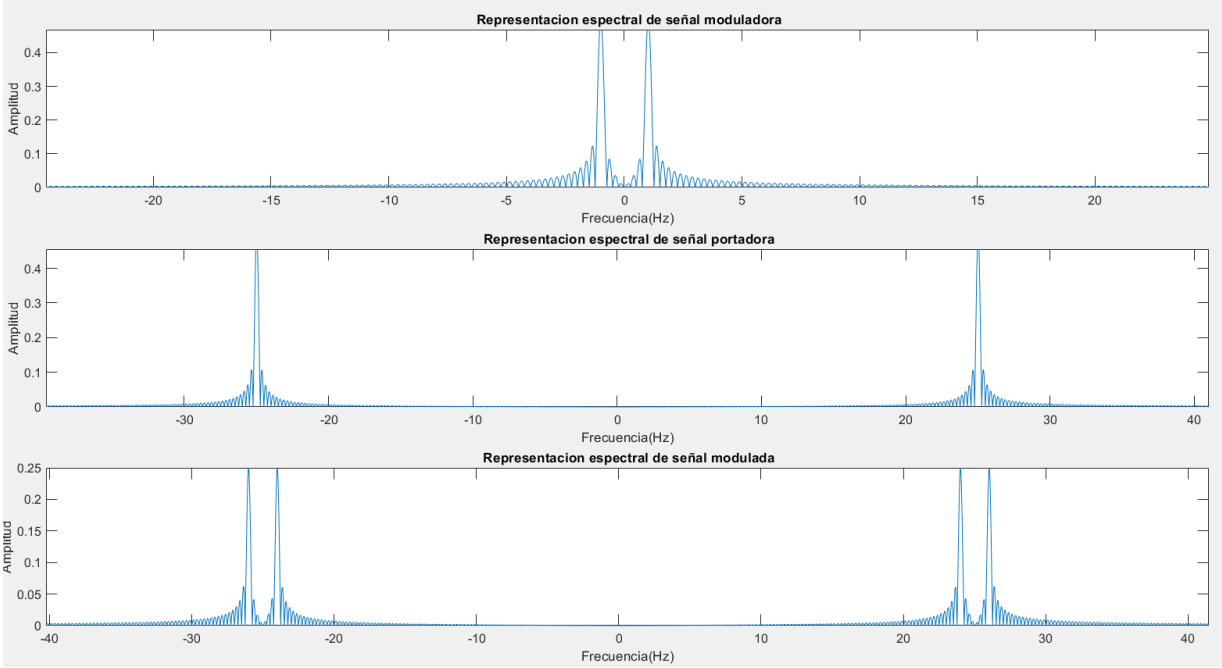


Figura 3.19 Representación espectral de la Señal Moduladora, Portadora y Modulada en DSB-SC

3.3.1.3 Modulación en Banda Lateral Inferior

La modulación LSB, es un tipo de modulación donde no solo no se transmite la portadora al igual que en DSB-SC, sino que tampoco se transmite la banda lateral superior, y es que, la transmisión de dos bandas laterales completamente idénticas es otro malgasto de potencia, son redundantes.

A continuación, se muestra la expresión de una modulación en LSB:

$$m_{LSB}(t) = A_c \cdot x(t) \cdot \cos(2\pi f_c \cdot t) - A_c \cdot TH\big(x(t)\big) \cdot \sin(2\pi f_c \cdot t) \qquad (25)$$

Siendo '*TH(x(t))*' la Transformada de Hilbert (*Hilbert Transform - HT*) de la moduladora.

La HT, no es en sentido estricto una '*Transformación*' ya que no cambia a otro dominio, realmente la HT podría verse mejor como una operación de filtrado, ya que va a permitir discriminar señales atendiendo al criterio de la fase, permitiéndonos separar señales.

La HT varía la información en fase de la señal, introduciendo un desfase a la misma de 90º, permaneciendo inalterado el módulo de la misma.

La expresión de la transformada de Hilbert vista desde los dominios tiempo, frecuencia corresponde a:

$$TH(x) = \hat{x}\,(t) = x(t)conv\left(\frac{1}{\pi t}\right) = \frac{1}{\pi}\cdot\int_{-\infty}^{+\infty}\frac{x(\tau)}{t-\tau}\,d\tau \qquad (26)$$

Siendo:

$conv$, la operación de convolución.

$$H_{TH}\,(f) = -j\cdot sgn(f) = \begin{cases} -j & f > 0 \\ +j & f < 0 \end{cases} \qquad (27)$$

Las componentes en fase y cuadratura para la modulación LSB, así como su envolvente compleja serían:

$$I(t)_{LSB} = A_c\cdot x(t) \qquad (28)$$

$$Q(t)_{LSB} = -A_c\cdot TH(x(t)) \qquad (29)$$

$$V_y(t)_{LSB} = A_c\cdot x(t) - jA_c\cdot TH(x(t)) \qquad (30)$$

A diferencia de los espectros mostrados anteriormente, a continuación, se muestra el espectro en frecuencia de la envolvente compleja (Figura 3.20), es decir, de la señal paso bajo equivalente a la señal paso banda:

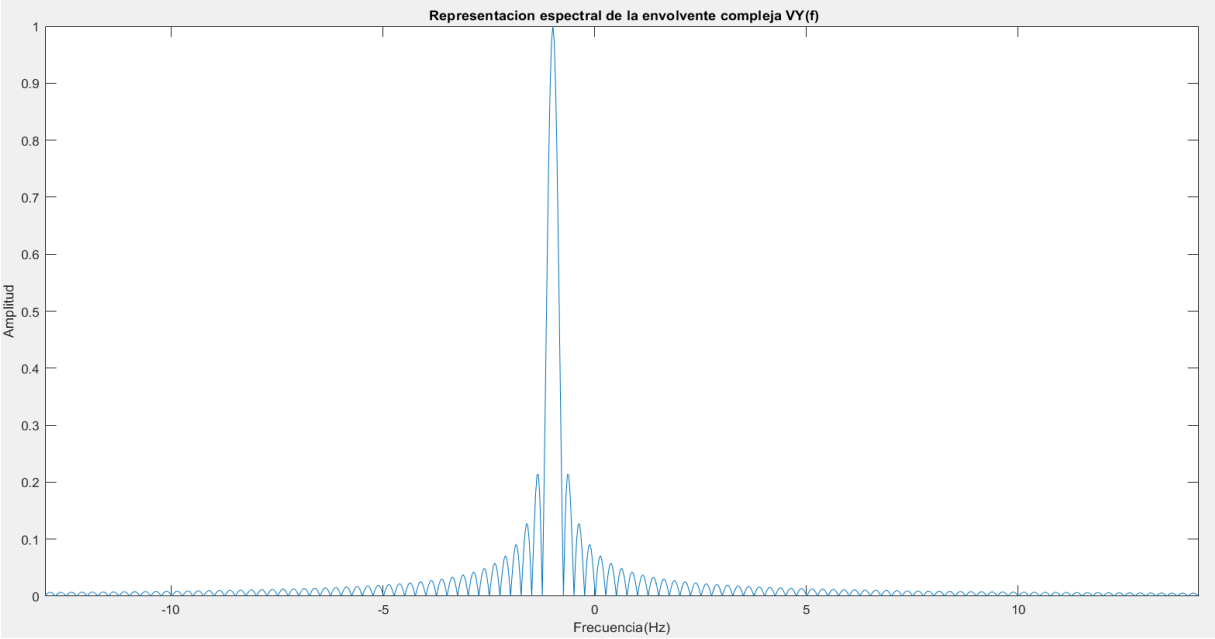


Figura 3.20 Espectro en frecuencia de la envolvente compleja VY(f) de la señal LSB

Como se observa en la figura, la mayor parte de la potencia queda contenido por debajo de la frecuencia de portadora.

3.3.1.4 Modulación en Banda Lateral Superior

Por último, en lo que respecta a modulaciones lineales, se encuentra la modulación USB, esta modulación es similar a la LSB a excepción de que ahora la mayor parte de la potencia quedará contenida por encima de la frecuencia de portadora.

A continuación, se muestra la expresión de una modulación en USB:

$$m_{USB}(t) = A_c \cdot x(t) \cdot \cos(2\pi f_c \cdot t) + A_c \cdot TH\big(x(t)\big) \cdot \sin(2\pi f_c \cdot t) \qquad (31)$$

Obteniendo de la expresión (ec.31) las componentes en fase y cuadratura para la modulación USB y, por ende, la envolvente compleja como:

$$I(t)_{USB} = A_c \cdot x(t) \qquad (32)$$

$$Q(t)_{USB} = A_c \cdot TH(x(t)) \qquad (33)$$

$$V_y(t)_{USB} = A_c \cdot x(t) + jA_c \cdot TH(x(t)) \qquad (34)$$

Mostrando a continuación (Figura 3.21) el espectro en frecuencia de la envolvente compleja paso bajo:

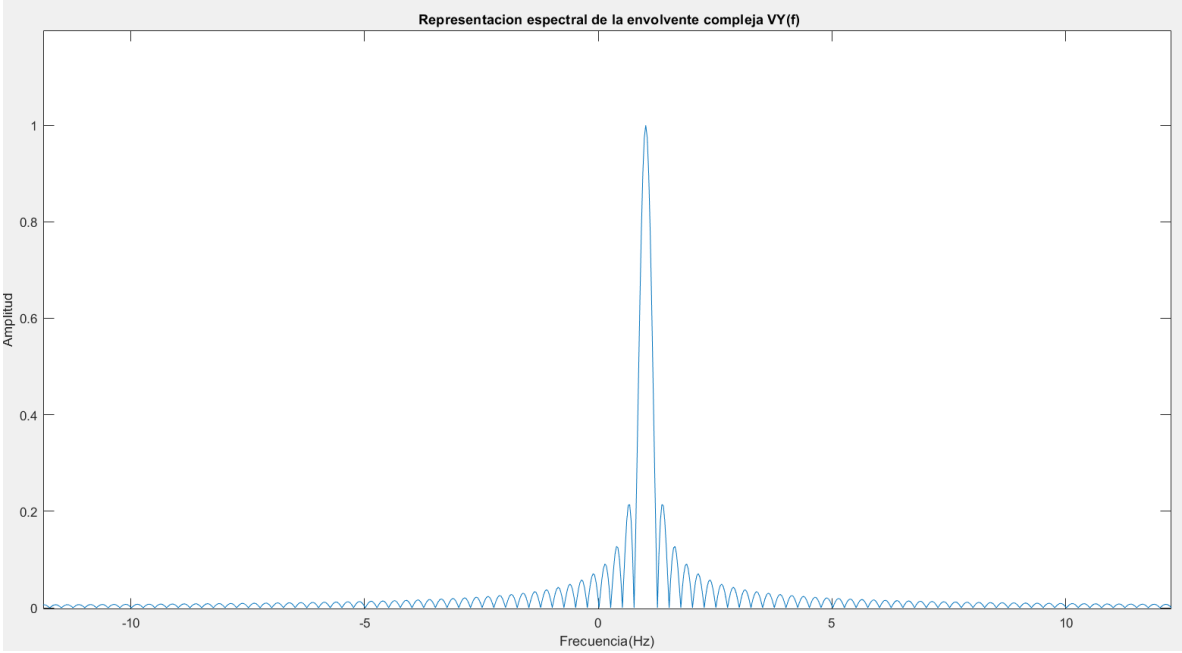


Figura 3.21 Espectro en frecuencia de la envolvente compleja VY(f) de la señal USB

3.3.2 Modulaciones angulares

Cualquier señal tiene tres parámetros fundamentales (Amplitud, Frecuencia, Fase).

Como se ha visto en el punto 3.3.1, las modulaciones lineales se producen cuando el parámetro que varía de la portadora es la amplitud y, dicha amplitud de la señal modulada era mayor en los instantes de tiempo donde la amplitud de la moduladora era también mayor. En el caso de las modulaciones angulares, el parámetro que variará en función de la moduladora será la frecuencia o lo que es lo mismo, la información irá encriptada en las modificaciones de frecuencia de la portadora, donde, los instantes de tiempo en los cuales la amplitud de la moduladora es elevada, serán traducidos a una mayor frecuencia en la portadora.

Las modulaciones angulares son procesos no lineales, esto es que, aparecerán componentes espectrales que antes de modular no estaban.

Se verá que el ancho de banda que se suele utilizar para una modulación angular es bastante mayor que para una modulación lineal.

3.3.2.1 Modulación de Frecuencia

Si recordamos, en algunas de las modulaciones lineales, la componente en cuadratura era 0, es decir, el seno de la portadora no aparece, sin embargo, ahora en FM tendremos tanto componente en fase como en cuadratura.

Partiendo de la expresión general de una modulación y particularizando para FM obtenemos:

$$m_{FM}(t) = A_c \cdot \cos\big(\omega_c(t) + \phi(t)\big) = A_c \cdot \cos\left(\psi(t)\right) \qquad (35)$$

Donde:

- $\psi(t) = \omega_c(t) + \phi(t)$ es la Fase Instantánea de la señal modulada

Aplicando de nuevo la identidad trigonométrica (ec.2) a la (ec.35) queda:

$$m(t) = A_c \cdot \cos\big(\phi(t)\big) \cdot \cos\big(\omega_c(t)\big) - A_c \cdot sin\big(\phi(t)\big) \cdot sin(\omega_c(t)) \qquad (36)$$

Obteniendo de la expresión (ec.36) las componentes en fase y cuadratura para la modulación FM, así como su envolvente compleja:

$$I(t)_{FM} = A_c \cdot \cos\big(\phi(t)\big) \quad (37)$$

$$Q(t)_{FM} = A_c \cdot sin\big(\phi(t)\big) \quad (38)$$

$$V_y(t)_{FM} = A_c \cdot \cos\big(\phi(t)\big) + jA_c \cdot sin\big(\phi(t)\big) \quad (39)$$

Siendo:

$$\phi(t) = 2\pi \cdot f_d \cdot \int_0^{+t} x(\tau)\, d\tau \quad (40)$$

$$\Delta f = f_d \cdot A_m \quad (41)$$

$$f_d = \beta \cdot \frac{f_m}{A_m} \quad (42)$$

Donde:

- β es índice de modulación en FM
- Δf es la máxima desviación de frecuencia producida por la moduladora
- f_d es la desviación instantánea de frecuencia
- A_m es la amplitud de la moduladora
- f_m es la frecuencia de la portadora

A continuación, se analizará el efecto que tiene el valor del índice de modulación β para la señal FM tanto en el dominio del tiempo como de frecuencia, para una frecuencia de moduladora de 1 Hz, una frecuencia de portadora de 25 Hz, empleándose una amplitud de moduladora normalizada, por tanto, $\Delta f = f_d$. Posteriormente se comentarán los resultados y se sacarán conclusiones sobre el efecto de este parámetro sobre la señal modulada.

- Caso β grande

  Hay diferentes formas de aumentar el valor de β, una de ellas es aumentando el valor de la desviación instantánea de la fase.

Con un valor elevado de β, obtenemos la siguiente figura en el tiempo de la señal modulada en FM. (Figura 3.22).

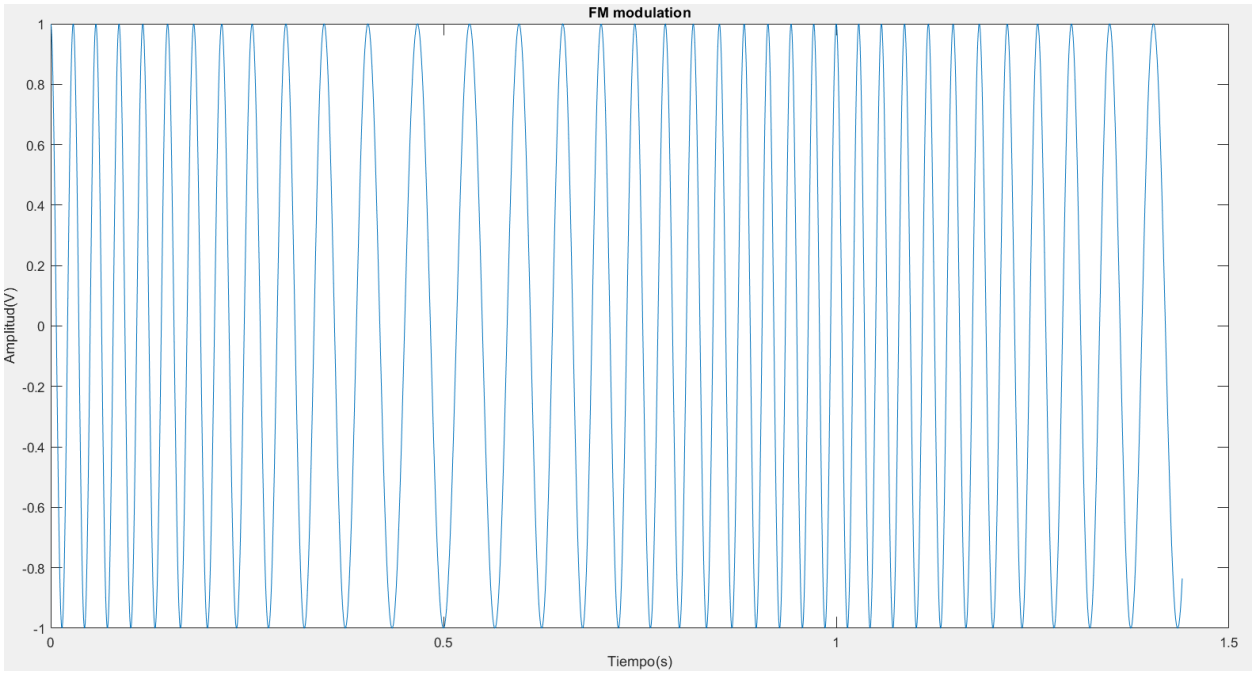


Figura 3.22 Forma de onda temporal de la señal modulada en FM con un valor de β elevado

Por otro lado, obtenemos la señal modulada en el dominio de la frecuencia (Figura 3.23).

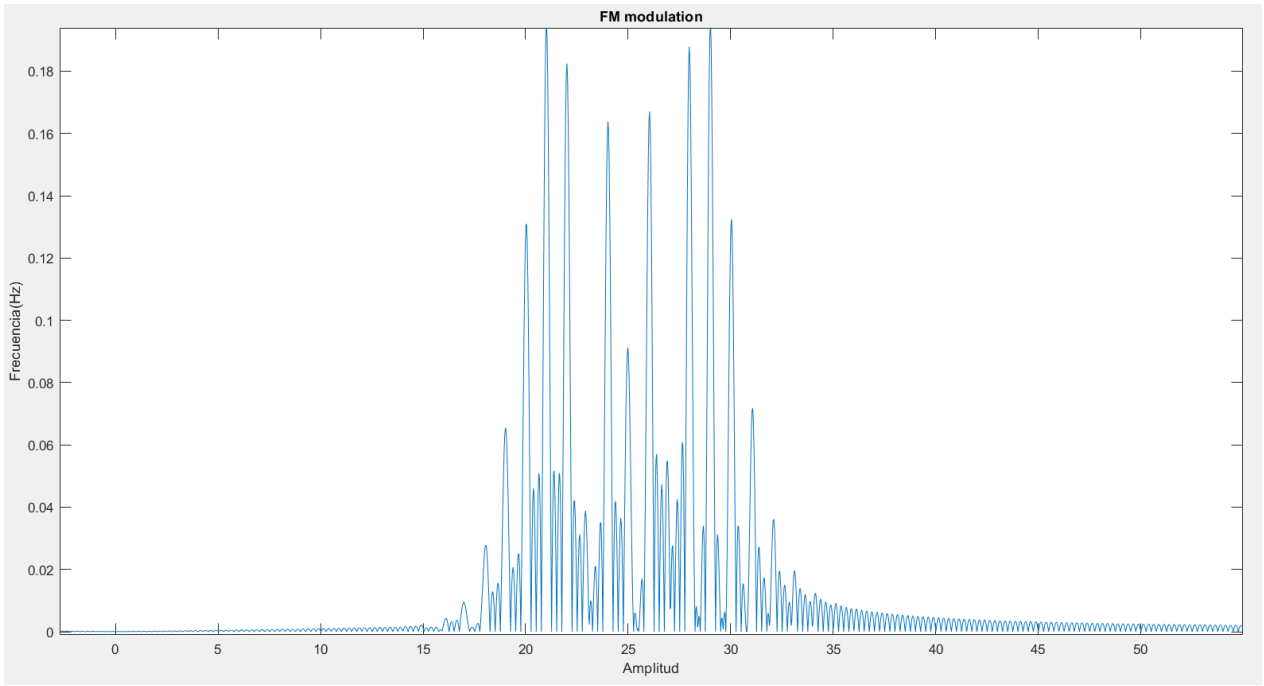


Figura 3.23 Espectro FM con valor de β elevado

- Caso β pequeño

  Para el caso opuesto, obtenemos la siguiente forma de onda temporal (Figura 3.24).

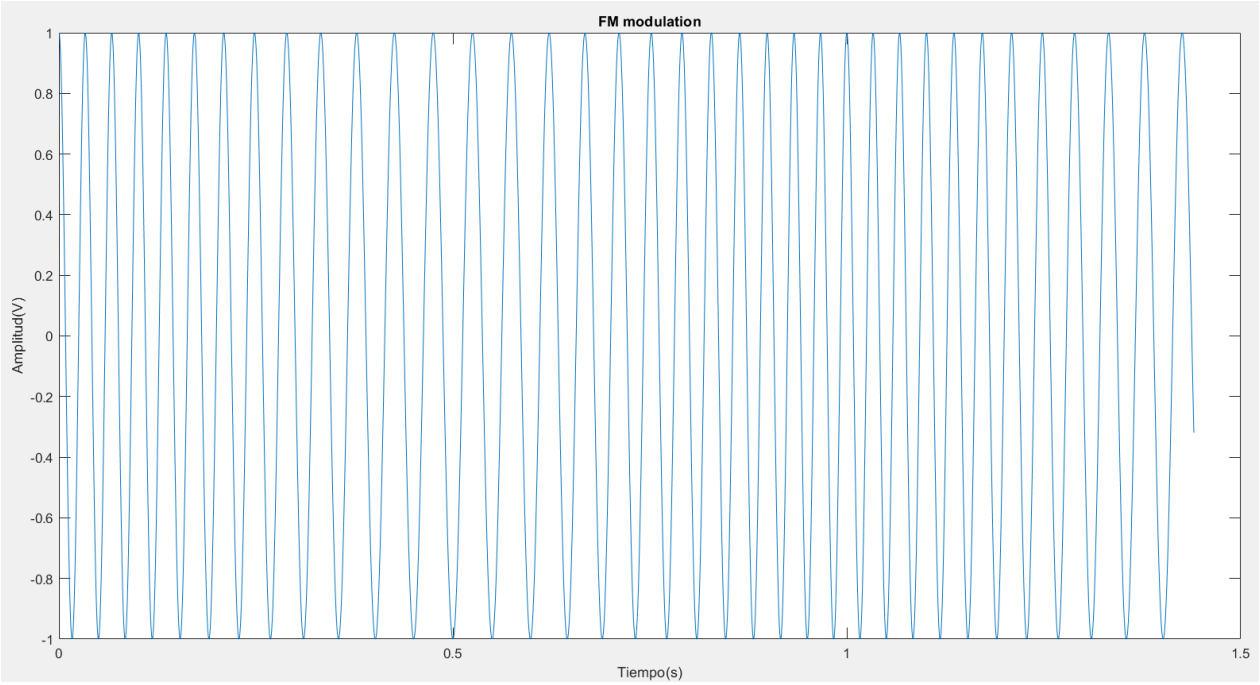


Figura 3.24 Forma de onda temporal de la señal modulada en FM con un valor de β pequeño

Por último, se muestra el espectro de la señal FM con un valor de β pequeño (Figura 3.25).

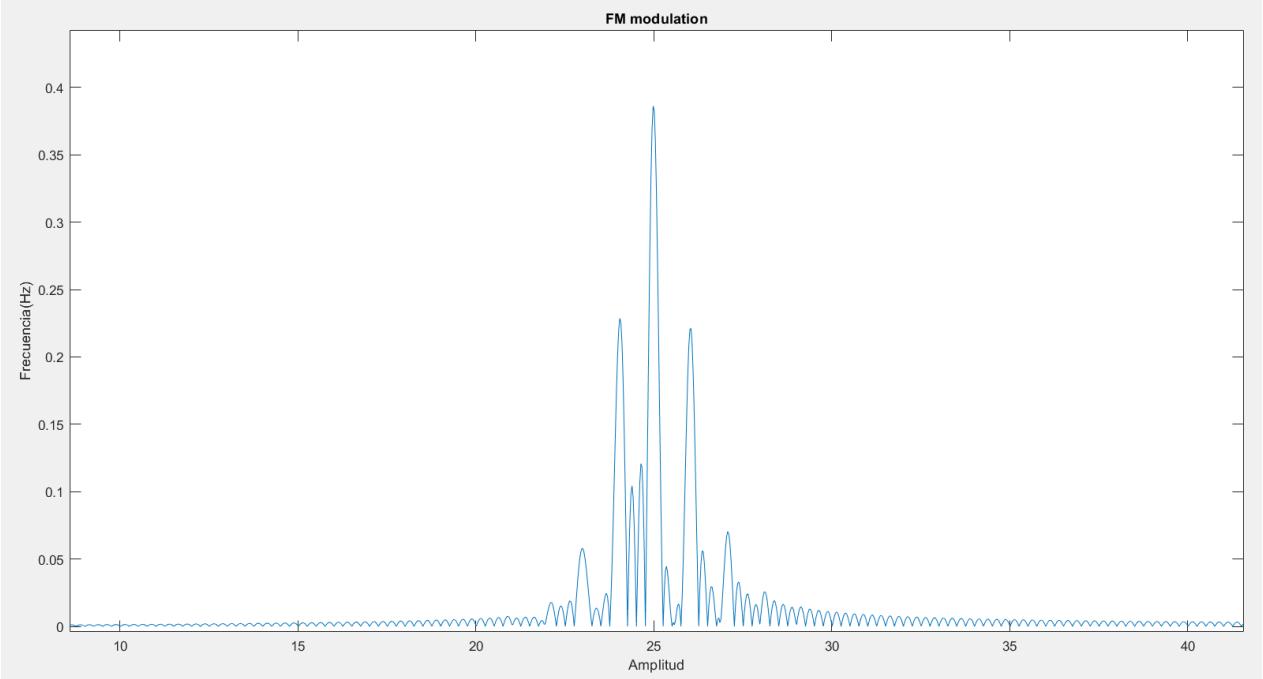


Figura 3.25 Espectro FM con valor de β pequeño

A continuación, se sacan una serie de conclusiones sobre el efecto del parámetro β en la señal modulada.

Analizando las formas de onda temporales, cuando tenemos un valor de β elevado, la frecuencia de la señal modulada en los intervalos donde la moduladora tiene una amplitud máxima, son mucho mayores que cuando usamos un valor de β pequeño como era de esperar.

Traducido al dominio de la frecuencia, se observa que cuando β es elevado, tenemos un mayor ancho de banda, de la señal modulada, apareciendo nuevas componentes espectrales, reduciéndose la potencia en la portadora, es decir, esa disminución de potencia de la portadora por el hecho de aumentar β, no se pierde, sino que es invertida en nuevas componentes espectrales, de ahí que obtenemos un mayor ancho de banda. Sin embargo, al disminuir β observamos como el ancho de banda de la señal modulada decrece, observando una mayor potencia en torno a la portadora, y es que, aunque se reduzca el ancho de banda, no estamos perdiendo energía al no aparecer esas nuevas componentes espectrales, sino que esa potencia está siendo invertida en la portadora.

Otra conclusión que puede sacarse a partir de los espectros mostrados, es que, si tomamos como moduladora un tono, aparecerán '*deltas*' localizadas en la f_c y en múltiplos de la frecuencia de moduladora separados por f_c, es decir:

$$f_c \pm n \cdot f_m \qquad (n = 0,1,2 \dots) \tag{43}$$

Conforme mayor sea el valor de 'n' menor potencia se observará.

3.4 Aprendizaje Automático

El aprendizaje automático de máquinas (*Machine Learning - ML*) tiene como principal objetivo desarrollar algoritmos que aprendan a partir de un conjunto de datos o de observaciones, de tal manera que, una vez el algoritmo haya sido capaz de aprender a partir de esos datos, sea posible establecer hipótesis o predicciones sobre nuevas observaciones, y de esta forma, tomar decisiones de forma automática.

El aprendizaje automático es usado cuando se desconoce la fórmula o algoritmo entre entrada y salida deseada a esa entrada.

Este tipo de algoritmos presentan de forma general tres fases fundamentales, aunque como se verá posteriormente, dependiendo del tipo de aplicación en cuestión, y del tipo de algoritmo automático seleccionado habrá o no más fases.

Estas tres fases fundamentales son: fase de entrenamiento, fase de evaluación y testeo.

La fase de entrenamiento consiste principalmente en proporcionar observaciones o ejemplos al sistema, a partir de los cuáles debe aprender.

Para llevar a cabo este entrenamiento, surgen dos alternativas [24]:

- Aprendizaje Automático Supervisado (*Supervised Machine Learning)*
- Aprendizaje Automático No Supervisado (*Unsupervised Machine Learning)*

El primero consiste en proporcionar al sistema un conjunto de datos de aprendizaje etiquetados, es decir, contienen la respuesta correcta o lo que es lo mismo, los datos de entrenamiento deben ser provistos en términos de entradas/salidas, para que de esta forma el sistema adapte sus parámetros intentando encontrar una relación óptima entre la entrada y la salida.

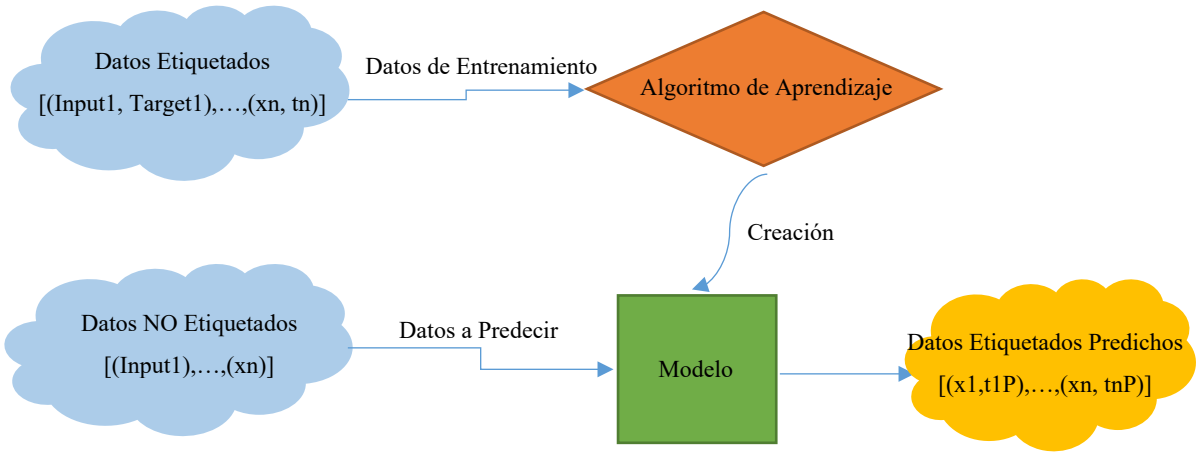


Figura 3.26 Esquemático de Funcionamiento del Aprendizaje Supervisado

Por el contrario, en el aprendizaje no supervisado, los datos de entrenamiento no están etiquetados, dando lugar a algoritmos de *clustering* que permitan descubrir patrones de clasificación a partir del conjunto de observaciones, es decir, el algoritmo deberá encontrar patrones a partir de los datos no etiquetados al agrupar e identificar similitudes entre ellos.

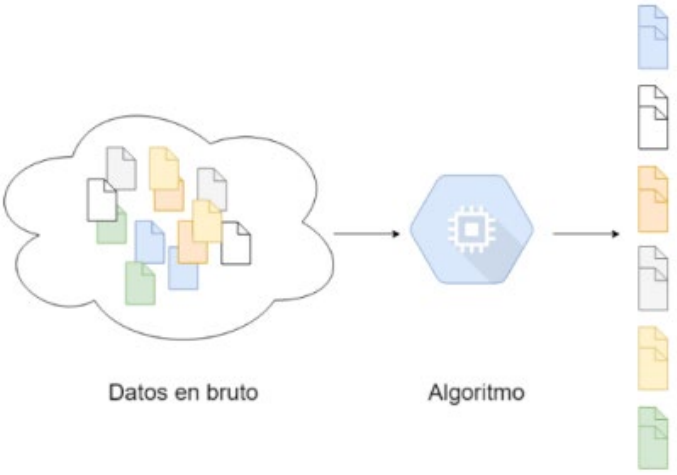


Figura 3.27 Esquemático de Funcionamiento del Aprendizaje No Supervisado

Cabe destacar que para el desarrollo del AMC implementado en el trabajo, se ha usado aprendizaje supervisado, ya que, dicho algoritmo a menudo es usado para clasificación de modulaciones [25], [2], [3].

La **fase de evaluación** y **testeo** comienza una vez el entrenamiento ha sido finalizado. Estas fases consisten en proporcionar al sistema unos datos de entrada, para que el propio sistema los etiquete, en la fase de evaluación se deberá saber a priori las verdaderas etiquetas de esos datos de entrenamiento, de esta forma podremos comprobar los resultados ofrecidos por el sistema, sin embargo, en la fase de testeo, no se conocerán dichas etiquetas.

En base a estas fases podremos diferenciar entre dos tipos de errores:

- Falsa Alarma (*False Alarm Rate*): Es la probabilidad que tiene el sistema de que una determinada entrada sea aceptada erróneamente como una determinada salida perteneciente a otra clase.
- Falso Rechazo (*False Rejection Rate*): Es la probabilidad que tiene el sistema de que una determinada entrada sea rechaza erróneamente como una determinada salida perteneciente a esa misma clase.

3.4.1 Algoritmos de Aprendizaje Supervisado

Una vez comentada de forma general la consistencia del aprendizaje automático, así como sus fases genéricas, se presentarán los diferentes tipos de algoritmos de aprendizaje supervisado.

Antes de presentar dichos algoritmos se hará un inciso detallando una fase necesaria para el desarrollo del proyecto.

Como se dijo previamente, dependiendo de la aplicación para la cual se use el algoritmo de aprendizaje automático, quedarán involucradas más fases, en nuestro caso, con

el objetivo de clasificar correctamente un tipo de señal modulada, se hace necesaria una fase de preprocesado de la información. Este procesado consiste en extraer las características clave (*Key Feature Extraction*) [25] de las señales paso bajo, para, posteriormente definir un data set de con el que entrenar el algoritmo.

En posteriores apartados se presentarán dichas características.

A continuación, queda definido el esquemático completo del proceso de desarrollo del clasificador supervisado a implementar (Figura 3.28).

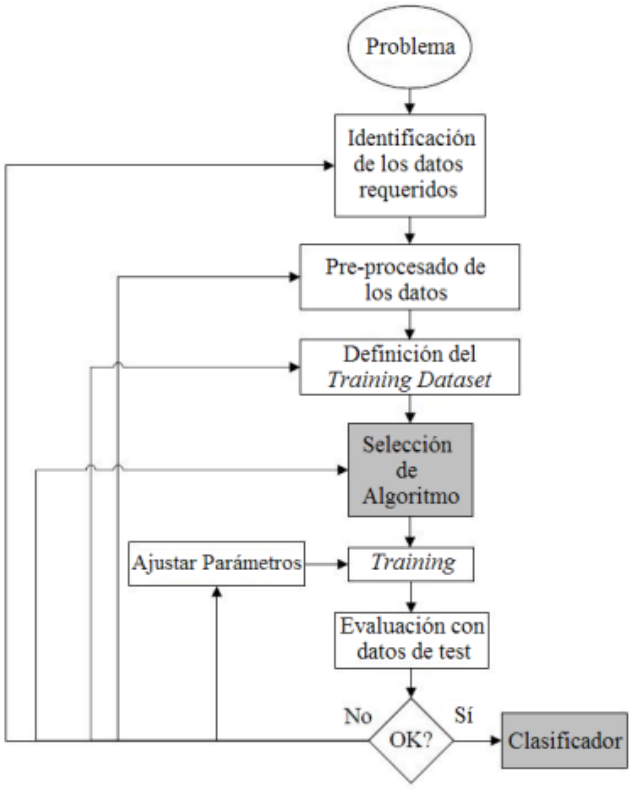


Figura 3.28 Proceso de Desarrollo del Clasificador Supervisado a Implementar

Por último, se debe seleccionar un algoritmo de aprendizaje supervisado.

Los algoritmos de aprendizaje supervisado, incluyen entre otros, los siguientes algoritmos:

- Algoritmos de redes neuronales artificiales (*Artificial Neuronal Network - ANN*)
- Máquina de Vectores de Soporte (*Support Vector Machine - SVM*)
- Árboles de Decisión (*Decision Trees*)

En base a numerosas búsquedas comparativas entre los diferentes algoritmos supervisados [26], [27], se han seleccionado las redes neuronales como algoritmo supervisado para la creación del AMC.

3.5 Redes Neuronales Artificiales

A continuación, se explicará y detallará el algoritmo supervisado seleccionado para la realización del AMC.

3.5.1 Introducción e Inspiración Biológica de la Red Neuronal Artificial

De forma simplicista, se podría considerar que el cerebro de un ser vivo, es el órgano que procesa la información que llega tanto desde el propio cuerpo como del exterior, para, en base a dicha información, actuar a partir de reacciones químicas e impulsos eléctricos.

Por otro lado, podemos ver esta semejanza aplicada a procesadores o microprocesadores, tratando la información recibida en base a su programación en forma de electricidad.

Hay operaciones en las que el cerebro humano es mucho más eficiente que los procesadores electrónicos, como el reconocimiento facial, sin embargo, para otro tipo de operaciones, como pueden ser cálculos, los microprocesadores son mucho más eficientes que el cerebro humano, pudiendo ser incluso un millón de veces más veloces.

Otra diferencia entre un computador digital y el cerebro humano es la forma en la que ambos gestionan la información. El cerebro humano, tiene un sistema de gestión de información *complejo, no lineal y paralelo*, por tanto, es capaz de realizar muchas operaciones de forma simultánea a diferencia de los computadores, que, gestionan la información de manera secuencial. [28]

Por estos motivos se han realizado arquitecturas en el mundo de la inteligencia artificial que se asemejan a los sistemas neurobiológicos, es decir, las redes neuronales están motivadas en modelar la forma de procesamiento de la información de sistemas nerviosos biológicos, donde, la base fundamental de estas redes artificiales son las neuronas o nodos que serán las encargadas de realizar los procesos de cálculo y transmisión de la información a través de la red, de forma **similar** a como un cerebro humano lo haría.

La palabra '*similar*' queda resaltada ya que, nunca una red neuronal artificial, por muchos datos de entrenamiento o por mucha complejidad que tenga, funcionará igual que los procesos '*analógicos*' que ocurren en el cerebro humano.

En conclusión, las redes neuronales artificiales son algoritmos que tratan de aprovechar la arquitectura, capacidad y funcionamiento único del cerebro para resolver problemas, aprovechando la gran velocidad de ejecución y cálculo de los sistemas electrónicos.

- **Inspiración Biológica**

Las redes neuronales artificiales, como su nombre indica, están inspiradas en las neuronas de los seres vivos.

Las neuronas son un tipo de células del sistema nervioso, típicamente poseen el aspecto y las partes mostrados en la Figura 3.29.

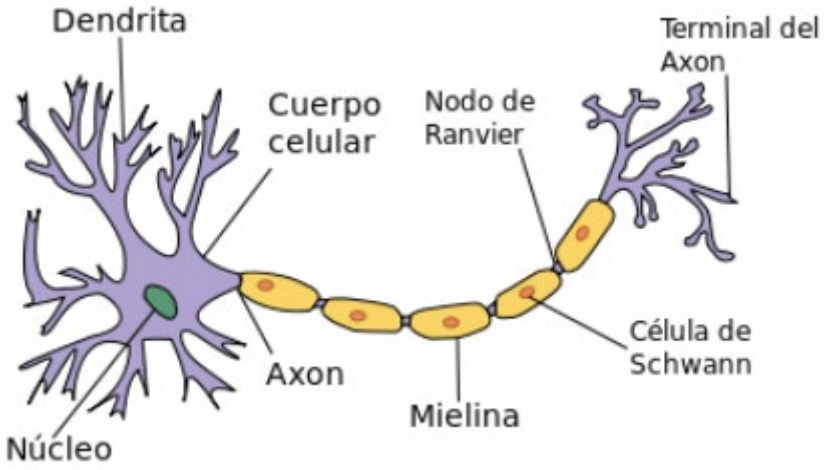


Figura 3.29 Modelo fisiológico de una neurona [29]

En las neuronas se distinguen tres partes fundamentales: Dendritas, soma o cuerpo celular y axón.

Las dendritas actúan como un canal de entrada de señales que provienen del exterior hacia el cuerpo celular de la neurona, mientras que el axón actúa como un canal de salida, ramificándose en filamentos mediante los cuales establecerá conexión con los cuerpos celulares de otras neuronas, esta conexión entre neuronas se llama sinapsis.

Una de las características más importantes de las conexiones sinápticas es la plasticidad, esta plasticidad es la capacidad para alterar a largo plazo la intensidad de las conexiones entre neuronas.

A continuación, se muestra una figura (Figura 3.30) de una neurona en el mundo computacional, para, posteriormente comentar las analogías entre ambos mundos, mundo biológico y mundo computacional, en base a los párrafos previos.

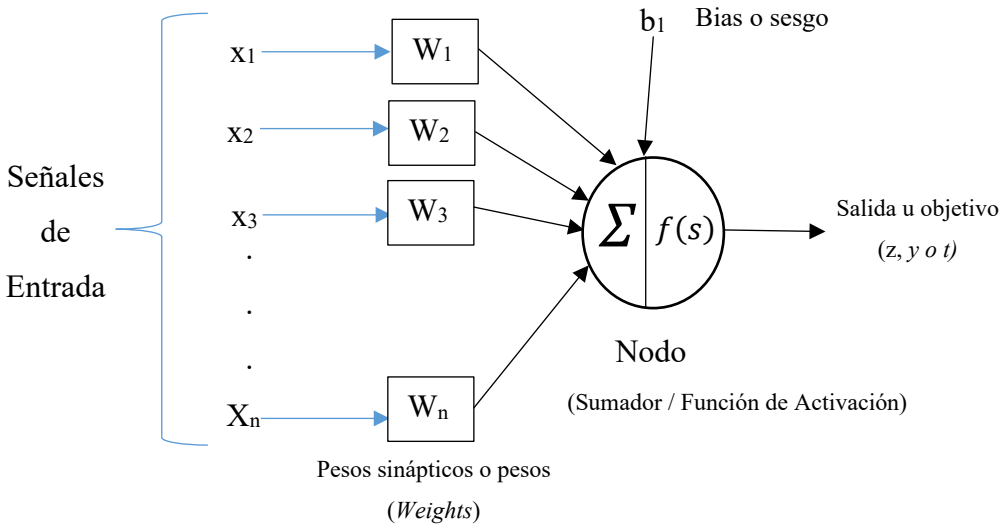


Figura 3.30 Neurona Artificial o Perceptrón Simple

Mediante la breve descripción del modelo fisiológico de una neurona biológica y, junto a ambas figuras se establecerán las similitudes a gran escala entre las redes neuronales artificiales y las biológicas.

Las señales que entran a la sinapsis, es decir, las señales externas que son transportadas a través de las dendritas hasta el axón, son las entradas de nuestra red neuronal.

Estas señales que entran a la sinapsis en el mundo biológico y, que se corresponden a las entradas de la red neuronal en el mundo computacional, serán ponderadas (atenuadas o excitadas) debido a esa plasticidad en el mundo biológico o, debido a unos pesos y bias en el mundo computacional, donde, el procesamiento de esa señal se realizará en el núcleo de la neurona en el mundo biológico, o bien en el nodo del mundo computacional.

A continuación, se explicará esa plasticidad o bien esos pesos y sesgos, con un ejemplo de forma que quede claro cuál es su función.

Cualquier persona, antes de comprar una casa por ejemplo, tiene en cuenta una serie de factores (entradas) los cuales determinarán el resultado de si la compra o no, cada uno de estos factores tendrá un peso distinto o similar, por ejemplo, el factor de precio de la casa, tendrá un mayor peso asignado que el factor número de cuartos de baño de la casa, por tanto al tener un mayor peso asignado, el resultado final se verá mayor influenciado por este factor debido a esa mayor ponderación o peso.

3.5.2 Características y aplicaciones

Las ANN, al margen de 'parecerse' al cerebro, presentan una serie de características propias del cerebro. Por ejemplo, las ANN aprenden de la experiencia, generalizan de

ejemplos previos a ejemplos nuevos, abstraen las características principales de una serie de datos y tienen una gran tolerancia a fallos.

A continuación, se muestran en detalle dichas características de las ANN:

- Las ANN aprenden: Adquieren conocimiento por medio del estudio, ejercicio o experiencia, quedando este almacenado, al igual que ocurre en el cerebro, además pueden cambiar su comportamiento en función del entorno.
- Las ANN generalizan: Estas redes pueden ofrecer dentro de un margen, respuestas correctas a entradas que presentan pequeñas variaciones debido a efectos del ruido o distorsión.

  Ampliando esto al proyecto, este ruido estará siempre presente en cada muestra de entrada a la ANN, siendo para cada muestra totalmente diferente al ser aleatorio, aun así, la ANN será capaz de clasificar correctamente los segmentos debido a esa generalización.
- Abstracción: Las ANN son capaces de aislar o considerar por separado las cualidades o peculiaridades de un objeto, es decir, son capaces de abstraer la esencia de un conjunto de entradas que aparentemente no presentan aspectos comunes entre sí.
- Poseen un alto nivel de tolerancia a fallas, es decir, pueden sufrir daños considerables y continuar teniendo un buen comportamiento, al igual como ocurre en los sistemas biológicos.

  La veracidad de esta característica depende principalmente del número total de características del objeto en cuestión usadas para entrenar la red, contra mayor número de características mayor tolerancia a fallos, traducido esto a un cerebro humano, una persona puede sufrir problemas de pérdida de memoria, contra mayores sean esas pérdidas de memoria, la tolerancia a fallos será mucho menor ya que, a la hora de tomar decisiones, contará con un menor número de características en las cuales basarse para decidir correctamente.

Estas características que presentan las redes neuronales, así como las ventajas que ofrecen los sistemas computacionales, permiten una extensa variedad de aplicaciones.

Algunas de las áreas de aplicación de las ANN son [30]:

- Análisis y Procesado de Señales: Permitiendo realizar predicciones, modelar sistemas o filtrar diferentes tipos de ruido.

- Reconocimiento de Imágenes y Caracteres.
- Compresión de datos de información como imágenes.
- Procesado del Lenguaje.
- Robótica.
- Diagnósticos Médicos.

3.5.3 Funcionamiento de las ANN

En los siguientes sub apartados, se detallará más aún el proceso de codificación de una neurona, explicando el proceso de suma que realiza una neurona para luego extenderse a él ¿Por qué se hace necesario usar redes neuronales? dando lugar a él ¿Por qué se hace necesario usar funciones de activación?, para finalmente concluir en ¿Cómo se ajustan los parámetros de la red para la minimización del error?

3.5.3.1 La Neurona

La complejidad de las redes neuronales, emerge de la interacción de muchas partes más simples trabajando de manera conjunta, en el caso de una red neuronal, a cada una de estas partes se le denomina neurona.

La neurona es la unidad básica de procesamiento que nos encontramos dentro de una red neuronal, que, como se ha comentado anteriormente es similar a una neurona biológica, estas tienen conexiones de entrada a través de las cuáles reciben estímulos externos, con estos valores, la neurona realizará una suma ponderada de ellos, donde esa ponderación de cada una de las entradas viene dada por el peso que se le asigna a cada una de esas conexiones de entrada, es decir, cada conexión que llega a nuestra neurona tendrá asociado un valor que servirá para definir la intensidad con la que cada variable de entrada afecta a la neurona.

Estos pesos y sesgos, son los parámetros de nuestro modelo y, serán los valores que se ajustarán para que nuestra red neuronal se adapte.

Ahora bien, dependiendo del número de entradas que se tengan en la neurona, se podría decir que internamente está realizando una regresión lineal simple o múltiple mediante ese sumatorio.

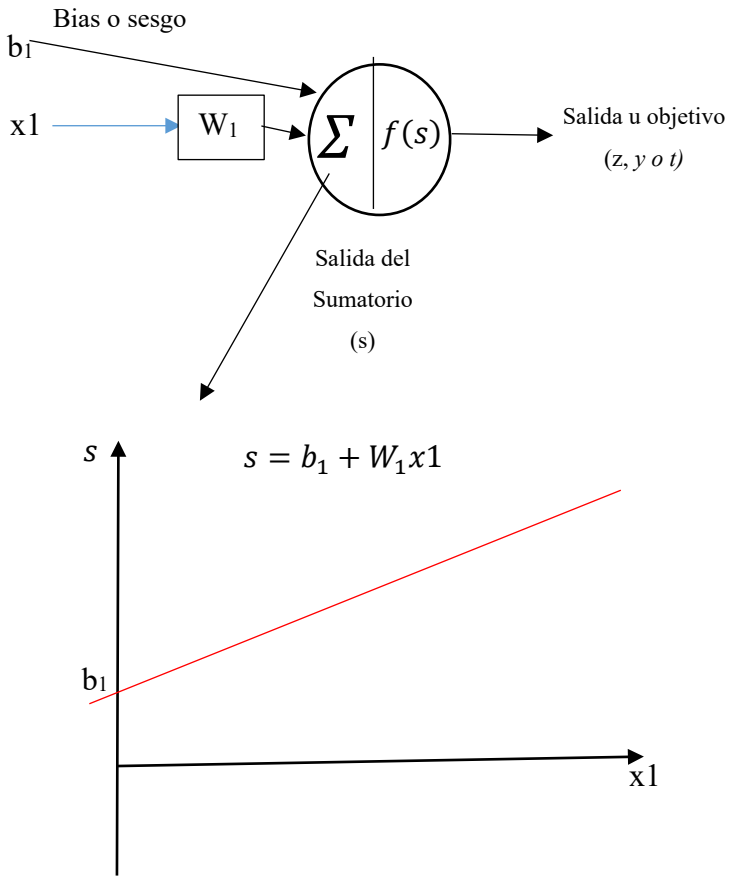


Figura 3.31 Modelo de Regresión Lineal Simple

En el espacio bidimensional contemplado la recta se ha definido mediante la suma del término independiente b_1 que muestra la altura a la cual la recta corta el eje *s*, por otro lado, se observa la pendiente w_1, la cual define la inclinación de la recta y la relación entre las variables *x*, *s*.

Esta recta sería creada por la neurona a partir del total de muestras usadas en el entrenamiento, permitiéndonos predecir valores incluso que no estaban presentes en las muestras de entrenamiento como era de esperar, por tanto, en este caso, cuantas más observaciones se tengan, mejor se ajustarán esos parámetros (pesos y sesgo) y, por ende, se realizará una mejor predicción.

Sin embargo, en el proyecto se contempla más de una única entrada, concretamente se tendrán cuatro entradas, si introducimos estas variables, no estaríamos ante un modelo de regresión simple, sino ante un modelo de regresión lineal múltiple.

Este modelo es similar al modelo anterior, salvo que la ecuación de salida del sumatorio de la neurona tendría más variables como se observa en (ec.44).

$$s = b_1 + W_1x1 + W_2x2 + W_3x3 + W_4x4 \quad (44)$$

Por tanto, el funcionamiento del sumatorio de la neurona ya no se trataría de encontrar la recta que mejor se ajuste a los datos bidimensionales, sino al mejor hiperplano que se ajuste a la '*nube*' de datos en espacios multidimensionales [31].

Aunque, como se verá posteriormente, se trabajará con formas **matriciales** tanto para los datos de entrada a la red neuronal, como para sus correspondientes salidas, incluyendo los parámetros. Esto hará que, a la hora de entrenar nuestro modelo, este se entrene de una forma mucho más eficiente, ya que las computadoras están especializadas para el procesamiento de matrices.

3.5.3.2 La Red Neuronal

Hasta ahora, únicamente se ha comentado el funcionamiento que realiza una neurona en cuanto a la suma ponderada, sin hacer hincapié en el funcionamiento de la función de activación, y es que, dichas funciones de activación, tienen un gran impacto o relevancia cuando se trabaja con redes neuronales como se comentará a continuación.

Hay dos formas distintas de añadir neuronas, una forma sería colocarlas en la misma capa, es decir, en la misma 'columna', las neuronas que se encuentren en la misma capa reciben la misma información de entrada de la capa anterior, otra forma, sería añadir neuronas colocando las mismas de forma secuencial, de esta forma la red neuronal puede adquirir un conocimiento jerarquizado, cuantas más capas sean añadidas, más complejidad tendrá el proceso que se elabore, esta profundidad recibe el nombre de Aprendizaje Profundo (*Deep Learning - DL*).

Para alcanzar el Aprendizaje Profundo debemos por tanto conectar múltiples neuronas de forma secuencial, aquí surge un problema, y es que, como se comentó anteriormente, el proceso que realiza cada neurona es una regresión lineal, y, se puede comprobar matemáticamente que, el efecto de sumar muchas operaciones de regresión lineal equivale únicamente a realizar una única operación. [32]

Aquí entran en juego las funciones de activación, básicamente si en la neurona se calculaba como valor de salida una suma ponderada de las entradas, ahora, esa suma ponderada pasará a ser la entrada a la función de activación, distorsionando dicha suma, añadiendo deformaciones no lineales. De esta forma se podrán concatenar de forma efectiva varias capas de neuronas entre sí, cosa que será esencial para el desarrollo del AMC, ya que, se adelanta que la red neuronal realizada en el proyecto, constará de tres capas de neuronas.

Por último, se presentarán en la tabla 3.5 distintos tipos de funciones de activación que normalmente son usadas en la práctica.

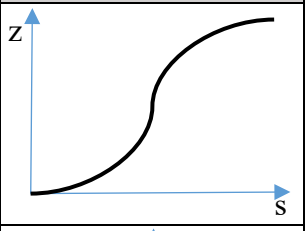
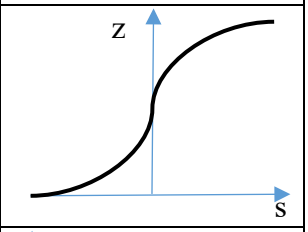
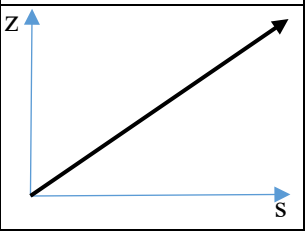
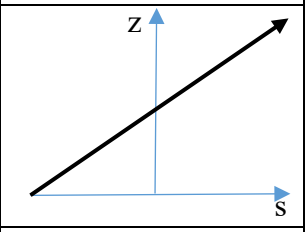
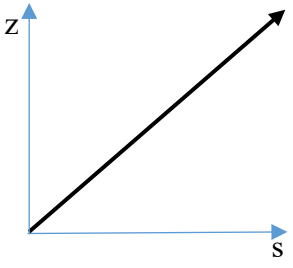
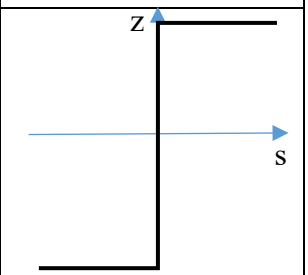
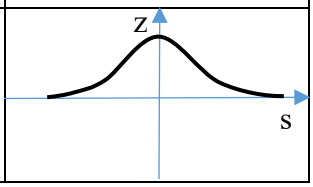
Nombre	Función $f(s)$	Rango	Forma
Log-Sigmoid	$z = \frac{1}{1+e^{-s}}$	$[0,+1]$	
Hyperbolic Tangent Sigmoid	$z = \frac{e^{s}-e^{-s}}{e^{s}+e^{-s}}$	$[-1,+1]$	
Positive Linear	$z = 0 \quad s < 0$ $z = s \quad s \geq 0$	$[0,+s]$	
Linear	$z = s$	$[-s,+s]$	
Saturating Linear	$z = 0 \quad s < 0$ $z = s \quad 0 \leq s \geq 1$ $z = 1 \quad s > 1$	$[0,+1]$	
Step	$z = sign(s)$ $z = Heaviside(s)$	$[-1,+1]$ $[0,+1]$	
Gaussian	$z = A \cdot e^{-\beta x^2}$	$[0,+1]$	

Tabla 3.5 Funciones de Activación

3.5.4 *Software para el desarrollo de la red neuronal artificial y elección*

A continuación, se comentarán las diferentes alternativas a la hora de desarrollar Machine Learning, además, se mostrará el software elegido.

Entre el software o lenguaje de programación más usado para aplicaciones de Machine Learning se encuentran los siguientes:

- Python
- C++
- Matlab

A forma comparativa se mostrará una tabla la cual recoge ventajas y desventajas de un software respecto de otro.

SOFTWARE	**Python**	**C++**	**Matlab**
Líneas de código	Tiende a tener cortas líneas de código	Tiende a tener largas líneas de código	Tiende a tener cortas líneas de código
Sintaxis	Abreviada, aunque puede llegar a ser confusa	Elevada sintaxis y estructuras predefinidas	Abreviada
Compilación	Interpretada	Precompilada	Interpretada
Velocidad	Alta	Muy Alta	Alta
Extensión	.py	.cpp	.m
Funciones	No tienen restricción sobre el tipo de argumento y el tipo de su valor de retorno	Puede aceptar y devolver el un tipo de valor ya definido	Si tiene restricciones sobre el tipo de argumento y su valor de retorno
Librerías de Deep Learning	✓✓✓✓	✓✓✓	✓✓✓
Gratis	✓✓✓	✓✓✓	✗
Habilidad y conocimiento propio	✓	✗	✓✓✓

Tabla 3.6 Diferencias entre el Software para el desarrollo de la red neuronal

Como conclusión se selecciona el lenguaje de Matlab para el desarrollo de la red neuronal, aunque Python podría ser una muy buena alternativa.

Esta elección ha venido influenciada en su mayor parte por la habilidad y conocimientos propios sobre Matlab.

Las Toolboxes o librerías de Matlab usadas en el proyecto han sido las siguientes:

- Audio Toolbox
- Deep Learning Toolbox
- Signal Processing Toolbox

3.6 Herramientas Software

A continuación, se detallarán más en profundidad las herramientas software usadas para la elaboración del proyecto.

3.6.1 Matlab

Matlab (*MATrix LABoratory*), es una plataforma de programación y cálculo numérico mediante vectores y matrices, aunque también puede trabajar con números escalares tanto reales como complejos, con cadenas de caracteres y con otras estructuras de información más complejas, este Software no es gratuito, el precio dependerá del tipo de licencia seleccionada.

Además, Matlab permite un gran abanico de prestaciones como [21]:

- Análisis de datos
- Gráficas: Visualizar y explorar datos
- Desarrollo de algoritmos
- Creación de aplicaciones
- Cálculo paralelo
- Cálculo en la nube
- Sistemas de control
- Deep Learning: Preparación de datos, diseño, simulación
- Machine Learning: Entrenar modelos, ajustar parámetros
- Robótica
- Procesamiento de señales

Dependiendo del tipo de prestaciones que se quieran usar, se requerirá usar una Toolbox u otra.

Figura 3.32 Icono de Matlab

Podría decirse que MATLAB es uno de los lenguajes de programación más extendidos en el mundo científico tanto a nivel de docencia como de investigación [22].

3.6.2 GNU radio

GNU radio es un software completamente gratuito y de código abierto que ofrece múltiples kits de herramientas para el procesamiento de señales y para el desarrollo de radios definidas por software, donde, estos kits de herramientas son fáciles de usar y muy intuitivos ya que, estos kits, no son otra cosa que bloques predefinidos realizados por usuarios.

Para alcanzar un procesamiento de señal en tiempo real de alto rendimiento, todos los bloques de procesamiento de señales están completamente desarrollados en el lenguaje de programación de C++. Sin embargo, los gráficos de flujo en GNU radio se compilan y ejecutan en Python. [23]

Además, en los casos en los que algunos bloques de procesado no se encuentren en la librería, o, el bloque existente no sea satisfactorio para una tarea en cuestión, GNU radio permite la posibilidad de que uno mismo desarrolle su propio bloque en Python, para posteriormente añadirlo a su librería de GNU radio.

GNU radio, también permite una gran cantidad de escenarios de simulación en los que no hará falta la necesidad de hardware real, por ejemplo, se podría implementar un sistema de radiocomunicaciones añadiendo incluso efectos del canal como ruido.

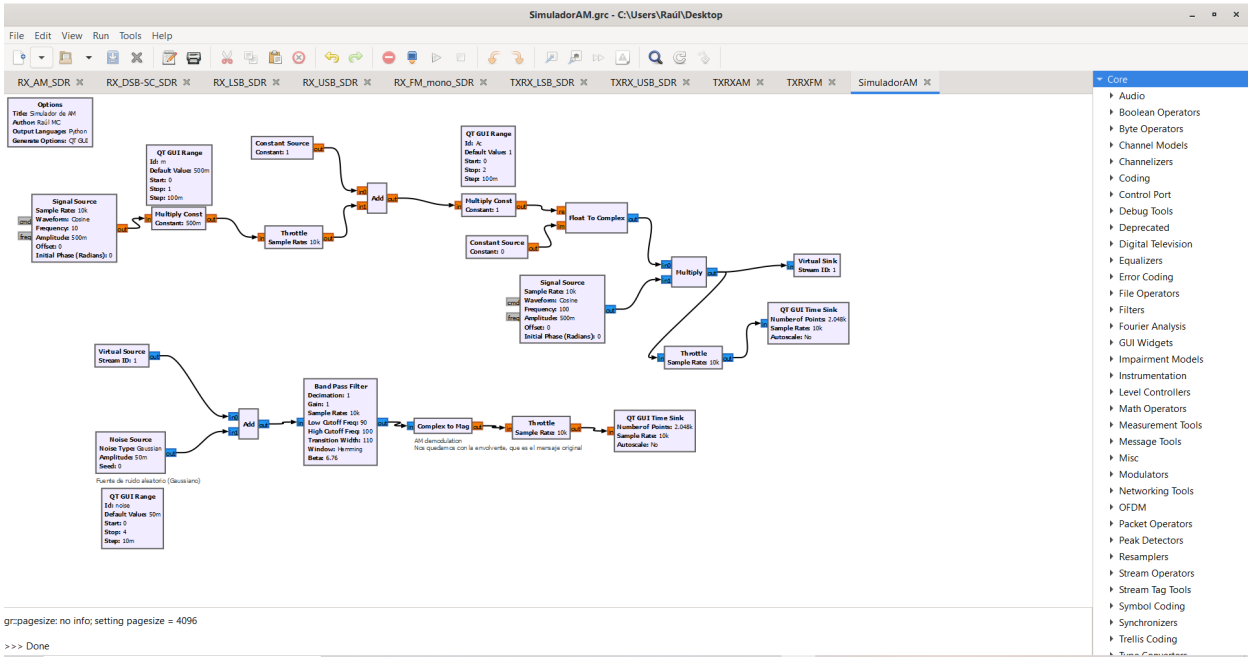


Figura 3.33 Interfaz de GNU radio y ejemplo de simulación

3.7 Tipo de Red Neuronal implementada y características

A continuación, se presentará el tipo de red neuronal implementada en el proyecto, así como sus características.

En cuanto al tipo de conexión usado, y, por tanto, tipo de procesamiento de la información, será el denominado FeedForward, es decir, un conexionado de capas de atrás hacia delante.

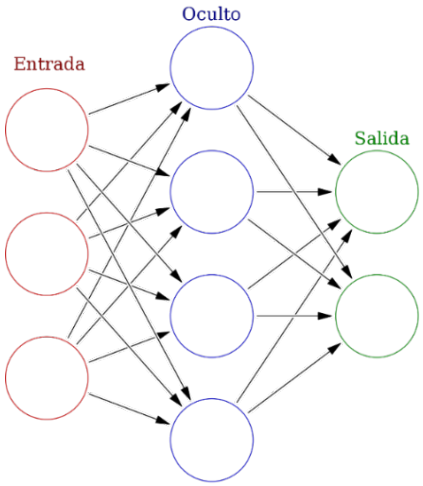


Figura 3.34 Red Neuronal Artificial FeedForward

Las capas de neuronas posicionadas entre la entrada y la salida se conocen como capas ocultas (*Hidden Layer – HL).*

Se obtiene la expresión final de salida para cualquier neurona 'j' de cualquier entrada 'i' de la red neuronal FeedForward como:

$$z_j = f\left(\sum_{Q=1}^{i} X_Q W_{Qj} + b_j\right) \quad (45)$$

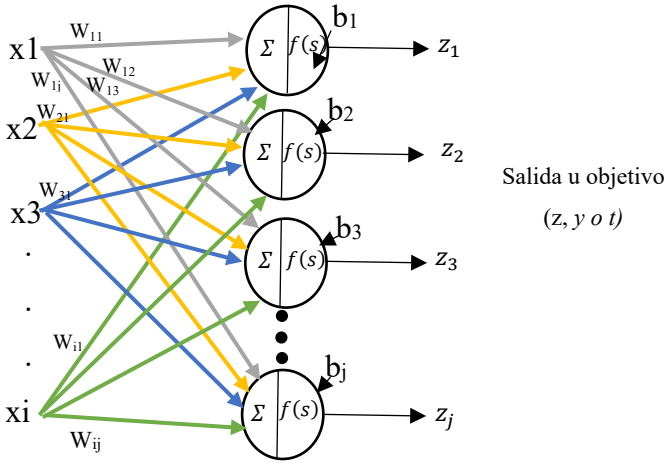


Figura 3.35 FeedForward en una capa

Por último, se muestra la configuración y número de nodos que poseerá la red neuronal implementada en el proyecto:

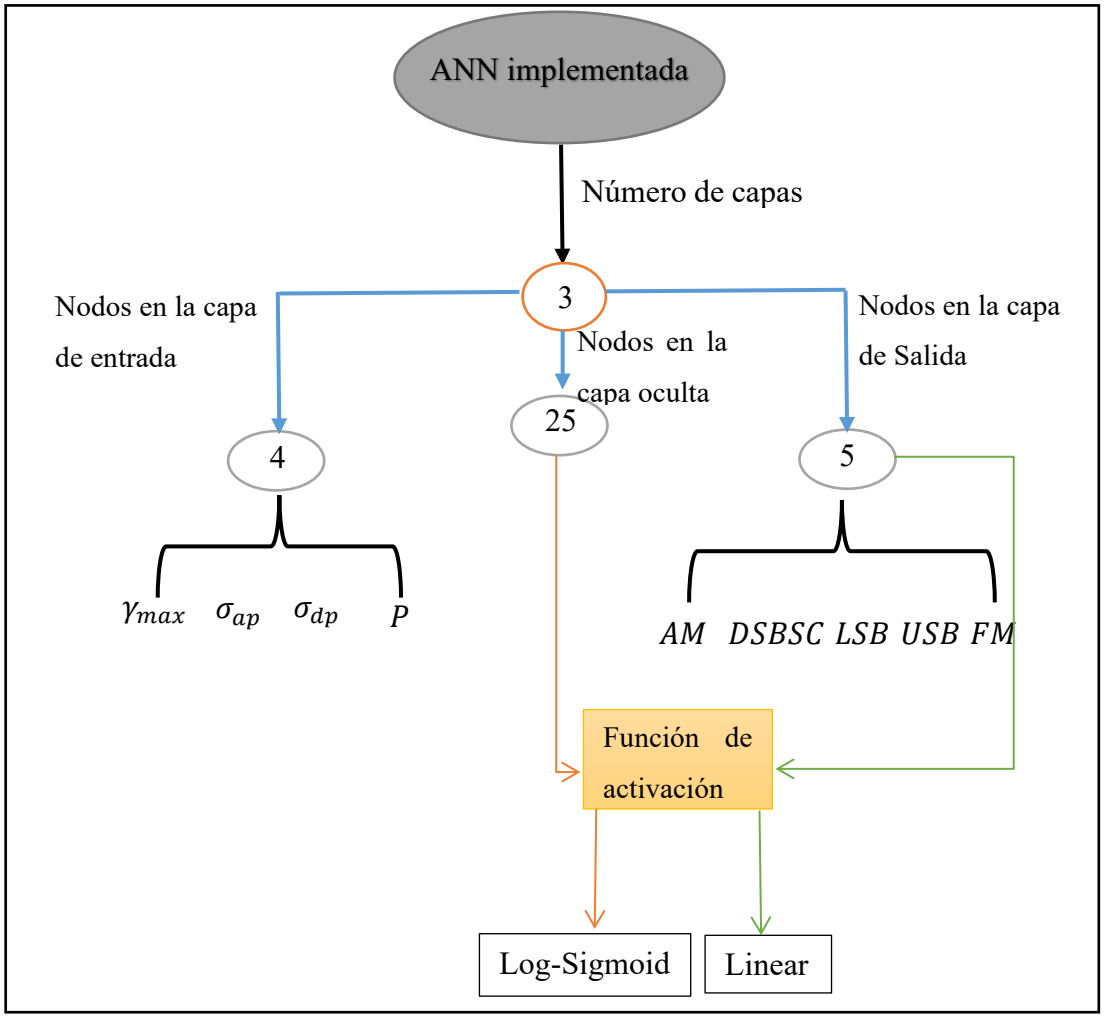


Figura 3.36 Características de la red neuronal a implementar

La elección de los 25 nodos conformantes de la capa oculta, así como las funciones de activación seleccionadas, han venido determinados por [38], donde se han realizado numerosas comparativas usando diferente número de nodos en la capa oculta, y, distintas funciones de activación, concluyendo que el número óptimo de nodos en la capa oculta son 25, ya que, a mayor número de nodos, el error resultante es mayor, de forma similar, para un menor número de nodos, el error ofrecido por la función de coste es también mayor.

Por otro lado, la combinación de otras funciones de activación, ha resultado en un mayor número de iteraciones.

3.8 Algoritmos para el entrenamiento empleados

Hasta ahora, se ha construido una herramienta (la red neuronal) en la que, variando sus parámetros, podemos resolver problemas muy complejos de **clasificación** o regresión.

A continuación, se procederá a describir la forma en la que este punto 3.8 será desarrollado.

El algoritmo usado en el proyecto para el entrenamiento de la red neuronal, es decir, para la minimización del error de la red neuronal mediante la variación de sus parámetros, ha sido el denominado **Descenso del Gradiente** (*Gradient Descent – (GD*) basado en **momentum** y **Ratio de Aprendizaje Adaptativo** con **BackPropagation**.

Por tanto, dicho sub apartado se comenzará con la explicación y desarrollo del Descenso del Gradiente y el Ratio de Aprendizaje para, a partir de ahí, comentar los algoritmos de optimización usados en el proyecto, que son momentum y ratio de aprendizaje adaptativo, para, finalmente explicar la consistencia del algoritmo de BackPropagation, ya que, BackPropagation se basa en el Descenso del Gradiente.

3.8.1 Descenso del Gradiente

Como se ha recalcado anteriormente, en un modelo cualquiera, ya sea de clasificación, regresión etc. Observamos unos parámetros (pesos y sesgos), mediante la variación de estos parámetros, se observará una variación en el error del modelo.

La **función de coste**, será la que ofrezca el error para cada una de las combinaciones de los parámetros, contra menor sea el error ofrecido por la función de coste para una combinación dada, mejores resultados se obtendrán, a continuación, se observan algunos tipos de funciones de costo.

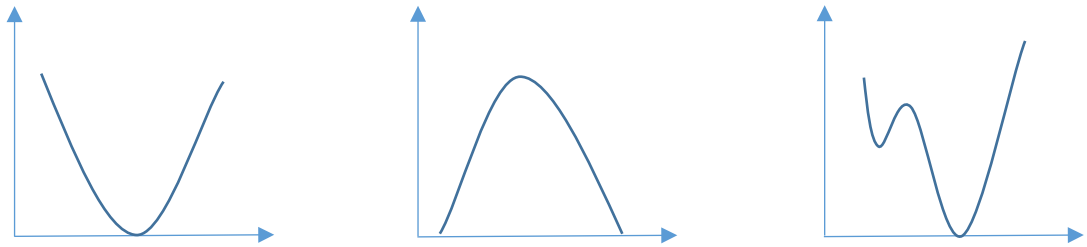


Figura 3.37 Tipos de Funciones de Coste

La primera función que aparece en la figura 3.35, se trata de una función convexa, en este tipo de funciones siempre observamos un valor mínimo que se corresponderá a un mínimo global, es decir, se asegura de que no se encontrará un valor inferior a ese mínimo global.

La segunda función es cóncava, este tipo de funciones no van a interesar como funciones de costo, ya que, como se observa, es similar a una función convexa invertida.

Por último, observamos un tipo de función no convexa, que serán las funciones a las cuales se deberá enfrentar, ya que, la mayoría de funciones de costo utilizadas en ML tendrán esa forma. Y esto es un problema, por un lado, existe el problema de que ya no se cumple la propiedad que interesaba de las funciones convexas, ahora es posible encontrarse con un punto mínimo (punto local) que no sea el punto mínimo global de la función, es decir,
la derivada de una función, nos indica la pendiente de la misma en el punto donde es evaluada, donde, si se iguala a 0, se encuentra el punto en el cual la pendiente es nula, con las funciones convexas, se sabe que solo hay un punto mínimo, y, por tanto, solo tendríamos que resolver una derivada, pero, con las funciones no convexas se pueden encontrar múltiples puntos mínimos y, por tanto, múltiples derivadas que resolver.

Este no es el único problema que se tiene con las funciones de costo no convexas (que serán las utilizadas para el proyecto), sino que, por otro lado, puede haber zonas de la función donde la pendiente sea nula.

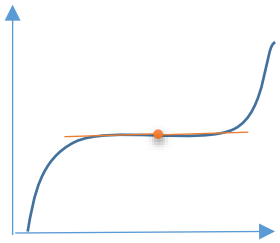


Figura 3.38 Punto de inflexión (Pendiente Nula)

Todos estos problemas que acarrean las funciones no convexas, obligan a tener que encontrar una solución para las mismas.

Para una mejor compresión, se procede a resumir todo lo explicado en el siguiente párrafo.

El entrenamiento de la red neuronal, se basa en la configuración de sus dos parámetros (pesos, sesgos), el cómo de bien se adaptan esos parámetros va a ser proporcionado por la función de coste, ya que, esta función, ofrece el error para una configuración u otra de estos parámetros, las funciones de coste a implementar, son del tipo no convexas, este tipo de funciones, presentan dos problemas, uno de ellos es la posibilidad de aparición de más de un mínimo local, es decir, se puede tener una configuración de parámetros que, a priori, parezca ser la más adecuada, cuando realmente esto no sea así, por tanto se debería de nuevo de volver a derivar la función para encontrar el verdadero mínimo global o mínimo error, el otro problema es la posibilidad de encontrar pendientes nulas en la función, conllevando de nuevo al problema anterior.

La solución a las funciones no convexas que se deben usar para el cálculo del error, será el uso del algoritmo del Descenso del Gradiente.

Las figuras y ecuaciones mostradas a continuación partirán de un modelo en el que solo se tienen dos variables por neurona, es decir un solo peso y sesgo, ya que, aunque el modelo neuronal del proyecto se corresponde a 1 sesgo y 4 pesos (ec.44) el proceso es exactamente similar a excepción del dimensionamiento.

Lo que hará el GD, será, aprovechar la información ofrecida por el cálculo de la derivada de la función en un punto para ayudarnos a encontrar los mínimos.

El funcionamiento del GD será el siguiente:

- Primeramente, se localiza la mayor pendiente para un punto en cuestión, este punto, para el comienzo del entrenamiento se inicializará de forma aleatoria. Dicha pendiente para cada uno de los parámetros de la red neuronal se obtiene mediante la derivada parcial del error ofrecido por la función de coste con respecto a cada uno de los parámetros.

$$\frac{\partial error}{\partial W}, \frac{\partial error}{\partial b} \qquad (46)$$

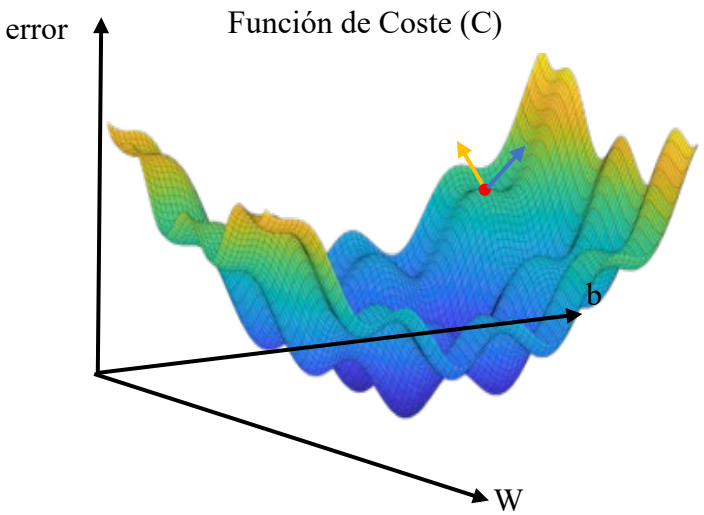


Figura 3.39 Derivada parcial del error con cada parámetro

- Conjuntamente el valor de las pendientes o direcciones obtenidas mediante las derivadas parciales conforman un vector que indica la dirección hacia donde la pendiente asciende, este vector es denominado **gradiente.**

  Lógicamente lo que se quiere es descender de tal manera que el error sea el menor posible, por tanto, se tomará el signo opuesto a ese vector gradiente, ya que el gradiente indica como se deben actualizar los parámetros para subir el error.

$$\begin{bmatrix} \frac{\partial error}{\partial W} \\ \frac{\partial error}{\partial b} \end{bmatrix} = \nabla_f \qquad (47)$$

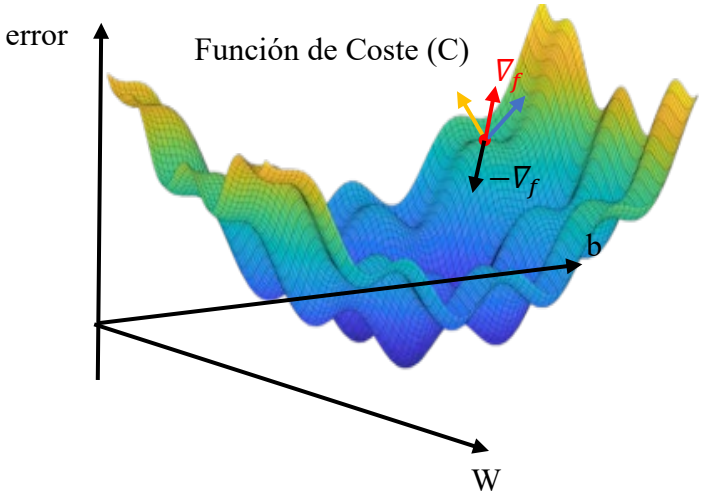


Figura 3.40 ± Vector Gradiente

- Al restar este gradiente, obtendríamos otro nuevo conjunto de parámetros los cuales tendrían un error menor. Este proceso se repetiría hasta convergencia, es decir, hasta una zona donde el movimiento no supondría una variación notable del coste, es decir, una zona con pendiente próxima a nula minimizando por tanto el coste del modelo.

$$(W, b)_{actualizados} = (W, b) - \nabla_f \qquad (48)$$

Por último, faltaría añadir un término a la expresión (ec.48), este término se conoce como Ratio de Aprendizaje (*Learning Rate – LR* o α), y definirá el afecto del gradiente a la actualización de los parámetros. Este término es realmente importante, ya que, definirá en su totalidad el comportamiento del algoritmo.

$$(W, b)_{actualizados} = (W, b) - \alpha\nabla_f \qquad (49)$$

A continuación, se analizarán tres comportamientos para diferentes valores de este LR sobre la actualización de los parámetros y por tanto sobre el error de coste, para la realización de las siguientes figuras se ha hecho uso de una herramienta interactiva gratuita disponible en [33] la cual nos permitirá visualizar el efecto de diferentes valores del LR sobre el error ofrecido por la función de coste.

La figura naranja representa la función de coste, los colores más fuertes representan los valores de error más pequeños, por el contrario, los valores más débiles representan los errores más elevados.

- Caso 1, valor del parámetro LR elevado

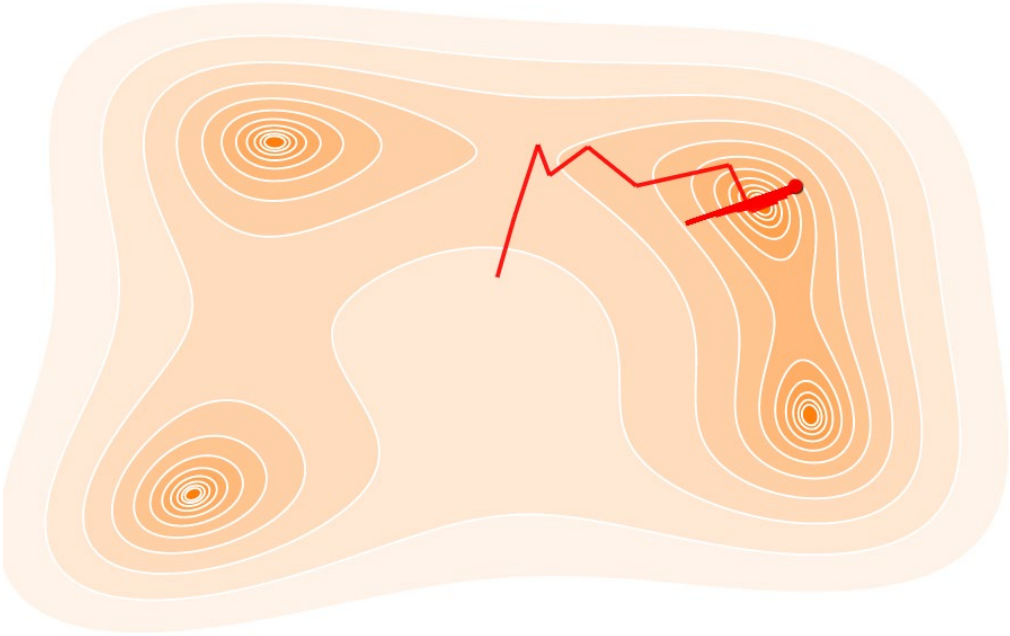


Figura 3.41 Convergencia del error cuando el valor del LR es elevado

El uso de un elevado LR da lugar a grandes variaciones en las actualizaciones de los parámetros para cada iteración, como resultado, se alcanzarán valores de error pequeños más rápidamente (en 6 iteraciones), aunque, como se observa en la figura, nunca habrá una convergencia, es decir, nunca se alcanzará el valor mínimo de error. Siendo imposible para el algoritmo converger en dicho punto mínimo, causando que el proceso de optimización quede en un bucle infinito.

- Caso 2, valor del parámetro LR pequeño

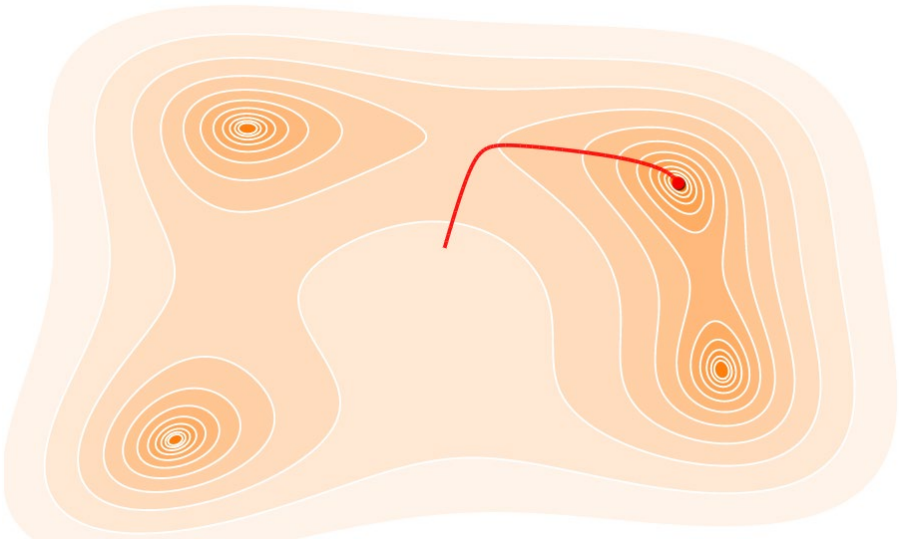


Figura 3.42 Convergencia del error cuando es usado un valor de LR pequeño

En el caso opuesto tenemos una convergencia con un error mínimo, aunque el número de iteraciones para llegar a ese estado es mucho mayor, con un total de 65 iteraciones. Esto podría hacer ineficiente al algoritmo, además, al ser tan pequeño el salto entre iteración, podría desencadenar el colapso del sistema, haciendo mucho más posible el atascamiento en mínimos locales.

- Caso 3, valor óptimo del parámetro LR

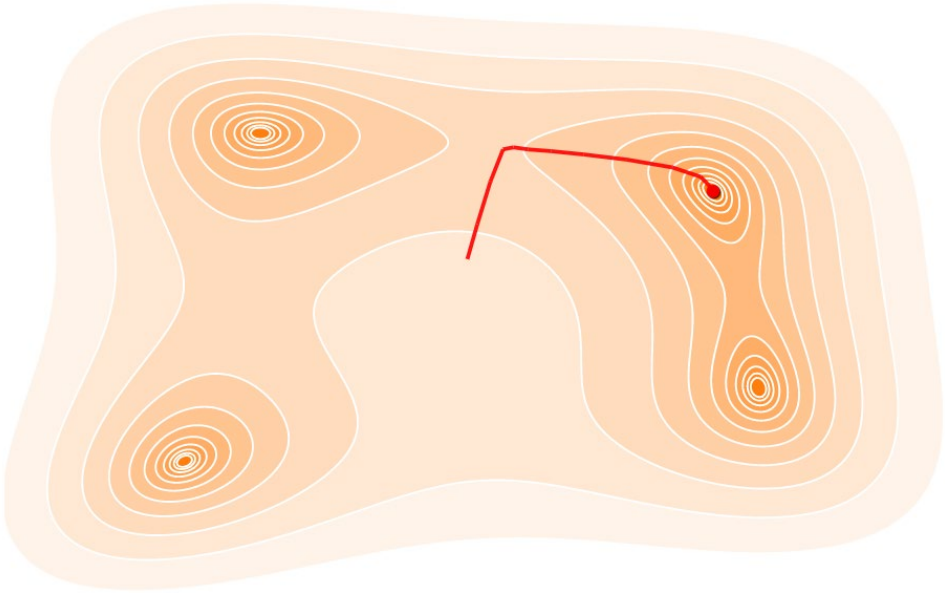


Figura 3.43 Convergencia del error cuando es usado un valor óptimo del LR

Cuando el valor usado del LR no es ni muy grande ni muy pequeño, se observa el comportamiento óptimo, llegando a converger en un error muy pequeño, sin un excesivo número de iteraciones (en 24 iteraciones).

La correcta configuración de la tasa de aprendizaje es fundamental para desarrollar un correcto y eficiente algoritmo, por ello, a continuación, se verán los algoritmos de optimización usados en el proyecto, estos algoritmos de optimización serán usados para optimizar el propio algoritmo del descenso del gradiente, y, para optimizar el valor de la tasa de aprendizaje ya que, existen diferentes técnicas para ajustar este parámetro de forma dinámica.

3.8.2 *Momentum y Ratio de Aprendizaje Adaptativo*

A continuación, se expondrán los dos algoritmos de optimización implementados en el desarrollo del AMC, estos algoritmos de optimización se implementan en la mayoría de casos de manera conjunta, ya que, de esta forma, se reducirá de forma descomunal el tiempo de entrenamiento de la red neuronal.

Como se ha visto, el descenso del gradiente puede entenderse con la analogía de un caminante que siempre da un paso constante en la dirección que tiene el mayor descenso, es decir, cada iteración es independiente a la anterior y, en cada una de estas iteraciones, la dirección tomada, será siempre la que posea un descenso mayor.

El algoritmo de optimización del descenso del gradiente basado en momentum, nos permitirá incluir una inercia en el paso del algoritmo, ahora, estas iteraciones o pasos del algoritmo no serán independientes entre sí, sino que, de alguna manera influirán en las siguientes iteraciones.

Lo que sucederá será lo siguiente. El error, acelerará su proceso de minimización cuando acumule varios pasos anteriores en descenso, es similar a dotar de un efecto de gravedad a esa actualización de los pesos y sesgo, por otro lado, el error disminuirá su proceso de minimización cuando acumule varios pasos anteriores en subida, llegando incluso a aumentar el error.

Este concepto podría asimilarse a una pelota lanzada a un cuenco, mientras más tiempo continue rodando hacia el fondo del cuenco (hacia el error mínimo), más velocidad adquirirá, por ende, más pronto llegará a la zona más baja (más rápidamente se alcanzará ese error mínimo), sin embargo, debido a esa velocidad con la que la pelota llega a la parte baja del cuenco, causará que nuevamente suba por la otra pared del cuenco (aumente de nuevo el error), aunque, esa subida será mucho menor que la bajada realizada (ese aumento de error, será mucho menor que el error de partida presente en la bajada), posteriormente la pelota bajaría de nuevo para quedarse situada de forma estática en la parte más baja del cuenco (el error disminuiría de nuevo para quedarse estático en su mínimo).

Para comprender este algoritmo se añade la siguiente figura (Figura 3.44), la cual muestra la convergencia usando un valor óptimo del LR, implementando la optimización del algoritmo mediante momentum.

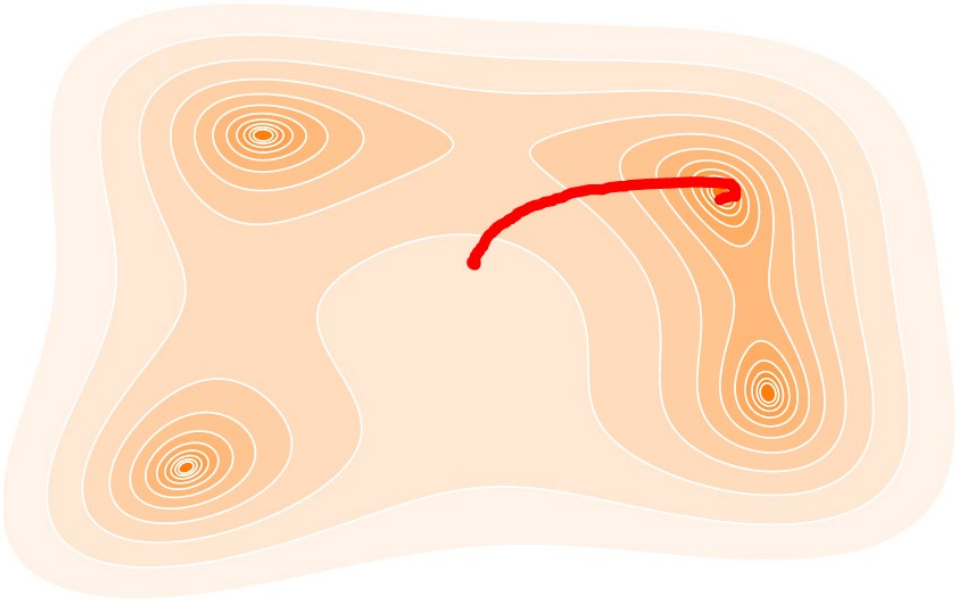


Figura 3.44 Convergencia del error cuando es usado momentum para optimizar el descenso del gradiente, usando un valor óptimo del LR

Como se observa en la figura, se dota a esa '*particula*' de error de una gravedad, la cual se asemeja al ejemplo de la pelota comentado anteriormente.

Por tanto, gracias a este algoritmo de optimización del gradiente, se alcanzaría ese mínimo error mucho más rápido (en 17 iteraciones), es decir, el entrenamiento de la red mediante la actualización de los parámetros necesitaría un menor número de iteraciones para alcanzar la adaptación óptima.

La ecuación que define el descenso del gradiente en el punto actual al incluir esa optimización basada en momentum sería la siguiente:

$$(W, b)_{actualizados} = (W, b) - \alpha M \qquad (50)$$

Siendo M:

$$M = \nabla_f + \eta {\nabla_f}^{t-1} \qquad (51)$$

Donde:

- ${\nabla_f}^{t-1}$ equivale al gradiente obtenido en la iteración anterior
- η es un valor que definirá el afecto de la iteración anterior sobre la iteración actual

Por último, faltaría por implementar el algoritmo de optimización basado en la tasa de aprendizaje adaptativo (α_{adpv}), y es que, como se ha visto y comentado, el algoritmo de momentum, tiene unos pequeños inconvenientes los cuales consistían por un lado en ese aumento del error en el último trayecto de recorrido debido a esa velocidad que traía consigo la partícula del error, por otro lado, a causa de ese incremento del error, se debe de nuevo de volver a minimizarlo traduciéndose esto en un mayor número de iteraciones .

Esta tasa de aprendizaje adaptativa no solo solucionará en gran parte estos problemas tomando valores muy pequeños (cercanos a 0) cuando el error pase de disminuir a incrementar, no produciéndose por tanto un incremento considerable del error, sino que también nos permitirá que cuando el error este en pleno descenso, esta tasa aumente su valor, por tanto, se alcanzarán valores de error mínimos mucho más rápido disminuyendo en ambos casos el número de iteraciones.

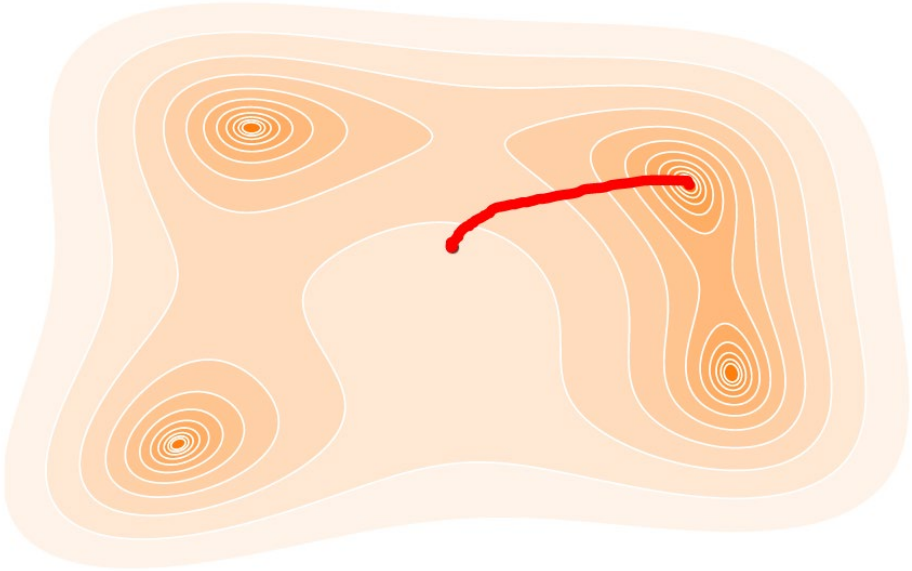


Figura 3.45 Convergencia del error cuando es usado momentum y LR adaptativa como algoritmos optimizadores del descenso del gradiente

El número total de iteraciones para alcanzar el error mínimo ha sido de 7.

A continuación, se muestra por tanto la ecuación final del algoritmo del descenso del gradiente siendo implementado mediante los dos algoritmos de optimización contemplados y usados en el desarrollo del AMC.

$$(W, b)_{actualizados} = (W, b) - \alpha_{adpv}(\nabla_f + \eta {\nabla_f}^{t-1}) \qquad (52)$$

Donde:

- α_{adpv} es el valor de la tasa de aprendizaje adaptativa, cuyo valor en cada iteración incrementa o decrementa en función de α_{inc} , α_{dec}

Dicha ecuación (ec.52) será la que se realice en cada una de las neuronas para el entrenamiento de sus parámetros.

3.8.3 BackPropagation

La red neuronal artificial es un algoritmo de **aprendizaje automático**, es decir, dicho aprendizaje de la red neuronal el cual será conseguido mediante la variación de sus parámetros, debe hacerse de forma automática evitando ser el usuario quien varíe o modifique dichos parámetros.

Será la propia red neuronal quien aprenda de por sí sola a partir de los datos de entrada, viéndose de esta manera el gran potencial del ML.

Mediante el uso del descenso del gradiente se conseguía un método perfecto para ajustar los parámetros de la neurona, evaluando el error del modelo en un punto, realizando las derivadas parciales y obteniendo el vector opuesto del gradiente, para, de esta forma minimizar el error.

Entonces, la pregunta ahora sería ¿Por qué no se usa directamente el algoritmo del descenso del gradiente para el entrenamiento de la red neuronal?

Para aplicar el descenso del gradiente, se necesita el gradiente, cuando se trabaja en un modelo de regresión lineal simple con una neurona, es decir, con solo dos variables de entrada a una neurona, el cálculo del vector gradiente no supone ningún problema, ya que, únicamente el coste varía en función de esos dos parámetros, sin embargo, como ya se comentó anteriormente, el modelo usado en el proyecto se corresponde con un modelo de regresión lineal múltiple (ec.44) en el que no solo tenemos una neurona, sino una red neuronal, por tanto, la forma en la que modificar un parámetro puede afectar al resultado final del error es mucho más complejo, ya que, esa variación influirá al resultado final mediante muchas más conexiones como se aprecia en la figura 3.46.

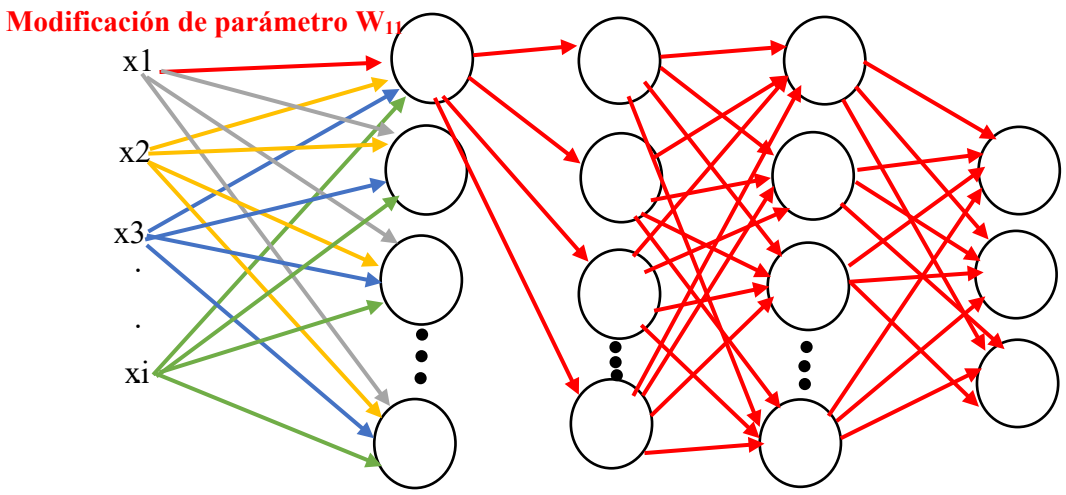


Figura 3.46 Efecto de modificar un parámetro en las conexiones de la ANN

Debido a la complejidad que presenta la estructura jerarquizada de la red neuronal, se procederá a usar el descenso del gradiente anteriormente comentado (incluidos los algoritmos de optimización) para el entrenamiento de la red neuronal haciendo uso de la técnica de BackPropagation para el cálculo de las derivadas parciales en cada neurona y la conformación del vector gradiente. Esta técnica nos permitirá de forma automática calcular las derivadas y por ende conformar el vector gradiente para toda la red neuronal, ya que, el descenso del gradiente requiere del gradiente para la minimización del error.

La importancia de este algoritmo es clave dentro de la arquitectura de una red neuronal.

El cálculo de las derivadas parciales como se ha comentado incluye el error, por tanto, al hacer uso de una red neuronal, cada neurona tendrá su error correspondiente, mediante BackPropagation se va a realizar un análisis de la responsabilidad que tiene cada neurona en el error, y, este análisis va a ser realizado hacia atrás, es decir, desde la última capa hasta las primeras, este procedimiento resultará muy útil y eficiente, ya que, en una red neuronal el error de las capas anteriores depende directamente del error en las capas posteriores, de esta forma, se podrá determinar la manera en la que una neurona afecta al resultado final mediante la retro propagación de errores ya que se realizará un reparto de errores entre las neuronas, y, dicho error de cada neurona será utilizado para determinar la cantidad la cual hay que modificar cada parámetro de la neurona, ya que, esos errores serán los usados para el cálculo de las derivadas parciales de cada parámetro de la red neuronal, conformando de esta manera el vector gradiente necesario para el entrenamiento de la red.

3.8.3.1 Formulación matemática del Algoritmo de BackPropagation [34]

Una vez comentado el funcionamiento del algoritmo de BackPropagation, se pasará a la formulación matemática del mismo.

Como se ha comentado anteriormente, para entrenar la red, se deben calcular las derivadas parciales del error o coste (C) respecto los parámetros de la red neuronal, donde estos parámetros son dos, los pesos y el sesgo, por tanto, habrá que calcular dos derivadas, una sobre el parámetro de peso (w) y, otra sobre el parámetro sesgo o bias (b), y, se empezará a trabajar desde la última capa hasta las primeras capas.

3.8.3.1.1 Derivadas en la última capa

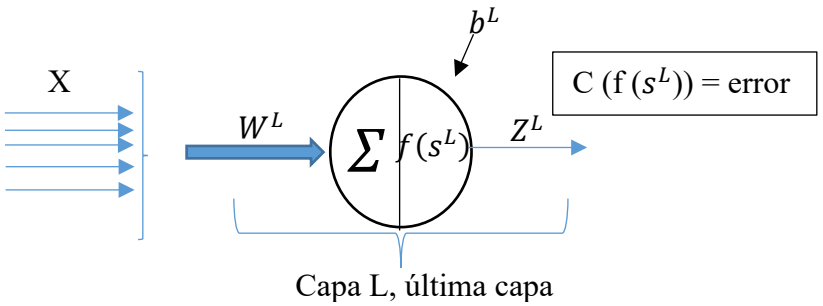


Figura 3.47 Neurona de la última capa

$$\frac{\partial C}{\partial W^L}, \frac{\partial C}{\partial b^L} \qquad (53)$$

Para el análisis de ambas derivadas, se debe conocer el camino que conecta el valor del parámetro y el coste final, siendo para cada parámetro de la última neurona los siguientes caminos:

$$\frac{\partial C}{\partial W^L} = \frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} \cdot \frac{\partial s^L}{\partial W^L} \qquad (54)$$

$$\frac{\partial C}{\partial b^L} = \frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} \cdot \frac{\partial s^L}{\partial b^L} \qquad (55)$$

Donde, el término $\frac{\partial C}{\partial f^L}$, similar en ambas ecuaciones, representa la derivada de la salida de la función de activación o salida de la neurona con respecto al coste, es decir, la

información que ofrece dicho término es la variación del coste o error cuando se varía la función de activación de la neurona.

El término $\frac{\partial f^L}{\partial s^L}$, nuevamente igual en ambas ecuaciones, representa la variación de la salida de la neurona en función de la suma ponderada.

Por último, se tienen los términos $\frac{\partial s^L}{\partial W^L}$ y $\frac{\partial s^L}{\partial b^L}$, estos términos informan sobre la variación de la suma ponderada con respecto a la variación de los parámetros de la red neuronal.

Siendo la suma ponderada en la última capa:

$$s^L = \sum_i {X_i}^L W_i^L + b^L = \sum_i {f_i}^{L-1} W_i^L + b^L \qquad (56)$$

Observándose que ${X_i}^L = {f_i}^{L-1}$, ya que, lógicamente, las entradas a las neuronas de la última capa se corresponden con las salidas de las neuronas de la capa inmediatamente anterior, siendo 'i' la conexión en cuestión.

Se obtienen los valores de las derivadas:

$$\frac{\partial s^L}{\partial W^L} = {f_i}^{L-1} \qquad (57)$$

$$\frac{\partial s^L}{\partial b^L} = 1 = cte \qquad (58)$$

$$\frac{\partial f^L}{\partial s^L} = f^L \cdot s^L \cdot (1 - f^L s^L) \qquad (59)$$

$$\frac{\partial C}{\partial f^L} \qquad (60)$$

Donde el resultado de la (ec.60), dependerá de la función de coste a usar.

De las (ecuaciones 54,55), el bloque compuesto por la multiplicación de $\frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L}$, representa la variación del error en función del valor de s^L, que es la suma ponderada

calculada dentro de la neurona, es decir, este bloque, informará sobre el grado de variación del error o coste de la neurona cuando se produce un cambio en dicha suma ponderada.

$$\frac{\partial C}{\partial s^L} = \frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} = \delta_L \qquad (61)$$

Si dicha derivada ofrece un valor elevado, indica que un pequeño cambio en la suma ponderada afectará mucho al error, por el contrario, si dicho valor es pequeño, indica que, ante grandes o pequeñas variaciones de la suma, el error no se verá prácticamente afectado.

Por tanto, la derivada de la (ec.61), informará sobre la **responsabilidad** que presenta una neurona sobre el error, pudiéndose ser interpretada dicha ecuación como el error imputado a la neurona en la última capa δ_L.

Sustituyendo (ec.57, 58, 61) en (ec.54, 55) obtenemos finalmente las derivadas buscadas en la última capa:

$$\frac{\partial C}{\partial W^L} = \delta_L \cdot f_i^{L-1} \qquad (62)$$

$$\frac{\partial C}{\partial b^L} = \delta_L \qquad (63)$$

Por tanto, han quedado deducidas las tres expresiones (ec.61, 62, 63) diferentes que permitirán obtener las derivadas parciales buscadas en la última capa, y, como se verá ahora, también serán necesarias para el cálculo de las derivadas en capas anteriores.

3.8.3.1.2 Derivadas en la capa anterior a la última, y, en el resto de capas

Para el cálculo de los parámetros de la capa anterior a la última, gran parte del trabajo ha sido ya realizado en la última capa, siendo las ecuaciones ahora:

$$\frac{\partial C}{\partial W^{L-1}} = \overbrace{\frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L}}^{\delta_L} \cdot \overbrace{\frac{\partial s^L}{\partial f^{L-1}}}^{W_L} \cdot \frac{\partial f^{L-1}}{\partial s^{L-1}} \cdot \frac{\partial s^{L-1}}{\partial W^{L-1}} \tag{64}$$

$$\frac{\partial C}{\partial b^{L-1}} = \underbrace{\frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} \cdot \frac{\partial s^L}{\partial f^{L-1}} \cdot \frac{\partial f^{L-1}}{\partial s^{L-1}}}_{\delta_{L-1}} \cdot \frac{\partial s^{L-1}}{\partial b^{L-1}} \tag{65}$$

Como se observan en ambas expresiones, los dos primeros términos $\frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L}$, ya han sido calculados previamente, el cuarto y quinto término de ambas expresiones $\frac{\partial f^{L-1}}{\partial s^{L-1}}$, $\frac{\partial s^{L-1}}{\partial W^{L-1}}$, $\frac{\partial s^{L-1}}{\partial b^{L-1}}$, se operan exactamente igual que anteriormente, solo que en la capa previa.

Por último, el tercer término $\frac{\partial s^L}{\partial f^{L-1}}$, sería la única nueva derivada a calcular en la capa anterior, este término informa sobre como varía la suma ponderada de una capa cuando se varía la salida de la neurona en la capa inmediatamente anterior, lógicamente, dicha derivada entre la suma ponderada realizada en la capa posterior y la salida de la neurona en la capa anterior es la matriz de parámetros w_L.

Ahora, la multiplicación en ambas ecuaciones de los cuatro primeros términos $\frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} \cdot \frac{\partial s^L}{\partial f^{L-1}} \cdot \frac{\partial f^{L-1}}{\partial s^{L-1}}$, se convierten en el error imputado de las neuronas en la capa inmediatamente anterior a la última.

$$\frac{\partial C}{\partial s^{L-1}} = \frac{\partial C}{\partial f^L} \cdot \frac{\partial f^L}{\partial s^L} \cdot \frac{\partial s^L}{\partial f^{L-1}} \cdot \frac{\partial f^{L-1}}{\partial s^{L-1}} = \delta_{L-1} \tag{66}$$

Y, aquí, es donde entra en juego la gran eficacia del algoritmo de BackPropagation, ya que, el proceso realizado en esta capa, es extensible para el **resto de capas** de la red neuronal, aplicando una lógica similar

$$\frac{\partial C}{\partial W^{L-n}} = \frac{\partial C}{\partial f^{L}} \cdot \frac{\partial f^{L}}{\partial s^{L}} \cdot \frac{\partial s^{L}}{\partial f^{L-1}} \cdot \frac{\partial f^{L-n}}{\partial s^{L-n}} \cdot \frac{\partial s^{L-n}}{\partial W^{L-n}} \quad (67)$$

$$\frac{\partial C}{\partial b^{L-n}} = \frac{\partial C}{\partial f^{L}} \cdot \frac{\partial f^{L}}{\partial s^{L}} \cdot \frac{\partial s^{L}}{\partial f^{L-n}} \cdot \frac{\partial f^{L-n}}{\partial s^{L-n}} \cdot \frac{\partial s^{L-n}}{\partial b^{L-n}} \quad (68)$$

Siendo:

- $L-n$ la capa en cuestión, donde $(n = 0,1 \dots L-1)$

3.9 Desarrollo e implementación del AMC

La ANN reconocedora de la modulación se compone de los siguientes bloques principalmente:

- Bloque de preprocesado: En el cual las características clave de entrada a la red neuronal serán calculadas y extraídas para cada segmento de señal complejo.
- Generación del Training Data Set.
- Entrenamiento y fase de aprendizaje de la ANN para ajustar sus parámetros, validación y testeo.

3.9.1 Características clave para la clasificación de señales analógicas

Para un perfecto reconocimiento de la modulación, se deben de seleccionar previamente un conjunto de características que permitan la discriminación entre las diferentes modulaciones, en general, la selección de características adecuadas depende principalmente de los tipos de modulación de interés, estas características seleccionadas serán utilizadas posteriormente por un clasificador de aprendizaje automático para determinar el tipo de modulación de una señal recibida.

En la literatura, se sugieren diferentes tipos de características de las señales recibidas [35] como:

- Amplitud Instantánea
- Fase
- Frecuencia
- Transformada de Fourier
- Momentos de orden superior
- Transformada de Wavelet
- Cumulantes de orden alto

- Cumulantes cíclicos de orden superior
- Estadísticas de muy alto orden

En base a las modulaciones a clasificar en el proyecto, y, en base al algoritmo seleccionado para el desarrollo del AMC, las características seleccionadas vienen dadas por la amplitud instantánea y la fase de las señales complejas recibidas, siendo un total de cuatro características a calcular.

A continuación, en los siguientes sub apartados se comentan y definen las características en cuestión.

3.9.1.1 γ_{max}

La primera característica (γ_{max}), es el valor máximo de la densidad de potencia espectral de la amplitud instantánea normalizada y centrada de la señal recibida.

$$\gamma_{max} = \max\left(\frac{\left|DFT\left(A_{cn}(i)\right)\right|^2}{N_s}\right) \quad (69)$$

Donde:

- DFT es la Transformada Discreta de Fourier, aunque, para el cálculo de la DFT se usarán algoritmos rápidos, es decir, la Transformada rápida de Fourier (*Fast Fourier Transform - FFT*), permitiendo reducir en gran medida el número de operaciones a realizar [36].
- N_s es el número de muestras por segmento de señal recibido.
- $A_{cn}(i)$ es la amplitud instantánea normalizada y centrada de la señal recibida en los instantes de tiempo $t = \frac{i}{f_s}$ $(i = 1,2, \dots, Ns)$.
- f_s es la frecuencia de muestreo.

Se expresa $A_{cn}(i)$ como:

$$A_{cn}(i) = A_n(i) - 1 \quad (70)$$

Siendo $A_n(i)$ la amplitud instantánea normalizada, expresándose esta como:

$$A_n(i) = \frac{A(i)}{m_a} \quad (71)$$

Obteniendo la amplitud instantánea $A(i)$ de cada segmento complejo recibido como:

$$A(i) = |I(i) + jQ(i)|^2 = \sqrt{I(i)^2 + Q(i)^2} \quad (72)$$

Por último m_a es la media de la amplitud instantánea para cada segmento de señal, expresándose esta como:

$$m_a = \frac{1}{N_s}\sum_{i=1}^{N_s} A(i) \quad (73)$$

Esta característica, γ_{max}, mide la varianza de la amplitud de una señal. Para modulaciones donde la información es transportada mediante variaciones de la amplitud de la señal (modulaciones lineales), el valor de γ_{max} no puede ser cero, por otro lado, para modulaciones donde la amplitud queda constante (modulaciones angulares), el valor de γ_{max} debe ser cero. Aunque, debido al ruido presente en el canal de transmisión, el valor de γ_{max} no será exactamente 0 en FM, sino que estará muy cercano a 0.

Por tanto, dicha característica va a permitir realizar una discriminación entre dos conjuntos de modulaciones, por un lado, el primer conjunto de modulaciones que nos va a permitir distinguir es:

- AM, DSB-SC, LSB, USB

Por otro lado:

- FM

3.9.1.2 σ_{ap}

La segunda característica (σ_{ap}), es la desviación estándar del valor absoluto de la componente no lineal de la fase instantánea después de ser centrada, evaluada sobre los intervalos no débiles del segmento de señal.

Se define como:

$$\sigma_{ap} = \sqrt{\left(\frac{1}{C}\sum_{A_n(i)>a_t} \phi_{NL}^2(i)\right) - \left(\frac{1}{C}\sum_{A_n(i)>a_t} |\phi_{NL}(i)| - \left(\frac{1}{C}\right)^2\right)^2} \quad (74)$$

Donde:

- $\phi_{NL}(i)$ es le valor de la componente no lineal de la fase instantánea en los instantes de tiempo $t = \frac{i}{f_s}$ $(i = 1,2, \dots, Ns)$
- a_t es el límite o umbral para $A_n(i)$, para valores de $A_n(i)$ por debajo de ese umbral la señal es muy sensible al ruido, afectando esto a la evaluación de la fase instantánea.
- C es el número de muestras en $\phi_{NL}(i)$ para el cual $A_n(i) > a_t$

Siendo $\phi_{NL}(i)$:

$$\phi_{NL}(i) = arctg(\frac{I(i)}{Q(i)}) \qquad (75)$$

La característica σ_{ap}, mide la varianza en la fase instantánea absoluta, dicha característica va a permitir realizar una discriminación entre las modulaciones que tienen fase y las que no la tienen, por ejemplo, nos podría discriminar entre FM y AM, o, AM y DSB-SC entre otras.

Sin embargo, con esta característica podría haber dificultades a la hora de diferenciar entre FM, LSB, USB entre sí, ya que, estos tres tipos de modulaciones, como se vio anteriormente, se componían tanto de componente en fase como de cuadratura.

3.9.1.3 σ_{dp}

La tercera característica (σ_{dp}), es la desviación estándar de la componente no lineal de la fase instantánea directa (no absoluta) evaluada sobre los intervalos no débiles del segmento de la señal, y, es definida como:

$$\sigma_{dp} = \sqrt{(\frac{1}{C}\sum_{A_n(i)>a_t}\phi_{NL}^2(i)) - \left(\frac{1}{C}\sum_{A_n(i)>a_t}\phi_{NL}(i) - \left(\frac{1}{C}\right)^2\right)^2} \qquad (76)$$

La única diferencia entre la característica 3 y 2 se encuentra en el segundo sumatorio, en la característica 2 se tomaba el valor absoluto de dicho sumatorio, sin embargo, aquí no, por tanto, el valor calculado de la característica 3 debe ser mayor que el valor obtenido para la característica 2, ya que, en el sumatorio se contemplarán sumas y restas de fases.

Esta tercera característica permitirá discriminar las mismas modulaciones que la característica anterior, sin embargo, es usada para añadir aún más robustez, ya que, contra mayor número de características y más homogéneas entre sí, mejor clasificación se podrá realizar.

3.9.1.4 Ratio P

Por último, la cuarta característica (Ratio P), va a informar sobre la simetría del espectro alrededor de la frecuencia de portadora, definiéndose como:

$$P = \frac{P_L - P_U}{P_L + P_U} \tag{77}$$

Donde:

$$P_L = \sum_{i=1}^{f_{cn}} |X_c(i)|^2 \tag{78}$$

$$P_U = \sum_{i=1}^{f_{cn}} |X_c(i + f_{cn} + 1)|^2 \tag{79}$$

Siendo:

- P_L la potencia en la banda lateral inferior
- P_U la potencia en la banda lateral superior
- $X_c(i)$ la secuencia del espectro de magnitud de la señal interceptada, es decir la transformada de Fourier de la señal recibida.
- $f_{cn} + 1$ es el número de muestra correspondiente a la frecuencia de portadora (0 Hz), ya que, el vector de muestras se inicia en la muestra 1.

El valor ofrecido por esta característica, nos permitirá discriminar entre diferentes modulaciones basadas en amplitud con diferentes propiedades en el dominio de la frecuencia, en nuestro caso, nos permitirá discriminar entre LSB y USB entre sí, además, también nos permitirá discriminar estas dos modulaciones del resto.

Siendo para LSB un valor de P de 1, sin embargo, para USB se obtendrá un valor de P de -1, para el resto de modulaciones, al tener un espectro simétrico respecto de la portadora, el valor de P será de 0, aunque nuevamente, al haber ruido en el canal, estos valores, en la práctica no serán exactamente los dispuestos aquí, pero sí muy cercanos.

3.9.2 Generación del Training Data Set

La obtención del training data set, requiere primeramente de una gran cantidad de señales moduladas, para, posteriormente, extraer de cada señal modulada las características clave detalladas previamente, obtenidas las características clave, se deberá de formalizar la matriz de objetivos.

Por tanto, este apartado, será desarrollado en una serie de sub apartados, en cada uno de los mismos se detallarán los procedimientos comentados, para, finalmente obtener el training data set en su totalidad.

3.9.2.1 Generación de señales moduladas y extracción de características

El entrenamiento de la red neuronal, ha sido llevado a cabo mediante una gran cantidad de valores de las características comentadas anteriormente.

Para ello, primeramente, debe generarse la señal modulada para la posterior extracción de dichas características. Dichas modulaciones han sido generadas mediante Matlab en su equivalente paso bajo, además, se ha simulado el efecto del ruido en el canal, añadiendo ruido paso bajo a las componentes en fase y en cuadratura de las señales moduladas.

La generación de señales moduladas en su equivalente paso bajo y la adición de dicho ruido debe realizarse ya que, las señales que posteriormente se reciban del SDR para su clasificación, vendrán dadas en su equivalente paso bajo como se comentó anteriormente junto a ruido.

Es decir, el entrenamiento de la red neuronal se realizará mediante la extracción de características de señales que han sido generadas simulando su llegada en recepción.

Siendo las componentes de ruido:

$$n(t) = n_I(t) + jn_Q(t) \tag{80}$$

Por tanto, las señales a las cuales se le extraerán dichas características serán:

$$V_{yn}(t) = I(t) + n_I(t) + jQ(t) + n_Q(t) \tag{81}$$

Con el objetivo de obtener una gran cantidad de datos además de variados, se ha procedido a generar las diferentes modulaciones variando los siguientes parámetros:

		MODULACIONES				
Parámetro	**Rango [mín-máx]**	AM	DSB-SC	LSB	USB	FM
Señal moduladora x(t)	-	✓	✓	✓	✓	✓
Ancho de banda del mensaje o moduladora	[f-f_s/2] Hz	✓	✓	✓	✓	✓
Relación señal a ruido (SNR)	[10-30] dB	✓	✓	✓	✓	✓
Índice de modulación (m)	[0.1-1]	✓	×	×	×	×
Desviación de frecuencia (f_d)	$\beta \cdot fm_{mín}$ Hz $\beta \cdot fm_{máx}$ Hz	×	×	×	×	✓

Tabla 3.7 Parámetros a variar en la generación de modulaciones

La variación de un parámetro u otro para la generación de modulaciones, dependerá del tipo de modulación que se esté generando en cuestión, es decir, si se genera una señal en AM, el parámetro de Desviación de frecuencia no se considerará.

Para la generación de la señal moduladora presente en cada modulación, se ha hecho uso de una función ofrecida por Matlab [37] la cual, a partir de un fichero de audio de larga duración, es dividido en miles de segmentos de 2048 muestras, es decir, esta función aumenta la frecuencia de muestreo de la señal original para, posteriormente, dividir en segmentos de 2048 muestras, cada uno de estos segmentos se corresponderá con una señal moduladora.

La frecuencia de muestreo usada para la generación de las señales moduladoras ha sido de 200 KHz, por tanto, la duración de cada segmento de señal moduladora es de 10.24 milisegundos.

Este aumento de la frecuencia de muestreo, y, esa división de la señal original en segmentos de mucha menor duración nos permitirá:

- Obtener muestras de cada segmento parecidas entre sí, es decir, habrá mayor correlación entre las muestras que componen un segmento.
- Obtener mayor cantidad de señales moduladoras.

Además, no solo se ha normalizado la señal original moduladora respecto su valor máximo, sino que cada uno de estos segmentos en los cuales ha quedado dividida la señal

original, han quedado normalizados, de esta manera se obtienen valores de amplitud más elevados.

A continuación, se muestra un ejemplo de la forma en la que se generan las modulaciones en AM, variando el índice de modulación (m), este ejemplo es aplicable para el resto de modulaciones del proyecto, siendo generadas de una forma similar con las variaciones de los parámetros correspondientes.

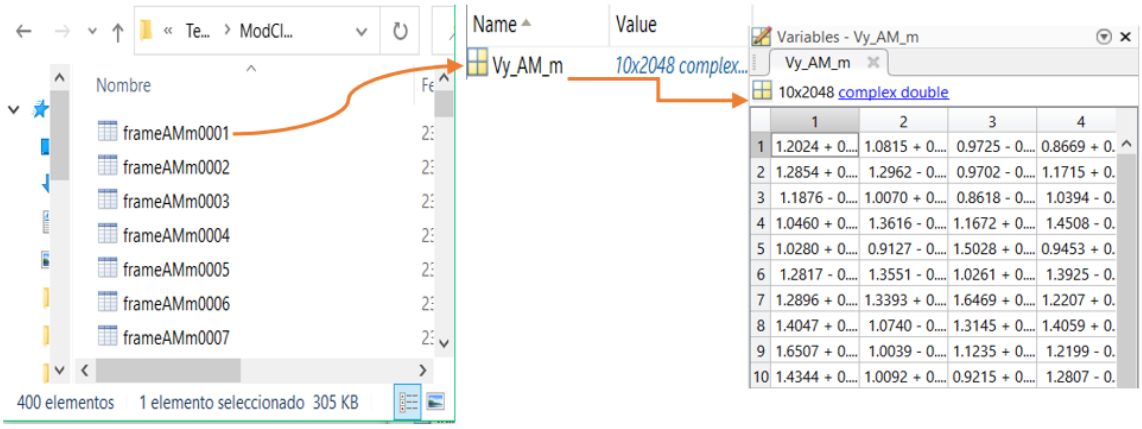


Figura 3.48 Ejemplo de generación de señal AM variando el parámetro m

En la figura mostrada, se han generado 400 ficheros con extensión .mat, cada uno de estos ficheros, contiene una variable en forma de matriz con 10 filas y 2048 columnas.

Cada una de las filas de cada fichero, se corresponde a una señal modulada en AM con un valor del parámetro m diferente, es decir, la primera fila se corresponde con una señal modulada en AM con un valor de m de 0.1, así sucesivamente hasta llegar a la fila 10 correspondiente a una señal generada en AM con un valor de m de 1.

Cada uno de los ficheros .mat o lo que es lo mismo, cada una de las variables matriciales, son generadas haciendo uso de una señal moduladora diferente, sin embargo, los datos de cada fichero son realizados para la misma señal moduladora, aunque para un valor diferente del parámetro m, es decir, en este ejemplo se han generado en total 400 señales moduladoras diferentes, cada una de estas 400 moduladoras, ha sido modulada 10 veces con un valor del parámetro m diferente, obteniendo en total 4000 señales moduladas.

Cabe destacar que, a la hora de variar un parámetro en la generación de una modulación, los demás parámetros vistos anteriormente en la tabla 3.7 han quedado con un valor por defecto, excepto la variación de la señal moduladora que siempre variará, por ejemplo, si en la generación de la modulación AM se ha variado el parámetro m, todos los parámetros restantes variables para este tipo de modulación (SNR, ancho de banda del

mensaje) han quedado fijados en un valor constante, siendo estos valores por defecto para todas las modulaciones los siguientes:

Parámetro	Valor por defecto
Ancho de banda del mensaje	$f_s/2$ Hz
SNR	10 dB
Índice de modulación (m)	0.5
Desviación de frecuencia (f_d)	$\beta \cdot \frac{f_s}{2}$ Hz

Tabla 3.8 Valores por defecto para los parámetros

Explicada la forma en la cual las modulaciones son generadas, se procederá a continuación a detallar el proceso de cálculo de las características mencionadas.

El cálculo de las 4 características, es realizado una vez han sido generadas todas las señales moduladas deseadas para su posterior clasificación.

Continuando con el ejemplo anterior (Figura 3.48) donde se modulaba una señal en AM mediante la variación del parámetro m, se obtenían en total 4000 señales moduladas, por tanto, al tener 4 características que calcular, se obtendrán 16000 valores de características.

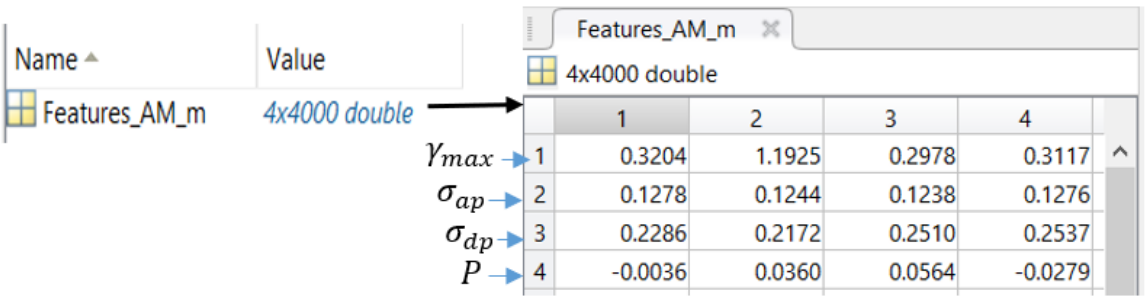


Figura 3.49 Ejemplo de almacenamiento de valores de las características

Como se puede observar, los valores de las características para cada señal modulada quedan almacenados en una matriz, donde, cada fila representa el valor de cada una de las cuatro características. El número de columnas representa el número de señales.

Para cada tipo de modulación se tiene una variable matricial, que, posteriormente se deberán de agrupar de manera conjunta en una única variable matricial para el entrenamiento de la red.

3.9.2.2 Inputs de la red neuronal

Una vez han sido extraídas las características correspondientes a todos los tipos de modulaciones deseadas, han sido agrupadas en una única variable matricial, obteniéndose de esta manera las entradas a la red neuronal, en este caso, para el entrenamiento de la red se han utilizado un total de 18080 segmentos de señales moduladas, y, por tanto, un total de 72320 valores de características.

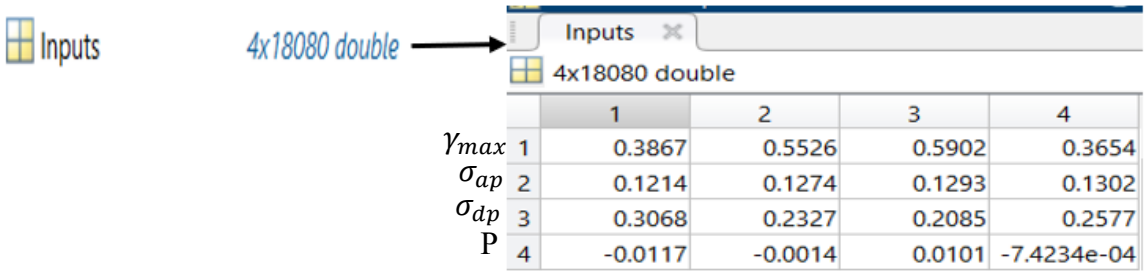


Figura 3.50 Conjunto de entradas usado para el entrenamiento de la red neuronal

3.9.2.3 Targets de la red neuronal

El training data set requiere de datos etiquetados (entradas y sus correspondientes salidas u objetivos) como se comentó en capítulos anteriores, en este caso, la variable de salidas se corresponde de nuevo con una matriz, en este caso de 5 filas y 18080 columnas.

Figura 3.51 Targets de la red neuronal

Donde, cada columna se corresponde con un tipo de señal modulada, y, se tienen 5 filas ya que se requieren en total 5 combinaciones de dígitos diferentes para poder asignar las entradas a 5 tipos de modulaciones diferentes a clasificar. La combinación de dígitos seleccionada para cada tipo de modulación ha sido la siguiente:

Modulación	**AM**	**DSB-SC**	**LSB**	**USB**	**FM**
COMBINACIÓN DE DÍGITOS	1	0	0	0	0
	1	1	0	0	0
	1	1	1	0	0
	1	1	1	1	0
	1	1	1	1	1

Tabla 3.9 Asignación de las salidas para las diferentes entradas a la red neuronal

3.9.2.4 Training Data Set

Obteniendo, por tanto, el data set correspondiente para el entrenamiento de la red neuronal.

Figura 3.52 Data Set de entrenamiento de la red neuronal

Para una mejor comprensión de las Inputs y Targets que componen el Data Set, se realiza la siguiente tabla:

Señal modulada	1	2	3	…	18080
γ_{max}	$F1_1$	$F1_2$	$F1_3$	…	$F1_{18080}$
σ_{ap}	$F2_1$	$F2_2$	$F2_3$	…	$F2_{18080}$
σ_{dp}	$F3_1$	$F3_2$	$F3_3$	…	$F3_{18080}$
P	$F4_1$	$F4_2$	$F4_3$	…	$F4_{18080}$
Clase	AM	FM	USB	…	FM

Tabla 3.10 Ejemplo del Training Data Set

3.9.3 Entrenamiento y aprendizaje de la red neuronal, validación y testeo

Obtenido el Training Data Set, el entrenamiento, validación y testeo ha sido llevado a cabo mediante la red neuronal presentada anteriormente.

En cuanto al entrenamiento, como se comentó anteriormente, los algoritmos usados han sido el Descenso del Gradiente basado en momentum y Ratio de Aprendizaje Adaptativo con BackPropagation, en base a dichos algoritmos se confeccionan los valores de los siguientes parámetros:

Parámetro	**Valor**
α_{adpv}	$1e-5$
α_{inc}	1.04
α_{dec}	0.7
Épocas a completar	1000
Error objetivo	$1e-3$

Tabla 3.11 Valores de los parámetros usados para el entrenamiento

Siendo una época un ciclo completo del conjunto de datos de entrenamiento a través de la red neuronal, y, el error objetivo, el error que se pretende conseguir.

La combinación de los valores presentados en la tabla 3.11, permitirá ofrecer el mínimo error con el menor número de iteraciones posible.

Como función de coste asociada al entrenamiento, ha sido elegida la función Error Cuadrático Medio (*Mean Squared Error -MSE*), siendo expresada como:

$$MSE = \overbrace{\frac{1}{N}\sum_{i=1}^{N} E(i)}^{\text{Mean}} \qquad (82)$$

Donde:

$$E = \overbrace{(Targets - Z^L)^2}^{\text{Error}}{}^{\text{Squared}} \qquad (83)$$

Siendo:

- N, número de datos de entrenamiento
- Targets (T), el objetivo actual
- Z^L, el objetivo calculado, es decir, la salida de las neuronas de la última capa (capa de salida)

La selección de la función de coste como la función del error cuadrático medio permitirá la penalización con mayor intensidad a los valores predichos por el modelo, que, tengan un mayor error, y, con menor intensidad a los valores que tengan un menor error, debido a esa elevación cuadrática.

Generada la red neuronal, e inicializados todos los parámetros correspondientes, se procede a realizar una división en el training Data Set, de manera que un 70% de los datos sean usados para entrenamiento, un 15% para validación y un 15% para testeo.

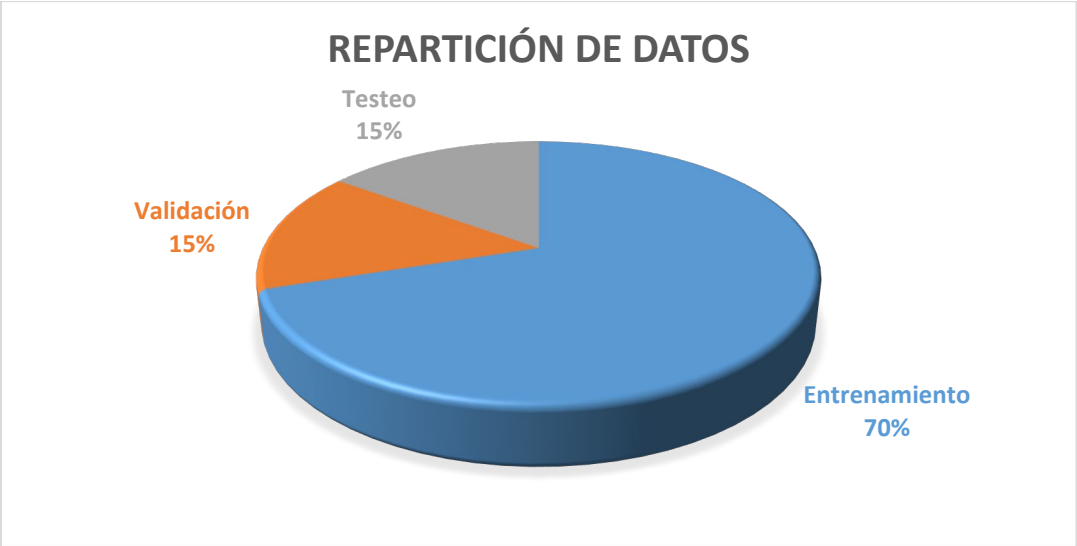


Figura 3.53 División del Training Data Set

Realizada la división de datos, se ha procedido al entrenamiento de la red neuronal implementada.

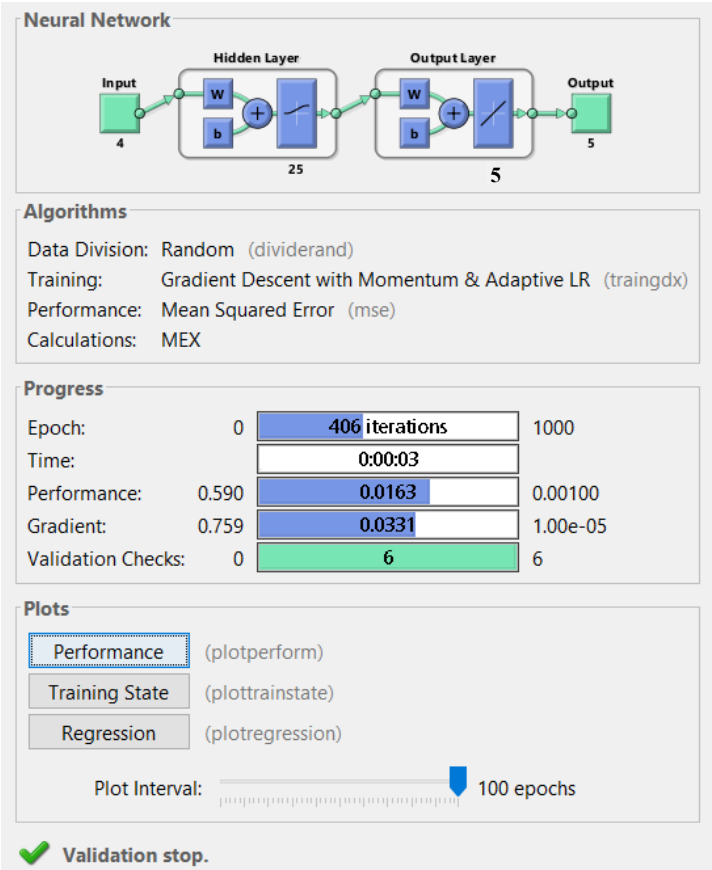


Figura 3.54 Entrenamiento de la ANN

Como se puede observar, en tan solo tres segundos, se ha alcanzado un error considerablemente bajo a partir del entrenamiento, mediante 406 iteraciones.

Donde, para cada época, el valor MSE calculado es comparado con el parámetro del error objetivo de la siguiente manera:

- Si MSE ≈ Error objetivo, el aprendizaje se detiene, en este caso, los pesos y sesgos de la red neuronal son guardados para el futuro proceso de la validación y testeo.
- Si MSE > Error objetivo, el algoritmo continúa entrenando la red hasta que el máximo número de épocas haya sido alcanzado.

En el caso visto, el error ofrecido por la MSE para la época 400 ha sido similar al error objetivo, deteniéndose el proceso de entrenamiento.

A continuación, se muestra el proceso del entrenamiento, validación y testeo de la red neuronal implementada.

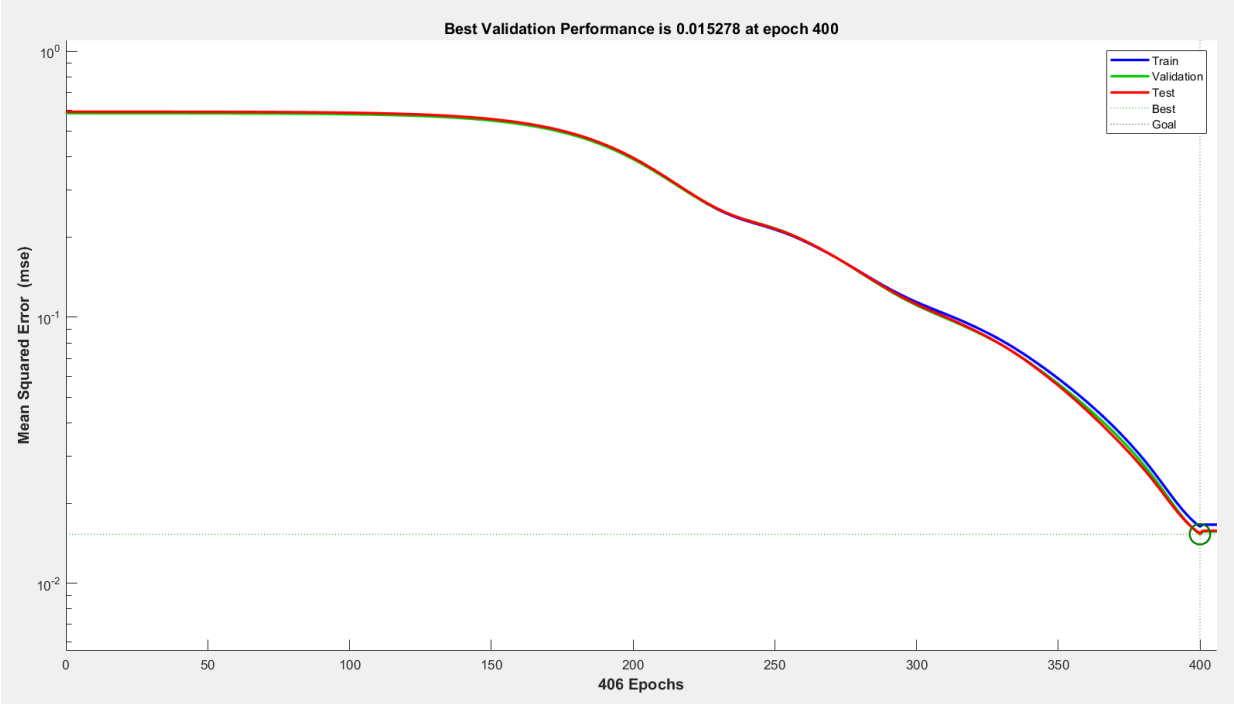


Figura 3.55 Proceso de entrenamiento, validación y testeo de la ANN

4 RESULTADOS Y DISCUSIÓN

Este capítulo tiene como finalidad mostrar los resultados obtenidos en la clasificación tanto de nuevas señales generadas a partir de Matlab, es decir, señales que no viajan sobre el '*aire*', como de señales que van sobre el '*aire*', es decir, transmitidas y recibidas con los SDRs.

Estos resultados serán mostrados a través de tablas y de matrices de confusión, esta herramienta permitirá visualizar el desempeño del clasificador desarrollado. Consiste en una matriz cuyas filas y columnas se corresponden con los tipos de modulación, el resultado será óptimo cuando cada muestra de señal modulada sea clasificada como la clase que realmente es, de forma gráfica esto situaría a la muestra en la diagonal de la matriz (misma fila, misma columna). Cuando se clasifica de forma errónea, aparecen dos tipos de error, falso rechazo y falsa alarma comentados en el punto 3.4, aunque lo interesante será el **error total**.

En base a los resultados obtenidos, se sacarán una serie de conclusiones para cada tipo de modulación clasificada.

Posteriormente, se comentarán aspectos muy interesantes para desarrollar en trabajos complementarios.

4.1 Resultados en la clasificación de señales que no van sobre el aire

Para el testeo del clasificador implementado sobre señales que no viajan sobre el aire, se ha propuesto la generación de un total de 18080 señales, las cuales serán moduladas de forma aleatoria entre los cinco tipos de modulación propuestas en el proyecto.

Al tratarse de una evaluación de la clasificación, se deberán de tener las salidas correspondientes a las entradas para determinar el acierto y el error en la clasificación.

Se muestra a continuación la matriz de confusión obtenida:

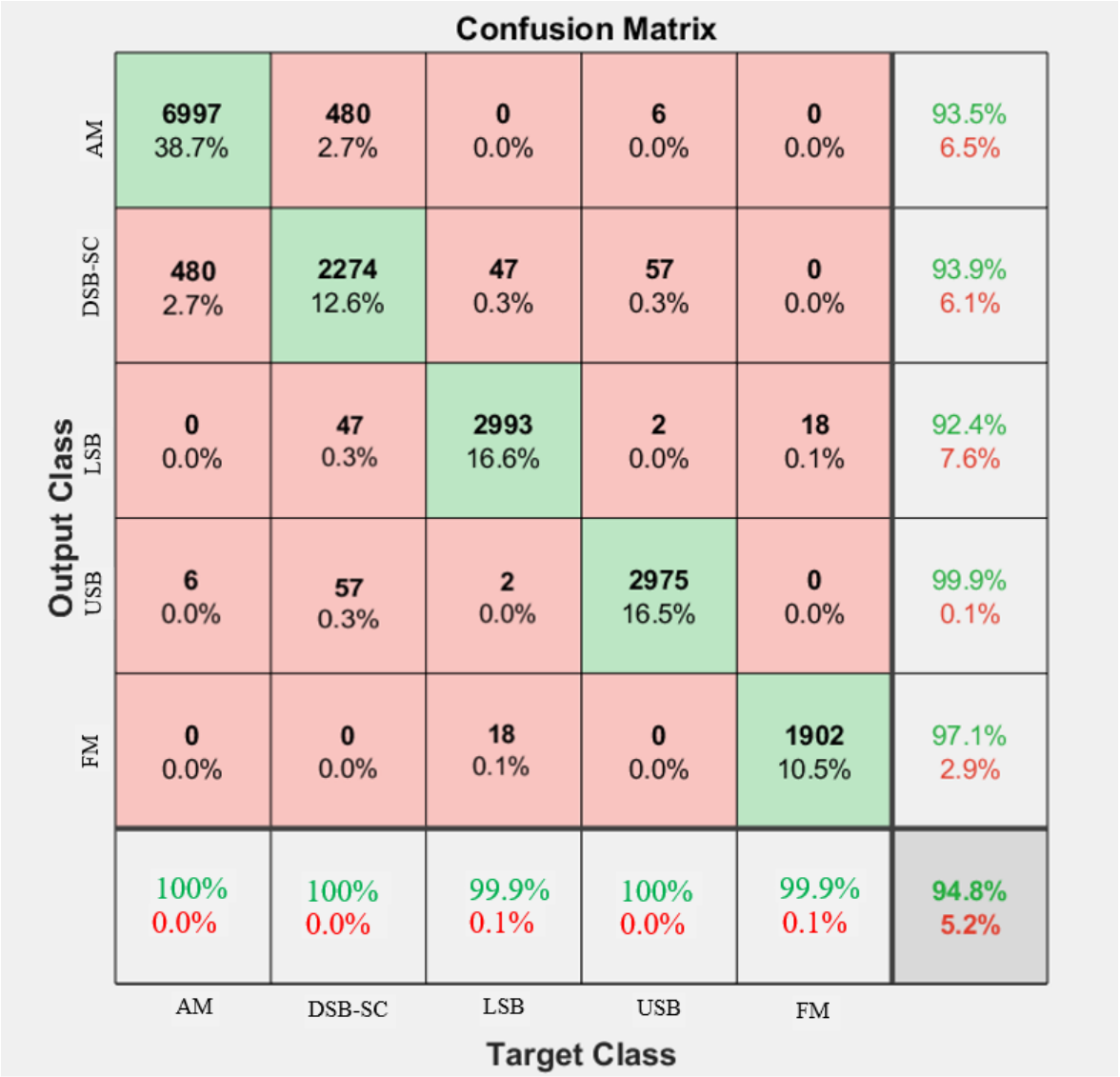


Figura 4.1 Matriz de Confusión obtenida para señales que no viajan sobre el aire

Se puede observar cómo un 94.8% de veces, la clasificación de modulación es correcta, sin embargo, un 5.2% de veces, se produce una clasificación errónea. Siendo este error el error total, es decir, la celda situada abajo derecha ofrece el error total.

Donde, los porcentajes de color negrita, es decir, los que aparecen en los datos en sí, cuando la celda es de un tono verdoso, ese porcentaje indica para el total de datos para una determinada modulación, el porcentaje acertado, sin embargo, el porcentaje de las celdas de color rojizo, indica para el total de datos, el porcentaje correspondiente a los datos que han sido clasificados erróneamente.

Por otro lado, los porcentajes que aparecen al final de cada columna, son los porcentajes de error y acierto vinculados a las clases de modulaciones, es decir, tenemos modulaciones lineales y angulares, por ejemplo, la primera columna correspondiente a AM, ofrece un 100% de aciertos a la clase de modulaciones lineales, es decir, aunque no todas las

muestras de la modulación AM se hayan clasificado como modulación AM, si se han clasificado como que pertenece a una modulación lineal.

Por último, los porcentajes que aparecen en cada fila se corresponde al error total para cada modulación, es decir, para la primera fila, se han obtenido que un 93.5% de aciertos en la clasificación de AM, sin embargo, un 6.5% de errores al clasificar.

Como último detalle, se puede observar que los errores de clasificación suelen ser más propensos a ocurrir para una misma clase, es decir, se observan menor número de errores en la clasificación entre FM y el resto de modulaciones.

4.2 Testeo con SDR

Este apartado, se desarrollará en una seria de sub apartados, en el primero de ellos, se dará a conocer tanto el proceso que se ha seguido a la hora de procesar las señales recibidas, como el proceso seguido para la clasificación de las mismas.

En un siguiente sub apartado se mostrará una tabla la cual muestra los valores de los diferentes parámetros usados en los bloques SDR para transmitir y recibir las señales.

Seguidamente, se comentarán una serie de problemas obtenidos mediante la transmisión y recepción de señales con los SDRs.

Posteriormente se definirán e implementarán una serie de soluciones a dichos problemas, para, finalmente mostrar los resultados obtenidos en la clasificación de señales que van sobre el aire.

4.2.1 Proceso de recepción y clasificación de señales que van sobre el aire

Primeramente, se ha procedido a la configuración de los equipos transmisor y receptor.

Seguidamente, con el objetivo de obtener diferentes fragmentos de señales moduladas, se han generado uno a uno los distintos esquemas de modulación en GNU radio, los cuales serán mostrados en el anexo correspondiente.

Comenzada la transmisión de señales moduladas mediante GNU radio y el SDR número 1, se pasa a la recepción de las mismas mediante Matlab y el SDR número 2, donde, la frecuencia de muestreo usada ha sido de 150 KHz, esta frecuencia de muestro ha sido escogida por un lado, para asegurar que las muestras recibidas se correspondan a fragmentos de señal modulada, por otro lado, la elección de este parámetro, ha venido impuesto por la mínima tasa de muestreo capaz de ofrecer los SDRs, presentada en la tabla 3.3.

La recepción de señales se realizará en intervalos de 3.1 segundos, es decir, cada 3.1 segundos se almacenará un bloque de señal recibida.

Cada intervalo de señal correspondiente a 3.1 segundos, se corresponderá con un vector fila compuesto por 465000 muestras complejas.

Con el objetivo de poder clasificar fragmentos de señal mucho más pequeños, es decir, clasificar señales moduladas transmitidas en un corto periodo temporal, cada intervalo de 3.1 segundos correspondientes a 465000 muestras se dividirá en 227 sub bloques, cada uno contendrá 2048 muestras complejas equivalente a una duración de 13.7 ms.

A cada uno de esos sub bloques se le calcularán las características clave para posteriormente introducirlos en la red neuronal ya entrenada.

Finalmente, se determinará para cada sub bloque el tipo de modulación que presenta, por ejemplo, en caso de transmitir una señal en LSB, en recepción, si todo funciona correctamente, se deberían de clasificar la mayoría de esos 227 sub bloques como LSB.

A continuación, se muestra una figura (Figura 4.2) en la que se presenta el proceso descrito previamente junto con una serie de detalles a tener en cuenta en la clasificación de las muestras recibidas.

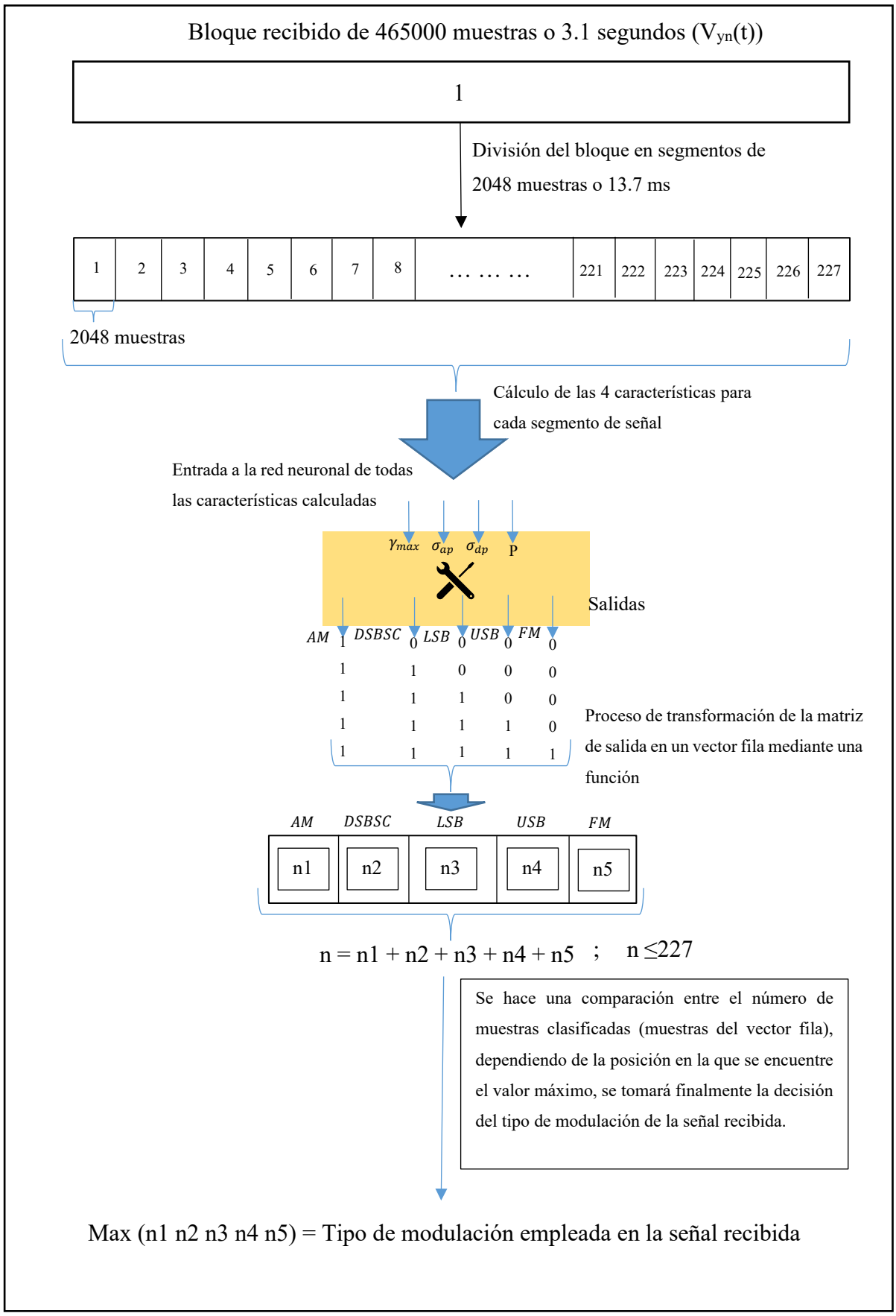


Figura 4.2 Proceso de clasificación de señales provenientes del aire

La función que transforma la matriz de salida de la red neuronal en un vector fila, permitirá saber el número de veces totales que la señal recibida ha sido clasificada como un tipo de modulación u otra, además, permitirá no contabilizar los segmentos de señal que solo sean ruido, es decir, el vector fila correspondiente a la transformación de la matriz de salida, ofrece para cada columna, el número de segmentos que han sido clasificados como un tipo de modulación, donde el valor de n, será menor que 227 en los casos en los que la red neuronal ofrezca una salida que no se corresponda con ningún tipo de modulación, ya que, en casos en los que se produzcan pausas en la transmisión, los segmentos recibidos serán únicamente ruido, en los casos donde esos segmentos recibidos sean ruido, no se contabilizarán como ningún tipo de modulación.

A diferencia de las señales moduladas generadas en Matlab, es decir, que no van sobre el aire, introducidas en la red neuronal para el testeo, siempre se consideraron señales moduladas, es decir, si se tenían 18080 segmentos, se obtendrían 18080 clasificaciones de algún tipo de modulación.

Ahora, con las señales que van sobre el aire, se podrían ocasionar pausas en la transmisión o, instantes de tiempo en los que la señal modulada tuviese una potencia muy baja, por tanto, si reciben 227 segmentos, podría ocurrir que la mitad de ellos se recibieran a partir de señales moduladas, y la otra mitad solo a partir de ruido quedando por tanto sin contabilizar estos segmentos de ruido.

A continuación, se presenta el sistema físico real usando los SDRs.

Figura 4.3 Sistema físico (TX-RX)

4.2.2 Valores de los bloques SDR usados en transmisión y recepción

Parámetro	SDR Transmisor	SDR Receptor
Tasa de muestreo	150000 Hz	150000 Hz
Frecuencia de portadora	915 MHz	915 MHz
Ganancia RF	30 dB	30 dB
Ganancia IF	-	-
Ganancia Banda Base	20 dB	20 dB
Canal	TX1	RX2
Ancho de Banda	1.5 MHz	1.5 MHz

Tabla 4.1 Valores usados en los bloques SDR

En cuanto al tipo de señales moduladoras usadas, han sido varias para variar, todas estas han sido señales de audio, limitadas en banda mediante un filtro paso bajo (*Low Pass Filter - LPF*), donde, finalmente, en transmisión, se debe de realizar una interpolación, aumentando la frecuencia de muestreo de las mismas, adaptándolas a las frecuencias de muestreo requeridas por los SDRs.

El ancho de banda de los bloques SDR, ha sido seleccionado a ser el valor mínimo disponible, ya que, las señales de audio normalmente están contenidas entre 20 y 20 KHz, por tanto, aunque se usen modulaciones que eleven el ancho de banda original, no superarán el ancho de banda del canal, por tanto, no se producirá ninguna pérdida de información.

4.2.3 Problemas con la transmisión y recepción de señales mediante SDR

A continuación, se comentarán una serie de problemas que presentan la transmisión y recepción de señales de mediante los SDRs usados, cabe destacar que este problema será relevante o no dependiendo del tipo de aplicación para la cual los SDRs son usados, en la aplicación desarrollada en este proyecto, será relevante a la hora de clasificar todas las señales moduladas, sobre todo AM y DSB-SC, **imposibilitando** la correcta clasificación de señales AM y dificultando la clasificación de señales DSB-SC.

Por último, estos problemas estarán presentes cuando se transmiten y reciben señales mediante el uso de dos SDRs, es decir, cuando se transmite y recibe una señal usando el mismo SDR estos problemas no están presentes.

A continuación, se explica el problema y su afecto sobre las modulaciones comentadas anteriormente.

El problema reside en que simplemente por el hecho de comenzar la transmisión en el SDR transmisor, aunque no se envíe ninguna información por el canal, es decir, se tienen

valores de 0, en el receptor, se observa la presencia de un tono en el tiempo o equivalente a un pico de continua en frecuencia desplazado a -f Hz, esto quiere decir que hay un pequeño desajuste entre SDR transmisor y receptor como se comentará más adelante, ocasionando tres problemas a la hora de clasificar modulaciones, sin embargo, dependiendo del tipo de modulación en cuestión, de estos tres problemas, podrían afectar todos, dos o únicamente uno.

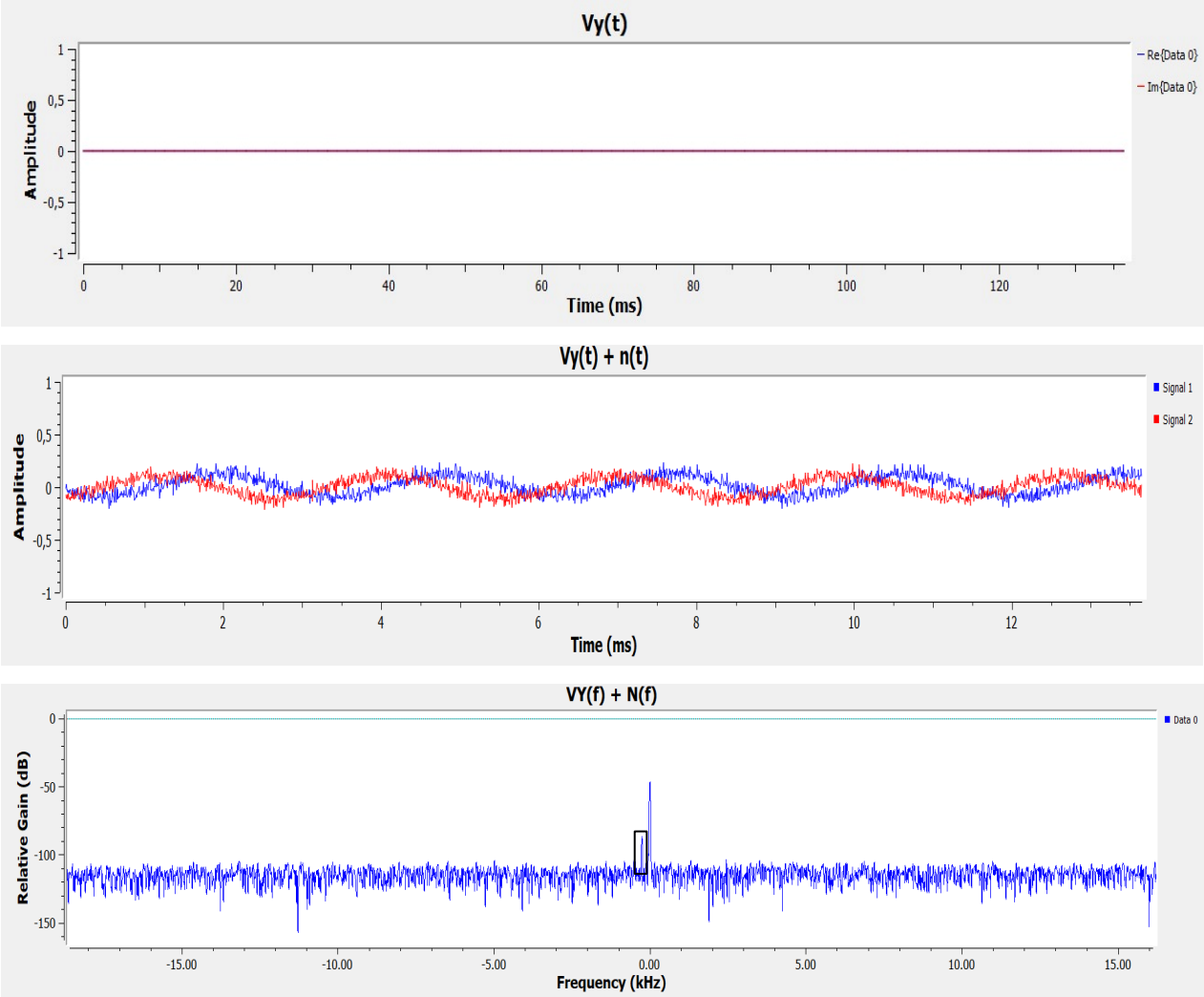


Figura 4.4 Problema 1 en la transmisión y recepción de señales con los SDRs

Esto dificultará completamente la clasificación de cualquier señal modulada por los siguientes motivos.

- **Problemas de clasificar señales AM transmitidas y recibidas mediante SDR (Afectan 3/3 problemas).**

  En AM, se observarán tres problemas, el primero de ellos será que la señal recibida, tendrá una **media nula**, es decir, una señal modulada en AM, siempre tiene una media, debido a esa suma que se produce en su expresión correspondiente (ec.16). Por tanto, el SDR al eliminar la media de las señales que recibe, no se podrán clasificar señales AM.

  A continuación, en la figura 4.5, se observa en el dominio del tiempo, la envolvente compleja correspondiente a una señal AM enviada y recibida, mediante los SDRs.

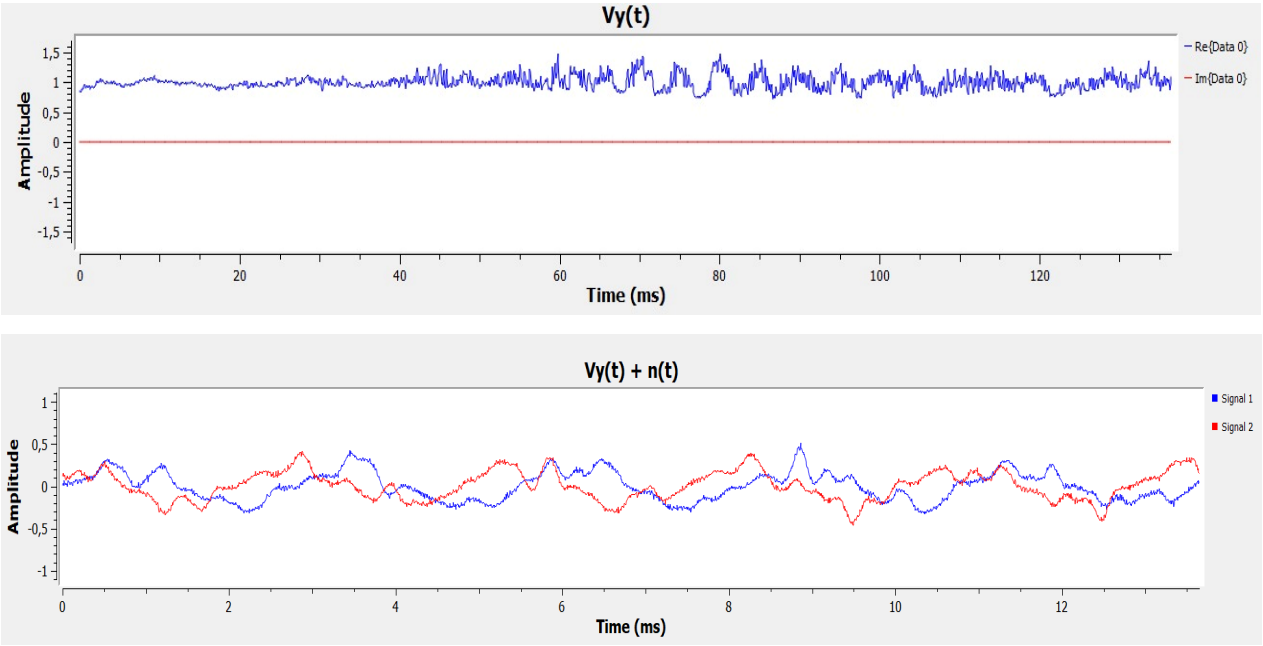


Figura 4.5 Problema 2 en la transmisión y recepción señales con los SDRs (TX-RX)

Como se puede observar, la señal enviada tiene una media en torno a 1, sin embargo, la señal recibida tiene una media en torno a cero, lo que causará tener valores de la envolvente compleja negativos como puede observase en la segunda figura, traduciéndose esto en valores de fase distintos a cero, por ende, se tendrá un cálculo erróneo de las características σ_{ap}, σ_{dp}, ya que, como se dijo anteriormente en el punto 3.3.1.1, la fase de una señal AM debe ser 0 o muy cercano a este puesto que los valores de la envolvente compleja siempre son positivos.

El segundo problema que se observa es debido a ese tono no deseado ya que, no solo se recibe señal de la componente en fase, sino que también se recibirá señal de la componente en cuadratura con valores de amplitud similares, es decir, cuando se obtuvieron las componentes en fase y cuadratura de la señal en AM (ec.16,17), se observaba que la componente en cuadratura era 0 o muy cercano a este debido al ruido presente en el canal, sin embargo, debido a ese tono no deseado, se están recibiendo valores de la componente en cuadratura no cercanos a 0.

Por último, el tercer problema será debido al cálculo de la característica P y es que, este desajuste entre transmisor y receptor, está haciendo que las muestras de señales moduladas recibidas estén desplazadas como se observa en la siguiente figura.

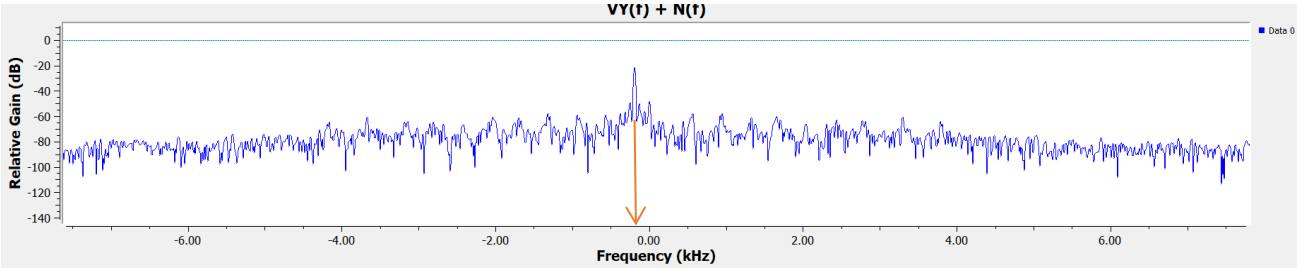


Figura 4.6 Problema 3 en la transmisión y recepción señales con los SDRs (TX-RX)

Por tanto, el valor del cálculo de la potencia en la banda lateral inferior será siempre mayor al valor ofrecido por la banda lateral superior, ya que como se observa, la frecuencia de portadora no se localiza justo en 0 Hz, sino que se encuentra desplazada a una frecuencia negativa.

Se han visto por tanto tres problemas que acarrea el uso de SDRs para la transmisión y recepción de señales, el primero de ellos, es que el SDR elimina la media de las señales, el segundo de ellos es debido a que el tono no deseado presente en la recepción de señales se compone no solo de componente en fase sino también de componente en cuadratura, afectando esto a las modulaciones cuyo valor de la componente en cuadratura sea 0, por último, el tercer problema es debido al cálculo de la característica P ya que, se están recibiendo las muestras desplazadas, donde, estos tres problemas afectan en su totalidad a la clasificación de las señales AM por los motivos comentados.

Estos problemas debido a los SDRs, causarán que la modulación en amplitud no se pueda clasificar correctamente como se observa a continuación, ya que, se han detectado los

fragmentos de señales recibidos como ruido, es decir, ninguno de los 227 fragmentos en los que se dividía la señal capturada ha sido clasificado correctamente.

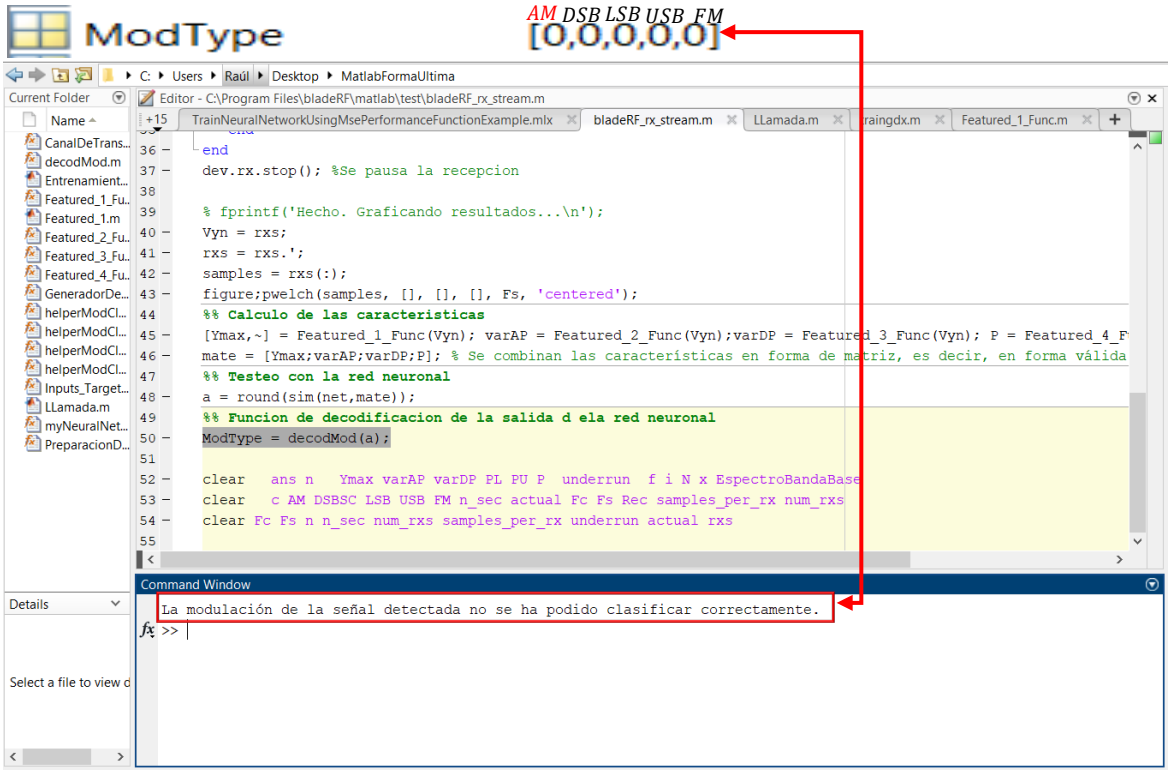


Figura 4.7 Resultado de la clasificación de señal AM transmitida y recibida mediante SDR

Obteniéndose para múltiples pruebas unos resultados de clasificación similares, para concluir una tasa de acierto de aproximadamente un 0 % frente a una tasa de error del 100 % para señales moduladas en AM.

- **Problemas de clasificar señales DSB-SC transmitidas y recibidas mediante SDR (Afectan 2/3 problemas).**

  El problema en DSB-SC será similar al contado anteriormente, y es que, aunque la media de la envolvente compleja de una señal DSB-SC sea nula o cercana a este valor, nuevamente se está recibiendo señal no deseada tanto en la componente en fase como en cuadratura, donde la componente en cuadratura para una señal modulada en DSB-SC debe ser cero (ec.23).

  Por último, el problema de la característica P también está presente, ya que se está recibiendo un espectro desplazado.

- **Problemas de clasificar señales LSB, USB, FM transmitidas y recibidas mediante SDR (Afectan 1/3 problemas).**

  El único problema que afectará en este tipo de modulaciones será el problema del valor de la característica P, a causa de la recepción de un espectro desplazado.

  Como se vio anteriormente, este tipo de modulaciones se componen tanto de componente en fase como en cuadratura, por tanto, el tono no deseado que se recibe junto a la señal deseada, será tratado como una señal más, de ahí que este problema no afecte a la clasificación de estas modulaciones.

Una vez comentados todos los problemas presentes en la transmisión y recepción de señales mediante los dos SDRs, se procederá a detallar las soluciones pertinentes.

4.2.4 ¿A qué se debe la aparición de esa señal no deseada desplazada a -f Hz?, ¿Soluciones?

Una parte del trabajo realizado ha estado dedicado especialmente a la búsqueda de información para una posterior solución de los problemas inesperados presentados.

Este sub apartado se centra en dos de los tres problemas comentados anteriormente, ya que, el problema que se tenía con la eliminación de la media para señales AM por parte de los SDR no ha podido ser solucionado, por tanto, dicha modulación no se podrá clasificar correctamente.

En cuanto a los otros dos problemas encontrados (aparición de un tono no deseado, espectro recibido desplazado), podrán ser resueltos de forma simultánea como se verá a continuación, permitiendo una correcta clasificación de las modulaciones DSB-SC, LSB, USB, FM.

Partiendo de que dichos problemas no están presentes cuando un único SDR es utilizado, se llega a la conclusión de que el problema puede deberse a desequilibrios IQ [39] entre el canal de transmisión del SDR número 1 y el canal de recepción del SDR número 2.

Desequilibrio IQ

Principalmente hay dos tipos de receptores de RF, heterodinos y homodinos.

El receptor heterodino, tiene como mínimo dos etapas de mezcladores, es decir, como mínimo dos etapas de conversión de frecuencia, la primera etapa de conversión de frecuencia traslada la señal RF recibida a una frecuencia intermedia (*Intermediate Frecuency - IF*), seguidamente, a la señal convertida a IF se le aplica un demodulador IQ donde los mezcladores I y Q conforman la segunda etapa, realizando una conversión de la señal IF a banda base. La desventaja principal de este tipo de receptores heterodinos es su compleja

estructura, aunque este tipo de receptores presenta menos problemas de desequilibrio IQ debido a esa doble etapa de conversión de frecuencia.

Por otro lado, los receptores homodinos, solo aplican una etapa de mezcladores, correspondiéndose esta etapa a la demodulación IQ, es decir, entre la señal recibida RF y la señal banda base, solo hay una única conversión de frecuencia.

Hoy en día, muchas plataformas de SDR incluyendo bladeRF SDR, adoptan la arquitectura homodina debido a su simplicidad, sin embargo, este tipo de receptores presenta un gran inconveniente, el desequilibrio IQ.

Un demodulador IQ ideal, debería tener una perfecta simetría en los canales IQ, donde la ganancia de amplificación a lo largo del canal I es idéntica a la ganancia de amplificación a lo largo del canal Q, la señal portadora del canal I tiene exactamente 90 grados de fase frente a la señal portadora del canal Q. Sin embargo, en la realidad, hay una discordancia entre los procesos seguidos entre los canales IQ, lo que conduce al desequilibrio IQ.

El desequilibrio IQ, por tanto, se puede dividir en tres tipos de problemas:

- Desequilibrio en la amplitud de las componentes IQ
- Desequilibrio de fase entre los canales IQ
- Compensación de corriente continua (*Direct Current – DC)*

Siendo el problema de la compensación de CC el que afecta en la clasificación de señales ya que, a causa del mismo, se desarrollan los dos problemas mencionados anteriormente.

Este problema existe debido al oscilador local, conduciendo a una desviación de las muestras recibidas, es decir, existe una **descalibración** entre la frecuencia de la señal generada por el oscilador del canal transmisor del SDR número 1 y la frecuencia del canal de recepción del SDR número 2, donde, para un mismo SDR ambos canales están calibrados entre sí de fábrica ya que, la generación de la portadora es realizada a través del mismo oscilador local para ambos sentidos de la comunicación, transmisión, recepción.

A continuación, se procede a detallar lo comentado anteriormente para una mejor compresión.

El bladeRF, tiene dos canales, uno para transmitir, otro para recibir, ambos canales del mismo SDR usan el mismo oscilador local para la modulación y demodulación, simplemente una de las dos señales generadas estará desfasada “90º” respecto la otra, ambos canales del mismo SDR están calibrados entre sí, esto no quiere decir que, si se transmite por ejemplo a la frecuencia de portadora de 915 MHz, se esté transmitiendo exactamente a

esa frecuencia, lo mismo se aplica para el canal receptor, que se esté recibiendo a 915 MHz, no implica que justamente se reciba a esa frecuencia, pudiendo transmitirse y recibirse por ejemplo a 915.000.100 Hz aquí es donde entra el concepto de **calibración**, y es que, el usar el mismo oscilador local para la transmisión y la recepción de señales, aunque exactamente no se esté ni transmitiendo ni recibiendo a 915 MHz, no habrá prácticamente ningún tipo de desviación en las muestras recibidas ya que, el oscilador usado para la generación de portadoras con ese pequeño error de frecuencia es el mismo usado para la modulación y la demodulación, es decir, ese pequeño error de frecuencia está siendo asumido en ambos sentidos de la transmisión para el mismo SDR.

Sin embargo, el uso de dos SDRs, podría acarrear el tener un error diferente en la frecuencia de la señal portadora generada, es decir, si se está transmitiendo con un SDR a la frecuencia de 915 MHz (realmente a 915.000.100 Hz, por ejemplo) y, se está recibiendo con otro SDR distinto a la frecuencia de 915 MHz (realmente a 915.000.300 Hz, por ejemplo), se está produciendo una desviación en las muestras recibidas, siendo esto lo que está ocurriendo con la transmisión y recepción de señales mediante dos SDRs.

Además, estas deficiencias deben corregirse tanto en **frecuencia** (como se verá posteriormente), como en **temperatura** [40].

Nuevamente se recalca que este problema debido a esa descalibración no sería muy relevante para otro tipo de aplicaciones, a no ser que dicha descalibración fuese relativamente elevada.

Por tanto, a continuación, se propone una solución cuya base ha sido tomada del repositorio oficial de los desarrolladores del bladeRF en GitHub [40]. Esta solución se trata de una corrección de compensación de CC, y, será realizada en el receptor.

La solución se basará en lo siguiente:

- Mediante el uso de GNU radio, dos ordenadores y, dos SDRs, se transmitirá y recibirá de forma simultánea valores de señal iguales a 0.
- Mediante GNU radio, se hará uso de un bloque en el esquema de recepción, el cual permitirá variar la frecuencia de portadora del canal de recepción del SDR receptor mediante un rango de valores de frecuencia predefinido.
- Mediante GNU radio, se visualizará el espectro en frecuencia de la señal que se está recibiendo en tiempo real, localizando el pico de CC originado por ese descalibramiento entre transmisor y receptor.

- A continuación, se ajustará la frecuencia de portadora en el receptor mientras se visualiza el espectro recibido, intentando realizar la calibración entre ambos SDRs, de esta manera se intentará reducir el pico de CC observado.
- Por último, se anotará el error existente de frecuencia entre la frecuencia de portadora a la que se transmite y la frecuencia de portadora (corregida) a la que realmente se recibe, para, de esta manera, siempre que se realice una simulación, recibir a la frecuencia de portadora corregida.

$$f_{c(corregidaRX)} = f_{c(RX)} \pm error \quad \begin{cases} +error & f_{c(realRX)} < f_{c(RX)} \\ -error & f_{c(realRX)} > f_{c(RX)} \end{cases} \qquad (84)$$

Donde:

- $f_{c(corregidaRX)}$, es la frecuencia de portadora que se usará en recepción para conseguir la calibración entre SDR transmisor y SDR receptor
- $f_{c(RX)}$, es la '*supuesta*' frecuencia de portadora a la cual se debería recibir
- $f_{c(realRX)}$, es la frecuencia de portadora que realmente se recibe

Siendo el $error$ o $f_{c(offsetRX)}$:

$$error = \left| f_{c(RX)} - f_{c(realRX)} \right| \qquad (85)$$

A continuación, se muestra una imagen del proceso seguido para la calibración:

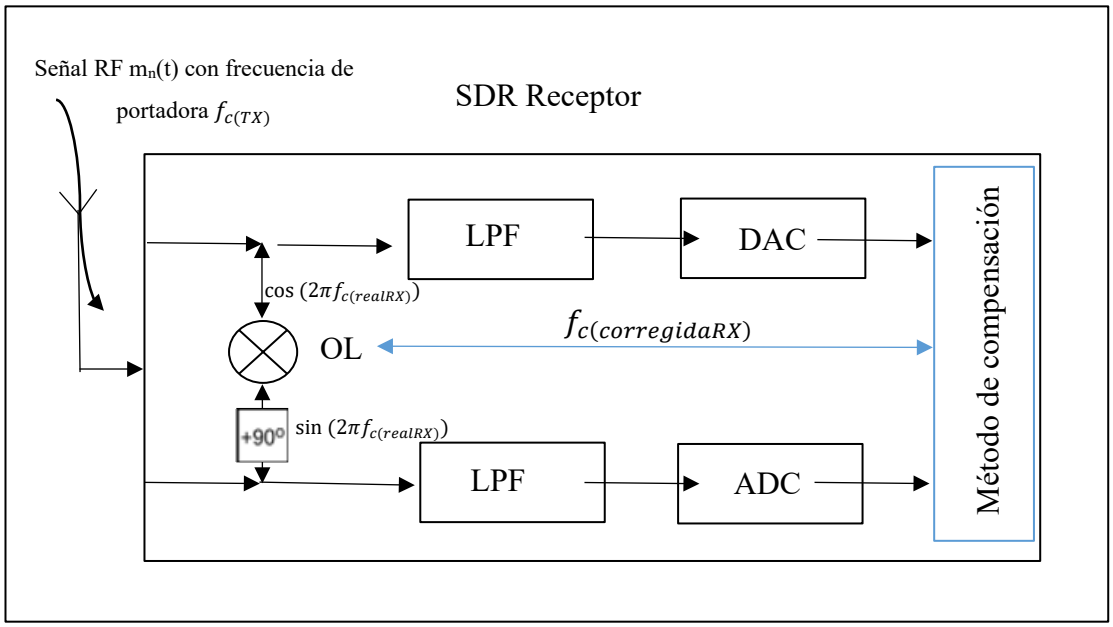


Figura 4.8 Proceso de calibración manual del SDR receptor

Donde, el método de compensación sería realizado como se ha comentado anteriormente mediante GNU radio.

Por último, se detalla el proceso seguido en el método de compensación, siendo este proceso el descrito anteriormente.

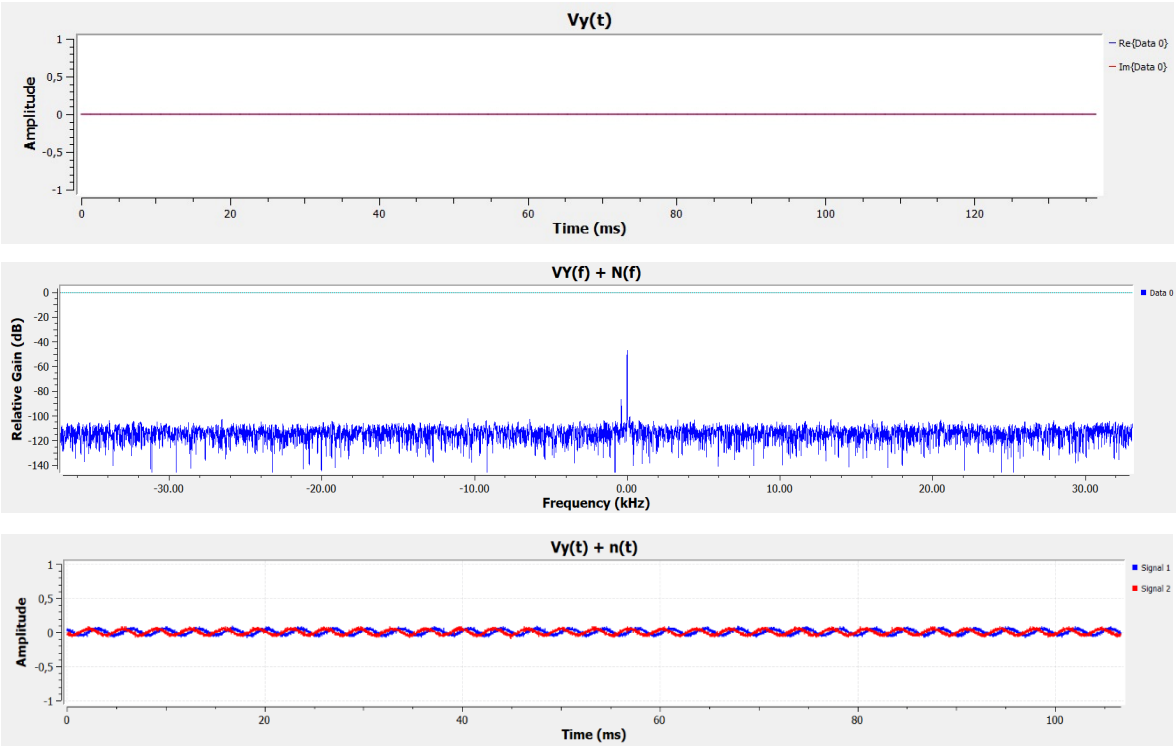


Figura 4.9 Proceso de calibración_1

La primera sub figura muestra en el dominio del tiempo la señal enviada al SDR para su posterior transmisión a la frecuencia de portadora $f_{c(TX)}$ de 915 MHz, la segunda sub figura muestra la señal recibida en el dominio de la frecuencia a la frecuencia de portadora $f_{c(RX)}$ de 915 MHz, aunque, el espectro es observado en su **equivalente paso bajo** en la frecuencia de portadora de 0 Hz, observándose ese pico de continua desplazado a -f Hz comentado anteriormente. La última sub figura muestra en el dominio del tiempo la señal recibida, observándose el tono en sus componentes en fase y cuadratura.

Lo que se hará a continuación será variar la frecuencia de portadora del SDR receptor $f_{c(RX)}$, para, de esta manera, calibrar ambos SDRs.

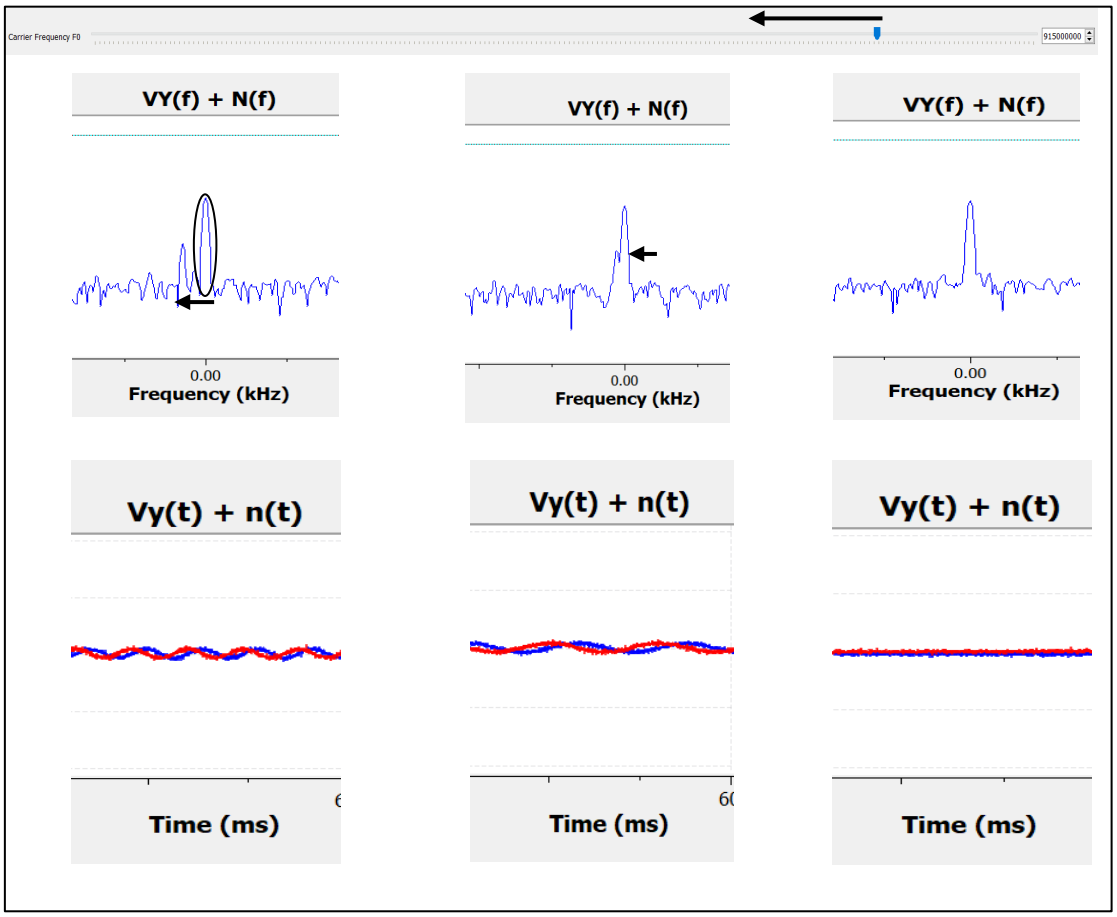


Figura 4.10 Proceso de calibración_2

Las sub figuras muestran el proceso de calibración en el dominio de la frecuencia y su correspondiente forma temporal.

El tono recibido se correspondía con un pico de continua en la frecuencia de -353 Hz, es decir, el SDR receptor, realmente está recibiendo no a la frecuencia $f_{c(RX)}$ de 915 MHz sino a la frecuencia de $f_{c(realRX)}$ 915.000.353 Hz, por tanto, se ha procedido a bajar la $f_{c(RX)}$ hasta compensar ese error de frecuencia, siendo:

$$f_{c(corregidaRX)} = f_{c(RX)} - error = 915\ MHz - 353\ Hz = \ 914.999.647 Hz$$

Por tanto, cuando se quiera transmitir a la frecuencia de 915 MHz, se deberá recibir a la frecuencia de 914.999.647 Hz.

De esta forma, siempre que se quiera sintonizar el receptor a una frecuencia de transmisión determinada, se deberá de bajar o subir la misma en 353 Hz, aunque como se dijo anteriormente, este error puede aumentar o disminuir, ya que, también depende de la temperatura, además, este será el error determinado para el SDR transmisor en cuestión.

De esta forma, aunque el tono persista, ha quedado cancelado al mínimo posible, por otro lado, el espectro que se recibirá ahora estará centrado **aproximadamente** en 0 Hz, la palabra '*aproximadamente*' queda resaltada debido a que como se ha comentado previamente, esto también depende de la temperatura a la cual ambos SDRs se encuentren, por tanto, aunque la modulación AM siga siendo imposible de clasificar ya que la señal que se recibirá seguirá llegando sin media en cualquier instante temporal, las demás modulaciones sí que se podrán clasificar correctamente como se verá en el punto 4.2.5 .

Por otro lado, como se ha comentado, la temperatura también influye en la calibración de ambos SDRs, no influirá tanto como para hacer que las muestras se reciban desplazadas en centenas de hercios, pero si en pocas unidades de hercios, es decir, variaciones en la temperatura de ambos SDRs influirán en si las muestras paso bajo que se reciben están **justo** centradas en 0 Hz o si estás se reciben un poco desplazadas a ± f Hz.

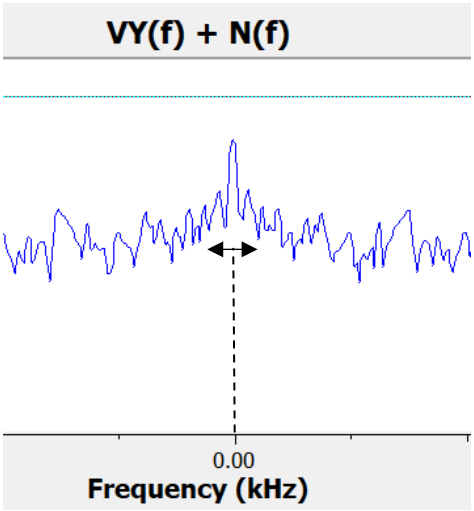


Figura 4.11 Efecto en las variaciones de temperatura de los SDRs sobre el espectro de señal recibido

Y esto realmente es un problema, ya que, aunque el espectro que se recibe ahora este prácticamente centrado en 0 Hz, esas pequeñísimas variaciones de las muestras recibidas seguirán haciendo que el cálculo de la característica P, no sea correcto, por tanto, para la solución de este problema se ha decantado por la realización de una función en Matlab, la

cual permita desplazar el espectro de la señal recibida justamente a 0 Hz para todos los segmentos en los cuales se divide la señal recibida (227 segmentos de señal), este proceso lógicamente sería realizado antes de que la característica P sea calculada, de esta manera se obtendrán los valores correctos de dicha característica.

4.2.5 *Resultados de la clasificación de señales moduladas transmitidas y recibidas mediante SDRs*

Antes de comenzar con los resultados obtenidos una vez han sido resueltos los problemas descritos, excepto el de la media, cuya problemática será la de no poder clasificar señales AM generadas y recibidas con los SDRs, se comentará un detalle simplemente a nivel de curiosidad.

Y es que, si se recuerdan los problemas que acarreaban la generación de señales DSB-SC mediante los SDRs (espectro recibido no centrado, recepción de señal no deseada en cuadratura), simplemente solucionando el problema de centrar el espectro, se llegaba a clasificar correctamente dicha modulación incluso con un problema.

Aunque muchos segmentos están siendo considerados como ruido, ya que, a partir de varias pruebas en las que se divide la señal recibida DSB-SC en 227 segmentos a clasificar, en su mayoría de veces está considerando aproximadamente un 90% de segmentos como ruido, es decir, como señales que no han sido involucradas en el entrenamiento de la red neuronal.

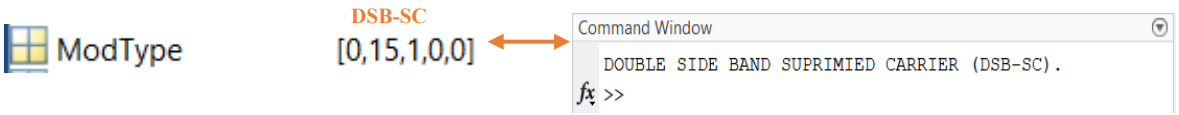


Figura 4.12 Resultado de clasificación de señal DSB-SC mezclada con un tono no deseado en sus componentes en fase y Cuadratura

Obteniéndose para múltiples pruebas unos resultados de clasificación similares

A diferencia de la clasificación realizada para señales AM, la clasificación de señales DSB-SC será correcta en los instantes de tiempo donde se tenga una componente en fase distinta de cero y, una componente en cuadratura cercana a cero, de ahí que se puedan clasificar como DSB-SC las muestras correspondientes a esos instantes de tiempo generando por tanto una salida correcta, sin embargo en AM no bastaría esta condición, ya que la señal recibida en AM es recibida sin media en todo el instante temporal, de ahí que no se clasifique ningún segmento como AM.

El problema realmente de clasificar señales DSB-SC emitidas y recibidas mediante los SDRs sin calibrar, ocurriría cuando la señal DSB-SC transmitida tuviese muy corta

duración temporal, es decir, segmentos de señal mucho menores de 3.1 segundos, en ese caso podría ocurrir que la señal recibida fuese considerada como ruido y, por tanto, no clasificada como ningún tipo de modulación.

Por tanto, gracias a dividir la señal en segmentos tan pequeños, aunque llegase a existir ese tono no deseado debido a la descalibración ya solucionada entre transmisor y receptor, se iba a permitir una correcta clasificación, aunque con el pequeño inconveniente comentado.

A continuación, se mostrarán los resultados obtenidos con el resto de modulaciones contempladas en el proyecto una vez transmisor y receptor han sido calibrados.

- Resultados de clasificación de señal LSB emitida y recibida mediante SDR.

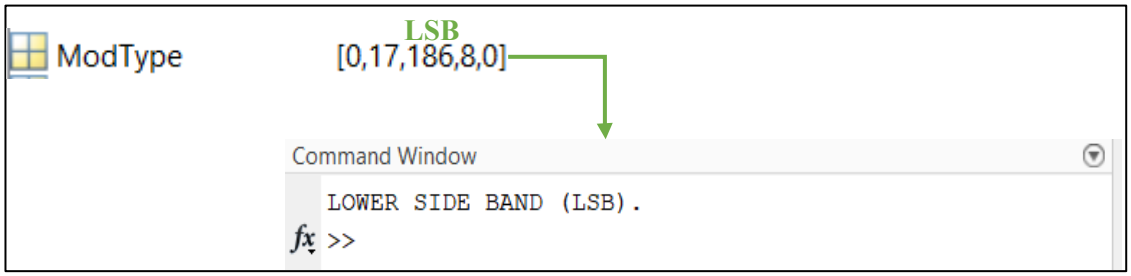


Figura 5.13 Resultado de clasificación de señal LSB transmitida y recibida mediante SDR

Como se puede observar la clasificación de la señal ha sido acertada exitosamente.

Obteniéndose para múltiples pruebas unos resultados de clasificación similares, para concluir una tasa de acierto de aproximadamente un 88.15 % frente a una tasa de error del 11.85 % para señales moduladas en LSB.

A continuación, se detalla el proceso seguido para la recepción y clasificación que, aunque se detalló en 4.2.1, se volverá de nuevo a comentar de forma más resumida para una mejor compresión.

Se ha recibido una señal proveniente del aire, se ha fragmentado en 227 segmentos de 2048 muestras cada uno, se han calculado las características de cada segmento y se han introducido en la red neuronal para su clasificación, la salida de la red neuronal es una matriz de 5 filas y 227 columnas, compuesta de valores binarios, seguidamente esa matriz ha pasado por una función de decodificación, ofreciendo el vector '*ModType*' el cual contiene en cada columna el número de veces que cada fragmento de esos 227 ha sido clasificado como un tipo de modulación u otra.

Se tienen en total 211 fragmentos clasificados como un tipo de modulación, ya que, los restantes fragmentos (227-211) han sido considerados como ruido.

De esos 211 fragmentos, 186 han sido clasificados como fragmentos correspondientes a una señal modulada en LSB, al ser este valor superior al resto de valores presentes en el vector, se ofrece como resultado de señal clasificada como LSB.

Este procedimiento es el seguido en cualquier otro tipo de clasificación de modulación.

Posteriormente se detallará la obtención de dichas tasas.

A continuación, se muestra el resultado de clasificar una señal transmitida en DSB-SC una vez ambos SDRs han sido calibrados.

- Resultados de clasificación de señal DSB-SC emitida y recibida mediante SDR.

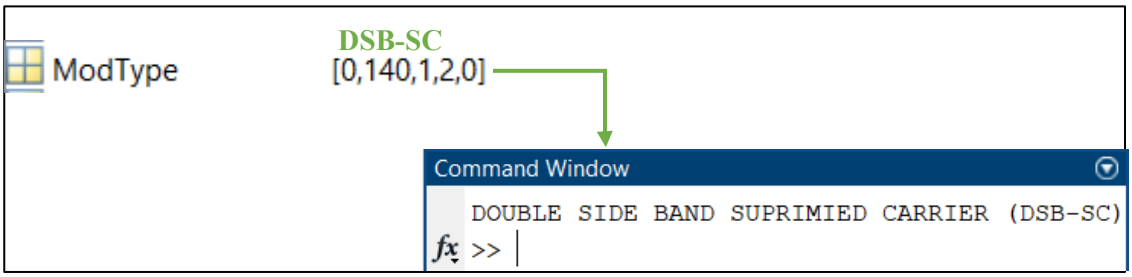


Figura 5.14 Resultado de la clasificación de una señal DSB-SC transmitida y recibida mediante SDR

Obteniéndose para múltiples pruebas unos resultados de clasificación similares, para, concluir una tasa de acierto en la clasificación de modulación de aproximadamente un 99.21 % frente a una tasa de error del 0.79 % para señales moduladas en DSB-SC.

- Resultados de clasificación de señal USB emitida y recibida mediante SDR.

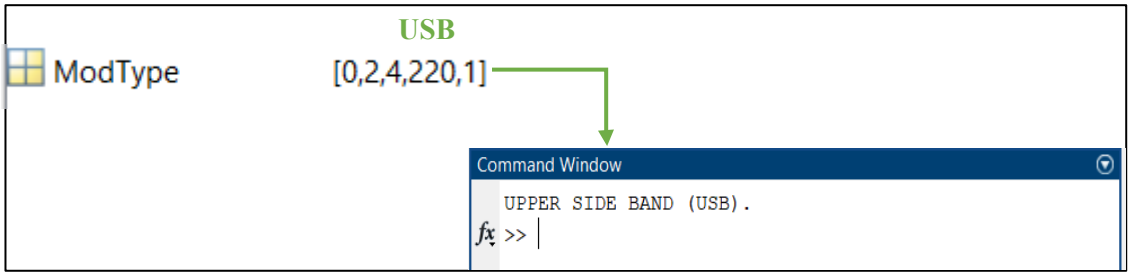


Figura 4.13 Resultado de clasificación de señal USB transmitida y recibida mediante SDR

Como se puede observar la clasificación de la señal ha sido acertada exitosamente.

Obteniéndose para múltiples pruebas unos resultados de clasificación similares, para concluir una tasa de acierto de aproximadamente un 94.9 % frente a una tasa de error del 5.1 % para señales moduladas en USB.

➢ Resultados de clasificación de señal FM emitida y recibida mediante SDR.

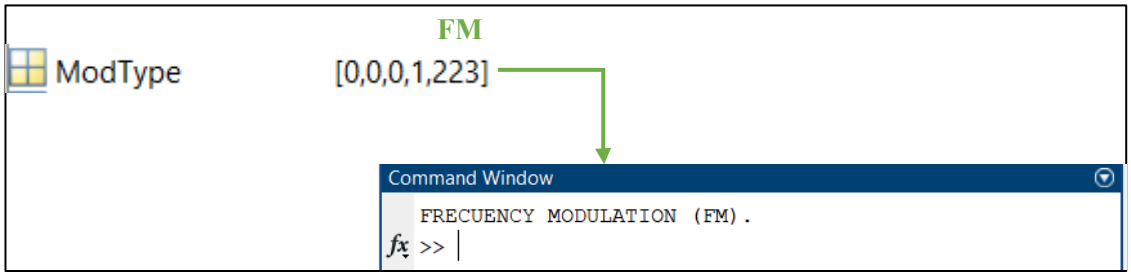


Figura 4.14 Resultado de clasificación de señal FM transmitida y recibida mediante SDR

Por último, se observa la clasificación de una señal FM.

Obteniéndose para múltiples pruebas unos resultados de clasificación similares, para concluir una tasa de acierto de aproximadamente un 96.44 % frente a una tasa de error del 3.56 % para señales moduladas en FM.

A continuación, se recogerá en una tabla el porcentaje total de aciertos y fallos para la clasificación de las señales moduladas transmitidas y recibidas mediante SDR.

Modulación	**Tasa de Acierto**	**Tasa de Error**
AM	0 %	100 %
DSB-SC	99.21 %	0.79 %
LSB	88.15 %	11.85 %
USB	94.9 %	5.1 %
FM	96.44 %	3.56 %
Total	**75.74 %**	**24.26 %**

Tabla 4.2 Tasa total de aciertos y fallos para señales transmitidas y recibidas mediante SDR

La tasa de acierto, ha sido considerada para cada modulación como el número medio obtenido en múltiples pruebas del total de fragmentos clasificados correctamente entre el número total de fragmentos clasificados multiplicado por 100, es decir, no se han considerado los segmentos de ruido ya que no han sido clasificados como modulación.

Obteniendo la tasa de error como 100 % - (% Tasa de Acierto en cuestión).

4.3 Conclusiones

Este apartado, quedará dividido en dos sub apartados, ya que, el trabajo realizado ha dado lugar a dos tipos de conclusiones, por un lado, se detallarán las conclusiones profesionales o científicas, por otro lado, las conclusiones a nivel personal o académico en general.

4.3.1 Conclusión científica

Una vez ha sido alcanzado el final del recorrido propuesto en este estudio, se podría concluir que las diferentes soluciones obtenidas han resultado en su mayor parte satisfactorias, y es que, a pesar de los problemas que han surgido, se han planteado e incorporado soluciones realmente efectivas como la incorporación de algoritmos optimizadores permitiendo reducir el tiempo de entrenamiento de la ANN o el calibramiento de los SDRs permitiendo una correcta clasificación de cuatro tipo de modulaciones.

Por otro lado, se debe recordar que la incorporación conjunta de equipos de radios definido por software y de clasificadores automáticos de modulaciones, no solo abaratarían en gran parte el coste de los sistemas de comunicaciones tradicionales, sino que también, permitirían incluir un gran dinamismo mediante la gran reprogramación que ofrecen estos sistemas, pudiendo ser útiles incluso con futuras invenciones de modulación de una señal, que, aunque es imposible predecir las tecnologías que existirán dentro de 10 o 20 años, se puede observar una tendencia similar en todos los tipos de modulación existentes.

4.3.2 Conclusión académica

El desarrollo e implementación de las diferentes soluciones ha requerido un aprendizaje y compresión adecuados para la documentación del problema. La proposición de las posibles soluciones, así como su implementación práctica y sus resultados de evaluación, permitirán que este aprendizaje contribuya de forma positiva a cualquier trabajo futuro.

Este trabajo ha permitido adquirir perspectiva a la hora de enfrentarse a un problema, tratando de relacionar y plasmar conceptos basados en problemas de disciplinas desconocidas (no estudiadas previamente), al campo de especialización propio, para, posteriormente, utilizar técnicas previamente adquiridas para solucionar un problema.

Se puede concluir en que este trabajo ha aportado muchas habilidades nuevas tanto a nivel personal como profesional, contribuyendo de forma positiva en todos los aspectos de forma general.

4.4 Líneas futuras

Aunque el objetivo principal del proyecto ha sido alcanzado, es decir, el desarrollo e implementación de un AMC, en este apartado se presentan una serie de mejoras o funcionalidades adicionales que pueden ser objeto de futuras líneas de trabajo.

Un AMC permite un amplio abanico de posibilidades en cuanto a su desarrollo y funcionamiento, a continuación, se muestran algunas posibilidades de desarrollo y funcionamiento diferentes a los usados en el desarrollo del AMC del proyecto.

- Posibilidad de clasificar modulaciones digitales.
  La principal línea futura del trabajo, sería expandir el tipo de modulaciones a clasificar, de esta manera se obtendría una mayor probabilidad de clasificar una señal entrante al receptor, siendo para este tipo de modulaciones 5 las características clave a calcular, donde 3 de ellas son usadas también para modulaciones analógicas.
- Uso de algoritmos no supervisados.
  Otra línea de futuro sería el uso de algoritmos no supervisados para el desarrollo del AMC, ya que, sería interesante observar resultados de clasificación cuando el conjunto de entrenamiento no es etiquetado.
- Despliegue del AMC.
  El despliegue del modelo entrenado junto con el cálculo de las características clave en sistemas integrados o dispositivos FPGA.

5 ANEXOS

Este apartado implementado de forma complementaria a los capítulos de la memoria, contendrá información adicional o accesoria que, aunque no resulte imprescindible para la compresión del proyecto, pueden resultar útiles para determinados lectores.

5.1 Anexo I: Instalación de librerías, controladores y soportes para el bladeRF

PROCEDIMIENTO DE INSTALACIÓN

En primer lugar, aclarar que el siguiente procedimiento de instalación ha sido realizado para el sistema operativo de Windows.

Descarga

Se podrá optar por descargar la última versión del instalador localizada en:

https://nuand.com/windows_installers/bladeRF-win-installer-latest.exe

No obstante, si se desea, se puede instalar una versión previa, disponibles estas en:

https://www.nuand.com/win_installers/

Instalación del ejecutable o .exe

Es importante antes de iniciar la instalación asegurarse de que el bladeRF NO está conectado al sistema, dicha conexión se podrá realizar una vez la instalación haya sido finalizada.

Dicho esto, se comienza ejecutando el ejecutable del instalador, una vez ejecutada, se presentará una pantalla de bienvenida como se muestra a continuación.

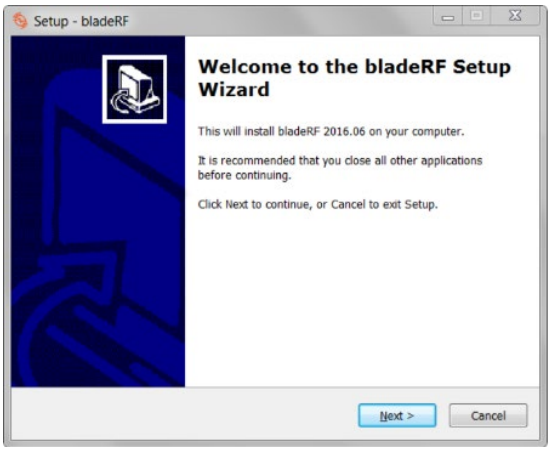


Figura 5.1 BladeRF Setup Wizard_1

Click en *next* para continuar.

Ubicación de destino

A continuación, se indica el destino de la instalación, se puede actualizar este campo si se desea.

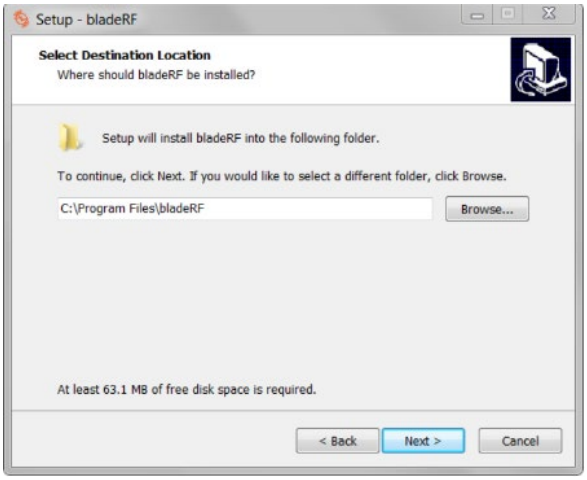


Figura 5.2 BladeRF Setup Wizard_2

Instalación de los Drivers

Esta pantalla presenta tres tipos de instalación de drivers. En caso de que sea la primera vez instalando el Software del BladeRF en el ordenador, el driver debe ser instalado, en otro caso, se podría omitir dicha instalación de drivers usando la última opción.

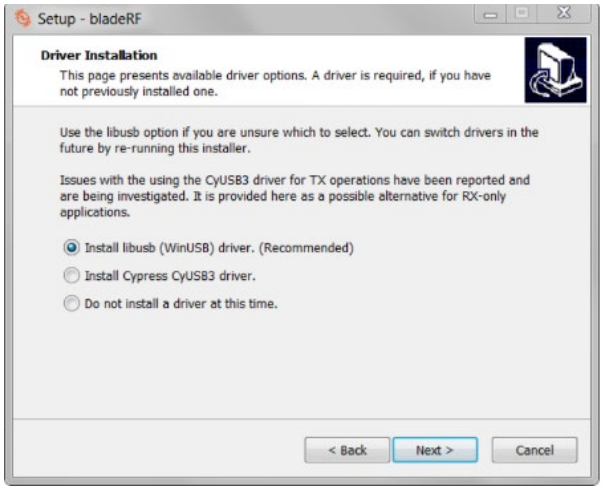


Figura 5.3 BladeRF Setup Wizard_3

Actualización del Firmware

La siguiente pantalla ofrece la opción de actualizar el firmware del bladeRF durante el proceso de instalación, esto es generalmente recomendado, ya que, las diferentes entregas de nuevas versiones incluyen la actualización de características y nuevos arreglos sobre problemas de versiones previas.

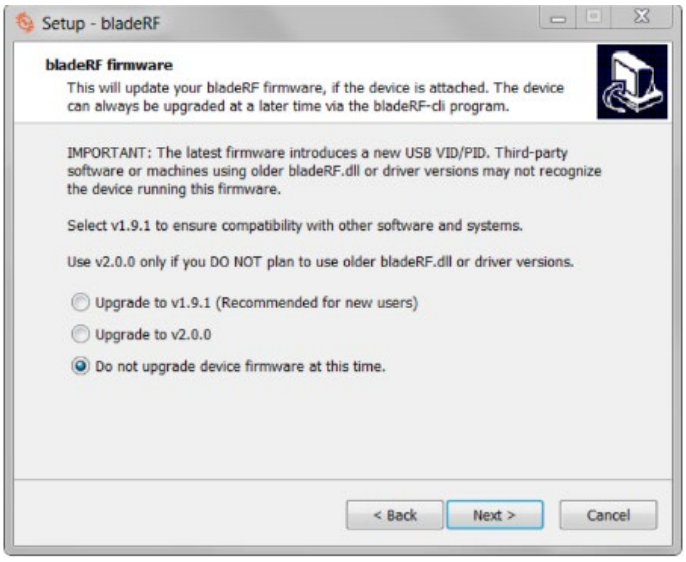


Figura 5.4 BladeRF Setup Wizard_4

Carpeta del menú de Inicio y Listo para Instalar

A continuación, dos nuevas pantallas serán mostradas:

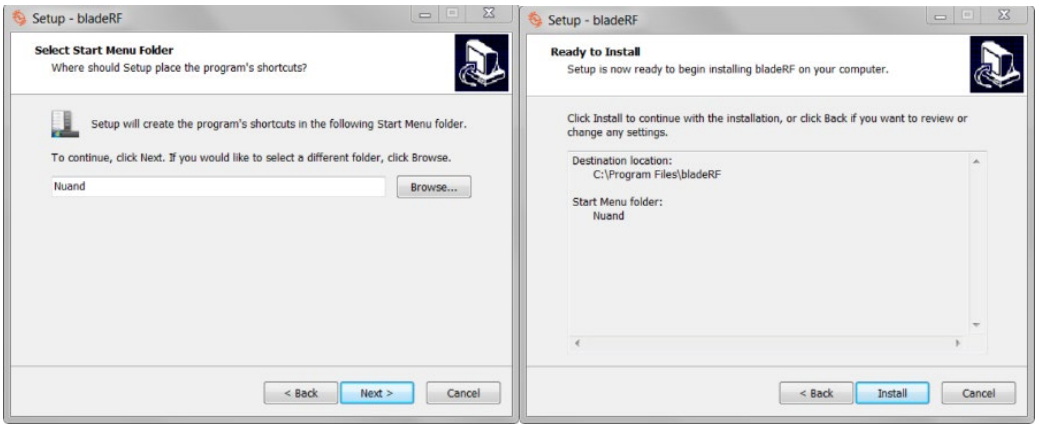


Figura 5.5 BladeRF Setup Wizard_5_6

La primera de ellas ofrece la posibilidad de customizar la localización de la carpeta en el menú de inicio generada por la posterior instalación del software.

El click en el botón *Install* encontrado en la segunda pantalla, comenzará la instalación de archivos en el sistema, siendo esta la última pantalla la cual permite volver hacia atrás y realizar cambios.

Progreso de Instalación

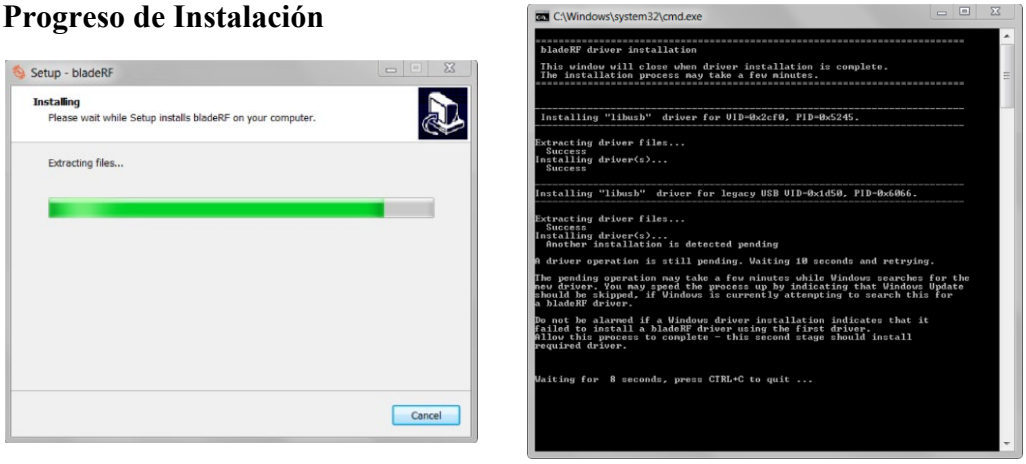


Figura 5.6 BladeRF Setup Wizard_7

La instalación de los drivers y de las librerías necesarias para el desarrollo del SDR, será el resultado de un progreso en la instalación del 100% en el caso de que anteriormente se ha ya seleccionado la opción de no actualizar el firmware.

Si anteriormente se marcó la opción de actualizar el firmware, el progreso en la instalación será del 80% cuando una nueva ventana se abra, mostrando el progreso de actualización del firmware.

En este caso, la actualización del firmware requiere una conexión entre el bladeRF y el sistema, por tanto, una vez se haya asegurado una correcta conexión entre ambos equipos se procederá a la actualización del firmware presionando cualquier tecla.

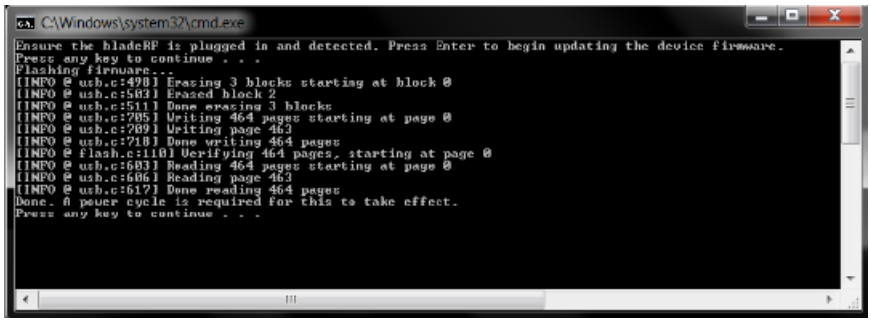


Figura 5.7 BladeRF Setup Wizard_8

Una vez se haya completado el progreso de la actualización del firmware, se pasará al siguiente nuevo proceso.

System PATH

En el final de la instalación, la siguiente ventana será presentada.

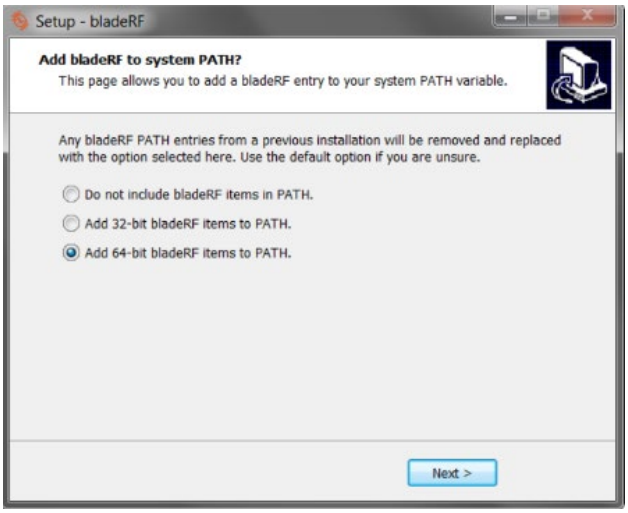


Figura 5.8 BladeRF Setup Wizard_9

Agregar los elementos de bladeRF al PATH de Windows permitirá que el software bladeRF-cli se ejecute desde cmd.exe y otros programas para la localización del bladeRF.dll, es decir, es muy importante marcar la segunda o tercera opción dependiendo del tipo de procesador en cuestión, ya que, la marcación de la primera opción no permitirá que otros softwares que no sean el bladeRF-cli puedan desarrollar el bladeRF.

POSIBLES PROBLEMAS y PROCEDIMIENTO DE TESTEO

Este procedimiento es fundamental antes de comenzar con el desarrollo del SDR, ya que, se pueden presentar dos problemas con el mismo, siendo estos presentados a continuación con sus respectivas soluciones.

El primer problema podría deberse a que el sistema no reconoce al SDR una vez han sido instalados en teoría todos los módulos necesarios para el funcionamiento del mismo, es decir, una vez completada la instalación y, abriendo el software bladeRF-CLI podría aparecer el siguiente aviso:

Figura 5.9 BladeRF Setup Problem_1

Esto está relacionado únicamente al proceso de instalación detallado anteriormente, es decir, si esto ocurre, desinstale por completo todos los archivos generados con la

instalación, posteriormente vuelva a ejecutar el instalador y siga las fases comentadas anteriormente en el proceso de instalación.

El segundo problema es algo más tedioso y es el problema que me ocurrió en cuestión. Se podría dar el caso de que el sistema reconozca perfectamente el dispositivo SDR usado en cuestión, no solo porque ya no aparece el aviso mostrado en el problema 1, sino porque se puede obtener información acerca del dispositivo, usando una serie de comandos básicos como '*version'*, '*info'*, '*print*', es decir, se puede realizar un testeo básico.

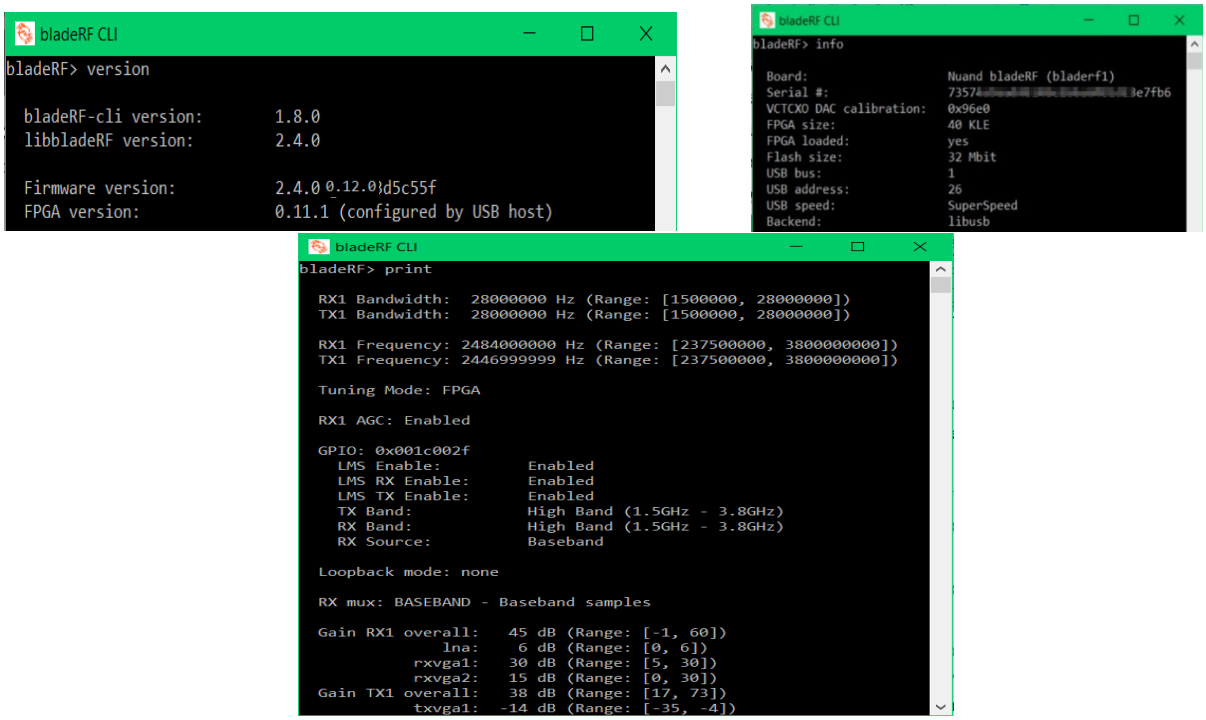


Figura 5.10 Testeo Básico del SDR

Aparentemente se podría pensar que todo está funcionando correctamente, pero, realmente, para comprobar que todo funciona correctamente, se debe de realizar un testeo más profundo, consistente en no solo realizar un testeo básico como el mostrado previamente, sino, también realizar una prueba de transmisión y de recepción, ya que, es aquí donde se presenta este segundo problema, y es que, se podría dar el caso de que aunque el testeo básico ha sido realizado con éxito, al realizar pruebas en la transmisión y recepción de señales se genera un error que muestra problemas con la FPGA del dispositivo '***FPGA not LOADED***', esto quiere decir que la FPGA del dispositivo no ha sido programada. Sin embargo, en la figura que muestra la información ofrecida por el comando '*version*', se puede observar que, si hay una versión de la FPGA.

En base a numerosas búsquedas en foros, se dio con el motivo y la solución del problema, y es que, las versiones de FPGA iguales o posteriores a la versión 0.12.0 presentan

problemas de compatibilidad con las versiones existentes del firmware y libbladeRF, la solución será la siguiente:

1. Acceda al siguiente enlace y descargue una versión de la FPGA anterior a la versión 0.12.0, estos archivos son llamados 'FPGA bitstream', usando una extensión .rbf, la selección del archivo dependerá del modelo de su bladeRF en cuestión, es decir, si usa un modelo de bladeRF x40, seleccione el archivo '*hostedx40.rbf*'.
2. Una vez descargado el archivo, cópielo y péguelo en la carpeta donde previamente ha instalado todo lo correspondiente al bladeRF.

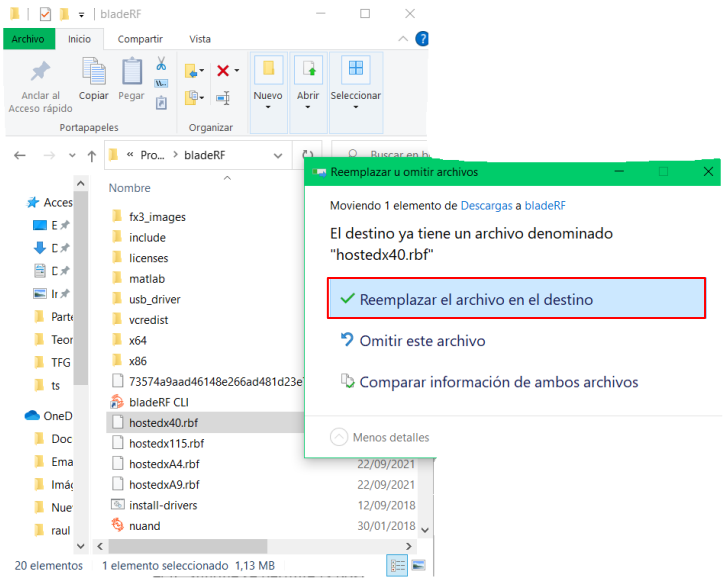


Figura 5.11 BladeRF Setup Solution2Problem_2_1

3. Por último, se procede a la instalación manual del archivo mediante el bladeRF-CLI, para ello simplemente escriba lo siguiente en la línea de comandos:

'**load fpga hosted**(modelo del bladeRF en cuestión)**.rbf**'

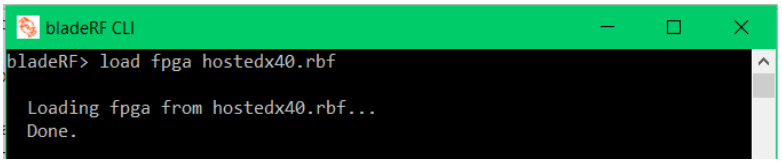


Figura 5.12 BladeRF Setup Solution2Problem_2_2

Detalle a tener en cuenta con el uso del software para el desarrollo del SDR

Como único detalle se añade la imposibilidad de desarrollar un mismo SDR de forma simultánea mediante más de un software diferente, es decir, aunque se permite la posibilidad de transmitir y recibir señales mediante dos canales independientes, ambos deben ser desarrollados con el mismo software.

5.2 Anexo II: Esquemas implementados en GNU radio

En el siguiente sub apartado se mostrarán los diferentes esquemas realizados en GNU radio para la transmisión de los cinco tipos de modulación empleadas en este proyecto, no obstante, y, aunque la demodulación de las señales moduladas no forma parte de los objetivos del proyecto, también han sido realizadas para cada tipo de modulación, la realización de los bloques demoduladores ha sido implementada a forma de chequeo de que la transmisión en una determinada modulación es correcta.

5.2.1 Esquemas implementados en transmisión

AM

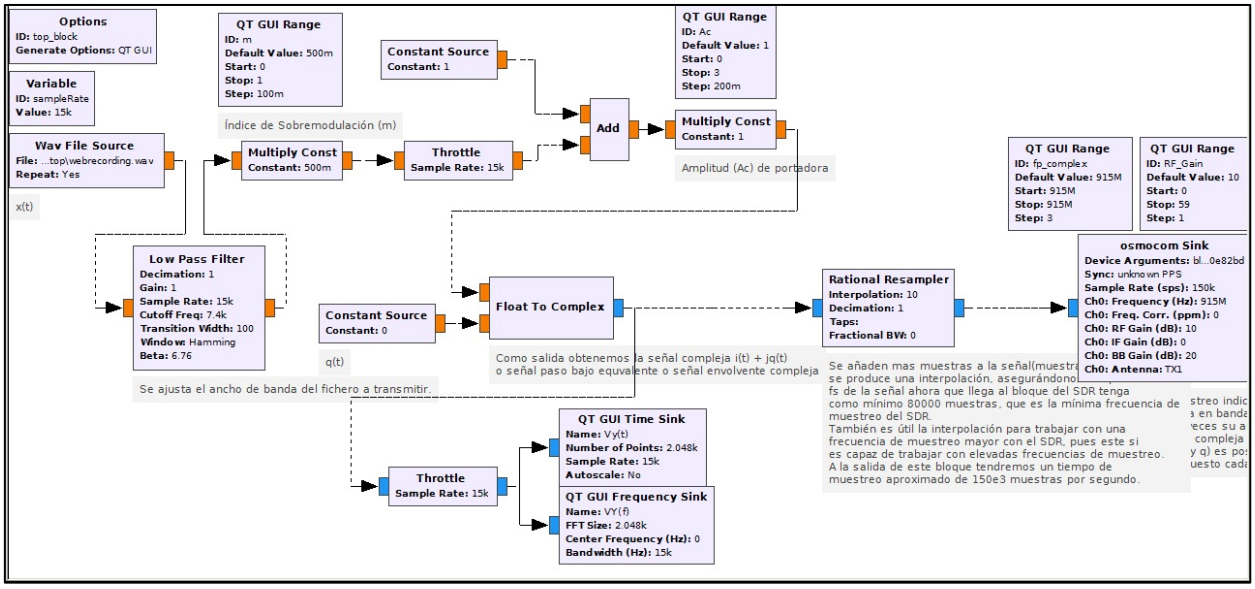


Figura 5.13 Esquema transmisor AM

A continuación, se explica el comportamiento de cada bloque utilizado, así como el proceso que seguirá la señal moduladora o x(t) hasta ser transmitida, siendo esta explicación válida para el resto de modulaciones y demodulaciones, únicamente en el resto de modulaciones o demodulaciones se comentarán los bloques que no se hayan comentado aquí.

En primer lugar, cabe destacar que el color de los bloques mostrados, índica el tipo de entrada o salida requerida u ofrecida por un bloque, es decir, en este caso, los bloques cuyo color de entrada o salida sea naranja, indica un tipo de dato flotante, por el contrario, los bloques cuyo color de entrada o salida sea azul, indica un tipo de dato complejo.

- Wav File Source: Este bloque es usado para leer una señal de audio con extensión .wav almacenada en nuestro ordenador en este caso, se corresponde con la señal moduladora o x(t) con frecuencia de muestreo de 11.000 Hz, como detalle, el archivo de audio a seleccionar no debe estar almacenado en

un directorio que contenga tilde, por el contrario, el archivo no podrá ser leído por dicho bloque.

- Low Pass Filter: Simula el efecto de un filtro paso bajo, este bloque es usado para contener la información de la señal en un ancho de banda exacto y conocido, en este caso, la frecuencia de corte del mismo ha sido elegida a ser 7.400 Hz, para, de esta forma cumplir el teorema de Nyquist del muestreo sabiendo que la frecuencia de muestreo del fichero de entrada es de 15.000 Hz.
- Constant Source, Multiply Const, Add: Estos bloques realizan de forma respectiva la generación de una señal constante con valor 1, la multiplicación de la señal de entrada filtrada por un valor de m, el cual varía entre 0 y 1, para, posteriormente sumar mediante el bloque Add, ambas señales y multiplicarlas nuevamente por un valor A_c, obteniendo la componente en fase correspondiente a AM, posteriormente se genera una nueva señal con valor 0, que representa la componente en cuadratura.
- Float To Complex: Este bloque recibe por un lado la parte real de una señal, por otro lado, recibe la parte imaginaria de otra señal, y, a su salida, ofrece la suma compleja de ambas señales.
- Throttle: Este bloque normalmente es usado para la posterior representación de señales, permitiendo que la tasa promedio de muestras por segundo no exceda la tasa especificada.
- Bloques QT GUI (Time, Frecuency) Sink: Estos bloques permiten respectivamente la visualización en tiempo real de una señal en el dominio del tiempo y de la frecuencia (FFT).
- Rational Resampler: Permite el aumento o disminución de la frecuencia de muestreo de la señal que recibe mediante la combinación de un interpolador seguido de un diezmador, permitiendo por tanto el cambio de frecuencia de muestro de la señal por un número racional. En este caso, la señal de entrada cuya frecuencia de muestro era de 15.000 Hz, se le aplica una interpolación de N = 10, por tanto, a la salida de este bloque, se obtiene una señal cuya frecuencia de muestro es igual a 150.000 Hz.
- Osmocom Sink: Este bloque es el encargado de realizar la conexión entre GNU radio y el SDR en cuestión, permitiendo desarrollar el canal de

transmisión del SDR mediante la modificación de los valores del mismo, a la entrada del bloque se tiene la señal compleja o señal paso banda equivalente conformada por sus componentes IQ la cual será modulada y transmitida por el SDR, como detalle se añade que los SDRs deben recibir señales en fase y cuadratura con valores de amplitud entre [-1,1], de no ser así, se producirá saturación.

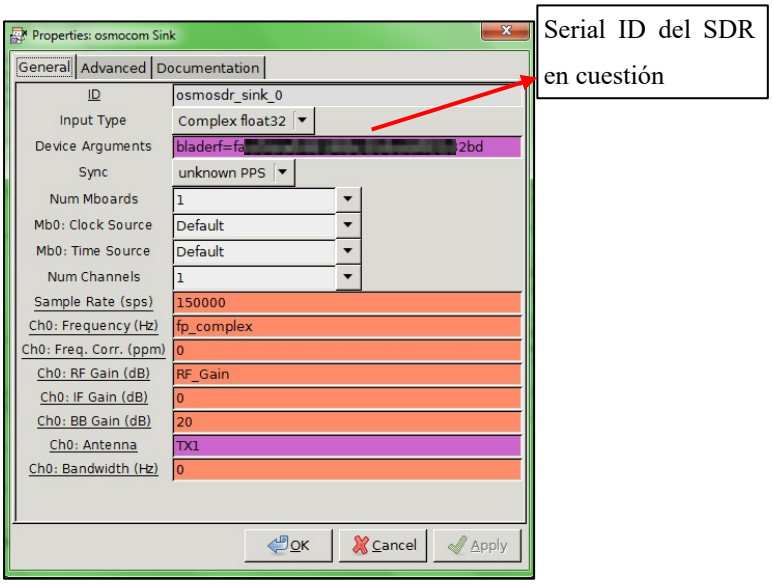


Figura 5.14 Propiedades del bloque SDR transmisor

El valor de 0 para el ancho de banda, indica una selección automática del mismo, siendo este valor el mínimo disponible, en este caso 1.5 MHz.

Cabe destacar que, a causa de la interpolación, es decir, a causa del aumento del número de muestras en la señal en el tiempo, en frecuencia se traduce en una disminución del ancho de banda, por lo que se va a producir el problema de que van a aparecer nuevas componentes espectrales que habrá que eliminar mediante un filtro paso bajo, aunque en este caso ningún filtro paso bajo será añadido, ya que, aparte de que dichas componentes aparecen con una potencia relativamente baja, el SDR, como se explicó al comienzo del proyecto filtra mediante un filtro paso bajo la señal recibida.

DSB-SC

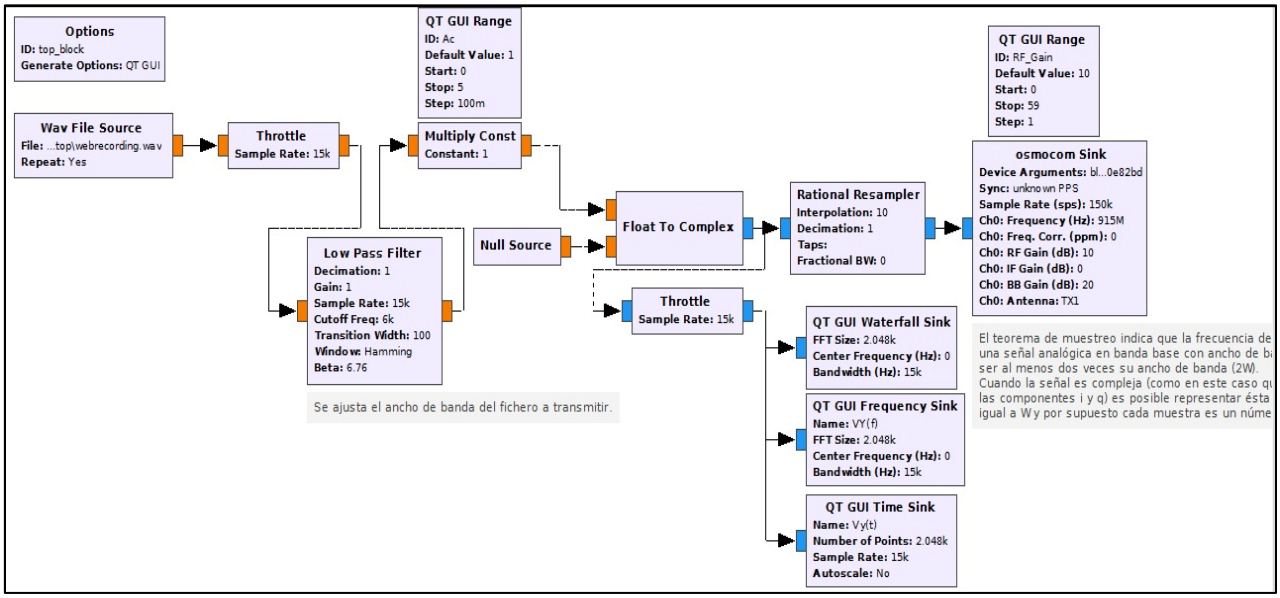


Figura 5.15 Esquema transmisor DSB-SC

LSB

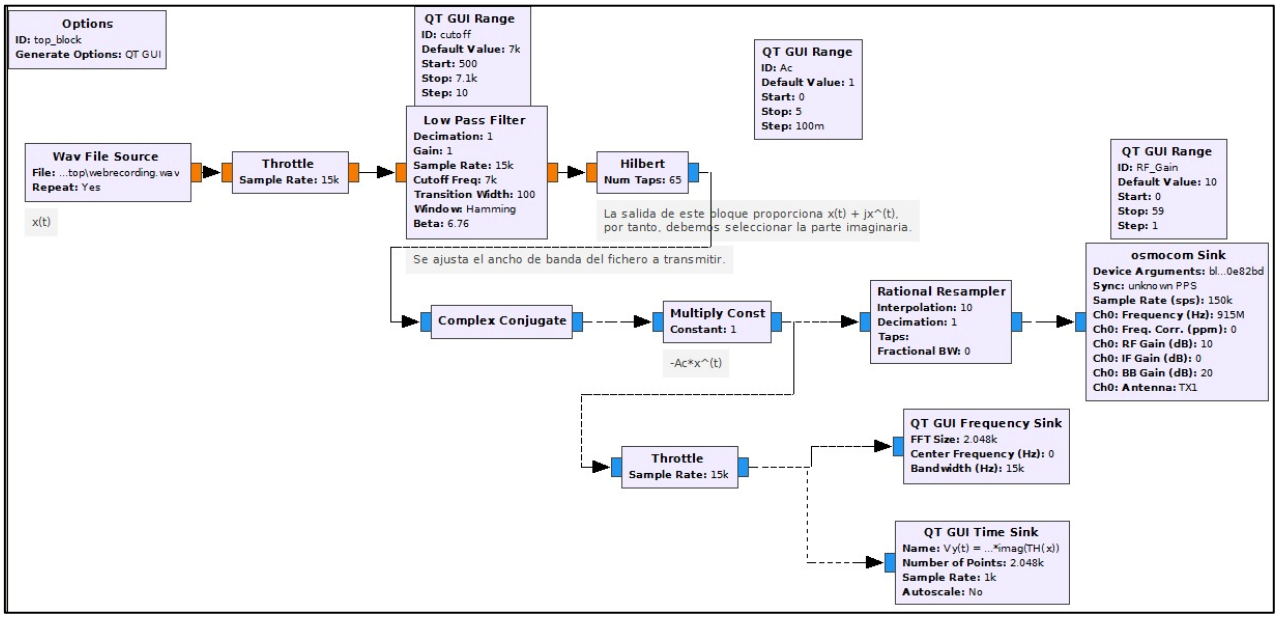


Figura 5.16 Esquema transmisor LSB

- Hilbert: Este bloque realiza la transformada de Hilbert de la señal de entrada y la suma a la misma señal de entrada sin transformar, es decir la salida de este bloque ofrece $x_{filtered}(t) + j\hat{x}(t)$.
- Complex Conjugate: Este bloque realiza la operación de conjugar la señal compleja de entrada.

USB

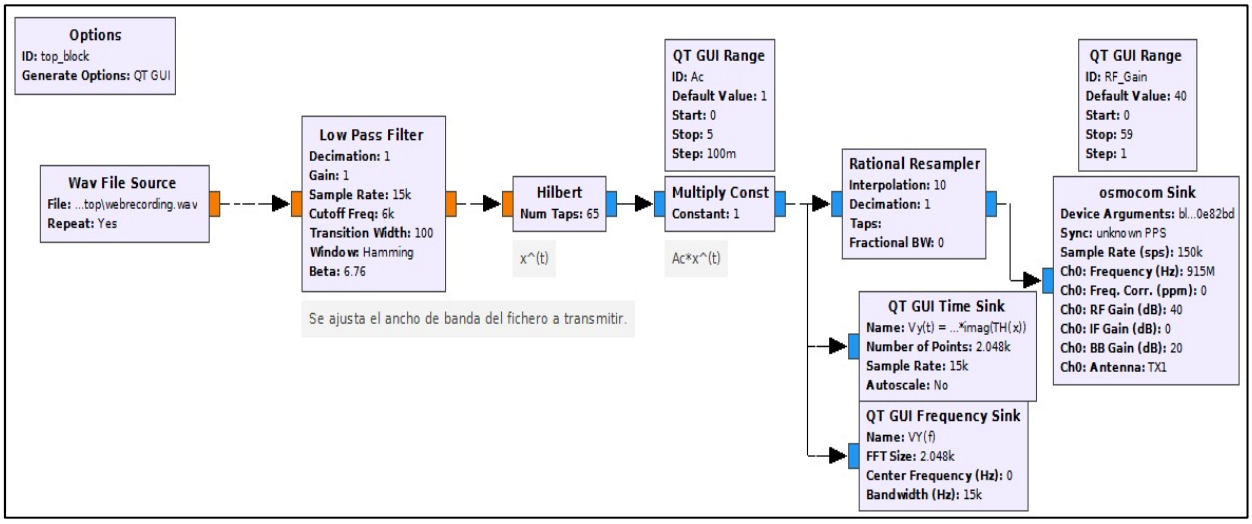


Figura 5.17 Esquema transmisor USB

FM

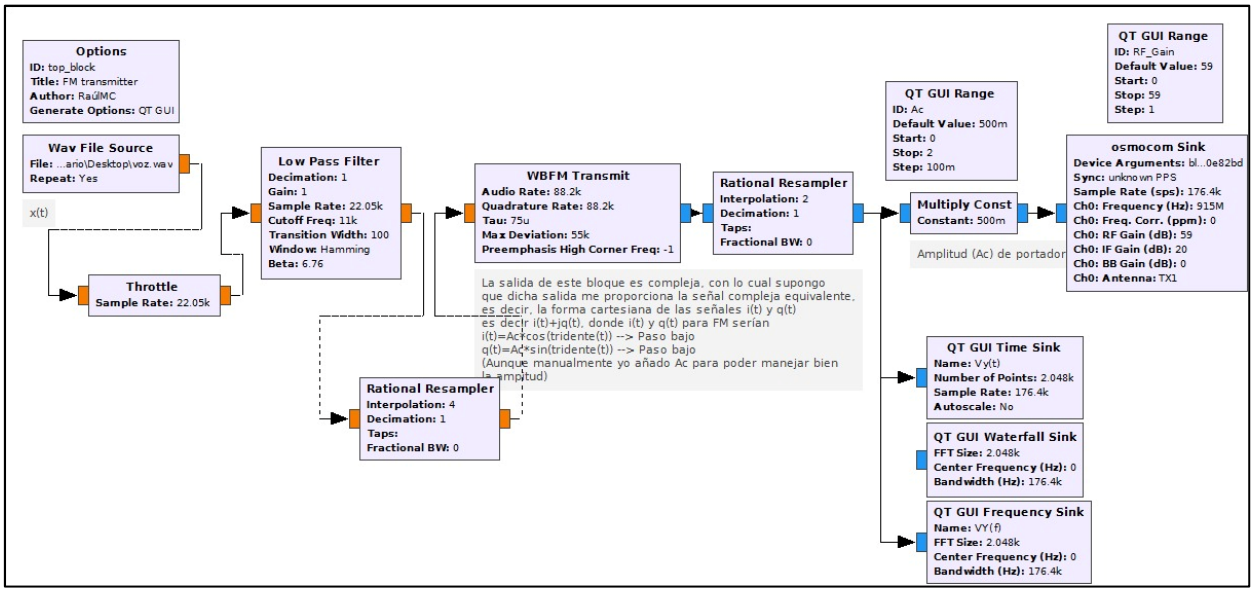


Figura 5.18 Esquema transmisor FM

- WBFM Transmit: Este bloque va a permitir generar directamente las componentes en fase y cuadratura correspondientes a una señal FM de banda ancha, es decir, con un valor del parámetro β elevado.

5.2.2 Esquemas implementados en recepción

AM

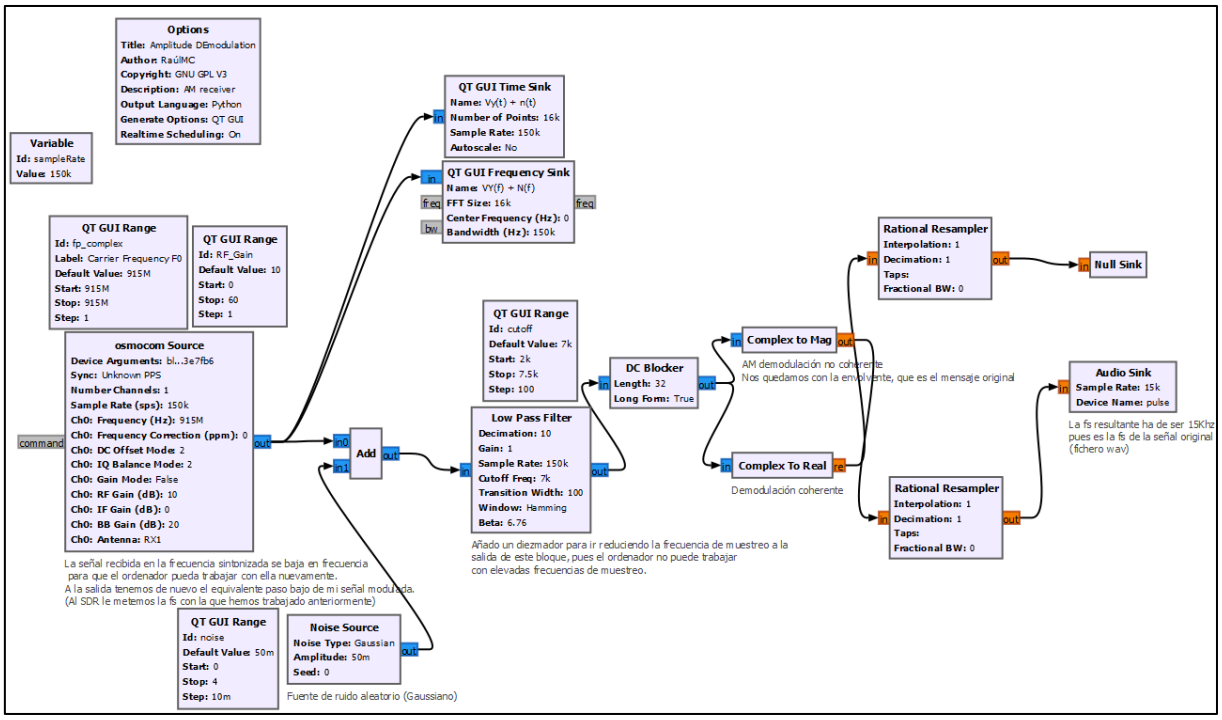


Figura 5.19 Esquema receptor AM

- Osmocom Source: Este bloque permitirá desarrollar el canal de recepción del SDR en cuestión, realiza el enlace entre la señal RF analógica recibida y la señal banda base compleja para su procesado en GNU radio.

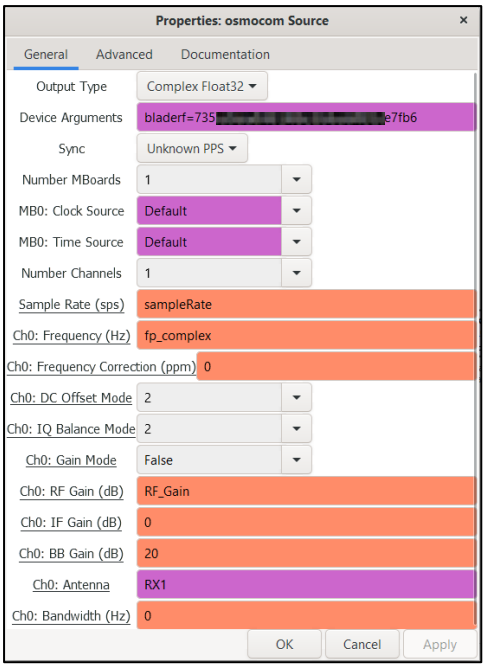


Figura 5.20 Propiedades del bloque SDR receptor

- Noise Source: Este bloque añade ruido blanco y Gaussiano paso bajo.

- Low Pass Filter: Este bloque simula el filtro paso banda que normalmente se tiene en recepción, en este caso al recibir las componentes paso bajo se pone este filtro paso bajo.
- DC Blocker: Usado para eliminar la componente de continua de la señal AM.
- Demoduladores (Complex to Mag, Complex to Real): El primer tipo de demodulador, es un demodulador no coherente, extrae la envolvente de la señal recibida que es la señal moduladora, en términos matemáticos este bloque obtiene el módulo de lo que recibe, el segundo tipo de demodulador es un demodulador coherente que se queda con la parte real de la señal compleja recibida, este tipo de demodulación permite obtener la señal original con menos ruido que la demodulación no coherente.

DSB-SC

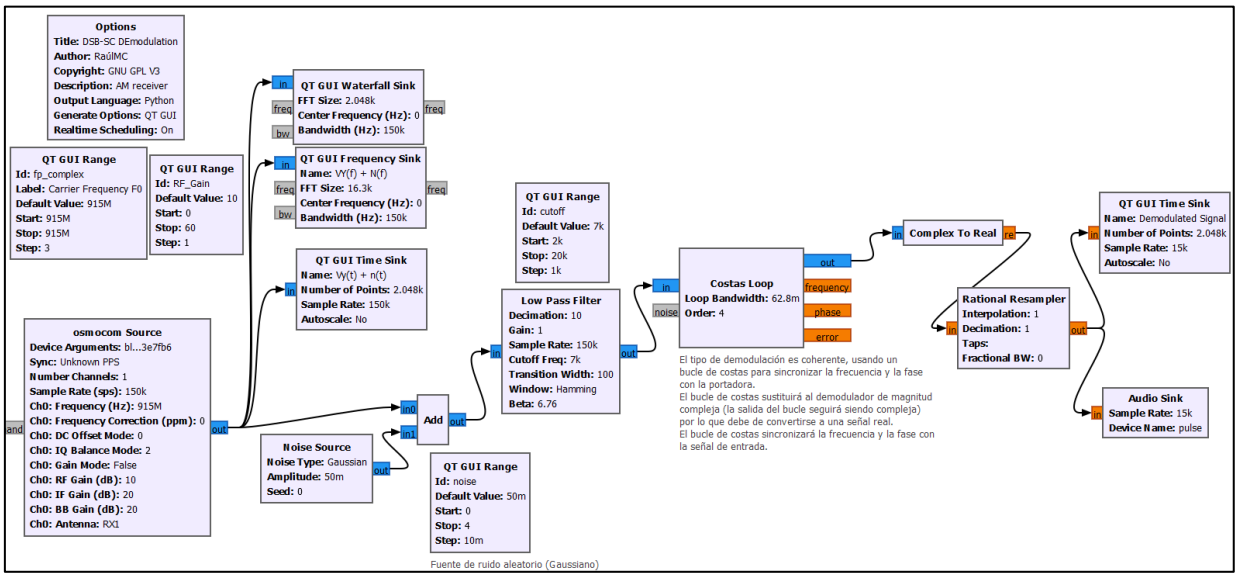


Figura 5.21 Esquema receptor DSB-SC

- Costas Loop: Este bloque junto con el bloque Complex To Real, permitirá la demodulación coherente de la señal DSB-SC sincronizando la frecuencia y la fase con la portadora, que, aunque realmente la señal que se recibe es bajada en frecuencia, este esquema puede ser útil a la hora de trabajar con receptores SDR baratos que no realicen la demodulación en fase y cuadratura.

LSB

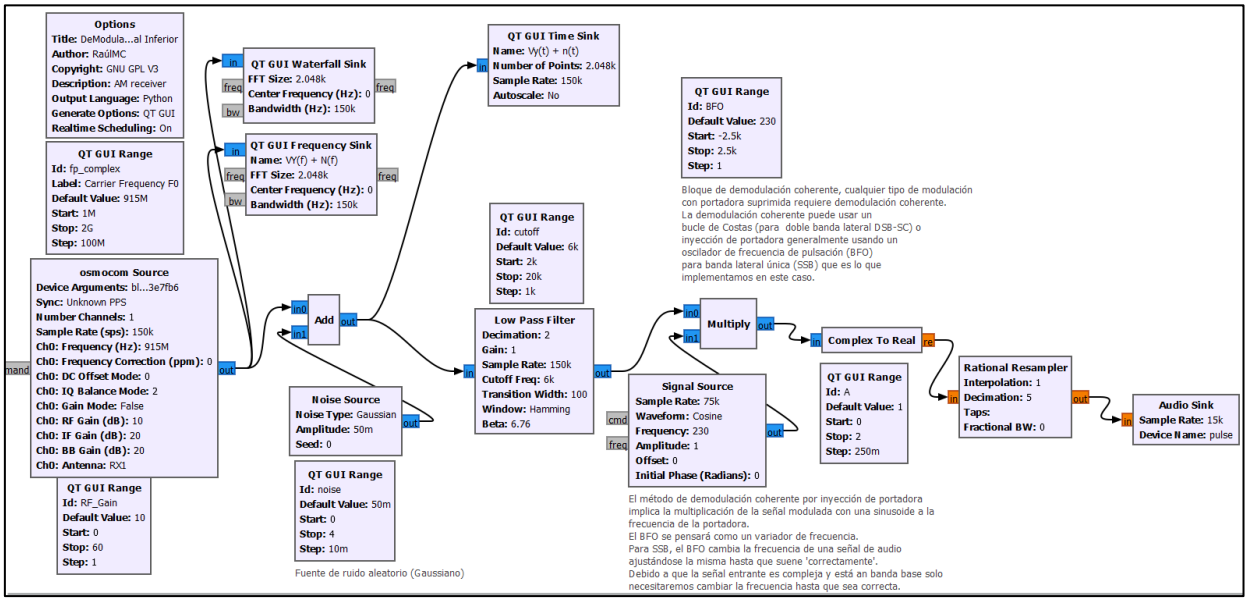


Figura 5.22 Esquema receptor LSB

USB

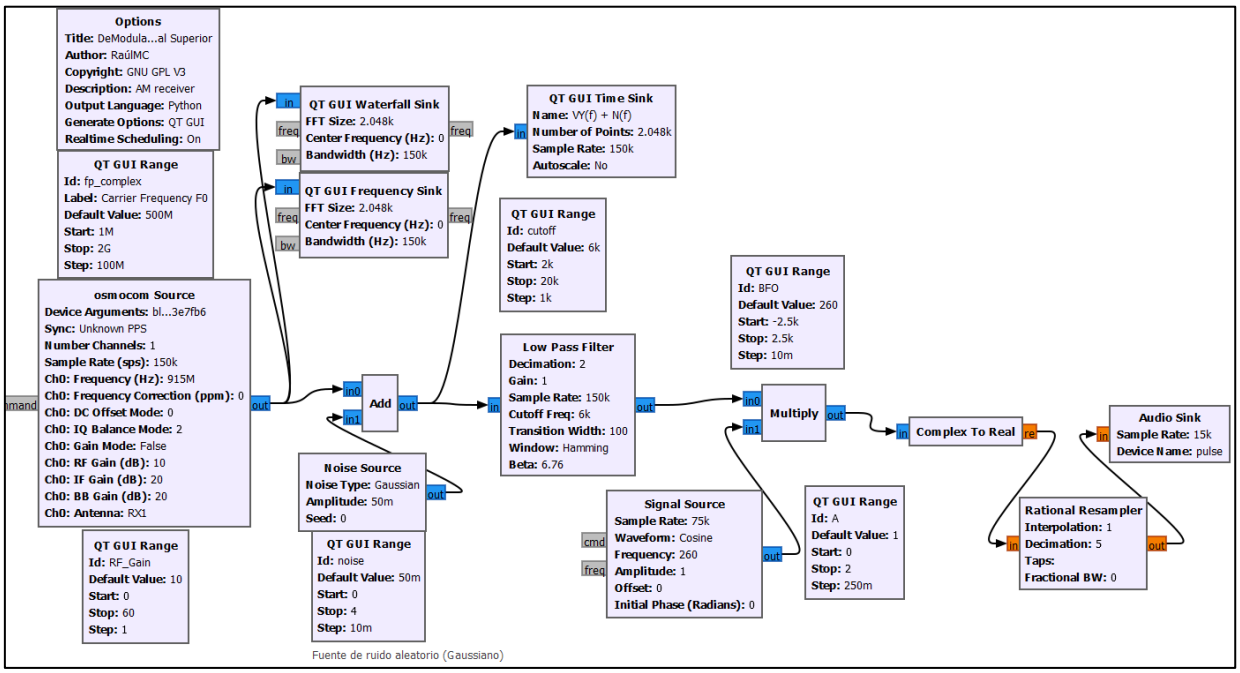


Figura 5.23 Esquema receptor USB

FM

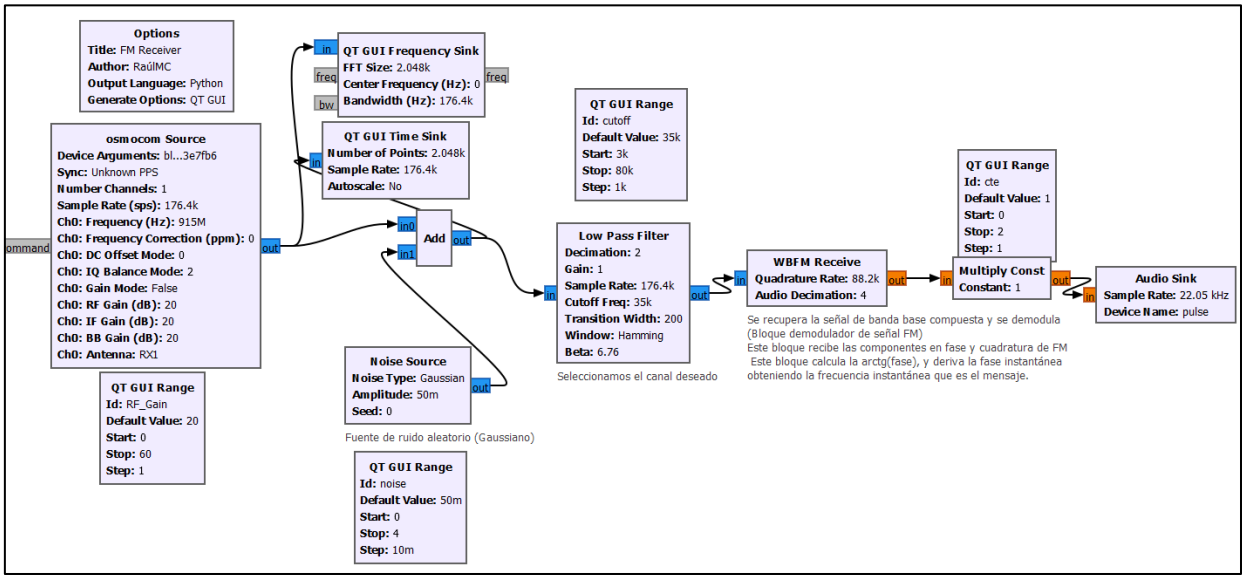


Figura 5.24 Esquema receptor FM

- WBFM Receive: Al igual que el bloque WBFM Transmit realizaba la modulación en FM, este bloque realizará la correspondiente demodulación de la señal FM.

5.3 Anexo III: Scripts y Funciones desarrolladas en Matlab

Debido a la gran cantidad de código desarrollado, en lugar de mostrarlo, se procederá a realizar primeramente unos diagramas de flujo que representen las relaciones entre todas y cada una de las funciones y scripts que han sido desarrollados, para, posteriormente, comentar en detalle el funcionamiento de cada script o función contemplada en dicho diagrama.

El número de diagramas que se mostrarán serán dos, el primero de ellos mostrará el procedimiento completo llevado a cabo para la clasificación de señales que no van sobre el aire. Una vez mostrado el primer diagrama se procederá a mostrar un segundo diagrama en el que se presente el procedimiento completo llevado a cabo para la clasificación de las señales que van sobre el aire.

Lógicamente muchas de las funciones son compartidas por ambos diagramas.

Posteriormente y, una vez mostrados ambos diagramas, se procederá a comentar el funcionamiento de cada función o script como se comentó anteriormente.

5.3.1 Diagrama de flujo del procedimiento completo llevado a cabo para la clasificación de señales que no van sobre el aire

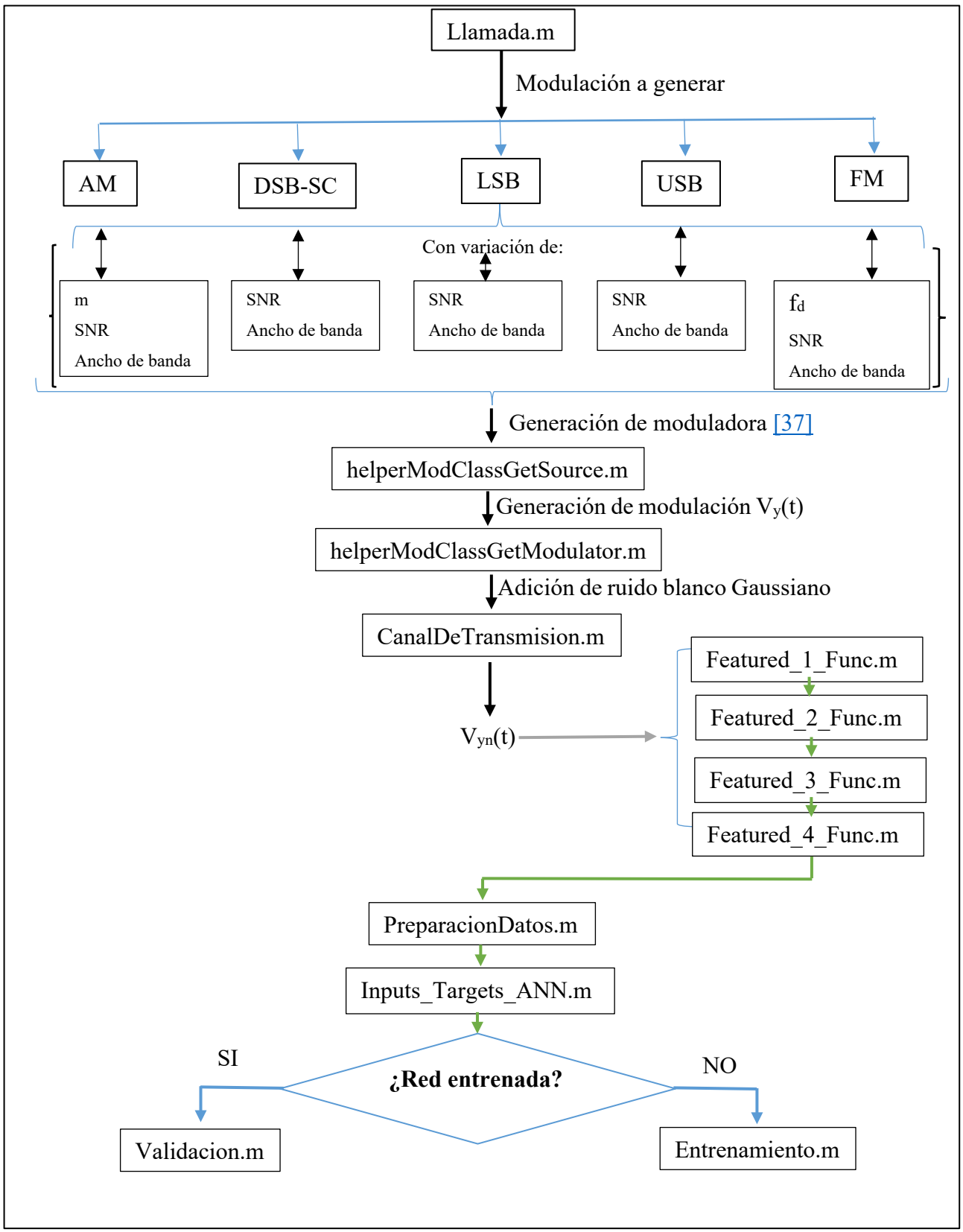


Figura 5.25 Diagrama de flujo sobre el procedimiento llevado a cabo sobre señales que no van sobre el aire

5.3.2 Diagrama de flujo del procedimiento completo llevado a cabo para la clasificación de señales recibidas por el SDR

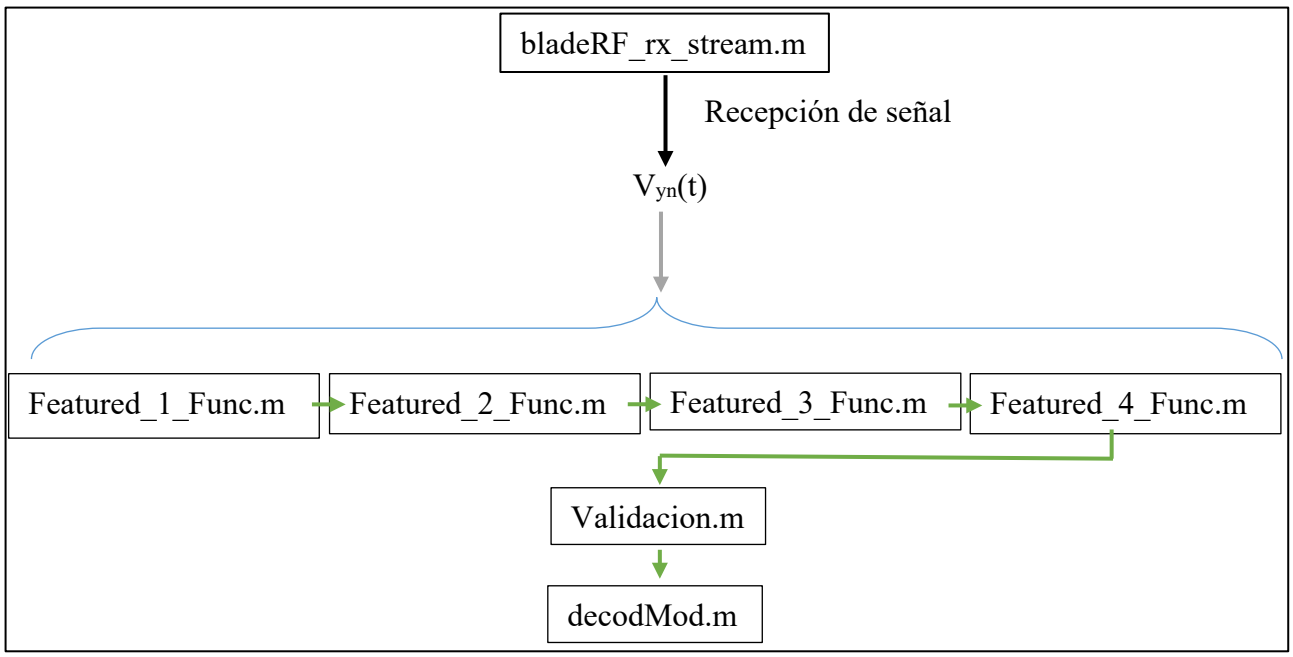


Figura 5.26 Diagrama de flujo sobre el procedimiento llevado a cabo sobre señales que van sobre el aire

5.3.3 Descripción de los Scripts y funciones desarrolladas en Matlab

La carpeta que contiene las funciones y scripts desarrollados en Matlab contiene un fichero Contents.m cuyo contenido se muestra a continuación.

```
% Llamada: General
%   helperModClassGetSource     - Generación de mensaje
%   helperModClassGetModulator  - Genera modulación
%   CanalDeTransmision          - Adición de AWGN
%   Featured_1_Func             - Cálculo de γmax
%   Featured_2_Func             - Cálculo de σap
%   Featured_3_Func             - Cálculo de σdp
%   Featured_4_Func             - Cálculo de P
%   PreparacionDatos            - Genera matriz de datos
%   Inputs_Targets_ANN          - Genera entrada-salida
%   Entrenamiento               - Entrenamiento ANN
%   Validacion                  - Validación de ANN
```

```
% bladeRF_rx_stream: General
%   Featured_1_Func              - Cálculo de γmax
%   Featured_2_Func              - Cálculo de σap
%   Featured_3_Func              - Cálculo de σdp
%   Featured_4_Func              - Cálculo de P
%   Validacion                   - Validación de ANN
%   decodMod                     - Genera cadena de texto
```

```
%Llamada.
%   El script Llamada o script principal será
%   responsable de la llamada a todas las funciones
%   necesarias para la generación y clasificación de
%   modulaciones, permitiendo entre otras cosas la
%   llamada a la función que genera las modulaciones
%   mediante la variación de los parámetros
%   correspondientes.
%   Además, permitirá la creación y almacenamiento en el
%   sistema de las señales moduladas, generando ficheros
%   .mat, cuyo directorio será
%    C:\Users\(Usuario)\AppData\Local\Temp\ModClassDataFiles
```

```
%helperModClassGetSource.
%   x = helperModClassGetSource(TYPE,SPF,FS)
%   HELPERMODCLASSGETSOURCE devuelve los
%    datos correspondientes a una señal moduladora de
%    audio para cualquier tipo TYPE de modulación
%    analógica, con un número de muestras por segmento
%    SPF, y una frecuencia de muestreo determinada FS en
%     Hz.
%
%   % Ejemplo 1:
%   %   Genere una moduladora para cualquier tipo de
%   %   modulación analógica con un número de muestras
%   %   por segmento de 2048 y una frecuencia de
%   %   muestreo de 200 kHz

%   x=helperModClassGetSource(AM,2048,200e3);
```

```
%helperModClassGetModulator.
%   Vy = helperModClassGetModulator(modType)
%   HELPERMODCLASSGETMODULATOR devuelve un
%    manejador de función de modulación Vy basado en
%    modType, que permitirá posteriormente modular una
%    señal en sus componentes en fase y cuadratura.
%    Siendo los mostrado a continuación los
%   identificadores de función.
%
%%Identificadores de función
% switch modType
%   case "AM"
%     Vy = @(x,m,Ac)AM_Transmitter(x,m,Ac);
%   case "DSB-SC"
%     Vy = @(x,Ac)DSBSC_Transmitter(x,Ac);

%   case "LSB"
%     Vy = @(x,Ac)LSB_Transmitter(x,Ac);
%   case "USB"
%     Vy = @(x,Ac)USB_Transmitter(x,Ac);
%   case "FM"
%     Vy = @(x,Ac,fs,fd)FM_Transmitter(x,Ac,fs,fd);
% end
%
%   % Ejemplo 1:
%   %   Genere una modulación AM con un valor
%   %   del parámetro m de 0.5 y una amplitud
%   %   de portadora con valor igual a 1.

%modulator=helperModClassGetModulator(AM);
%ModulatedSignal = modulator(x,0.5,1);
```

```
%CanalDeTransmision.
%   Vyn = CanalDeTransmision(Vy,SNR)
%   CANALDETRANSMISION añade ruido
%   AWGN paso bajo a las componentes en fase y
%   cuadratura de la señal modulada de entrada Vy.
%   Siendo SNR, el valor de la relación señal a ruido
%   en dB que tendrá la señal resultante a la
%   salida de esta función.
%
%   % Ejemplo 1:
%   %   Añada ruido blanco Gaussiano a una señal
%   %   modulada de tal forma que el SRN resultante
%   %   sea de 20 dB.
%   Vyn = CanalDeTransmision(Vy,20)
```

```
%Featured_1_Func.
%    [Ymax,An] = Featured_1_Func(Vyn)
%    FEATURED_1_FUNC devuelve el valor de la
%     característica Ymax y del parámetro An.
% Ymax
%    Valor máximo de la densidad de potencia
%    espectral de la amplitud instantánea
%    normalizada y centrada de Vyn.
% An
%    Amplitud instantánea de Vyn normalizada
%    respecto a la media de Vyn para cada
%    instante temporal. Este valor es necesario
%    para el cálculo de posteriores características.
```

```
%Featured_2_Func.
%    varAP = Featured_2_Func(Vyn)
%    FEATURED_2_FUNC realiza el cálculo de la
%    característica σap de la señal Vyn. Ofreciendo a
%    su salida el valor del mismo.
% σap
%    Es la desviación estándar del valor absoluto de la
%    componente no lineal de la fase instantánea después
%    de ser centrada evaluada sobre los segmentos no
%    débiles de señal Vyn.
```

```
%Featured_3_Func.
%    varDP = Featured_3_Func(Vyn)
%    FEATURED_3_FUNC realiza el cálculo de la
%     característica σdp de la señal Vyn.
%     ofreciendo a su salida el valor del mismo.
% σdp
%    Es la desviación estándar del valor directo
%    de la componente no lineal de la fase
%    instantánea después de ser centrada
%    evaluada sobre los segmentos no débiles
%    del segmento de señal Vyn.
```

```
%Featured_4_Func.
%    P = Featured_4_Func(Vyn)
%    FEATURED_4_FUNC realiza el cálculo de la
%     característica P de la señal Vyn
%     ofreciendo a su salida el valor del mismo.
% P
%    Informa sobre la simetría alrededor del
%    espectro de la frecuencia de portadora de
%    la señal Vyn.
```

```
%PreparacionDatos.
%   DATOS = PreparacionDatos(modTypeC)
%   PREPARACIONDATOS devuelve un
%   manejador de función DATOS basado en
%   modTypeC, que posteriormente permitirá
%   mediante su llamada, generar matrices de
%   datos de dimensiones (4 x Q) para cada tipo
%   de modulación analógica.
%
%   Cada una de las cuatro filas de la matriz,
%   ofrece el valor de una característica clave.

%   El número de columnas Q, depende del
%   número de segmentos de señal generados.
%%Identificadores de función
% switch modTypeC
%   case "AM_m"
%     DATOS = @(Ymax,varAP,varDP,P)AM_m(Ymax,varAP,varDP,P);
%   case "AM_BW"
%     DATOS = @(Ymax,varAP,varDP,P)AM_BW(Ymax,varAP,varDP,P);
%   case "AM_SNR"
%     DATOS = @(Ymax,varAP,varDP,P)AM_SNR(Ymax,varAP,varDP,P);
%   case "DSBSC_BW"
%     DATOS = @(Ymax,varAP,varDP,P)DSBSC_BW(Ymax,varAP,varDP,P);
%   case "DSBSC_SNR"
%     DATOS = @(Ymax,varAP,varDP,P)DSBSC_SNR(Ymax,varAP,varDP,P);
%   case "LSB_BW"
%     DATOS = @(Ymax,varAP,varDP,P)LSB_BW(Ymax,varAP,varDP,P);
%   case "LSB_SNR"
%     DATOS = @(Ymax,varAP,varDP,P)LSB_SNR(Ymax,varAP,varDP,P);
%   case "USB_BW"
%     DATOS = @(Ymax,varAP,varDP,P)USB_BW(Ymax,varAP,varDP,P);
%   case "USB_SNR"
%     DATOS = @(Ymax,varAP,varDP,P)USB_SNR(Ymax,varAP,varDP,P);
%   case "FM_BW"
%     DATOS = @(Ymax,varAP,varDP,P)FM_BW(Ymax,varAP,varDP,P);
%   case "FM_SNR"
%     DATOS = @(Ymax,varAP,varDP,P)FM_SNR(Ymax,varAP,varDP,P);
% end
%
%   % Ejemplo 1:
%   %    A partir de las siguientes 4 variables vectoriales
%   %   (Ymax, varAP, varDP, P), de dimensiones cada una(1 x Q),
%   %   correspondientes al cálculo de las características
%   %   de una señal modulada en AM donde ha sido variado
%   %   el parámetro m, formalice una matriz que contenga
```

```
%    %    en cada fila una de esas variables vectoriales.

%    DATOS = PreparacionDatos(AM_m);
%    MatrizAM_m = DATOS(Ymax,varAP,varDP,P); % Matriz de (4 x Q)
```

```
%Inputs_Targets_ANN.
%    [Inputs, Targets] = Inputs_Targets_ANN(MatrizsDatos)
%    INPUTS_TARGETS_ANN devuelve
%    dos matrices correspondientes a la
%    matriz de entradas de la red neuronal
%    y a la matriz de salidas de la red neuronal
% Inputs
%    Matriz de dimensiones (4 x R), cada fila
%    se corresponde con un valor de una característica
%    clave. El número de columnas R, depende del número
%    total de segmentos de señal generados,
%    esta matriz agrupa el conjunto de matrices
%    de dimensiones (4 x Q) obtenidas mediante la
%    función PreparacionDatos.m para cada modulación.
% Targets
%    Matriz de dimensiones de (5 x R), cada columna se
%    corresponde con un tipo de modulación analógica
%    de la siguiente manera.
%    AM --> [1;1;1;1;1]
%    DSB-SC --> [0;1;1;1;1]
%    LSB --> [0;0;1;1;1]
%    USB --> [0;0;0;1;1]
%    FM --> [0;0;0;0;1]
```

11 entradas, cada entrada es una matriz de (4 x Q)

```
%Entrenamiento.
%    [net] = Entrenamiento(Inputs,Targets)
%    ENTRENAMIENTO, entrena una red neuronal con
%     parámetros previamente definidos, dicho
%     entrenamiento es realizado mediante unas Inputs y
%     Targets ofrecidas por la función
%     Inputs_Targets_ANN.m Además, esta función almacena
%     en un archivo .mat la red neuronal entrenada.
%    El directorio del fichero guardado será
%    el mismo donde se encuentre esta función.
```

```
%Validacion.
%    [BinaryOutput] = Validacion(net,Inputs)
%    VALIDACION, valida una red neuronal entrenada a
%     partir de una matriz de entradas o de datos de
%     dimensiones (4 x Q) o, (4 x R) o bien cualquier
%     otra dimensión (4 x N) en el número de columnas.
%
%    BinaryOutput se corresponde a la matriz
%    de salida generada por la red neuronal,
%    cuyas dimensiones serán de (5 x Q) o,
%    (5 x R) o bien cualquier otra dimensión (5 x N)
%    en el número de columnas.
%    Esta matriz de salida, estará compuesta por
%    diferentes combinaciones de ceros y unos.
%    Las siguientes combinaciones de dígitos,
%    corresponderán a una señal analógica modulada.
%    AM --> [1;1;1;1;1]
%    DSB-SC --> [0;1;1;1;1]
%    LSB --> [0;0;1;1;1]
%    USB --> [0;0;0;1;1]
%    FM --> [0;0;0;0;1]
%    Cualquier otra combinación de dígitos, corresponderá
%    con una señal ruidosa, por ejemplo.
%    ruido --> [1;1;0;1;1]
```

```
%bladeRF_rx_stream.
%    El script bladeRF_rx_stream o script principal para
%    la recepción y clasificación de señales transmitidas
%    y recibidas mediante SDR, será responsable de la
%    llamada a todas las funciones que permitan la
%    clasificación de la señal recibida.
%    Además, permitirá la conexión y configuración del
%    SDR receptor para la recepción y división de las
%    señales en fragmentos de menor duración temporal.
```

```
%decodMod.
%    [ModType] = decodMod(BinaryOutput)
%    DECOMOD realizara la decodificación de la
%    matriz binaria BinaryOutput ofrecida como salida
%    por la red neuronal, permitiendo para cada columna
%    de dicha matriz, determinar si se trata de una
%    modulación en concreto o si se trata de ruido.
%    Ademas, esta función no solo devuelve un vector,
%    sino que genera un aviso en el command window
%    indicando el tipo de modulación de la señal recibida
%    o bien si esta señal no ha podido clasificarse como
%    modulación.
% ModType
%    En un vector de dimensiones (1 x 5), el número
%    de columnas indica el número de modulaciones
%    a clasificar, el valor ofrecido por cada columna
%    indica el número de veces que un fragmento ha sido
%    clasificado como un tipo de modulación.
%    ModType = [AM DSB-SC LSB USB FM];
%
%    % Ejemplo 1:
%    %    A partir de la siguiente matriz BinaryOutput
%    %    obtenida a la salida de la red neuronal
%    %    para una señal de entrada recibida por el SDR,
%    %    determine el tipo de modulación que se trata.

%   BinaryOutput = [1 1 1 1 1;1 1 1 1 1 ;1 1 1 1 1; 1 0 0 1 1; 1 1 1 1 1]
%   ModType = decodMod(BinaryOutput)
%  Modtype = [3 0 0 0 0] -> Señal detectada AM
```

6 BIBLIOGRAFÍA

[1] HUYNH-THE, Thien, PHAM, Quoc-Viet, NGUYEN, Toan-Van, NGUYEN, Thanh Thi, RUBY, Rukhsana, ZENG, Ming and KIM, Dong-Seong. Automatic Modulation Classification: A deep architecture survey. *IEEE Access*. 2021. Vol. 9, p. 142950–142971. DOI 10.1109/access.2021.3120419.

[2] AZZOUZ, E.E. and NANDI, A.K. Automatic identification of digital modulation types. *Signal Processing*. 1995. Vol. 47, no. 1p. 55–69. DOI 10.1016/0165-1684(95)00099-2.

[3] NANDI, A.K. and AZZOUZ, E.E. Automatic analogue modulation recognition. *Signal Processing*. 1995. Vol. 46, no. 2p. 211–222. DOI 10.1016/0165-1684(95)00083-p.

[4] NANDI, A.K. and AZZOUZ, E.E. Algorithms for automatic modulation recognition of Communication Signals. *IEEE Transactions on Communications*. 1998. Vol. 46, no. 4p. 431–436. DOI 10.1109/26.664294.

[5] KIM, Byeoungdo, KIM, Jaekyum, CHAE, Hyunmin, YOON, Dongweon and CHOI, Jun Won. Deep Neural Network-based automatic modulation classification technique. *2016 International Conference on Information and Communication Technology Convergence (ICTC)*. 2016. DOI 10.1109/ictc.2016.7763537.

[6] DILEEP, P, DAS, Dibyaiyoti and BORA, Prabin Kumar. Dense layer dropout based CNN architecture for Automatic Modulation Classification. *2020 National Conference on Communications (NCC)*. 2020. DOI 10.1109/ncc48643.2020.9055989.

[7] LEE, Jung Ho, KIM, Kwang-Yul and SHIN, Yoan. Feature image-based automatic modulation classification method using CNN algorithm. *2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 2019. DOI 10.1109/icaiic.2019.8669002.

[8] TUTTLEBEE, W.H.W. Software-defined radio: Facets of a developing technology. *IEEE Personal Communications*. 1999. Vol. 6, no. 2p. 38–44. DOI 10.1109/98.760422.

[9] Home. *Nuand* [online]. [Accessed 6 July 2022]. Available from: https://www.nuand.com/.

[10] WANG, Zhiyuan. *Slow wireless communication testbed based on software-defined radio*. 2017. Master's thesis. University of Twente.

[11] RTL-SDR blog V3 datasheet. [online]. [Accessed 6 July 2022]. Available from: https://www.rtl-sdr.com/wp-content/uploads/2018/02/RTL-SDR-Blog-V3-Datasheet.pdf

[12] *Great scott gadgets* [online]. [Accessed 7 July 2022]. Available from: https://greatscottgadgets.com/hackrf/.

[13] Features - nuand. [online]. [Accessed 7 July 2022]. Available from: https://www.nuand.com/bladeRF-brief.pdf

[14] PENG, Fusheng, HUANG, Xianhe, LI, Yimei and HU, Jianguo. Realization of voltage controlled temperature compensated crystal oscillator with single varactor. *2018 IEEE International Frequency Control Symposium (IFCS)*. 2018. DOI 10.1109/fcs.2018.8597501.

[15] LANDAU, H.J. Sampling, data transmission, and the Nyquist rate. *Proceedings of the IEEE*. 1967. Vol. 55, no. 10p. 1701–1706. DOI 10.1109/proc.1967.5962.

[16] MISHALI, Moshe and ELDAR, Yonina C. From theory to practice: Sub-Nyquist sampling of sparse wideband analog signals. *IEEE Journal of Selected Topics in Signal Processing*. 2010. Vol. 4, no. 2p. 375–391. DOI 10.1109/jstsp.2010.2042414.

[17] IQ complex tutorial. *IQ Complex Tutorial - GNU Radio* [online]. [Accessed 12 July 2022]. Available from: https://wiki.gnuradio.org/index.php/IQ_Complex_Tutorial#sampled_signals

[18] PROAKIS, J. *Digital Communication.* Singapur : McGraw Hill Series in Electrical and Computer Engineering, 1989.

[19] GALLAGER, R. Principles of Digital Communication. UK : University Press Cambridge, 2008.

[20] BENEDETTO, S. y BIGLIERI, E. Principles of digital transmission: with wireless applications. NY : Kluwer Academic/Plenum Publishers, 1999.

[21] Matlab - El Lenguaje del Cálculo Técnico. *El lenguaje del cálculo técnico - MATLAB & Simulink* [online]. [Accessed 14 July 2022]. Available from: https://es.mathworks.com/products/matlab.html

[22] CAÑADAS QUESADA, Francisco Jesús. *Seminario TCT MATLAB 2020_2021* [online]. [Accessed 16 July 2022]. Available from: https://dv.ujaen.es/goto_docencia_file_1145270_download.html

[23] GNU Radio Community. *GNU Radio* [online]. [Accessed 16 July 2022]. Available from: https://wiki.gnuradio.org/index.php/Main_Page

[24] SORAYA, Sedkaoui. Supervised versus unsupervised algorithms: A guided tour. *Data Analytics and Big Data*. 2018. P. 123–151. DOI 10.1002/9781119528043.ch7.

[25] HOSSEN, Abdulnasir, AL-WADAHI, Fakhri and JERVASE, Joseph A. Classification of modulation signals using statistical signal characterization and Artificial Neural Networks. *Engineering Applications of Artificial Intelligence*. 2007. Vol. 20, no. 4p. 463–472. DOI 10.1016/j.engappai.2006.08.004.

[26] ZHU, Zhechen and NANDI, Asoke Kumar. *Automatic Modulation Classification: Principles, algorithms, and applications*. Chichester, West Sussex, United Kingdom : Wiley, 2015.

[27] KHATRI, Samidha, ARORA, Aishwarya and AGRAWAL, Arun Prakash. Supervised machine learning algorithms for credit card fraud detection: A comparison. *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. 2020. DOI 10.1109/confluence47617.2020.9057851.

[28] HOPFIELD, J.J. Artificial Neural Networks. *IEEE Circuits and Devices Magazine*. 1988. Vol. 4, no. 5p. 3–10. DOI 10.1109/101.8118.

[29] Redes Neuronales Artificiales - Franciscocruz.org. [online]. [Accessed 25 July 2022]. Available from: https://franciscocruz.org/lectures/sistint/rna.pdf

[30] OLABE, Xabier Basogain. Redes neuronales artificiales y sus aplicaciones. *Publicaciones de la Escuela de Ingenieros*, 1998.

[31] SANTANA VEGA, Carlos. Regresión lineal y Mínimos Cuadrados ordinarios | DotCSV. *YouTube* [online]. 16 December 2017. [Accessed 27 July 2022]. Available from: https://www.youtube.com/watch?v=k964_uNn3l0&ab_channel=DotCSV

[32] SOUSA, S, MARTINS, F, ALVIMFERRAZ, M and PEREIRA, M. Multiple linear regression and artificial neural networks based on principal components to predict ozone concentrations. *Environmental Modelling & Software*. 2007. Vol. 22, no. 1p. 97–103. DOI 10.1016/j.envsoft.2005.12.002.

[33] FREDERICKSON, Ben. An interactive tutorial on numerical optimization. *Ben Frederickson* [online]. [Accessed 31 July 2022]. Available from: http://www.benfrederickson.com/numerical-optimization/.

[34] SANTANA VEGA, Carlos. ¿Qué es una red neuronal? parte 3.5 : Las Matemáticas de Backpropagation | DotCSV. *YouTube* [online]. 14 October 2018. [Accessed 2 August 2022]. Available from: https://www.youtube.com/watch?v=M5QHwkkHgAA&ab_channel=DotCSV

[35] GALLINDO MIER, Rafael. Progreso en la Clasificación Automática de Modulación para Radios Cognitivos. *Revista Digital de las tecnologías de la Información y las Telecomunicaciones*. 2 May 2019. Vol. 18, p. 96–115.

[36] CAÑADAS QUESADA, Francisco Jesús. *Aplicaciones de la Transformada Discreta de Fourier* [online]. [Accessed 4 August 2022]. Available from: https://dv.ujaen.es/goto_docencia_file_1140266_download.html

[37] MATLAB & Simulink. *Modulation Classification with Deep Learning* [online]. [Accessed 6 August 2022]. Available from: https://www.mathworks.com/help/deeplearning/ug/modulation-classification-with-deep-learning.html

[38] NANDI, A. K. and AZZOUZ, E. E. UK : *Signal Processing.* UK : s.n, 1997

[39] LI, Wei, ZHANG, Yue, HUANG, Li-Ke, COSMAS, John, MAPLE, Carsten and XIONG, Jian. Self-IQ-demodulation based compensation scheme of frequency-dependent IQ imbalance for wideband direct-conversion transmitters. *IEEE Transactions on Broadcasting*. 2015. Vol. 61, no. 4p. 666–673. DOI 10.1109/tbc.2015.2465138.

[40] GHILDUTA, Robert. DC offset and IQ imbalance correction · Nuand/Bladerf Wiki. *GitHub* [online]. [Accessed 8 August 2022]. Available from: https://github.com/Nuand/bladeRF/wiki/DC-offset-and-IQ-Imbalance-Correction#Generating_a_DC_offset_table.