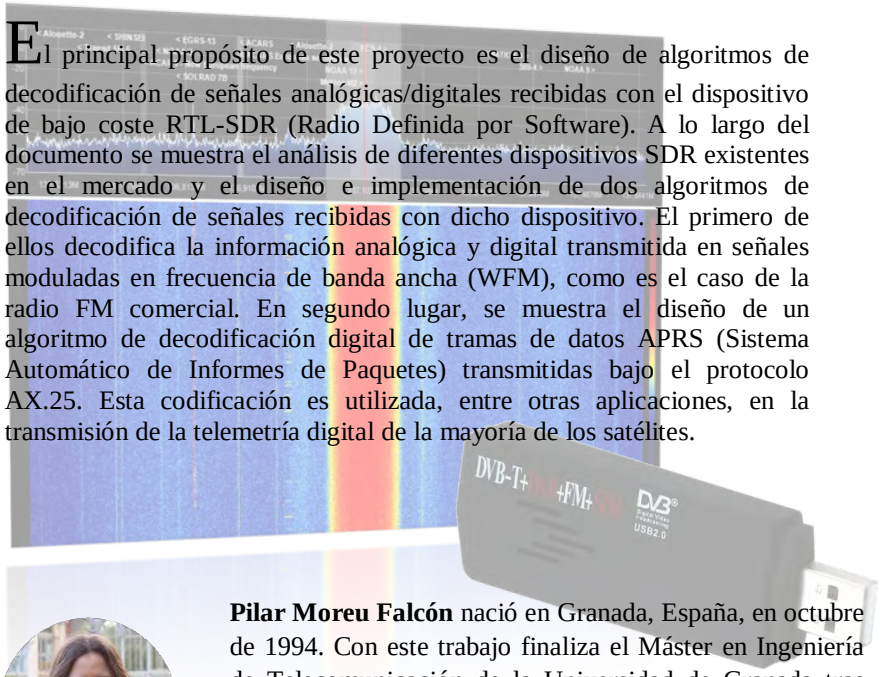


El principal propósito de este proyecto es el diseño de algoritmos de decodificación de señales analógicas/digitales recibidas con el dispositivo de bajo coste RTL-SDR (Radio Definida por Software). A lo largo del documento se muestra el análisis de diferentes dispositivos SDR existentes en el mercado y el diseño e implementación de dos algoritmos de decodificación de señales recibidas con dicho dispositivo. El primero de ellos decodifica la información analógica y digital transmitida en señales moduladas en frecuencia de banda ancha (WFM), como es el caso de la radio FM comercial. En segundo lugar, se muestra el diseño de un algoritmo de decodificación digital de tramas de datos APRS (Sistema Automático de Informes de Paquetes) transmitidas bajo el protocolo AX.25. Esta codificación es utilizada, entre otras aplicaciones, en la transmisión de la telemetría digital de la mayoría de los satélites.

Pilar Moreu Falcón nació en Granada, España, en octubre de 1994. Con este trabajo finaliza el Máster en Ingeniería de Telecomunicación de la Universidad de Granada tras haber terminado satisfactoriamente el Grado de Ingeniería de las Tecnologías de Telecomunicación con mención en Sistemas de Comunicación en julio de 2016 en la misma universidad.

Andrés María Roldán Aranda es el profesor ingeniero a cargo del presente proyecto, así como el tutor del alumno. Actualmente es profesor del departamento de Electrónica y Tecnología de Computadores de la Universidad de Granada.

TRABAJO FIN DE MÁSTER

Algoritmos de decodificación de señales radio analógicas/digitales sobre SDR

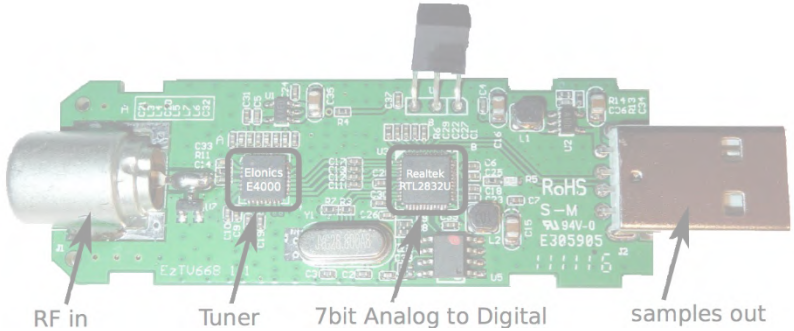
Pilar Moreu Falcón

INGENIERÍA DE TELECOMUNICACIÓN

2017/18

UNIVERSIDAD DE GRANADA

Máster en Ingeniería de Telecomunicación



TRABAJO FIN DE MÁSTER

Algoritmos de decodificación de señales radio analógicas/digitales sobre SDR

Pilar Moreu Falcón

Año académico 2017/2018

Tutor: Andrés María Roldán Aranda

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

"Algoritmos de decodificación de señales analógicas/digitales sobre SDR"

CURSO: 2017/2018

Pilar Moreu Falcón

—

Granada, 14 de septiembre de 2018

MÁSTER EN INGENIERÍA DE TELECOMUNICACIÓN

"Algoritmos de decodificación de señales analógicas/digitales sobre SDR"

REALIZADO POR:

Pilar Moreu Falcón

DIRIGIDO POR:

Andrés María Roldán Aranda

DEPARTAMENTO:

Electrónica y Tecnología de los Computadores

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

D. Andrés María Roldán Aranda, Profesor del departamento de Electrónica y Tecnología de los Computadores de la Universidad de Granada, como director del Trabajo Fin de Máster de D^{a}. Pilar Moreu Falcón,

Informa:

que el presente trabajo, titulado:

"Algoritmos de decodificación de señales analógicas/digitales sobre SDR"

ha sido realizado y redactado por la mencionada alumna bajo mi dirección, y con esta fecha autorizo a su presentación.

Granada, a 14 de septiembre de 2018

Fdo. Andrés María Roldán Aranda

Los abajo firmantes autorizan a que la presente copia de Trabajo Fin de Máster se ubique en la Biblioteca del Centro y/o departamento para ser libremente consultada por las personas que lo deseen.

Granada, a 14 de septiembre de 2018

Fdo. Pilar Moreu Falcón

Fdo. Andrés María Roldán Aranda

Algoritmos de decodificación de señales analógicas/digitales sobre SDR

Pilar Moreu Falcón

PALABRAS CLAVE:

GranaSAT, procesamiento de señales, Software-Defined Radio, SDR, demodulación, satélites, MatLab, decodificación, Open-Source, GranaSDR, RF, APRS, AX.25, receptor, RDS, WFM, FM, VHF, Software, Hardware, analógico, digital, AFSK, RTL-SDR.

RESUMEN:

El principal propósito de este proyecto es el diseño de algoritmos de decodificación de señales analógicas/digitales recibidas con el dispositivo de bajo coste RTL-SDR basado en la tecnología Radio Definida por Software.

A lo largo del documento se muestra el análisis de diferentes dispositivos SDR existentes en el mercado y el diseño e implementación de dos algoritmos de decodificación de señales recibidas con dicho dispositivo. El primero de ellos decodifica la información analógica y digital transmitida en señales moduladas en frecuencia de banda ancha (WFM), como es el caso de la radio FM comercial.

En segundo lugar, se muestra el diseño de un algoritmo de decodificación digital de tramas de datos APRS (Sistema Automático de Informes de Paquetes) transmitidas bajo el protocolo AX.25. Esta codificación es utilizada, entre otras aplicaciones, en la transmisión de la telemetría digital de la mayoría de los satélites.

Design of analog/digital signal decoding algorithms on SDR

Pilar Moreu Falcón

KEYWORDS:

GranaSAT, signals processing, Software-Defined Radio, SDR, demodulation, satellites, MatLab, decoding, Open-Source, GranaSDR, RF, APRS, AX.25, rceiver, RDS, WFM, FM, VHF, Software, Hardware, analogic, digital, AFSK, RTL-SDR.

ABSTRACT:

The main purpose of this project is the design of decoding algorithms for analog and digital signals received with the low cost RTL-SDR device (based on Software-Defined Radio).

Throughout the document, the analysis of different SDR devices existing in the market and the design and implementation of two signal decoding algorithms received with said device are shown. The first of them, decodes the analog and digital information transmitted in signals modulated in broadband frequency (WFM), as is the case with commercial FM radio.

Secondly, the design of a digital decoding algorithm for APRS (Automatic Packet Reporting System) data transmitted under the AX.25 protocol is shown. This coding is used, among other applications, in the transmission of digital telemetry of most satellites.

A todo el que ha hecho posible que este proyecto salga adelante

Agradecimientos:

En primer lugar, me gustaría agradecer a mi familia y amigos, gracias a quienes soy quien soy hoy. También quiero agradecer a mis compañeros de mi etapa universitaria, quienes me han ayudado a superar cada obstáculo de la mejor forma posible con su incansable espíritu de compañerismo.

Por último, quiero agradecer a mi tutor su dedicación y apoyo durante la realización de este trabajo con el que finalizo el Máster. Además, agradezco los consejos instructivos que me ha dado durante esta fase y que no sólo me han servido para sacar adelante este proyecto sino que estarán presentes en mi futura vida profesional.

Gracias a todos ellos por apoyarme y confiar en mi durante la etapa académica que hoy culmina.

Acknowledgments:

Firstly, I would like to thank my family and friends, thanks to who I am who I am today. I also want to thank my colleagues at my university stage, who have helped me overcome each obstacle in the best way possible with their tireless spirit of camaraderie.

Finally, I would like to thank to my advisor for his dedication and support during the realization of this project, with which I finish the Master. In addition, I appreciate the instructive comments that has given me during this project, useful to take forward it and it will be present in my future professional life.

Thanks to all of them for supporting me and trusting me during the academic stage that today culminates.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

GNU/Linux GNU/Linux es uno de los términos empleados para referirse a la combinación del núcleo o kernel libre similar a Unix denominado Linux, que es usado con herramientas de sistema GNU. Su desarrollo es uno de los ejemplos más prominentes de software libre; todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera bajo los términos de la GPL (Licencia Pública General de GNU, en inglés: General Public License) y otra serie de licencias libres.

GranaSAT GranaSAT es un proyecto académico de la \acrshort{UGR} que consiste en el diseño y desarrollo de un picosatélite (Cubesat). Este proyecto multidisciplinario está coordinado por el profesor Andrés María Roldán Aranda y está formado por un grupo de estudiantes de diferentes Grados, que pueden adquirir y ampliar los conocimientos necesarios para afrontar un proyecto aeroespacial real.

Hardware Conjunto formado por todas los componentes físicos de un producto electrónico.

MatLab Herramienta de software matemático con un lenguaje de programación propio (M), desarrollado por MathWorks y Cleve Moler en 1984. Se trata de un software propietario de MathWorks que continua en desarrollo actualmente y que cuenta con versiones gratuitas para estudiantes.

Modulación Conjunto de técnicas utilizadas para transportar información sobre una señal portadora, que suele ser un seno.

Ohms Unidad de resistencia eléctrica.

RTL-SDR Dispositivo multipropósito perteneciente a la tecnología SDR. Compuesto por un chipset RTL2832U que permite, entre otras aplicaciones, experimentar en el amplio mundo del espectro radioeléctrico.

SDRSharp También llamado SDR#, es el software compatible RTL-SDR gratuito más popular en uso en la actualidad. Su arquitectura es modular y cuenta con numerosos complementos desarrollados por terceros.

Software Conjunto formado por todos los programas que permiten que un producto electrónico realice ciertas tareas.

Software-Defined Radio Sistema de radiocomunicación donde los componentes (filtros, moduladores/demoduladores, detectores,...) son implementados por software. Un dispositivo SDR se conecta al PC del usuario a través de su puerto USB para la transmisión y recepción de señales de RF.

Windows Sistema operativo realizado por Microsoft

ACRÓNIMOS

2G Segunda Generación

3G Tercera Generación

5G Quinta Generación

ADC Analog Digital Conversor

AFSK Audio Frequency Shift Keying

AGC Automatic Gain Control

AM mplitude Modulation

AMSAT-UK AMateur SATélite en Reino Unido

APRS Automatic Packet Reporting System. Sistema Automático de Informes de Paquetes

ATSC Advanced Television Systems Committee

AX.25 Amateur X.25. Protocolo de capa de enlace de datos

bps Bit Per Second

BPSK Binary Phase Shift Keying

COFDM Coded Orthogonal Frequency Division Multiplexing

CSMA Carrier Sense Multiple Access

DAB Digital Audio Broadcasting

DAC Digital Analog Conversor

dBm Decibelio-milivatio

DDC Digital Down Converter

DSBSC Double-Sideband Suppressed-Carrier Transmission

DSP Digital Signal Processor. Procesadores Digitales de Señal

DTMB Digital Terrestrial Multimedia Broadcast

DUC Digital Up Converter

DVB-T Digital Video Broadcasting – Terrestrial. Difusión de Video Digital - Terrestre

ECC Extended Country Code

ECU Engine Control Unit. Unidad de Control Electrónica

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

FCC Federal Communications Commission

FCS Frame Check Sequence

FD-MIMO Full Dimensional Multiple Input Multiple Output

FM Frequency Modulation

FSK Frecuency Shift Keying

GNU GNU is Not Unix

GPS Global Positioning System. Sistema de Posicionamiento Global

GRC GNU Radio Companion

GSM Global System for Mobile communications

HPSDR High Performance Software Defined Radio

Hz Hercio

IF Intermediate Frequency

ISDB-T Integrated Services Digital Broadcasting Terrestrial

ISO International Standards Organization

LNA Low-noise amplifier

LTE Long Term Evolution

MatLab MATrix LABoratory

MIMO Multiple-input Multiple-output

NBPF Narrow Band Pass Filter

NFM Narrowband Frequency Modulation

NRZ-I Unipolar non-return-to-zero Inversion

OFDM Orthogonal Frequency Division Multiplexing

Open-Source Programa de Código Abierto

OSI Open System Interconnection

PC Personal Computer

PCIe Bus PCI Express

PI Programme identification

PM Phase Modulation

ppm Partes Por Millón

PSK Phase Shift Keying

PTY Programme Type

RCF Radio Communications Foundation

RDS Radio Data System

RF Radio Frequency. Radiofrecuencia

SCA Autorización de Comunicaciones Subsidiarias

SDR Software-Defined Radio

SMA SubMiniature version A

sps Samples Per Second

SSID Service Set IDentifier

TA Traffic announcement

TAPR Tucson Amateur Packet Radio Corporation

TFM Trabajo Fin de Máster

TP Traffic Programme

TV TeleVision

UGR Universidad de Granada

UHF Ultra High Frequency. Rango del espectro electromagnético comprendido entre 300 y 3 GHz.

USB Universal Serial Bus

USRP Universal Software Radio Peripheral

VGA Variable Gain Amplifier

VHF Very High Frequency. Rango del espectro electromagnético comprendido entre 30 y 300 MHz.

WFM Wideband Frequency Modulation

WiFi Wireless Fidelity

WiMAX Worldwide Interoperability for Microwave Access

WLAN Wireless Local Area Network

X.25 X.25. Protocolo de capa de enlace de datos

XOR OR exclusiva

CAPÍTULO

1

INTRODUCCIÓN

El presente proyecto se desarrolla en el marco del Trabajo Fin de Máster que pone fin al Máster Universitario en Ingeniería de Telecomunicaciones cursado por la alumna en la *Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación* de la *Universidad de Granada*. El principal objetivo del mismo es demostrar los conocimientos y capacidades adquiridos por la alumna durante la realización de dicho posgrado. Para ello, se ha decidido diseñar diferentes algoritmos de decodificación de señales analógicas/digitales sobre la filosofía *Software-Defined Radio*.

Este TFM se realiza en colaboración con el proyecto académico GranaSAT. Se trata de un proyecto aeroespacial llevado a cabo en la *Universidad de Granada* y coordinado por el Profesor Andrés María Roldán Aranda. GranaSAT, cuyo logo se muestra en la figura 5.1 tiene como objeto construir un *Cubesat*. Está formado por un grupo de estudiantes de diferentes Grados que quieren adquirir conocimientos en el campo aeroespacial y vivir una experiencia real en este área. El laboratorio está ubicado en el *Instituto Mixto Universitario de Deporte y Salud (iMUDS)* de la UGR.

Parte de la necesidad de integrar los algoritmos necesarios para la recepción y decodificación de señales procedentes de satélites con el dispositivo SDR en el Software Open-Source *GranaSDR* desarrollado por el proyecto académico GranaSAT por el supervisor del desarrollo Software y web Alberto Marín Navarro. Todo esto se comentará en la sección 1.1 del presente trabajo.

Los satélites transmiten señales analógicas/digitales, que contienen tramas de datos APRS, bajo el protocolo AX.25. Se deberá diseñar un algoritmo capaz de demodular las

1

Figura 1.1 – *Logotipo del proyecto académico GranaSAT [14].*

señales recibidas y decodificar la información digital contenida en ellas. Estos conceptos serán estudiados en el capítulo 5, donde se verá que la codificación de las tramas de datos es muy similar a la empleada para codificar los servicios RDS de la radio FM comercial, detallado en el capítulo 4. Esta concurrencia servirá como precedente para el estudio de la codificación de señales de radio FM comercial y para la implementación de su decodificación. De este modo, el presente trabajo plantea diseñar los algoritmos necesarios para implementar dos decodificadores, uno para FM comercial y otro para APRS AX.25.

En resumen, se emplearán los conocimientos de procesamiento de señales junto con los de programación para implementar los algoritmos que resuelven la problemática mencionada.

El presente trabajo en su totalidad está basado y parte de la tecnología SDR. Como se verá en el capítulo 2, la tecnología SDR está orientada a un uso más cotidiano ya que únicamente se necesita un ordenador y el propio dispositivo SDR para recibir y decodificar señales a diferentes frecuencias. Este dispositivo electrónico tiene un precio muy reducido por lo que resulta ser una herramienta accesible para cualquier usuario.

En cuanto a la metodología de trabajo, con este proyecto se pretende integrar una tecnología innovadora como es el *Software-Defined Radio* con las técnicas de información y procesamiento de señales tradicionales para reducir el coste económico de la recepción de señales de RF clásica. Con esta labor se desarrollarán las habilidades de análisis, diseño y evaluación de la alumna mediante la implementación de técnicas modernas y baratas para resolver una problemática tradicional.

1.1 Motivación

En primer lugar, la principal motivación de este Trabajo Fin de Máster es poner fin a los estudios de posgrado que terminan la etapa académica de la alumna en la Ingeniería de Telecomunicación.

Por otra parte, la posibilidad de explotar y evaluar los algoritmos implementados con un dispositivo SDR y señales reales es una motivación extra para la alumna para la realización de este proyecto debido a su pasión por el trabajo experimental.

El proyecto engloba conocimientos de las principales ramas de la telecomunicación: *Sistemas de Telemática*, *Sistemas de Comunicación* y *Sistemas Electrónicos*. La mención de Grado de la alumna es *Sistemas de Comunicación* y por este motivo gran parte del proyecto se centra en el procesamiento de señales y en los sistemas de comunicaciones radio. Este hecho refuerza la confianza de la alumna en los conocimientos y capacidades adquiridos durante el Grado y el Máster y permite profundizar en aquellos conceptos que no se han visto durante sus estudios. También son necesarios conocimientos de la especialidad de *Sistemas de Telemática*, rica en conocimientos de programación y redes, para apoyar el desarrollo del algoritmo Software de decodificación de señales y para el entendimiento de los protocolos de envío de información. Por último, la rama de *Sistemas Electrónicos* abarca la parte Hardware del proyecto. Con estos conocimientos se comprenderá la arquitectura Hardware del dispositivo SDR y cada uno de sus componentes, muy útil para determinar el comportamiento del dispositivo bajo ciertas condiciones externas. De este modo, tras la finalización del proyecto la alumna habrá aumentado sus conocimientos en las principales áreas de estudio de la Ingeniería de Telecomunicación, lo que resulta motivador ya que el TFM no se centra solo en su especialización, siendo ésta la filosofía del Máster Universitario cursado.

Desde el punto de vista técnico, el diseño e implementación de los algoritmos de decodificación de señales analógicas/digitales sobre SDR del presente trabajo responde a la necesidad del proyecto académico GranaSAT de ampliar el desarrollo del Software Open-Source de recepción y decodificación de señales de RF capturadas con diferentes dispositivos SDR: RTL-SDR, HackRF One, BladeRF o FUNcube Dongle Pro+. La decisión de desarrollar este Software surge a partir del estudio de las herramientas Software de decodificación de señales de RF existentes en el mercado. La mayoría de ellas implementan algoritmos que decodifican una codificación y/o modulación concreta, en un rango de frecuencias específico o para un único dispositivo SDR. De este modo, desde el proyecto GranaSAT se está desarrollando un Software Open-Source, denominado *GranaSDR*, que reúna todas estas condiciones de forma que con una única herramienta se pueda recibir y decodificar cualquier señal analógica/digital de la banda radio del espectro electromagnético. En la siguiente figura se muestra el logotipo de la herramienta Software *GranaSDR*:

Figura 1.2 – *Logotipo de la herramienta Software GranaSDR.*

El procesamiento de señales para desarrollar este Software es muy amplio, parte ya se

ha implementado en otros proyectos. En concreto, el presente documento se centra en la decodificación de dos tipos de señales: las señales de la radio FM comercial (WFM) y las señales de satélites (APRS bajo AX.25).

1

1.2 Objetivos del proyecto

Como objetivo principal del proyecto se tiene el diseño y la implementación de dos algoritmos de decodificación de señales analógicas/digitales, el decodificar FM comercial y el deodificador AX.25. Para el correcto desarrollo de estos algoritmos es necesario dividir y organizar el TFM en objetivos de menor tamaño para que la alumna sea capaz de resolverlos con éxito.

Los principales objetivos del presente trabajo son:

- Conocer y analizar las necesidades de los algoritmos requeridos de forma que se puedan extraer los requisitos principales y secundarios previos a su desarrollo.
- Analizar y comprender la arquitectura Hardware del dispositivo SDR y su funcionamiento.
- Evaluar el comportamiento del dispositivo SDR de bajo coste *RTL-SDR* cuando se somete a ciertas condiciones.
- Realizar un estudio para decidir qué modelo de SDR es más adecuado al sistema.
- Realizar un análisis de las tecnologías disponibles mediante las cuales se pueda resolver cada uno de los requisitos definidos.
- Diseñar un algoritmo capaz de decodificar de forma automática la información transmitida en las señales de la radio FM comercial.
- Diseñar un algoritmo que decodifique automáticamente la información digital APRS transmitida por los satélites bajo el protocolo AX.25.
- Realizar los test de validación y las pruebas necesarias para determinar que la fase de diseño ha sido exitosa.
- Acercar a la alumna al trabajo real con proyectos tecnológicos y a su organización.
- Formar a la alumna en el diseño de sistemas de ingeniería orientados a productos Software.
- Poner de manifiesto los conocimientos adquiridos por la alumna durante el Máster en Ingeniería de Telecomunicación y reforzar los adquiridos en la especialidad de *Sistemas de Comunicación* en el Grado en Ingeniería de Tecnologías de Telecomunicación.
- Superar la asignatura de Trabajo Fin de Máster con éxito y finalizar el Máster.

1.3 Definición de requisitos

Tras definir el tema de estudio del presente Trabajo Fin de Máster, a continuación se describirán los principales requisitos del sistema de ingeniería orientado a producto Software a implementar. Estos requisitos son los planteados por el coordinador del proyecto GranaSAT y la alumna durante las reuniones iniciales y se basan en el cliente final, que será aquel desee recibir la información de la radio FM comercial y la telemetría enviada por los satélites desde su propio ordenador. A partir de estos requisitos principales derivan una serie de requerimientos secundarios, evaluados por la alumna desde el punto de vista ingenieril. Estos requerimientos serán técnicos y dependerán del Software y Hardware utilizado durante el desarrollo del proyecto.

1.3.1 Requisitos principales

Este conjunto de requisitos responde a las necesidades del mercado, analizadas por los integrantes del grupo GranaSAT dedicados al desarrollo del Software Open-Source *GranaSDR*. Estas necesidades se plantearon en la sección 1.1 del presente capítulo y definen los requisitos del sistema a implementar a nivel funcional. Sin embargo, dado que este trabajo se centra en una parte muy concreta, la decodificación WFM y AX.25, los requisitos a continuación expuestos se centrarán en estos subsistemas del Software.

- **Recepción de señales procedentes de satélites sobre SDR**. Para la recepción de las señales satelitales APRS es necesario, además del dispositivo SDR, una herramienta Software que permita sintonizar el SDR a la frecuencia deseada para captar las señales recibidas.
- **Implementación Software para la decodificación de señales APRS procedentes de satélites**. Para la decodificación de la información digital o telemetría de las señales satelitales es necesaria la implementación de un decodificador AX.25 dedicado. Este decodificador deberá leer las señales captadas y obtener los datos contenidos en ellas.
- **Recepción de señales procedentes de la radio FM comercial sobre SDR**. Al igual que ocurre con al recepción de señales satelitales, para la recepción de las señales WFM de la radio comercial es necesario el dispositivo SDR y la herramienta Software que permita sintonizar el SDR a la frecuencia deseada para captar las señales recibidas.
- **Implementación Software para la decodificación de señales WFM procedentes de la radio FM comercial**. Para la decodificación de la información analógica y digital contenida en este tipo de señales se requiere la implementación de un decodificador WFM dedicado. Este decodificador deberá leer las señales captadas y obtener los datos contenidos en ellas.
- **Hardware empleado**. Para captar las señales se deberán utilizar dos modelos de dispositivo SDR: SDR de bajo coste *RTL-SDR* y el dispositivo *FUNcube Dongle*

1

Pro+. Se empleará un receptor SDR por su fácil integrabilidad a los sistemas de recepción de diferentes señales. Resulta ventajoso implementar los equipos propios de radiocomunicaciones en Software ya que no habrá que modificar el Hardware del sistema en función de la señales que se desean recibir.

1.3.2 Requisitos secundarios

En base a los requisitos anteriores, se establecen especificaciones técnicas necesarias para el correcto funcionamiento de los algoritmos implementados. A continuación se mencionan los requisitos secundarios del proyecto:

- **Conocimiento de las características del sistema SDR**. Para poder recibir las señales correctamente será necesario conocer en detalle la arquitectura de dicho dispositivo, su funcionamiento y su comportamiento bajo diferentes condiciones de trabajo.
- **Corrección de la desviación de frecuencia del sistema SDR**. El dispositivo SDR cuenta con un cristal de cuarzo utilizado para sintonizar la frecuencia de recepción de la señal de RF. Este cristal se calienta y provoca desviaciones en frecuencia a la hora de sintonizar la frecuencia de recepción. Por tanto, se requiere que se evalúe dicha desviación en frecuencia para cada uno de los modelos de SDR utilizados.
- **Banda de FM comercial**. Las emisoras de radio transmiten las señales moduladas en FM a frecuencias de RF dentro del siguiente rango: desde los 87.5 MHz a los 108 MHz.
- **Conocimiento de la modulación en frecuencia**. Para poder decodificar la información contenida en las señales de radio FM comercial se deberá comprender los fundamentos de la modulación en frecuencia para su demodulación.
- **Conocimiento de la señal múltiplex FM o stereo**. Al igual que antes, para decodificar la señal WFM es necesario conocer previamente el formato de la señal transmitida en el emisor de radio.
- **Decodificación de datos RDS**. Atendiendo a las características de la señal stereo estudiada, se decodificará la información digital del servicio RDS teniendo en cuenta el formato de los datos descrito en el estándar *Specification of the radio data system (RDS) for VHF/FM sound broadcasting in the frequency rango from 87,5 to 108,0 MHz* [31].
- **Conocimiento del sistema APRS**. El Sistema Automático de Informes de Paquetes transmite los datos a 144.8 MHz. Además de conocer la frecuencia a la que se deberá sintonizar el SDR para recibir las señales satelitales se debe comprender las características de dicho sistema como la modulación y codificación empleada, entre otros.

- **Conocimiento del protocolo AX.25**. El sistema APRS utiliza para transmitir las tramas de datos el protocolo AX.25. Es por ello por lo que se deberá estudiar el formato de los mensajes, descritos en la última versión del estándar *AX.25 Link Access Protocol for Amateur Packet Radio* [30].

- **Algoritmos de decodificación de coste reducido**. Al tratarse de un proyecto en colaboración con el grupo GranaSAT es necesario adaptar los costes a los medios económicos de dicho proyecto académico. Por ello, el Software implementado debe desarrollarse optimizando los recursos para reducir en lo máximo posible el coste económico.

- **Algoritmos de decodificación con el mínimo coste computacional**. Los algoritmos diseñados se deberán desarrollarse reduciendo el coste computacional para que su funcionamiento sea óptimo.

1.4 Estructura del proyecto

El proyecto se divide en 8 capítulos y 3 anexos que describen las diferentes fases del desarrollo del producto Software propuesto. Cada uno de los capítulos pretenden describir cronológicamente y de forma lógica el trabajo realizado por la alumna durante la realización del TFM.

Los capítulos del presente documento son los siguientes:

- El presente capítulo, numerado como 1, pretende ser una introducción al proyecto tanto en su faceta de Trabajo Fin de Máster como en su faceta de proyecto de colaboración tecnológica con el proyecto académico aeroespacial GranaSAT. En este capítulo se describen los aspectos principales para comprender la finalidad del trabajo así como su organización por parte de la alumna. Además, describe las necesidades principales y secundarias del producto Software a desarrollar en forma de requisitos. Estas necesidades han sido detectadas por la alumna durante las reuniones con el coordinador del grupo GranaSAT.

- El capítulo 2 introduce y describe la filosofía *Software-Defined Radio*, en la que se basa el proyecto. Se verá la evolución y el funcionamiento de este concepto y su situación en el mercado actual. Además, la arquitectura Hardware de los dispositivos SDR utilizados durante el desarrollo del TFM será analizada en detalle. Todo esto ayudará a poner en situación el trabajo dentro de esta tecnología y a comprenderla.

- Los dispositivos SDR utilizados durante el trabajo se evalúan en el capítulo 3. En primer lugar, se calibran y se estudia su desviación en frecuencia. Para ello, este capítulo describe el comportamiento de cada dispositivo electrónico bajo diferentes condiciones como el tiempo de trabajo del SDR o su temperatura.

- A continuación, el capítulo 4 explica los fundamentos de la modulación en frecuencia y los tipos de modulación FM existentes: NFM y WFM. Describe la señal stereo múltiplex FM y la información que contiene, detallando en profundidad la información digital transmitida en el servicio RDS.

- El capítulo 5 introduce las comunicaciones vía satélite y los protocolos de transmisión de las tramas de datos. Estos conceptos son necesarios para comprender el formato de las tramas de datos a decodificar y cómo se codifican y modulan para su envío. En concreto, en este capítulo se describe el sistema de radio aficionado para comunicaciones digitales en tiempo real APRS y el protocolo de capa de enlace de datos AX.25. Además, se detalla en que parte de la comunicación satelital toma protagonismo el dispositivo SDR.

- Tras introducir los conceptos necesarios para el correcto desarrollo de los algoritmos propuestos, el capítulo 6 describe la implementación del receptor WFM. En este capítulo se explican las técnicas de procesamiento de señal empleadas para demodular y decodificar la señal stereo múltiplex FM. Además, se verá el proceso de captación de las señales de la radio FM comercial. Hecho esto, se realizarán las pruebas y test de evaluación de los diferentes receptores para comprobar su funcionamiento.

- Análogamente, el capítulo 7 detalla la implementación del receptor AX.25 y la captación de las señales APRS procedentes de satélites. Al igual que antes, muestra el diseño del algoritmo de decodificación AX.25 y se realizan las pruebas y test de evaluación necesarios para comprobar su funcionamiento.

- Finalmente, el capítulo 8 incluye las conclusiones obtenidas del documento y algunas líneas de trabajo futuro planteadas durante el proceso de desarrollo de los algoritmos de decodificación descritos.

Como complemento, el proyecto cuenta con los siguientes apéndices de apoyo al documento:

- En el anexo A se describen las diferentes técnicas de calibración de los receptores SDR así como el procedimiento necesario para instalar los drivers del dispositivo *RTL-SDR*. Estos drivers son necesarios para poder utilizar el dispositivo como receptor de señales.

- En el anexo B se adjunta el algoritmo de decodificación de los parámetros del servicio RDS con el objeto de no cargar de contenido la parte del capítulo 6 dedicada a la decodificación de esta información digital.

- Por último, el anexo C detalla el presupuesto y los costes asociados a este proyecto.

De acuerdo a la estructura del proyecto descrita, a los objetivos definidos en la sección 1.2 del presenta capítulo y a los requerimientos técnicos evaluados (sección 1.3), la siguiente figura muestra la planificación del proyecto.

Se trata de un *Diagrama de Gantt* que ayuda a seguir la organización de las tareas y a su control temporal. Este diagrama se divide en periodos, cada uno de ellos tiene una duración de medio mes. El proyecto se ha desarrollado durante un curso académico completo. El siguiente diagrama cuenta con una leyenda que muestra el código de colores empleado.

1

Algoritmos de decodificación de señales analógicas/digitales sobre SDR

Leyenda: Duración de la actividad · Inicio real · % Completado · Real (fuera de plazo) · % Completado (fuera de plazo)

PERIODOS (Cada periodo tiene una duración de medio mes)

ACTIVIDAD	INICIO	DURACIÓN	INICIO REAL	DURACIÓN REAL	PORCENTAJE COMPLETADO
Inicio del proyecto					
Reuniones con el coordinador del proyecto GranaSAT y tutor del TFM	1	1	1	1	100%
Aproximación al problema	1	2	1	2	100%
Análisis					
Extracción de requerimientos	2	1	2	1	100%
Evaluación de los recursos Hardware	3	1	4	2	100%
Evaluación de los recursos Software	3	2	3	2	100%
Documentación sobre la radio FM comercial y las señales transmitidas	4	1	4	1	100%
Estudio de la modulación en frecuencia	5	1	5	1	100%
Caracterización de la señal stereo FM	5	2	5	2	100%
Documentación sobre las comunicaciones satelitales y las	6	1	11	1	100%
Estudio del sistema APRS	7	1	12	1	100%
Estudio del protocolo AX.25	8	1	13	1	100%
Caracterización de la señal satelital AX.25	9	2	14	2	100%
Calibrado del SDR	4	2	4	3	100%
Evaluación del comportamiento de los modelos SDR del laboratorio	6	2	6	3	100%
Desarrollo e implementación					
Reuniones con el coordinador del proyecto GranaSAT y tutor del TFM	8	1	8	1	100%
Captación de señales WFM	8	1	8	1	100%
Diseño del algoritmo de demodulación FM	9	1	9	1	100%
Evaluación del algoritmo de demodulación	10	1	10	1	100%
Diseño del algoritmo de decodificación FM	11	2	11	2	100%
Pruebas y testeo del algoritmo	13	1	13	1	100%
Captación de señales satelitales APRS	10	1	15	1	100%
Diseño del algoritmo de demodulación AX.25	11	2	16	2	100%
Evaluación del algoritmo de demodulación AX.25	13	1	18	1	100%
Diseño del decodificador de tramas de datos APRS	14	2	19	2	100%
Pruebas y testeo del algoritmo anterior	16	1	20	1	100%
Documentación					
Documentación del Software implementado	8	13	8	13	100%
Escritura de la memoria	2	21	4	19	100%
Preparación de la defensa	22	2	22	2	100%

Figura 1.3 – *Diagrama de Gantt del desarrollo del proyecto.*

CAPÍTULO

2

ESTADO DEL ARTE: *RADIO DEFINIDA POR SOFTWARE*

La Radio Definida por Software, SDR (Software-Defined Radio) es un sistema de radiocomunicaciones en el que la mayoría de las funciones de la capa física se implementan mediante Software en lugar de Hardware [29] y [28]. Esta tecnología permite implementar gran variedad de componentes (mezcladores, filtros, amplificadores, moduladores/demoduladores, detectores, ...) e incluso sistemas de comunicaciones completos como transmisores, receptores o transceptores, entre otros. Los parámetros de estos sistemas se configuran fácilmente de forma dinámica en Software aportando una gran flexibilidad a la hora de diseñar el sistema de radicomunicación [47]. De este modo, se puede tener un sistema radio completo con un dispositivo SDR, un ordenador doméstico con un puerto USB y una antena. Esta antena dependerá de las necesidades del usuario pero normalmente se utiliza una antena omnidireccional ADS-B que viene incluida en los kit SDR que venden en el mercado, como puede verse en la figura 4.1.

El dispositivo SDR se puede definir como un sistema de RF, donde la mayoría de los componentes son implementados mediante Software, que recibe información y la transmite al ordenador (PC) al que está conectado para su procesamiento. Dado que el procesamiento de señal se realiza en el PC del usuario, el sistema cuenta con un microprocesador de propósito general, tal y como se verá en la sección 2.4. Este hecho reduce la complejidad del sistema SDR ya que no es necesario implementar en Hardware el bloque de procesamiento completo. El coste del dispositivo embebido es bastante bajo en comparación con los sistemas tradicionales, con todos los elementos implementados en

Hardware. Como consecuencia, esta tecnología permite a cualquier usuario tener un sistema radio completo debido a su bajo precio.

Figura 2.1 – *Kit que incluye el dispositivo SDR y una antena ADS-B.*

Un sistema basado en Software-Defined Radio es muy flexible ya que un mismo dispositivo Hardware admite gran variedad de configuraciones mediante Software brindando al usuario muchas posibilidades de trabajo.

A continuación se verá la evolución de la tecnología SDR desde sus inicios hasta la actualidad, su funcionamiento y la arquitectura Hardware de los modelos que se utilizarán en el presente trabajo. En resumen a lo descrito en las líneas anteriores, un sistema SDR está compuesto por dos elementos: un dispositivo Hardware que recibe las señales de RF y un Software que configura este dispositivo Hardware.

2.1 Historia y evolución

La filosofía de la Radio Definida por Software no es algo novedoso. El concepto básico del SDR trata sobre la reducción de costes y el aprovechamiento de los dispositivos de propósito general. Esta idea lleva en la industria mucho tiempo, sin embargo la evolución de la tecnología hace que estos conceptos se vuelvan más aplicables tecnológica y comercialmente hablando hoy en día.

Antiguamente, los sistemas de radiocomunicaciones implementaban en Hardware cada una de las etapas de RF. En los *años 80* se introdujo la posibilidad de controlar estos equipos a través de un ordenador. Para ello, era necesario añadir al dispositivo SDR al menos un puerto de comunicación para su conexión con el PC. No es hasta la *década de los 90* cuando se empezaron a desarrollar los chips DSP que implementaban filtros paso baja y de supresión de ruido a través de técnicas digitales. A pesar de esto, los sistemas seguían siendo Hardware.

En el año 1991 el departamento de defensa de los Estados Unidos desarrolló el primer proyecto que sienta las bases de lo que hoy en día se considera SDR aunque todavía no se había introducido este concepto como tal. El proyecto *SPEAKeasy* pretendía aumentar la interoperabilidad entre diferentes equipos para posibilitar la compatibilidad de un solo

terminal con 10 protocolos distintos, gracias a su arquitectura modular enfocada al Software [47] y [56]. Se desarrollaron diferentes versiones hasta el año 1995 y todas ellas permitían que el Hardware digital de propósito general se comunicara en una amplia gama de frecuencias, técnicas de modulación, métodos de codificación de datos, tipos criptográficos y otros parámetros de comunicación [44].

Se trata de un equipo programable, como el de la figura 2.2 que integraba 10 tecnologías de comunicaciones inalámbricas diferentes operando en el rango de frecuencias de los 2 MHz a los 200 MHz inicialmente, y más tarde ampliado al rango 4 MHz a 400 MHz en la segunda fase de diseño [47]. Además, el prototipo era capaz de actualizar su código de forma que podía contemplar los estándares futuros. A pesar de todas las versiones desarrolladas, solo se consiguió realizar una comunicación simultánea por lo que fue necesario modificar el alcance del equipo. Debido a esta problemática surgió la segunda fase del *SPEAKeasy*, en la que se trabajó para disminuir el peso y costo del prototipo, aumentar su capacidad de procesamiento, diseñar basándonse en arquitecturas Open-Source y resolver la simultaneidad de comunicaciones [52].

Figura 2.2 – *Prototipo del proyecto militar SPEAKeasy [47] [56] [52].*

A mediados de los *años 90* el científico *Dr. Joseph Mitola III* comenzó a desarrollar e investigar un nuevo concepto de equipos de RF [4]. En 1992 introdujo el concepto Software-Defined Radio y en 1996 fundó la corporación sin ánimo de lucro *SDR Forum*, actualmente denominada *Wireless Innovation Forum* [29]. Esta sociedad fue creada para impulsar el desarrollo de aplicaciones civiles y militares de los sistemas SDR. Se trata de un foro de colaboración entre empresas caracterizado por hacer un uso innovador del espectro radioeléctrico e impulsar el avance de la tecnología de RF que soporta las comunicaciones mundiales [29].

Wireless Innovation Forum, cuyo logotipo se muestra en la figura 2.3, sigue activo actualmente y se encarga del desarrollo de estándares y especificaciones en el campo de la SDR para su divulgación.

A partir de este momento y hasta la actualidad se han ido desarrollando diferentes iniciativas de Software-Defined Radio [48]. A continuación, se describen las más

Figura 2.3 – *Logotipo de la corporación Wireless Innovation Forum [29].*

2

importantes:

- *Proyecto GNU Radio.* En 2001 se desarrolló la herramienta Software Open-Source GNU Radio como implementación de sistemas SDR a través de bloques de procesamiento de señales [33]. Más tarde, en 2009 se creó la interfaz gráfica de usuario para desarrollar aplicaciones GNU Radio en el lenguaje de programación Python, denominada GRC, que permitió hacer mucho más intuitiva la programación del SDR. No fue hasta la comercialización del dispositivo USRP cuando se hizo popular este Software. En la figura 2.4 se muestra un ejemplo de un programa desarrollado en esta interfaz gráfica.

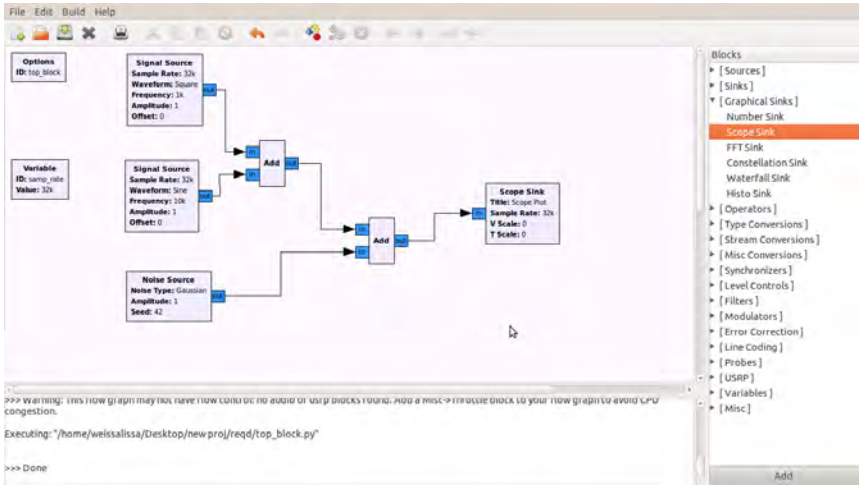


Figura 2.4 – *Interfaz gráfica de usuario GRC para el desarrollo de aplicaciones GNU Radio [33].*

- *Creación del primer periférico de la familia USRP.* La empresa *Ettus Research* produjo en 2005 por primera vez un periférico de la familia USRP, bajo en nombre *USRP1*. El principal objetivo de la empresa era crear una plataforma Hardware SDR de bajo coste para potenciar esta tecnología [35]. El receptor *USRP1*, mostrado en la figura 2.5, se suele utilizar en laboratorios de investigación, universidades y por aficionados con el paquete Software GNU Radio para crear sistemas de radio completos.

- *Proyecto SDR de alto rendimiento HPSDR.* En 2005 Phil Covington, Phil Harman y

Figura 2.5 – *Primer periférico de la familia USRP, denominado USRP1.*

Bill Tracey formaron el grupo HPSDR como proyecto de Hardware y Software Open-Source para el uso de la tecnología SDR por radioaficionados. El proyecto se creó con la idea de dividir el diseño general del SDR en varios módulos diseñados por diferentes usuarios. La conexión entre estos módulos se realizó utilizando un bus predefinido y común, como el de la conexión de placas a una placa base de PC. Esta idea permitió a los usuarios incorporar en el sistema únicamente los módulos que les interesan.

Figura 2.6 – *Dispositivo HPSDR.*

- *Creación del primer transceptor de la familia RadioFlex.* Más tarde, en 2009 *FlexRadio Systems* [8] creó una Radio Definida por Software para su uso entre los radioaficionados permitiendo transmitir señales en algunos segmentos de las bandas de radioaficionado de HF. Este dispoitivo, denominado *Flex-5000A*, se caracterizaba por su reducido consumo, costo y tamaño debido a la optimización del espacio [22]. A diferencia del resto de receptores, permitía la recepción de señales débiles gracias a un preamplificador con alta ganancia y bajo consumo que incorporaba.

Actualmente son muchos los proyectos y prototipos desarrollados bajo la tecnología Software-Defined Radio. Muchos de estos proyectos han nacido y nacerán en un futuro de foros donde los usuarios informan sobre las pruebas, problemas y nuevas funciones que encuentran de los programas y dispositivos SDR del mercado. Este hecho no sería posible si los receptores SDR no fueran tan económicos.

Figura 2.7 – *Primer transceptor de FlexRadio Systems, denominado Flex-5000A [8].*

Un ejemplo de innovación tecnológica haciendo uso del concepto SDR es el diseño de los nuevos prototipos de las redes móviles 5G. Dicha investigación se está realizando mediante tecnología Software-Defined Radio utilizando el receptor USRP por su facilidad de trabajo en múltiples bandas de frecuencia y protocolos. La principal ventaja de usar esta tecnología es el aumento de la velocidad a la hora de realizar las pruebas de dichas redes móviles. Investigadores de *Samsung* están trabajando en el diseño de un sistema MIMO tridimensional denominado FD-MIMO y sus pruebas las realizan en un receptor de la familia USRP. Utilizar este equipo les facilita la operación en las bandas de frecuencias necesarios debido a su configuración por Software [37].

La Infraestrutura celular *OpenBTS* [19] es un ambicioso proyecto que se está llevando a cabo actualmente. Pretende implementar una estación base de telefonía GSM en Software a partir de la arquitectura *GNU Radio* y el receptor *USRP* [46]. En esta infraestructura la señal radio será transmitida o captada por la antena y enviada a la tarjeta secundaria, en la que se produce la conversión D/A o A/D. A continuación se realiza un procesado de señal para transmitir la señal por la interfaz USB del PC para su procesado en la la arquitectura GNU Radio.

Por otra parte, los dispositivos transmisores y receptores SDR también han evolucionado y están en auge hoy en día. Esto se debe a la facilidad de que un mismo dispositivo Hardware pueda transmitir o recibir señales en función de como se programe su Software. Un ejemplo de este equipo sería *Noctar*, diseñado por la empresa *Per Vices* [18] y mostrado en la figura 2.8. Se trata de un SDR dúplex de alto rendimiento diseñado para colocarlo en el PC mediante la interfaz PCIe. Proporciona un ancho de banda más elevado, de hasta 250 MHz y abarca desde GPS hasta Bluetooth pasando por 3G. Se comercializa con fines industriales y de investigación debido a su alto precio.

Es claro ver que la tecnología Software-Defined Radio tiene dos vertientes muy definidas: por un lado los equipos receptores de bajo coste y por otro los sistemas de transmisión y recepción a un precio asequible. Como ya se ha dicho, estos precios tan bajos van a permitir que la tecnología evolucione a pasos agigantados ya que cualquier usuario puede adquirir fácilmente estos equipos y, entre todos ellos, guiarán el camino de la evolución del SDR.

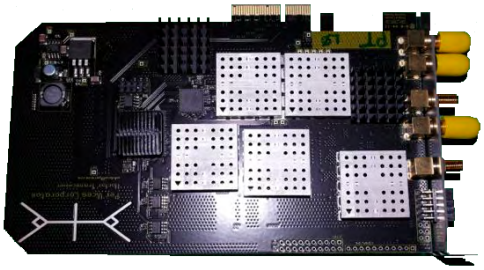


Figura 2.8 – *Sistema SDR dúplex Noctar de la empresa Per Vices [18].*

2.2 SDR en el mercado actual

Hoy en día, cualquier usuario puede acceder a un sistema SDR y esto ha hecho que el mercado de esta tecnología aumente exponencialmente en los últimos años.

Industria aeroespacial

La idea de que un mismo Hardware soporte diferentes funciones configurables por el usuario en Software resulta muy interesante para la industrial aeroespacial. A pesar de lo conservador que es el mercado satelital, está realizando un importante estudio sobre la viabilidad y estabilidad de la tecnología Software-Defined Radio para introducirla en él. Este proceso es necesario para evitar una alta inversión en recursos humanos y económicos en vano puesto que el desarrollo de un sistema aeroespacial es muy costoso. Sin embargo, hoy en día la tecnología SDR tiene un coste de desarrollo a largo plazo mucho menor que los sistemas de radio basados en Hardware.

Los satélites pueden clasificarse en dos grandes grupos: satélites de comunicaciones y de investigación. Los satélites de investigación además de funcionar como repetidores, como es el caso de los satélites de comunicaciones, se encargan del post-procesado de la información capturada. Es por ello, que el transceptor SDR es considerado como un dispositivo ideal para la comunicación entre el satélite y la estación terrestre para el envío de telecomandos, lectura e interpretación de telemetría, detección de la posición de la estación, etc.

Tecnologías de Internet Inalámbrica y Telefonía Celular

Los dispositivos SDR son la opción más clara para la segmentación de redes celulares y de acceso inalámbrico de banda ancha. Con esta tecnología se puede escoger la forma de onda deseada para así conseguir la operación simultánea de varios sistemas [43]. Diferentes tecnologías están desarrollando iniciativas para incorporar la Radio Definida por Software (SDR), como es el caso de WiFi, WiMAX y Beyond 3G. Se utilizaría el SDR junto con la técnica de transmisión OFDM para conseguir esta simultaneidad en los terminales móviles.

La aplicación del SDR en las redes celulares posibilita el aumento de la vida útil de las estaciones base y evita la congestión en el espectro. Cuando las operadoras de telefonía

despliegan su infraestructura pretenden que las estaciones base instaladas tengan una vida útil de varias décadas. Sin embargo, con la constante evolución de los terminales móviles se está acortando la vida útil de estas estaciones ya que la tecnología cada vez es más cambiante. Con la incorporación de la Radio Definida por Software en las estaciones base, éstas podrían incorporar los nuevos protocolos inalámbricos con tan solo una descarga de Software. De este modo, la infraestructura sería mucha más flexible y los operadores ahorrarían significativamente en costes. Esta idea está siendo adoptada por diversos proveedores de telefonía celular desde 2004 para mejorar sus redes [57]. En la actualidad los proveedores de equipos de redes están desarrollando estaciones base SDR que soportan diferentes estándares inalámbricos.

Del mismo modo, los terminales móviles están incorporando la tecnología SDR ya que necesitan soportar varios protocolos (2G, 3G, WLAN, GPS, Bluetooth, televisión móvil...). Al adoptar esta tecnología se disminuye enormemente el costo de desarrollo puesto que cada vez que se quiera incluir un protocolo nuevo solo hay que cambiar el Software del chip en lugar de fabricar uno nuevo.

Industria automotriz

La aplicación de la Radio Definida por Software en los automóviles mejora la experiencia de los usuarios en los viajes mediante la utilización de las tecnologías de comunicaciones. Debido al aumento de vehículos, a la congestión de estos en carreteras y autovías y a la imposibilidad de construir nuevas vías, se ha decidido optar por la telemática para solventar este problema [43]. En Europa la principal preocupación del gobierno para paliar la problemática mencionada es la información de tráfico y navegación. Por ello, en algunas áreas geográficas se ha popularizado los servicios de información de tráfico como el *Canal de Mensajes de Tráfico* usando subportadoras FM.

Los automóviles emplean sistemas ECU para el control de sus funciones mecánicas, hidráulicas y electrónicas tradicionales siendo así más efectivos en costos. Hoy en día, los vehículos tienen cada vez más sistemas ECU, entre unos 20 y 80 ECUs los vehículos más modernos. La industria automotriz está considerando minimizar el número de ECUs en los automóviles y hacerlas mejores. Además, mejoraría la comunicación entre estos dispositivos a través de buses en el vehículo utilizando la tecnología SDR. Con esto, se aprovecharían al máximo los potenciales de los sistemas ECU.

Por otra parte, se han desarrollado iniciativas como *Sistemas Inteligentes de Transportación*. Una de las más populares es la arquitectura Software Open-Source estandarizada en 2003, *AUTOSAR*. Pretende crear estándares Open-Source para arquitecturas eléctricas/electrónicas en automóviles para promover el desarrollo del Software destinado a vehículos [32].

A pesar de todos estos avances, actualmente la tecnología Software-Defined Radio no ha conseguido grandes éxitos en el sector automovilístico [41].

Ámbito académico

Debido a la gran repercusión de la tecnología Software-Defined Radio en las comunicaciones a nivel mundial, la mayoría de universidades han decidido incluir el estudio de esta tecnología en los planes de estudios de la Ingeniería de Telecomunicación (Grado y Máster). En estos centros, los estudiantes reciben formación teórica y práctica sobre SDR. En concreto, en la Universidad de Granada existen varias asignaturas del Máster de Ingeniería de Telecomunicación que forman a los estudiantes en esta tecnología en prácticas de laboratorio utilizando los equipos USRP y *RTl-SDR*.

La corporación *Wireless Innovation Forum* ha creado una competición, patrocinada por los desarrolladores de MatLab, en la que grupos de estudiantes diseñan, desarrollan y testean la Radio Definida por Software orientada a resolver los problemas de la tecnología inalámbrica del mundo real [41].

2.2.1 Sistemas SDR implementados

En los últimos diez años se han diseñado numerosos prototipos de sistemas SDR por parte de empresas y organizaciones. A continuación, se detallan los dispositivos más populares en el mercado actual:

- *FUNcube Dongle Pro+*. Receptor terrestre de señales de RF desarrollado en 2012 gracias a dos organizaciones sin ánimo de lucro: la Comunidad de Satélites Aficionados de Reino Unido AMSAT-UK [5] y la Fundación de Radiocomunicaciones RCF [**?**]. Es capaz de recibir señales en un amplio rango de frecuencias manteniendo la estabilidad: de 150 kHz a 260 MHz y de 410 MHz a 2.05 GHz. Algunos aspectos técnicos de interés son: velocidad de muestreo de 192 kHz, filtros selectivos *SAW* en 2 m y 70 cm, conversores ADC de 16 bits o una desviación en frecuencia de 0.5 ppm [55].

  En la figura 2.9 se muestra el dispositivo, donde se puede ver que se comunica con el ordenador del usuario a través del puerto USB. Este equipo es compatible con la mayoría de Software de configuración del SDR como SdrSharp, Gqrx SDR, Sdr-Radio V2 avance, FDC-DF, SdrDX, DRM o SeeDeR. Normalmente suele utilizarse para la recepción de señales de satélites y su precio es de aproximadamente 150 €.

Figura 2.9 – *Receptor FunCube Dongle Pro+.*

- *RTL-SDR (RTL2832U R820T).* Receptor de señales de RF diseñado en 2010. Se caracteriza por emplear un demodulador DVB-T, el chip *RTL2832U* de la empresa *Realtek Semiconductor Corp.* [39] [50]. Se descubrió que este chip se podía utilizar como receptor SDR de banda ancha debido a sus bondades [50] y como se trata del primer modulador el concepto RTL-SDR lleva su nombre. Por otra parte, este dispositivo se diseñó utilizando el sintonizador de frecuencia más popular del mercado, *Rafael Micro R820T*.

  Este receptor es capaz de captar señales en la banda de frecuencias de los 500 kHz (con prestaciones reducidas) a los 1.7 GHz. Admite frecuencias de muestreo de hasta 3.2 MHz y su desviación en frecuencia debido al aumento de la temperatura del dispositivo varía en función del modelo y fabricante. A diferencia del resto de productos del mercado, el RTL-SDR ofrece las mismas prestaciones a un precio mucho más reducido, entre 9 € y 20 €. De este modo, permite acercar la tecnología SDR a todos los usuarios.

  Es compatible con la mayoría de herramientas Software SDR como SdrSharp, HDSDR, SeeDeR, Gqrx, SdrDx, Sdr-Touch, entre otros.

Figura 2.10 – *Dispositivo RTL-SDR.*

- *HackRF One.* Continuando la tendencia iniciada con el USRP1 se desarrollaron diferentes modelos de transceptores basados en la tecnología SDR para trabajar con un mismo dispositivo diversas tecnologías radio a bajo coste. A lo largo de los años se han ido desarrollando muchos dispositivos de este tipo. Cabe destacar el periférico Open-Source *HackRF*, lanzado en 2014 por Micheal Ossman, por su capacidad para transmitir y recibir señales de RF en la banda de frecuencias de 1 MHz a 6 GHz con una potencia del puerto de la antena (salida) configurable en el rango 30 mW a 1 mW mediante una comunicación semidúplex [36].

  Este transceptor es compatible con las principales herramientas Software de configuración de SDR como GNU Radio y SDRSharp. Su precio en el mercado puede variar entre 270 € y 460 € en función del modelo.

Figura 2.11 – *Dispositivo HackRF.*

- *Mini Airspy.* Es uno de los receptores más conocidos del mercado ya que ofrece un rango de recepción continuo de los 25 MHz a los 1.8 GHz con un ancho de banda de hasta 6 MHz libres de espurios. Salió a la venta en 2016 por la empresa *iTead* y es una versión mini del SDR que fabrican.

  Este receptor tiene una desviación en frecuencia muy reducida, de 0.5 ppm, por lo que se trata de un dispositivo de alta precisión. Además, está diseñado con un conversor ADC de 12 bits. Es compatible con la mayoría de herramientas Software de configuración de SDR y se puede encontrar en el mercado por un precio entre 85 € y 120 € [55].

Figura 2.12 – *Dispositivo Mini Airspy.*

- *NooElec NESDR.* Receptor de banda ancha y de bajo costo diseñado por la empresa *NooElec.* Capaz de recibir señales de RF en el rango de frecuencias 25 MHz a 1.7 GHz debido al chip demodulador que utiliza: *RTL2832U.* A diferencia del dispositivo RTL-SDR, está diseñado con un sintonizador *E4000* capaz de sintonizar un rango de frecuencias mayor, desde 500 MHz hasta 2.35 GHz. A cambio de aumentar el rango de frecuencias de sintonización, este modelo de SDR aumenta su precio a 15-40 €. Dependiendo del modelo, la desviación en frecuencia varía pero siempre entorno a 0.5 ppm [55].

Figura 2.13 – *Dispositivo NooElec NESDR.*

Todos estos dispositivos tienen aplicaciones similares: escaneo de bandas de frecuencia, monitorización radio, APRS, recepción de telemetría, recepción de radio Amateur, etc. Normalmente en el mercado se encuentran en kits junto con una antena con la conexión adecuada al sistema para la recepción de señales.

2.3 Funcionamiento del SDR

Como se ha visto a lo largo del presente capítulo, existen muchas variantes de la Radio Definida por Software de mayor o menor calidad, equipos transmisores y/o receptores, etc. Todos ellos tienen el mismo funcionamiento básico, basado en el esquema de la figura 2.14.

El concepto SDR está compuesto por tres etapas funcionales: sección de RF, sección de IF y sección Banda Base. Las dos primeras etapas se implementan en Hardware mientras que la etapa Banda Base se implementa en Software [46].

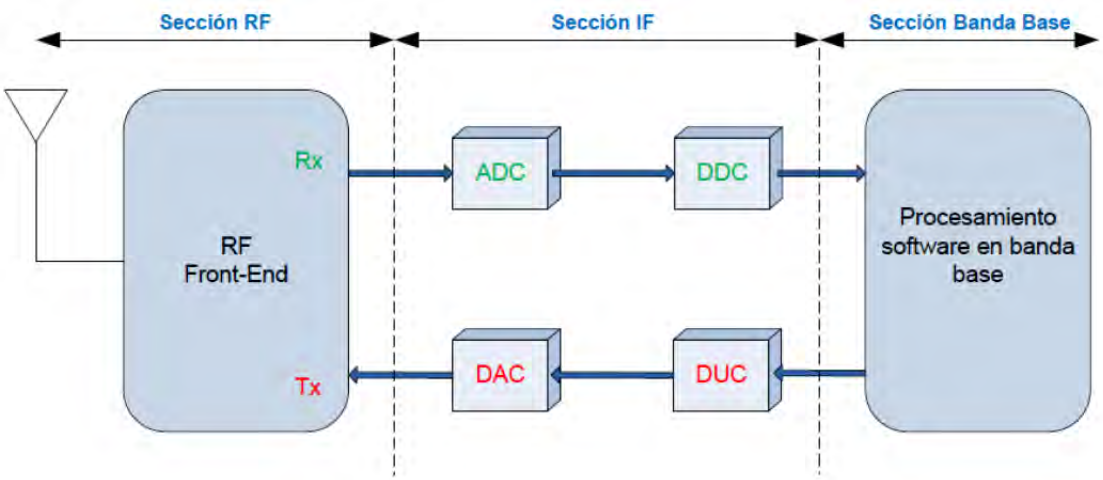


Figura 2.14 – *Diagrama de bloque genérico de un dispositivo SDR.*

Se comprenderá mejor el funcionamiento de estas etapas y del dispositivo SDR dividiendo la descripción en función de si se trata de recepción o transmisión de señales. Esta división es necesaria ya que los dispositivos RTL-SDR son únicamente receptores [50].

Funcionamiento en transmisión:

1. Etapa de Banda Base. Esta primera fase se encarga del procesamiento Software de la señal como la modulación, el análisis en frecuencia de la señal, filtrado de señal, dectectores, etc.

2. Etapa de IF. Pasa la señal de banda base a una frecuencia intermedia (IF) mediante el módulo DUC y convierte la señal digital en analógica para su envío mediante un conversor DAC [50] [48].

3. Etapa de RF *RF Front-End*. Se encarga de transmitir las señales de RF. Para ello, antes amplifica y modula las señales de IF que recibe de la etapa anterior. Esta frecuencia intermedia puede ser 0 Hz, tratándose así de un conversor *Zero-IF* [46] [48].

Funcionamiento en recepción:

1. Etapa de RF *RF Front-End*. Esta primera etapa se encarga de recibir las señales de RF, acondicionarlas para la siguiente etapa y convertirlas de RF a IF [50].

2. Etapa de IF. Recibe una señal de IF y la convierte a banda base a través del módulo DDC, que se encarga de disminuir la tasa de muestreo de la señal. A continuación, digitaliza la señal mediante un conversor ADC [50] [46].

3. Etapa de Banda Base. Por último, la señal banda base recibida es procesada mediante Software, en el que se implementa la demodulación, filtrado o el análisis espectral de la señal, entre otros [50] [46].

2.4 Arquitectura Hardware

El proyecto GranaSAT utiliza diferentes modelos de dispositivos SDR para la recepción de señales de los satélites y de la radio FM comercial, entre otras aplicaciones. En concreto, cuenta con varios dispositivos **RTL-SDR** y un dispositivo *FUNcube Dongle Pro+*. Por tanto, en esta sección del capítulo se describirá únicamente la arquitectura Hardware de estos dos modelos de SDR.

Una vez conocido el funcionamiento y las etapas de un dispositivo SDR será más sencillo comprender la arquitectura Hardware de cada uno de los modelos utilizados.

2.4.1 RTL-SDR

El dispositivo embebido *RTL-SDR* es uno de los modelos SDR más baratos del mercado actualmente. A esta gran ventaja se le unen las buenas prestaciones que tiene, descritas en esta sección.

Como ya se ha mencionado a lo largo del capítulo consta de un chip demodulador ***RTL2832U*** junto con el sintonizador ***R820T***, un conector USB 2.0 y un conector de antena de TV hembra o un conector SMA hembra, según el fabricante. En la figura 2.15 se muestra el interior de un dispositivo RTL-SDR. En ella pueden identificarse fácilmente los chips mencionados.

Figura 2.15 – *Interior del dispositivo RTL-SDR.*

Chip RTL2832U

Diseñado por la empresa *Realtek Semiconductor Corp.*, se trata de un demodulador COFDM. Implementa esta técnica de modulación ya que permite la transmisión de información digital, motivo por el cuál se utiliza para la recepción de televisión DVB-T, radio DAB/DAB+, banda aérea,radioaficionados en bandas altas, telefonía GSM y LTE, GPS y radio FM [50] [55].

Figura 2.16 – *Chip demodulador Chip RTL2832U.*

Este chip, mostrado en la figura 2.16 tiene una interfaz USB 2.0 para su conexión con el PC, de ahí la *"U"* en su nombre [24]. Sus principales características técnicas son la siguientes:

- Permite la recepción de señales capturando sus muestras en cuadratura I/Q.
- Soporta diferentes modos de transmisión (2k y 8k) con un ancho de banda de 6, 7 o 8 MHz con 18.284544 MHz de tasa de muestreo. El modo de transmisión *2k* tiene 1705 portadoras con espaciado de 4464 Hz y el modo *8k* 6817 portadoras con espaciado de 2232 Hz.
- Es capaz de detectar automáticamente los principales parámetros de modulación: razón de código e intervalo de guarda. Y admite los esquemas de modulación 4QAM (QPSK), 16QAM y 64QAM.
- Soporta diferentes frecuencias intermedias utilizando un oscilador de cristal sencillo de

bajo coste de 28.8 MHz que genera una señal de reloj de ±100 ppm, manteniendo así la estabilidad en la recepción. De este modo soporta los siguientes sintonizadores: IF (36.125 MHz), baja-IF (4.57 MHz) y Zero-IF (conversión directa o 0 Hz). Para ello utiliza

El diagrama de bloques funcionales del chip de demodulación *RTL2832U* se muestra en la figura 2.17 y ayudará a entender mejor su funcionamiento [38]. Puede dividirse en el conversor analógico-digital ADC y el procesador de señal digital DSP.

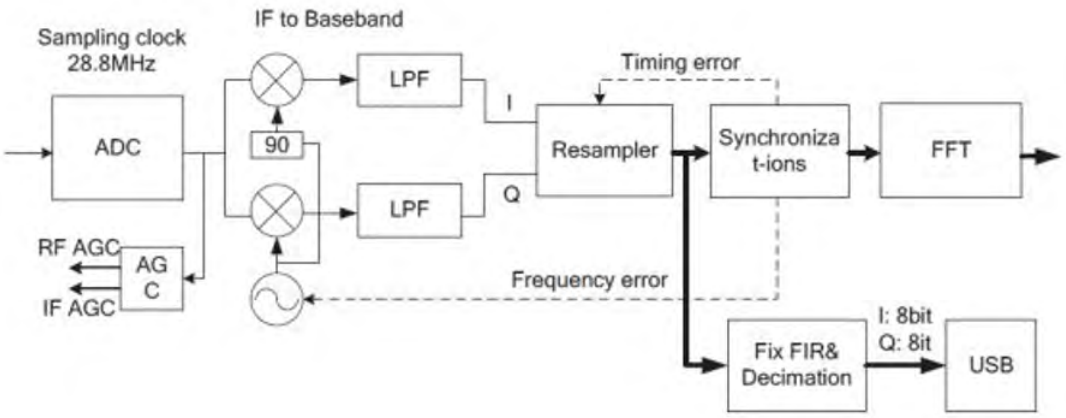


Figura 2.17 – *Diagrama de bloques del chip demodulador RTL2832U.*

El *conversor analógico digital* está formado por el módulo ADC, se encarga de medir el nivel de señal de RF recibido para convertirla en una señal digital con una resolución de 8 bits. Para ello, toma muestras en intervalos periódicos dados por la señal de reloj a 28.8 MHz. A continuación realiza la cuantificación y codificación de la señal con los siguientes parámetros:

- Resolución de 8 bits.
- Tasa de datos de salida de 3.2 MS/s (millones de muestras por segundo), que equivale a una frecuencia de 3.2 MHz.
- Throughput de 25.6 Mbps.

El *procesador de señal digital* está formado por el resto de módulos del diagrama y puede dividirse en las siguientes etapas:

1. *Conversión digital IF a Banda Base.* Esta primera etapa está compuesta por el módulo AGC, un oscilador de cristal y dos mezcladores para obtener la componente en fase (I) y en cuadratura (Q) de la señal, desfasadas 90º.

2. *Filtrado paso baja digital.* La señal en cuadratura se filtra paso baja para eliminar las componentes a frecuencias elevadas para quedarse únicamente con la señal en banda base.

3. *Remuestreo.* Esta señal se remuestrea a una frecuencia menor, de 2.4 MHz, para mejorar la resolución espectral.

4. *Envío de datos.* Por último, se envían la señal banda base como un número complejo con las componentes en fase y cuadratura (ecuación 2.4.1) a través de la interfaz USB 2.0 con una resolución de 8 bits.

$$x(t) = x_I(t) + jx_Q(t) \tag{2.4.1}$$

Chip R820T

El sintonizador *R820T* de TV digital DVB-T de sicilio fue diseñado por la empresa *Rafael Microelectronics* y se muestra en la figura 2.18. Soporta todos los estándares de televisión digital: DVB-T en Europa, ATSC en Estados Unidos, DTMB en China e ISDB-T en Japón [50]. Es capaz de sintonizar señales dentro del rango de frecuencias de 24 MHz a 1.766 GHz con un ancho de banda de 3.2 MHz únicamente para su recepción. Destaca entre el resto de sintonizadores del mercado por sus prestacioens así como su alto rendimiento y coste reducido [34]. Entre sus características técnicas [34] destaca:

- Consumo de energía muy reducido, inferior a 178 mA.
- Cuenta con una interfaz serie I2C de dos hilos para la comunicación con el resto de dispositivos electrónicos.
- Su interfaz de entrada tiene una figura de ruido de 3.5 dB y una potencia máxima de recepción de 10 dBm cuando el modo es 8k, la salida FFT, la modulación 64QAM y la razón de código 7/8.
- Recibe la señal de RF en banda base.

Figura 2.18 – *Chip demodulador Chip R820T.*

El diagrama de bloques funcionales del chip sintonizador *R820T* se muestra en la figura 2.19 y se divide en las siguientes etapas:

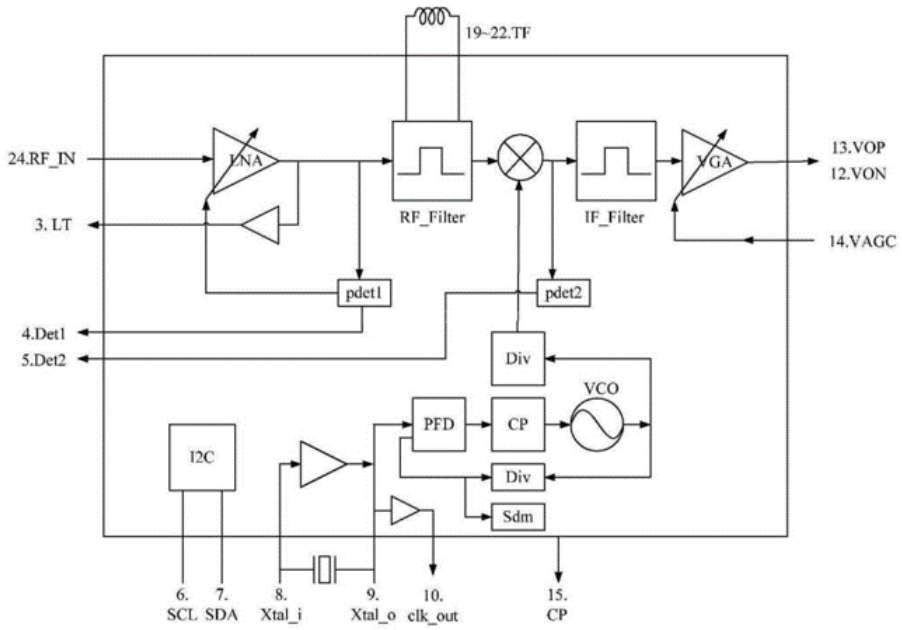


Figura 2.19 – *Diagrama de bloques del chip demodulador R820T.*

- Amplificación de bajo ruido LNA para lograr una buena calidad de imagen. Cuando la potencia de entrada de RF aumenta, la ganancia LNA se atenúa y viceversa. Esta ganancia varía en el rango 0 dB a 40 dB [34].

- Filtrado RF paso banda centrado a la mitad de la frecuencia de muestreo y con un ancho de banda pasante del 80 % de la frecuencia de muestreo. Para modificar la frecuencia a la que se centra el filtro RF se utiliza un cristal de cuarzo con una frecuencia típica de 16 MHz y una desviación frecuencia dentro del rango ±30 dB a ±50 dB. Esta desviación en frecuencia se debe al aumento de la temperatura del cristal de cuarzo debido al tiempo de trabajo del dispositivo [34].

- Mezclador. A continuación la señal pasa por un mezclador para trasladarla de RF a IF. Esta frecuencia intermedia IF no es fija sino que depende de la aplicación, sus valores típicos son 3.57 MHz y 4.57 MHz.

  Para trasladar la señal en frecuencia es necesario un oscilador que genere la frecuencia intermedia IF, implementado mediante un circuito *PLL fraccional*. Dicho circuito tiene un tiempo de bloqueo de 5 segundos y genera la señal a frecuencia intermedia dada por la ecuación 2.4.2 [34].

$$e^{-j2f_{IF}t} \tag{2.4.2}$$

- Filtrado paso banda centrado a la frecuencia intermedia IF con anchos de banda típicos de 6, 7 y 8 MHz. Elimina la frecuencia imagen debida al traslado en frecuencia de la señal.
- Amplificación VGA. Por último, la señal pasa por un amplificador de ganacia variable dentro del rango 1 dB a 48 dB para obtener señales de salida con un nivel de amplitud entre 06 V y 2.5 V [34].

Comprendido el funcionamiento de los chips demodulador y sintonizador del RTL-SDR será mucho más rápido estudiar su diagrama de bloques. A continuación, en la figura 2.20, se puede ver en detalle cada una de las etapas de este receptor y el comportamiento de las señales en cada una de ellas.

Por último, cabe mencionar el hecho de que los receptores RTL-SDR contienen la librería *librtlsdr* y la herramienta de comandos necesaria para poder realizar el intercambio de datos con el dispositivo. Algunos de estos comandos son *rtl_test*, *rtl_eeprom*, *rtl_sdr*, *rtl_fm* y *rtl_tcp*. Este Software es Open-Source y se puede descargar en [23]. Eso será de gran utilidad en el próximo capítulo.

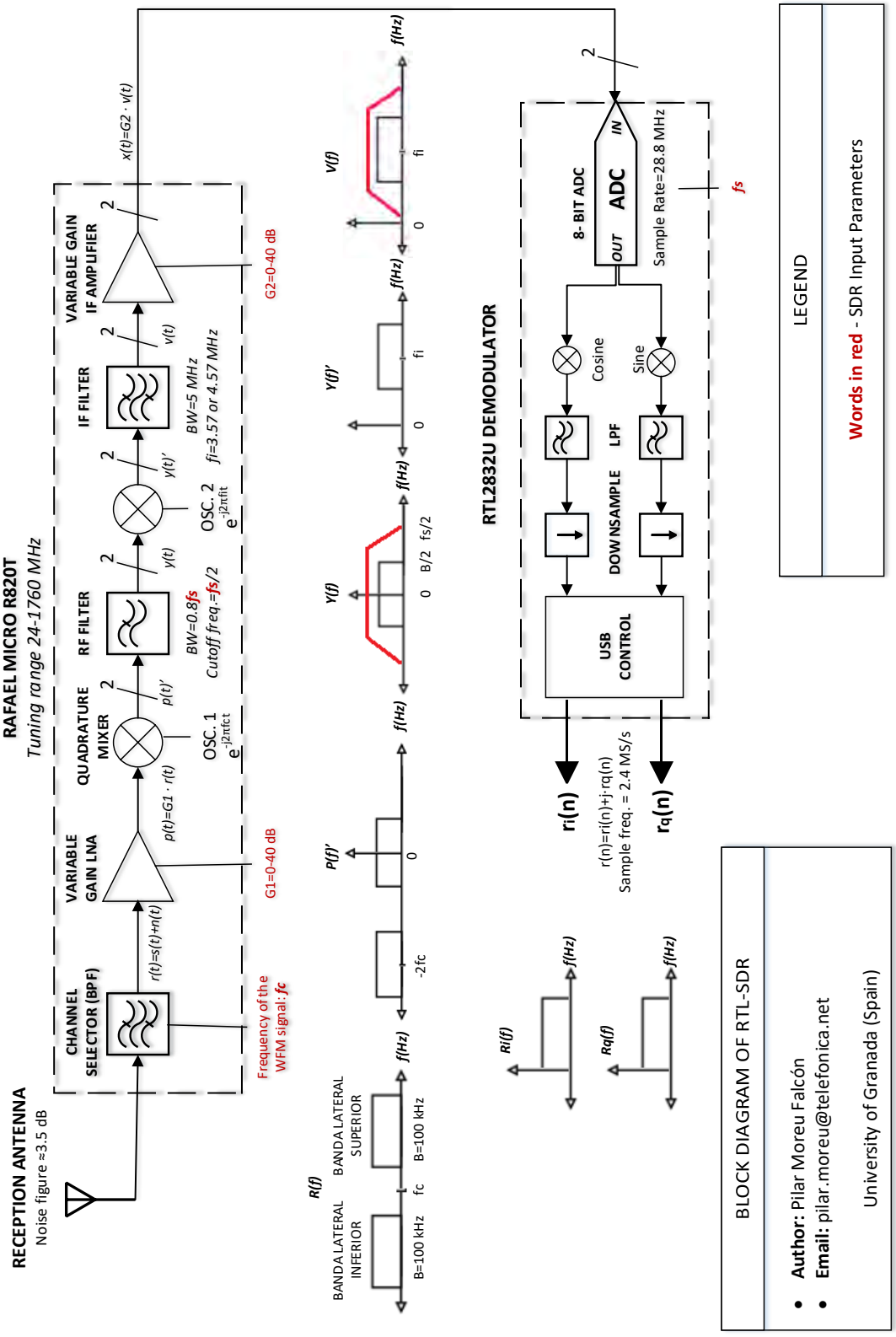


Figura 2.20 – *Diagrama de bloques del receptor RTL-SDR.*

2.4.2 FUNcube Dongle Pro+

A continuación, se detalla la arquitectura Hardware del modelo de SDR *FUNcube Dongle Pro+*. Este dispositivo es considerado en la industria como un receptor SDR avanzado y está diseñado especialmente para la recepción de señales satelitales, con un precio algo más elevado de 150 € aproximadamente [55].

A lo largo de este capítulo se han detallado algunas de características de este receptor de señales SDR. Una de ellas es el rango de cobertura de frecuencia de 150 kHz - 260 MHz y de 410 MHz - 2.05 GHz. Sin embargo, en al práctica se ha demostrado que el rango de frecuencias garantizado es algo menor, de 150 kHz - 240 MHz y de 420 MHz - 1.9 GHz. A pesar de esto, el rango de cobertura sigue siendo mayor que en el receptor RTL-SDR y esto es debido al chip sintonizador que contiene [27]. El resto de características técnicas [7] se enumeran a continuación:

- Para la recepción de señales cuenta con un puerto de antena SMA hembra estándar con una impedancia de 75 Ohmios.
- Interfaz de salida USB 1.X tipo A con conexión macho para la comunicación con el PC.
- La desviación en frecuencia teórica del cristal de cuarzo es mínima, de 0.5 ppm. En la práctica se han obtenido valores mínimos de 1.5 ppm.
- Recibe las señales con una tasa de muestreo de 192 kHz, igual al ancho de banda de recepción. Esto es posible ya que el muestreo es complejo (I/Q) y, por tanto se cumple que $f_s = B$ en vez de $f_s = B/2$.

En la siguiente figura se puede ver el interior de un dispositivo *FUNcube Dongle Pro+*, que consta principalmente de dos chips: sintonizador *E4000* y codec de audio *TLV320AIC3204*.

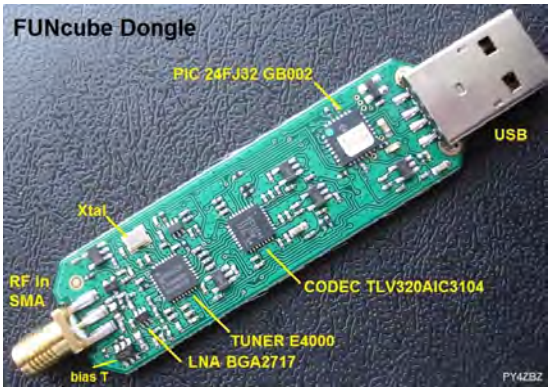


Figura 2.21 – *Interior del receptor FUNcube Dongle Pro+.*

Chip E4000

Diseñado por *Elonics*, se trata de un sintonizador de silicio que afina dos veces más rápido que el chip Rafael Micro *R820T*. Tiene un rango de cobertura de frecuencia mayor, de 65 MHz a 2.3 GHz [1]. Sin embargo, el comportamiento del chip hace que el *E4000* de mejores resultados a frecuencias inferiores a 450 MHz, mientras que el chip *R820T* funciona mejor por encima de esta frecuencia.

Figura 2.22 – *Chip E4000.*

En su diagrama de bloques funcionales, figura 2.23, se puede ver que este sintonizador está compuesto por una serie ordenada de filtros y amplificadores consecutivos que se encargan de captar la señal deseada.

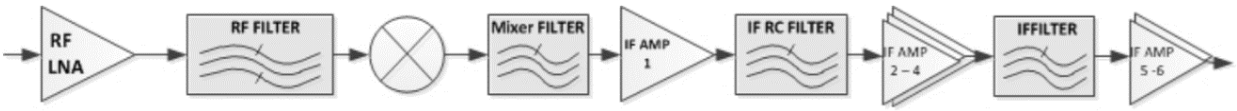


Figura 2.23 – *Diagrama de bloques del chip E4000.*

La etapas por las que pasa la señal en el chip *E4000* son las siguientes:

- Amplificación de bajo ruido en RF. El módulo LNA se encarga de mejorar la sensibilidad general del dispositivo SDR y la figura de ruido.
- Filtrado paso banda en RF para eliminar las componentes en frecuencia indeseadas. Este filtro estará centrado a la frecuencia central a la que el usuario desea recibir señales.
- Traslado de la señal a la frecuencia intermedia IF mediante un mezclador y un oscilador que genera una señal de referencia de 24.576 MHz. Si esta frecuencia intermedia es 0 Hz implementaría *Zero-IF*.
- Filtrado paso banda para eliminar la frecuencia imagen resultante del traslado en frecuencia de la señal.

A partir de aquí se repiten las etapas de amplificación y filtrado paso alta para atenuar las señales fuertes por debajo de la frecuencia mínima de operación. Esta frecuencia está definida entre 54 MHz y 2147 MHz en intervalos de 1100 a 1250 MHz.

Chip TLV320AIC3204

El chip *TLV320AIC3204*, diseñado por *Texas Instruments*, se trata de un codec de audio estéreo de bajo voltaje y baja potencia con entradas y salidas reprogramables.

Se encarga del procesamiento de la señal de audio recibida, dando como salida una señal compleja I/Q como 2.4.1. Este procesador de señal tiene una resolución de 32 bits y opera con una señal de reloj dentro del rango 512 kHz a 50 MHz.

Figura 2.24 – *Chip TLV320AIC3204.*

En la figura 2.25 se puede ver el diagrama de bloques del receptor *FUNcube Dongle Pro+*, compuesto por una antena de recepción, el sintonizador E4000, el codec de audio TLV320AIC3104 y un microcontrolador. Ya se han descrito las características de los chips de sintonización y codec de audio, por lo que a continuación se detallarán los detalles técnicos de la antena y del microcontrolador. De esta forma se comprenderá el funcionamiento y la arquitectura Hardware completa del receptor *FUNcube Dongle Pro+*.

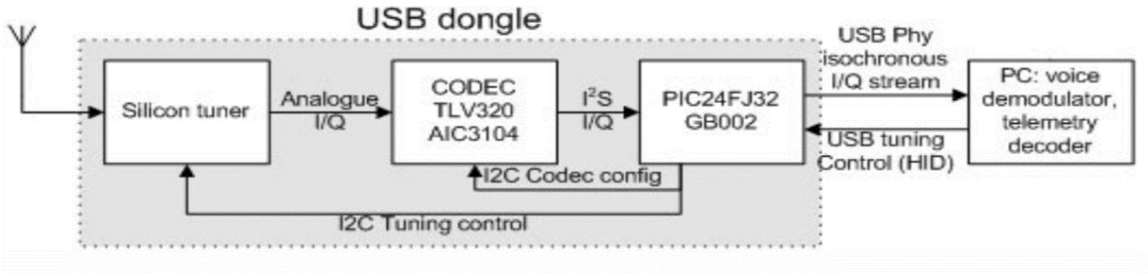


Figura 2.25 – *Diagrama de bloques del receptor FUNcube Dongle Pro+.*

- La antena tiene una figura de ruido variable entre 2.5 dB a 50 MHz y 5.5 dB a 1.296 GHz.
- Incorpora un microcontrolador de 16 bits *PIC24FJ32GB002* con una interfaz de salida de datos USB y un consumo extremadamente bajo, como el de la figura 2.26.

Figura 2.26 – *Microcontrolador PIC24FJ32GB002.*

Se encarga de controlar la señal digital y enviarla al ordenador del usuario. Para ello, implementa dos conversores, analógico digital y viceversa. El convertidor analógico/digital ADC peude ser de 10 bits con una tasa de conversión de 500 ksps o de 12 bits con una tasa de 200 ksps. En cambio, el conversor digital/analógico DAC es de 10 bits y tiene una velocidad de actualización de 1Msps [12].

2

CAPÍTULO

3

EVALUACIÓN DE LOS DISPOSITIVOS SDR EMPLEADOS

El grupo GranaSAT cuenta con diferentes dispositivos SDR en su laboratorio. En concreto, tiene un receptor *FUNcube Dongle Pro+* y varios receptores de bajo coste RTL-SDR de distintos fabricantes. La alumna trabajará con dichos sistemas SDR durante el desarrollo del presente trabajo, motivo por el cuál es necesario evaluarlos previamente para así conocer sus características y comprender su funcionamiento.

En las secciones siguientes se evalúa el funcionamiento de los dos modelos de SDR del laboratorio y posteriormente se realiza un estudio comparativo entre estos para decidir qué modelo tendrá mejor rendimiento. En primer lugar, se calibra cada dispositivo para conocer la desviación de frecuencia que presenta cuando está trabajando. Posteriormente se evaluará esta desviación en frecuencia, medida en ppm, en función del tiempo de trabajo del dispositivo. De este modo, se obtendrá una curva que permita al usuario futuro del dispositivo corregir su desviación en frecuencia en función del tiempo. Esta gráfica será muy útil para determinar qué modelo de SDR ofrece mejores resultados. Por otra parte, se medirá la temperatura del dispositivo en el momento de la medición de la desviación en frecuencia de forma que se comprenda la relación entre esta desviación y la temperatura.

Los dispositivos SDR utilizan una frecuencia de conversión fija controlada por un cristal de cuarzo para la sintonización de la frecuencia de RF de recepción. A medida que aumenta el tiempo de trabajo del dispositivo SDR, aumenta su temperatura de forma que se calienta el cristal de cuarzo. En función de la calidad y precisión del cristal de cuarzo este aumento

de temperatura tendrá mayor o menor efecto en el funcionamiento del receptor SDR. Una temperatura del cristal de cuarzo suficientemente elevada provoca una mala sintonización de la frecuencia de recepción. El oscilador de cristal de cuarzo no sintoniza la frecuencia de forma exacta sino que sufre un desplazamiento en frecuencia. Esta desviación es la medida que caracteriza cada dispositivo SDR y se suele expresar en partes por millón (ppm).

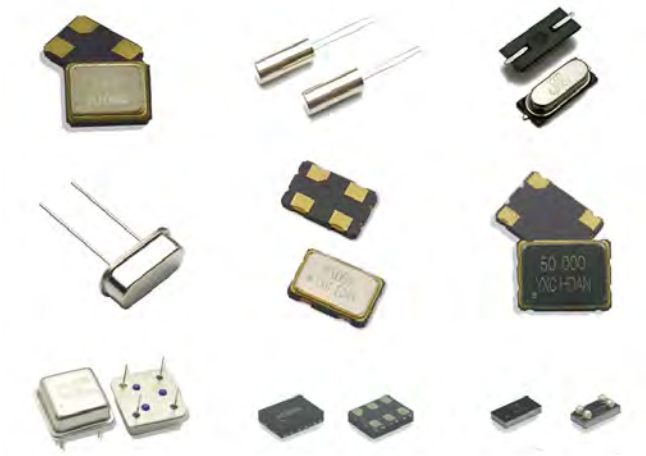


Figura 3.1 – *Ejemplos de osciladores de cuarzo.*

En el mercado existen osciladores de cuarzo de diferentes tolerancias y precios, Figura 3.1. El receptor RTL-SDR utiliza un cristal de cuarzo de baja precisión, de ahí su bajo coste. Como consecuencia de este coste tan reducido, la tolerancia del cristal es elevada tomando valores dentro del rango de ±30 a ±50 ppm. En cambio, el receptor *FUNcube Dongle Pro+* está fabricado con un cristal de cuarzo de alta precisión que tiene una variación en frecuencia de 0.5 ppm teóricas y 1.5 ppm en la práctica. Una variación en frecuencia de 1.5 ppm equivale a una variación de 1.5 Hz por cada MHz de la señal de salida del oscilador de cuarzo.

3.1 Calibración

El primer dispositivo a evaluar es el modelo de receptor RTL-SDR, caracterizado por tener tolerancias de ±30 a ±50 ppm y explicado en el capítulo 2. Este parámetro varía en función del fabricante y modelo de RTL-SDR por lo que es necesario calibrar todos los modelos de RTL-SDR disponibles en el laboratorio. A lo largo del presente trabajo se utilizan los siguientes modelos de RTL-SDR:

- *Andoer® Mini Sintonizador Receptor Soporte a DVB-T+DAB+FM+SDR RTL2832U+R820T Digital Portátil USB 2.0* en blanco como el de la figura 3.2 y en negro como el de la figura 3.3 [3]. Disponible en el mercado por un precio aproximado de 13 €, utiliza un oscilador de cuarzo de baja precisión y cuenta con una interfaz USB 2.0 para su conexión con el PC y una interfaz IEC Coaxial para su conexión con la antena de de TV.

Figura 3.2 – *Receptor RTL-SDR de Andoer® blanco.*

Figura 3.3 – *Receptor RTL-SDR de Andoer® negro.*

- *Allwin Sintonizador Receptor USB DVB-T RTL-SDR Realtek RTL2832U y R820T PAL IEC Input* como el de la figura 3.4, disponible en el mercado por 12 € [2]. Su oscilador de cuarzo también es de baja precisión y cuenta con las mismas interfaces que el dispositivo de Andoer®.

Figura 3.4 – *Receptor RTL-SDR de Allwin.*

El receptor *FUNcube Dongle Pro +* (figura 3.5) está disponible por 107 € en su web oficial [13] y utiliza un oscilador de cuarzo de alta precisión, mejorando sus prestaciones en la recepción de señales de RF. En el capítulo 2 se detalla sus características y funcionamiento.

Figura 3.5 – *Receptor SDR FUNcube Dongle Pro+.*

El calibrado del dispositivo SDR consiste en calcular la desviación en frecuencia medida en ppm de forma que se ajuste el oscilador para la sintonización a la frecuencia RF deseada. Las técnicas de calibración del receptor SDR son muy variadas y las más populares se describen en el anexo A. La siguiente tabla recoge los resultados obtenidos tras implementar las técnicas de calibración del anexo A. Muestra el dispositivo SDR utilizado y su tolerancia o desviación en frecuencia obtenida en cada técnica.

Tolerancia (ppm)	Técnicas de calibración del dispositivo receptor		
SDR	**Software *kalibrate***	**Software *SDRSharp***	**Macro de *MatLab***
Figura RTL-SDR 3.2	62.467	60	98.3641
Figura RTL-SDR 3.3	85.8937	75	38.4918
Figura RTL-SDR 3.4	27.962	24	20.9234
Figura SDR 3.5	-	-1.4	-

Tabla 3.1 – *Resultados de la calibración de los dispositivos SDR.*

La tabla anterior muestra la desviación en frecuencia para cada dispositivo. Para todos ellos vemos que los resultados obtenidos por las diferentes técnicas no son siempre iguales. La tolerancia obtenida por la herramienta *kalibrate* y el Software SDRSharp son muy similares y por el contrario, la macro de MatLab ofrece un resultado más dispar. Estas diferencias se deben a los siguientes motivos.

El Software SDRSharp calcula la desviación en frecuencia mediante la transmisión directa de un tono sin interferencias ni atenuaciones por propagación por lo que el valor obtenido es bastante preciso. Por otra parte, la herramienta *kalibrate* obtiene la desviación en frecuencia como el valor promedio del offset obtenido en diferentes mediciones temporales, de forma que se obtiene un valor de tolerancia preciso. En cambio, la macro de MatLab mide en tiempo real el offset en frecuencia pero no obtiene su valor promedio, sino que devuelve la última medición realizada. La desviación en frecuencia oscila dentro de un rango relativamente amplio, normalmente entre 30 y 50 ppm. Es por este motivo por lo que en la macro de MatLab no se obtienen resultados precisos ya que el rango de tolerancia típica es grande.

Cabe destacar el hecho de que el receptor *FUNcube Dongle Pro+* solo se ha podido calibrar mediante la tercera técnica del anexo A con ayuda del generador de señal y el Software SDRSharp. Esto ocurre porque las técnicas restantes sólo son válidas para aquellos receptores que incluyen los chips *R820T* y *RTL2832U*, esto es, receptores de bajo coste RTL-SDR.

A continuación se ha realizado un estudio de la evolución de la desviación en frecuencia del dispositivo a lo largo del tiempo. Para ello se han tomado medidas cada 20 minutos durante 2 horas de trabajo del receptor a través de la herramienta *kalibrate* debido a la precisión de los resultados que ofrece y a la sencillez de su uso. Para el caso del receptor *FUNcube Dongle Pro+* se ha empleado el Software SDRSharp y el generador de señales para medir la desviación en frecuencia en la recepción.

Tolerancia (ppm)	Tiempo de trabajo					
SDR	**20 min**	**40 min**	**60 min**	**80 min**	**100 min**	**120 min**
Figura 3.2	62.467	62.489	61.281	59.717	63.085	61.099
Figura 3.3	85.8937	82.2821	79.4863	81.3294	82.401	82.983
Figura 3.4	27.962	26.07	24.565	24.872	26.811	24.69
Figura 3.5	-1.4	-1.4	-1.3	-1.2	-1.3	-1.4

Tabla 3.2 – *Desviación en frecuencia del SDR a lo largo del tiempo.*

A partir de estos datos se ha realizado la curva que describe la evolución de la desviación en frecuencia a lo largo del tiempo para cada uno de los receptores:

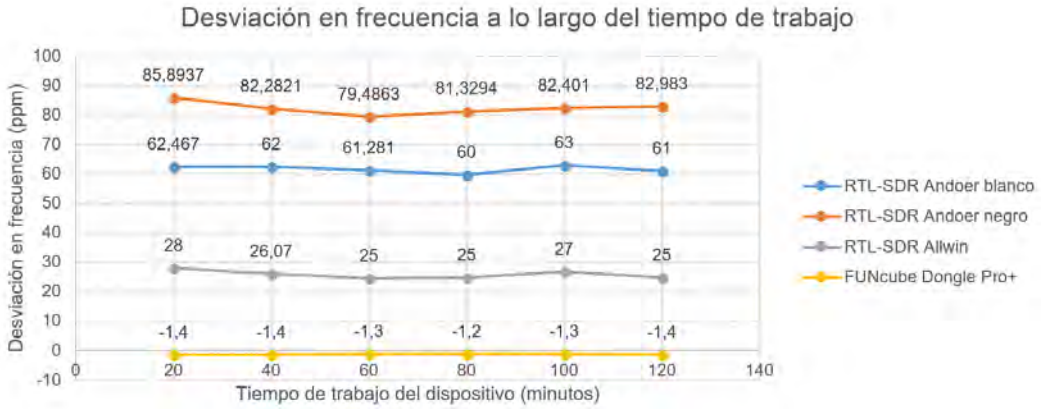


Figura 3.6 – *Desviación en frecuencia del SDR a lo largo del tiempo.*

A medida que aumenta el tiempo de trabajo del dispositivo, la temperatura de este aumenta y eso provoca que la desviación en frecuencia varíe. Los valores entre los cuales oscila la tolerancia del receptor son muy próximos entre sí formando un rango de amplitud 5 ppm como máximo. De la gráfica anterior se puede sacar las siguientes conclusiones.

La primera y más sencilla de detectar es que el dispositivo que presenta una menor desviación en frecuencia es, como era de esperar, el SDR *FUNcube Dongle Pro+*, diseñado con un oscilador de cuarzo de alta precisión. De todos los receptores RTL-SDR, con cristal de cuarzo de bajo coste, destaca el Sintonizador Receptor USB DVB-T RTL-SDR Allwin Realtek RTL2832U y R820T PAL IEC Input de la figura 3.4. Este receptor fue adquirido por el grupo GranaSAT recientemente y a pesar de que su arquitectura Hardware es igual que el resto de RTL-SDR se obtiene una desviación en frecuencia mucho menor. Eso se debe a que el tiempo de trabajo total de los equipos más antiguos es muy superior y la sensibilidad de sus componente se ve afectada. De este modo, las condiciones de temperatura varían y provocan que el dispositivo se caliente más fácilmente. Otro factor implicado en la diferencia de la desviación en frecuencia para el mismo modelo de receptor son los componentes utilizados por el fabricante. La calidad de los componentes es proporcional a la calidad de los resultados.

La variación en la tolerancia del dispositivo es reducida, siendo menor en el SDR *FUNcube Dongle Pro+* debido a que está fabricado con componentes de alta precisión. Al igual que antes, el receptor RTL-SDR más antiguo tiene mayor oscilación debido a la calidad de sus componentes. A pesar de esto, se puede ver que un valor típico de oscilación para un RTL-SDR es de 3 ppm.

3.2 Consumo eléctrico del receptor

A continuación, se ha estudiado la curva de consumo eléctrico del dispositivo respecto al tiempo de trabajo. Para ello, se ha utilizado el sistema *Charger Doctor* como el de la figura 3.7, que no es más que un medidor de tensión y de corriente de dispositivos USB.

El consumo eléctrico es la cantidad de energía eléctrica que necesita un dispositivo para que funcione. En cambio, la potencia eléctrica es la proporción por unidad de tiempo con al cual la energía eléctrica se transfiere al dispositivo. Con el equipo *Charger Doctor* se puede calcular esta potencia ya que ofrece al usuario la tensión (V, en voltios) y la corriente (I, en amperios) en cada instante. Para ello se hace uso de la siguiente expresión:

$$P = V \cdot I \tag{3.2.1}$$

La resistencia equivalente (R, en Ohmios) del dispositivo es conocida y su valor es 0.05 Ohmios por lo que se puede obtener la potencia eléctrica del receptor SDR como:

$$P = R \cdot I^2 = \frac{V^2}{R} \tag{3.2.2}$$

Figura 3.7 – *Detector de voltaje y amperaje Charger Doctor.*

Se han tomado medidas del voltaje y amperaje dado por el equipo *Charger Doctor* cada 20 minutos durante 2 horas de trabajo del receptor para obtener el consumo eléctrico del dispositivo.

Modelo	Tiempo de trabajo						
SDR	Parámetro	20 min	40 min	60 min	80 min	100 min	120 min
Figura 3.2	Tensión (V)	5.15	5.15	5.16	5.16	5.16	5.15
Figura 3.2	Corriente (A)	0.29	0.29	0.29	0.29	0.29	0.29
Figura 3.2	Potencia eléctrica (W) 3.2.1	1.4935	1.4935	1.4964	1.4964	1.4964	1.4935
Figura 3.3	Tensión (V)	5.15	5.14	5.15	5.15	5.14	5.16
Figura 3.3	Corriente (A)	0.28	0.28	0.29	0.29	0.29	0.28
Figura 3.3	Potencia eléctrica (W) 3.2.1	1.442	1.4392	1.4935	1.4935	1.4906	1.4448
Figura 3.4	Tensión (V)	5.15	5.15	5.16	5.16	5.15	5.15
Figura 3.4	Corriente (A)	0.29	0.29	0.29	0.29	0.28	0.28
Figura 3.4	Potencia eléctrica (W) 3.2.1	1.4935	1.4935	1.4964	1.4964	1.442	1.442
Figura 3.5	Tensión (V)	5.14	5.16	5.17	5.19	5.18	5.15
Figura 3.5	Corriente (A)	0.17	0.17	0.17	0.18	0.17	0.17
Figura 3.5	Potencia eléctrica (W) 3.2.1	0.8738	0.8772	0.8789	0.9342	0.8806	0.8755

Tabla 3.3 – *Consumo eléctrico del SDR a lo largo del tiempo.*

A partir de estos datos se ha realizado la curva que describe el consumo eléctrico del dispositivo en función del tiempo de trabajo:

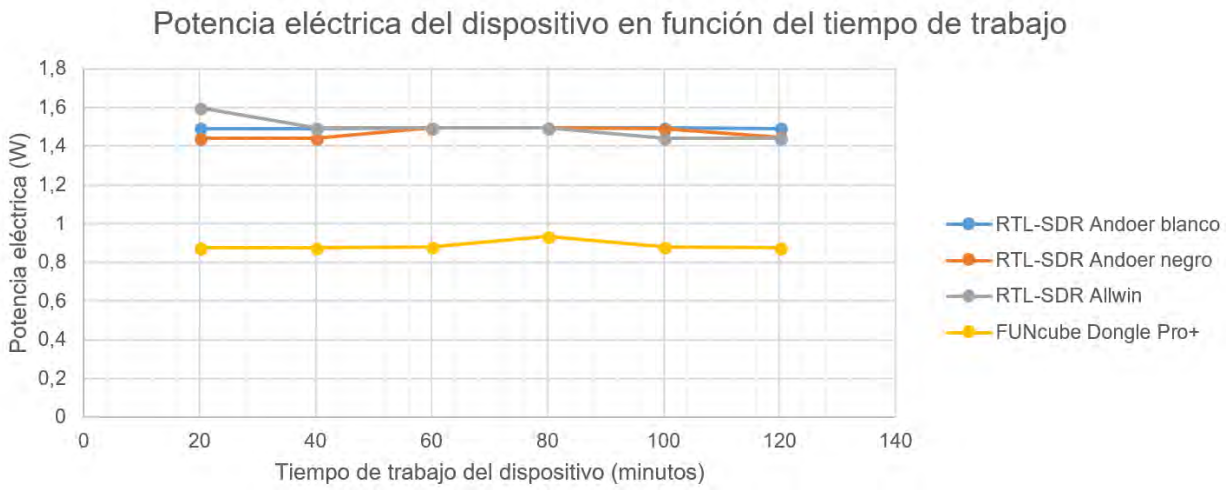


Figura 3.8 – *Consumo eléctrico del SDR a lo largo del tiempo.*

Observando esta curva se puede ver que el dispositivo SDR consumo mucho menos potencia eléctrica debido a la calidad de sus componentes. Para su funcionamiento no requiere tanta potencia como los equipos de bajo coste RTL-SDR puesto que las condiciones de trabajo de sus componentes son mejores. Dentro de este modelo de receptor SDR existen diferentes curvas de evolución del consumo eléctrico debido a, como se ha mencionado, el tiempo de uso del dispositivo y a la calidad de sus componentes de fabricación. Para observar mejor estas diferencias se ha realizado un zoom en la gráfica de la figura 3.8.

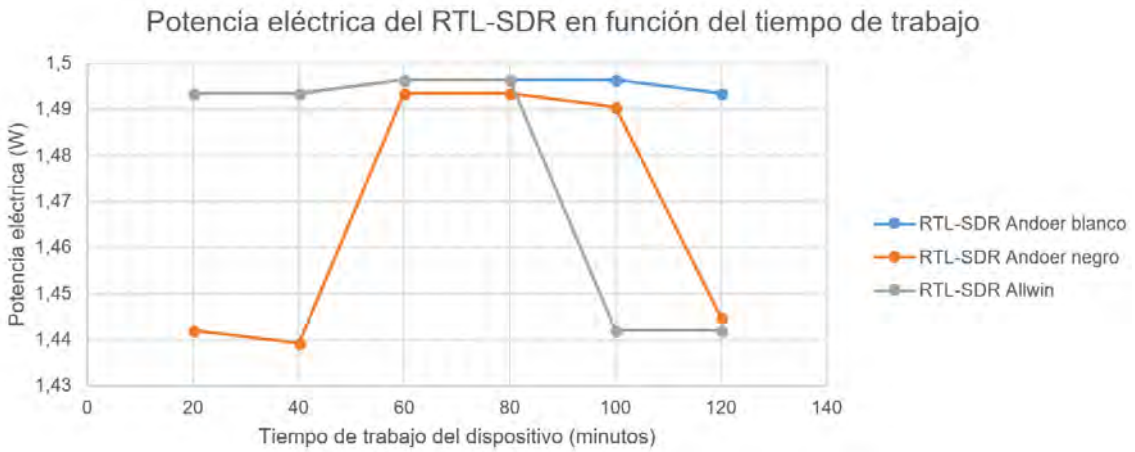


Figura 3.9 – *Consumo eléctrico del SDR a lo largo del tiempo.*

3

En esta gráfica se puede ver como a lo largo del tiempo el consumo eléctrico tiene una tendencia a disminuir. De todos los receptores de bajo coste, el dispositivo de la figura 3.4 es el que consigue reducir el consumo. Idealmente, la potencia eléctrica necesaria por un equipo para su funcionamiento debería ser constante en el tiempo y viendo los valores obtenidos se puede ver que esto se prácticamente se consigue ya que la variación máxima es del orden de la centésima para el RTL-SDR con peor comportamiento y de la milésima para el RTL-SDR con mejores prestaciones.

3.3 RTL-SDR vs. FUNcube Dongle Pro+

Para concluir la evaluación de los receptores SDR disponibles en el laboratorio del grupo GranaSAT se realiza una comparación entre las prestaciones de los dispositivos de bajo coste RTL-SDR y el receptor de alta precisión *FUNcube Dongle Pro+*.

En el estudio realizado en el presente capítulo puede verse fácilmente que el rendimiento general del dispositivo de alta precisión es mucho más alto. Esto hace que las interferencias a la hora de recibir señales en dicho receptor sean significativamente menores que las que tienen los equipos RTL-SDR. Dada una señal de RF débil, el *FUNcube Dongle Pro+* es capaz de recibirla sin problemas y en cambio un RTL-SDR recibiría dicha señal afectada por las interferencias producidas por la señales de frecuencias adyacente más fuertes. Además, es fácil ver que el receptor de alta precisión gana al de bajo coste en lo que respecta al consumo ya que requiere una potencia eléctrica mucho menor para su funcionamiento. En cuanto a la tolerancia del receptor, el *FUNcube Dongle Pro+* tiene menor desviación en frecuencia lo que hace que la sintonización en frecuencia sea mucho más precisa.

Desde el punto de vista Hardware, el receptor *FUNcube Dongle Pro+* cuenta con una arquitectura más sofisticada que ofrece mejores prestaciones como filtrados más estrechos que permiten recibir señales con menos interferencias o el uso de menos conversiones en frecuencia que debilitan la señal.

A cambio de todas estas ventajas el dispositivo *FUNcube Dongle Pro+* tiene un precio de mercado mucho más elevado por lo que su uso está menos extendido que el receptor RTL-SDR entre los usuarios que se inician en esta tecnología y, sobre todo, en el ámbito académico. Sin embargo, para las investigaciones interesa mucho más que las prestaciones sean buenas a cambio de un precio más alto a corto plazo. Ambos equipos tienen un coste a largo plazo muy bajo por lo que resultan muy útiles.

3

CAPÍTULO

4

FM COMERCIAL

Tal y como se ha adelantado, la codificación de las señales satelitates es muy similar a la empleada para transmitir el servicio RDS de la radio FM comercial. Es por ello que se detalla en el presente capítulo la radio FM comercial y sus técnicas de modulación y codificación empleadas de forma que se pueda comprender el algoritmo de demodulación implementado en este Trabajo Fin de Máster.

La radio FM comercial consiste en el envío de información como radioemisión a través de la modulación en frecuencia. La transmisión de esta información se realiza en lo que se denomina la banda de FM comercial y comprende las frecuencias de 88 MHz a 108 MHz. Debido a que la transmisión radio se realiza mediante la modulación en frecuencias e puede decir que se trata de una tecnología analógica puesto que este tipo de modulación lo es, como se verá en la sección 4.1. La radio FM comenzó a transmitirse en la banda de frecuencias mencionada a partir de los *años 60* y ha ido evolucionando a lo largo de los años desde el envío de sonido mono (radio convencional) hasta el estéreo (radio FM) o la transmisión de datos en la misma señal FM [21].

El rango de cobertura mencionado (88 MHz a 108 MHz) se divide asignando frecuencias separadas 200 kHz a cada estación FM, de forma que no se produzca interferencias entre las señales enviadas por estaciones consecutivas. La primera frecuencia central asignada es 88.1 MHz y teniendo en cuenta la separación entre señales de 200 kHz se tiene un total de 100 frecuencias disposibles para estaciones FM [11]. De este modo, cada estación FM puede transmitir a la frecuencia central asignada con un ancho de banda de 200 kHz. Teniendo en cuenta que las estaciones tienen una desviación máxima de su frecuencia central de 75 kHz, la banda de guarda es de 25 kHz. Esta banda de guarda es el rango de frecuencias

en el cuál no hay transmisión de señales de estaciones adyacentes, lo que ayuda a evitar las interferencias entre estas señales, denominado *aliasing*.

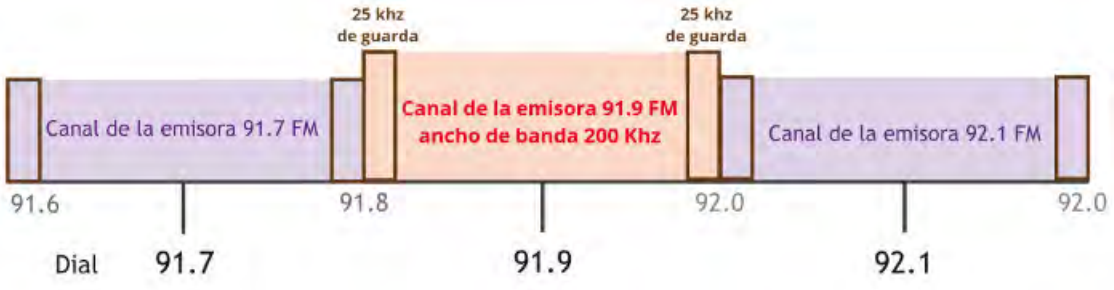


Figura 4.1 – *Kit que incluye el dispositivo SDR y una antena ADS-B.*

Originalmente, el sistema FM fue diseñado para el envío de señales de audio. Sin embargo, a finales de los *años 80* se descubrió la capacidad de transmisión de datos en estas señales lo que hizo que también se utilizara para el envío de los servicios de datos RDS, descritos en la sección 4.5 [21].

4

4.1 Modulación en frecuencia (FM)

La modulación es una técnica que consiste en la transmisión de una señal mensaje o información en una señal portadora, normalmente senoidal, mediante la variación de alguno de sus parámetros. Existen diferentes tipos de modulaciones, siendo de interés en el presente capítulo únicamente la modulación angular. Se caracteriza porque la señal mensaje se transmite variando la frecuencia o la fase de la portadora manteniendo el resto de parámetros constantes. La modulación angular se denomina como modulación de envolvente constante ya que la amplitud de la señal portadora no varía a lo largo del tiempo [49].

Antes de entrar en detalle en la modulación en frecuencia, en la siguiente clasificación se pueden ver las diferentes técnicas de modulación angular existentes hoy en día:

- *Modulación angular analógica*: de frecuencia FM y de fase PM.
- *Modulación angular digital*: por desplazamiento de frecuencia FSK y de fase PSK.

La modulación en frecuencia (FM) se trata de una modulación angular analógica en la que, como su propio nombre indica, la señal de información se transmite en las variaciones de frecuencia de la señal portadora, manteniéndose el resto de parámetros de la señal constantes [49]. La principal ventaja de esta modulación es que es muy inmune al ruido y a cambio resulta más compleja de modular y demodular.

La señal que contiene la información que se desea transmitir se denomina señal *modulante* puesto que será la que module los parámetros de la señal portadora obteniendo

como resultado la señal *modulada*. Ya se ha adelantado que la señal portadora, ecuación 4.1.1, se trata de una señal senoidal por lo que su expresión se define normalmente como una función senoidal generalizada. Se ha escogido una señal coseno porque su manejo matemático es más cómodo pero es arbitrario elegir un seno o un coseno.

$$x_c(t) = A_c sen\left(\omega t + \phi\right) = A_c cos\left[\phi(t)\right] \tag{4.1.1}$$

Donde A_c es la amplitud máxima de la señal, ω la frecuencia angular en *rad/s* y ϕ el ángulo de fase en *radianes*. Debido a que se trata de una modulación angular, la frecuencia y la fase serán instantáneas ya que varían en función de la señal modulante. De este modo, la frecuencia angular instantánea será:

$$\omega(t) = \omega_c + k_1 x_m(t) \tag{4.1.2}$$

Donde $x_c(t)$ es la señal de información o modulante. La frecuencia angular instantánea $\omega(t)$ y el ángulo instantáneo de fase $\phi(t)$ se relacionan mediante las expresiones 4.1.3 y 4.1.4.

$$\omega(t) = \frac{d\phi(t)}{dt} \tag{4.1.3}$$

$$\phi(t) = \int \omega(t)dt \tag{4.1.4}$$

La señal modulada se obtiene a partir de la señal portadora 4.1.1 y la señal mensaje $x_m(t)$ mediante una breve demostración matemática. Partiendo de la expresión de la señal portadora 4.1.1 y sustituyendo la expresión del ángulo instantáneo de fase 4.1.4 se obtiene la expresión de la señal modulada.

$$x_c(t) = A_c cos\left[\phi(t)\right] = A_c cos\left[\int \omega(t)dt\right] \tag{4.1.5}$$

$$\int \omega(t)dt = \int_0^t \left[\omega_c + x_m(t)\right] dt = \omega_c t + k_1 \int_0^t x_m(t)dt \tag{4.1.6}$$

Si se sustituye 4.1.6 en 4.1.5 se tiene la expresión de la señal modulada en frecuencia:

$$x_{FM}(t) = A_c \cos\left[\omega_c t + k_1 \int_0^t x_m(t)dt\right] \tag{4.1.7}$$

Para obtener una expresión más compacta de la señal modulada se define una señal mensaje o modulante de la forma:

$$x_m(t) = A_m \cos(\omega_m t) \tag{4.1.8}$$

De este modo, para la señal modulante definida se obtiene una señal modulada de la siguiente forma sustituyendo 4.1.8 en 4.1.7.

$$x_{FM}(t) = A_c \cos[\omega_c t + k_1 A_m sen(\omega_m t)] = A_c \cos[\omega_c t + \beta sen(\omega_m t)] \tag{4.1.9}$$

El término β es el índice de modulación de frecuencia e indica la profundidad de la modulación. Se calcula como la relación entre la desviación en frecuencia máxima y la frecuencia central ω_c [45].

$$\beta = \frac{\Delta\omega}{\omega_m} \tag{4.1.10}$$

En la expresión 4.1.9 se puede ver que la amplitud de la señal modulada en frecuencia A_c se mantiene constante y que las variaciones ocurren en su fase. La frecuencia de la señal modulada varía proporcionalmente a la amplitud de la señal mensaje A_m y la rapidez de la desviación de frecuencia lo hace con la frecuencia de la señal modulante ω_m. Todo esto puede verse de forma más clara en la representación gráfica de las señales.

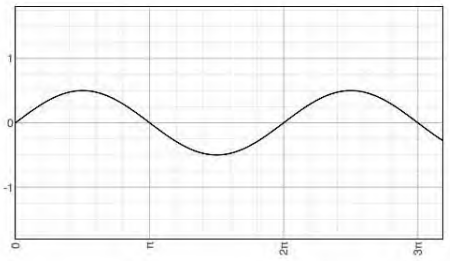


Figura 4.2 – *Señal modulante o de información* $x_m(t)$.

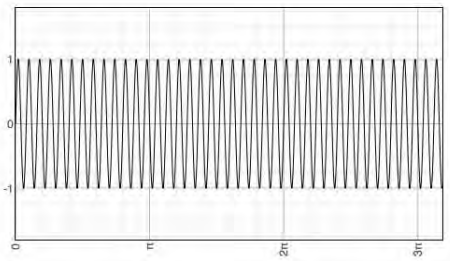


Figura 4.3 – *Señal portadora* $x_c(t)$.

En la figura 4.2 puede verse la señal cosenoidal con los datos que se desean transmitir. Para ello, se va a utilizar una señal portadora como la de la figura 4.3, cuya frecuencia

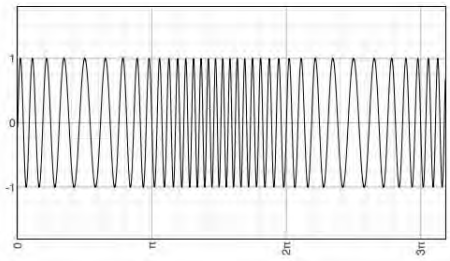


Figura 4.4 – *Señal modulada* $x_{FM}(t)$.

variará proporcionalmente a la información. De este modo, se obtiene la señal modulada en frecuencia, figura 4.4, como resultado de la modulación en frecuencia con un índice de modulación β. Es fácil comprobar que la amplitud se mantiene constante y la frecuencia varía a lo largo del tiempo.

En función del valor que toma este índice de modulación en frecuencia se tienen dos tipos de modulación: en banda estrecha NFM y en banda ancha WFM.

4.1.1 Narrow FM

Cuando el índice de modulación en frecuencia es mucho menor que 1 el ancho de banda necesario para transmitir la señal modulada FM depende únicamente de la frecuencia de la señal modulante $\omega_m = 2\pi f_m$.

La señal de información es una señal banda base cuya frecuencia es relativamente reducida, por lo que este tipo de modulación se denomina **FM de banda angosta**, puesto que el ancho de banda para transmitir la señal modulada es muy estrecho y se aproxima a la modulación en amplitud AM [26]. Este ancho de banda necesario para transmitir la señal modulada se define como:

$$\beta << 1 : B_{NFM} = 2f_m \tag{4.1.11}$$

4.1.2 Wide FM

Por el contrario, cuando el índice de modulación es superior a la unidad depende principalmente de la desviación en frecuencia y esto hace que el ancho de banda sea mucho más amplio. Esto es necesario ya que para cada frecuencia portadora ω_c, tras su modulación, aparecen varias componentes laterales separadas de la frecuencia de la portadora múltiplos de ω_m. En función del valor que tome el índice de modulación, habrá más o menos componentes espectrales laterales. De este modo, el ancho de banda de la señal modulada será:

$$\beta > 1 : B_{WFM} = 2\left(\beta + a\right) f_m \tag{4.1.12}$$

El término a es una constante con valor comprendido entre 1 y 2. La radiodifusión comercial de la FM emplea este tipo de modulación para transmitir sus señales y se tiene como parámetros típicos $a = 2$ y el índice de modulación $\beta = 5$ [26]. La expresión anterior se conoce como la *Regla de Carson* [26] [45] y proporciona el ancho de banda de la señal modulada con exactitud solo cuando $\beta >> 1$, esto es, en la modulación WFM.

En la figura 4.5 se puede ver que ahora el ancho de banda debe ser mayor ya que se transmiten más componentes en frecuencia: la componente central a la frecuencia de la portadora ω_c y las componentes laterales a $\omega_c + n\omega_m$ y $\omega_c - n\omega_m$ con $n = 1, 2, 3, ...$

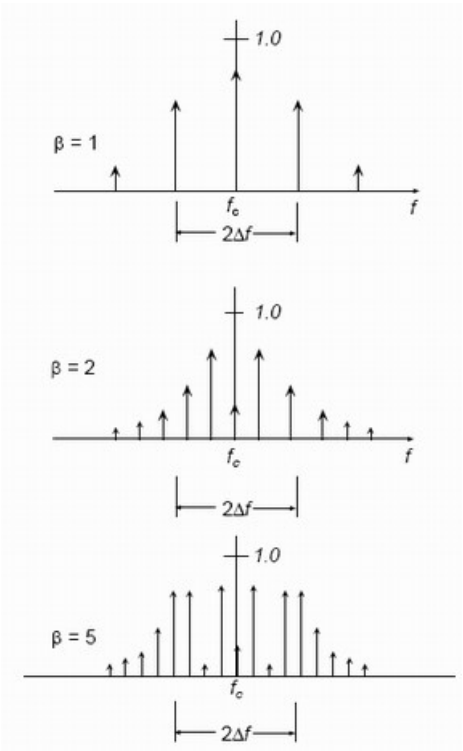


Figura 4.5 – *Espectro de la señal modulada FM de banda ancha o WFM.*

La principal ventaja de utilizar la modulación WFM es que el ruido es mucho menor. Esto ocurre ya que a medida que aumenta el índice de modulación β disminuye la potencia de la señal portadora concentrándose la máxima potencia en las bandas laterales que es donde se transmite la información [17].

4.2 Demodulación en frecuencia (FM)

El proceso contrario a la modulación es la demodulación y es el proceso en el cuál se recupera la información transportada por una señal portadora. Para poder demodular la señal es necesario conocer la modulación que se utilizó en el transmisor.

El proceso de demodulación en frecuencia es muy sencillo, basta con evaluar las variaciones de la fase de la señal recibida para obtener la información. Esto se puede realizar con diferentes técnicas, existiendo así distintos tipos de demoduladores FM. El dispositivo SDR utiliza un procesador digital de señal (DSP), explicado en el capítulo 2.

La forma más simple y rápida de implementar un demodulador FM se muestra en el diagrama de bloques de la figura 4.6.

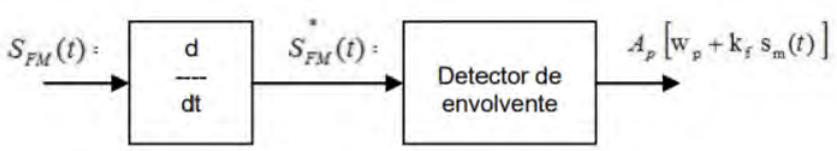


Figura 4.6 – *Diagrama de bloques funcionales de la demodulación en frecuencia.*

En primer lugar, la señal modulada en FM, 4.1.9, pasa por un diferenciador ideal que la deriva respecto al tiempo, obteniendo la expresión 4.2.1.

$$\frac{d}{dt}\left[A_c cos\left(\omega_c t + \beta sen(\omega_m t)\right)\right] = -A_c sen\left[\beta sen\left(\omega_m t\right) + \omega_c t\right]\left[\beta\omega_m cos\left(\omega_m y\right) + \omega_c\right] \quad (4.2.1)$$

Esta señal pasa por un detector de envolvente que no es más que un filtro paso baja que cancela las frecuencias elevadas. Se trata de un filtro de *Butterworth* de orden 4 y frecuencia de corte $f_c = 10 f_m$ para que únicamente pase la señal transmitida [58].

4.3 Señal mono FM

La radio FM comercial transmitía en sus inicios señales de audio monofónicas moduladas en frecuencia y emitidas con un único canal. El espectro de la señal FM mono es muy simple, sólo se transmite una señal en el rango de frecuencias de 30 Hz a 15 kHz, denominada señal *Suma* o *L+R*.

4.3.1 Receptor mono

De este modo, los receptores monofónico solo tenían que demodular la señal FM recibida para decodificar la señal *Suma*. Debido a que esta señal de audio (*L+R*) se transmite en banda base su decodificación es muy sencilla.

1. *Filtrado paso baja de la señal modulada FM.* La señal mono recibida se filtra con un filtro paso baja con frecuencia de corte 15 kHz para quedarse únicamente con la señal *Suma* y eliminar las componentes de frecuencia superiores, interpretadas como ruido

por el receptor.

2. *Filtrado de-énfasis.* En el transmisor se aplicó un filtrado de pre-énfasis a la señal de audio de los canales L y R antes de formar la señal *Suma* para reducir el ruido en la transmisión. La amplitud a frecuencias altas es más sensible al ruido por lo que se aumentó su amplitud en el transmisor para hacer más robusta la transmisión. De este modo, es necesario que en el receptor se reduzca la amplitud de estas componentes en frecuencia altas. Para el diseño de este filtro se emplea una constante de tiempo de 50 μs en Europa.

3. Por último, una vez que ha decodificado la señal de audio *Suma* es necesario que remuestrearla para poder reproducirla. La frecuencia de muestreo de audio es de 48 kHz para que sea audible por el oído humano.

4.4 Señal estéreo FM

Desde finales de los *años 50*, investigadores de Estados Unidos comenzaron el estudio del a emisión de señales de radio estéreo sobre una modulación en frecuencia. No es hasta 1961 cuando la FCC evalúa y aprueba el sistema de transmisión de señales de radio estéreo como el estándar para la radiodifusión estéreo en FM en Estados Unidos. Más tarde, en 1975 se produce la primera emisión de radio FM en estéreo [9].

El paso de la emisión monofónica a estereofónica no supuso un problema para aquellos usuarios con receptores mono puesto que la señal estéreo seguía emitiendo la señal *Suma*, denominada también *Señal Mono*. Esta señal mono es la suma de dos canales de audio L y R. Por el contrario, los receptores estéreo son capaces, como se verá a continuación, de obtener estos dos canales por separado mediante la recepción de otras componentes de audio en la señal estéreo [9].

Cada estación de radiodifusión FM comercial transmite una señal de audio estereofónica modulada en frecuencia. Esta señal, denominada *MPX* o *estéreo múltiplex*, tiene un ancho de banda de 100 kHz y unas características muy específicas. La radio FM estéreo, además de emitir las señales de audio necesarias para decodificar los canales L y R transmite datos con información sobre la emisora.

La señal estéreo múltiplex transmite la siguiente información, como puede verse en la figura 4.7:

- *Señal de audio Suma.* En el rango de frecuencias de 30 Hz a 15 kHz se transmite la **señal *Mono*, *Suma* o *canal L+R***. Esta señal se transmite en banda base y es la única parte de la señal *MPX* que decodifican los receptores mono.

- *Señal de audio Diferencia.* Desde los 23 kHz y hasta los 53 kHz se transmite la **señal *Resta* o canal *L-R***. Esta señal está centrada al doble de la portadora piloto (38 kHz), tiene un ancho de banda de 15 kHz y está modulada en DSBSC. Con esta modulación

se ahorra enviar la portadora que no contiene ningún tipo de información utilizándose una modulación con portadora suprimida. De este modo, se ahorra en energía emitida y se aprovecha más la potencia efectiva del emisor.

- *Piloto estéreo.* A 19 kHz se transmite la **portadora *piloto* o *piloto estéreo***. Como su propio nombre indica, solo se puede ver cuando el receptor es estéreo. Se trata de un tono que tiene la misma fase que la señal *Diferencia* y una amplitud del 10% de la amplitud total de la señal. Las funciones de la portadora piloto son: informar al receptor que la emisión es estéreo y regenerar las subportadoras de la señal *Diferencia* a 38 kHz y la señal *RDS* a 57 kHz.

- *Servicios de datos RDS.* Algo parecido a la transmisión de la señal *Diferencia* ocurre con los **servicios de datos RDS**, que se transmiten modulados con DSBSC a 57 kHz. Estos servicios contienen información digital sobre la emisora y otros aspectos de interés para el usuario y su codificación la dicta el estándar *"Specification of the radio data system RDS for VHF/FM sound broadcasting in the frequency rango from 87,5 to 108,0 MHz"* [31].

- *Servicios SCA.* A 67 kHz se modulan los **servicios SCA**, orientados al sector profesional y que no se pueden sintonizar con radios convencionales. Algunos ejemplos de estos servicios son: la emisión de hilos musicales de pago, enlaces de radio o telemetría.

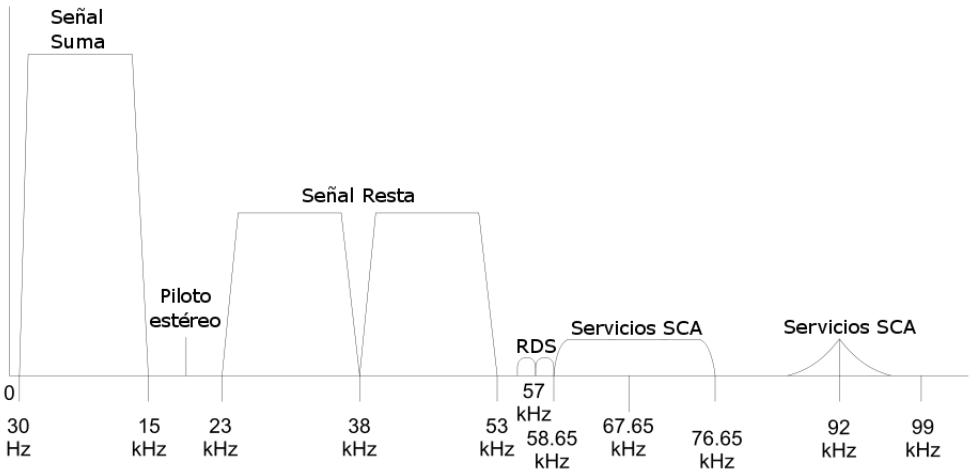


Figura 4.7 – *Espectro de la señal estéreo múltiplex.*

4.4.1 Receptor estéreo

El receptor FM estéreo demodula en frecuencia la señal recibida y es capaz de decodificar la señal tanto si es mono o estéreo. Si se trata de una emisión mono actúa como un receptor

monofónico decodificando la señal de audio *Suma* o *L+R*. Cuando la emisión es estéreo el proceso de decodificación es más complejo [9].

1. En primer lugar, el receptor comprueba si existe la portadora piloto a 19 kHz. Si recibe este tono, sabe que la señal recibida es estéreo y que debe activar el circuito decodificador correspondiente. En ese caso, filtra la señal portadora piloto con un filtro paso banda centrado a 19 kHz y con una banda pasante muy estrecha.

2. Decodifica la señal *Suma* (*L+R*). Para ello, filtra la señal recibida eliminando las componentes a frecuencia superiores a 15 kHz y le aplica el de-énfasis, de forma análoga a como se realiza en el receptor mono.

3. A continuación, decodifica la señal *Diferencia* (*L-R*). Esta señal se filtra paso banda con un filtro centrado al doble de la portadora piloto (38 kHz) y una banda pasante de 30 kHz. cuando la señal *Diferencia* es filtrada se tiene un espectro con información en el rango de 23 kHz a 53 kHz. Esta señal se traslada a banda base para poder decodificarla. Para ello, se implementa un oscilador que genera un tono al doble de la portadora (38 kHz), que mediante un mezclador trasladará la señal *L-R* a banda base. A la señal resultante se le aplica el de-énfasis explicado en 4.3.1.

4. Por último se combinan las señales *Suma* y *Diferencia* para obtener los canales L y R originales. Este proceso es muy sencillo, basta con combinar las señales como se indica en 4.4.1 y 4.4.2.

$$(L+R)+(L-R)=2L \rightarrow L=\frac{(L+R)+(L-R)}{2} \tag{4.4.1}$$

$$(L+R)-(L-R)=2R \rightarrow R=\frac{(L+R)-(L-R)}{2} \tag{4.4.2}$$

   Al igual que ocurría en el receptor mono, es necesario remuestrear las señales de audio de los canales L y R a 48 kHz para que sea audible por el oído humano.

Tras este proceso, descrito en la figura 4.8, se ha obtenido la señal de audio FM recibida. Si además, el receptor estéreo permite mostrar la información del servicio RDS, en caso de que se esté transmitiendo, el proceso de decodificación continuará. Antes de explicarlo, es necesario profundizar en el concepto SDR.

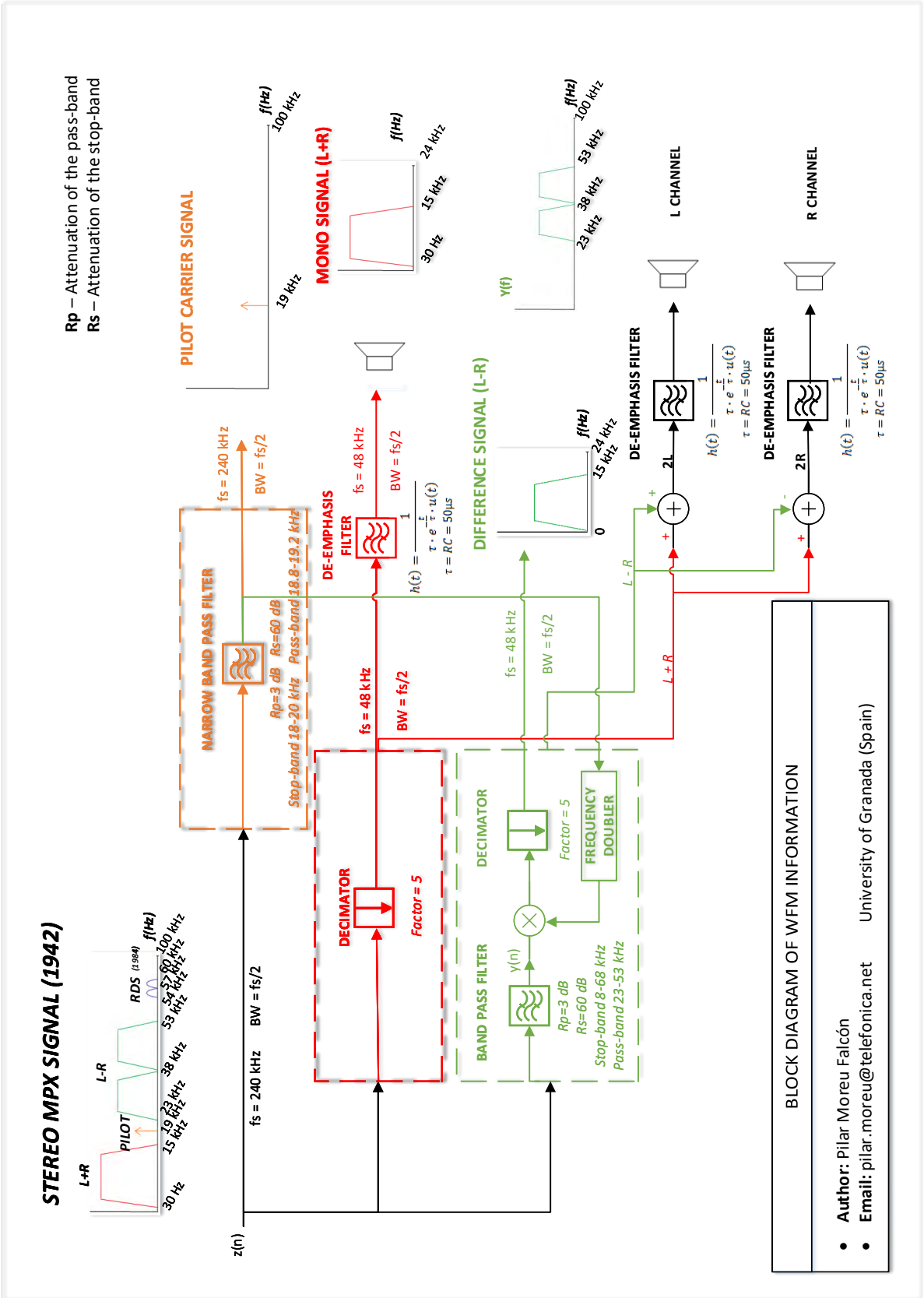


Figura 4.8 – *Diagrama de bloques del receptor FM estéreo.*

4.5 Servicio RDS

El servicio de datos RDS es un sistema de transmisión de información de las emisores en sus canales de emisión regular que no afecta a la calidad del audio transmitido [51]. Se trata de una transmisión de datos digital junto a la señal de radio FM que da al usuario información muy variada como: identificación de la emisora, frecuencias alternativas de la emisora, frecuencias de emisoras hermanas, información sobre los programas emitidos, radiotexto, servicio de buscapersonas, telecontrol,etc.

La ventaja de usar la radiodifusión FM comercial para enviar estos datos es que la información llega a un gran número de usuarios gracias a la amplia cobertura de la red de emisoras FM. Además, el coste de estas emisoras es mínimo [51].

El sistema RDS utiliza una modulación en subportadora de 57 kHz sincronizada en cuadratura con el tercer armónico de la portadora piloto de la señal FM estéreo (19 kHz).La señal RDS tiene un ancho de banda del 3% de la señal de audio completa, éste es así de reducido para evitar interferencias que degraden la calidad de audio de la señal [51].

4.5.1 Codificación RDS

Tal y como se indica en el estándar europeo [31], la información se modula con un codificador binario a 1187.5 bps, tasa relacionada con la portadora piloto mediante la expresión $19kHz/48 = 1187.5$ bps. Esta velocidad de transmisión de datos proporciona la compatibilidad y robustez necesaria para las aplicaciones actuales de este servicio de datos [51].

La información RDS se codifica en dos etapas: codificador diferencial y codificador bifase. Utilizar el codificador diferencial resulta de utilidad ya que permite recuperar los 0 y 1 en el receptor a pesar de que la señal llegue invertida. En la siguiente tabla se puede ver la codificación diferencial BPSK puesto que se transmiten dos símbolos de un bit (0 o 1).

Bit actual	Bit anterior	Bit codificado
0	0	0
0	1	1
1	0	1
1	1	0

Tabla 4.1 – *Codificación BPSK diferencial.*

La codificación bifase permite la sincronización de los datos mediante una señal de reloj con una frecuencia el doble de la señla de reloj de transmisión $1187.5 \cdot 2 = 2375$ Hz [51] [31]. En la figura 4.9 puede verse que se transmite una señal positiva seguida de una negativa cuando el símbolo vale 1 y viceversa cuando vale 0.

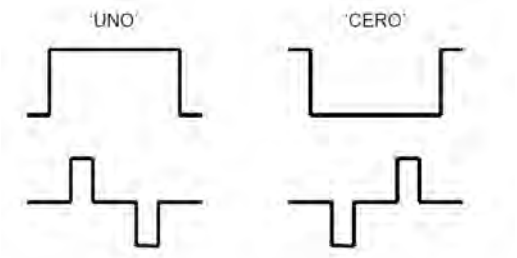


Figura 4.9 – *Codificación bifase de los símbolos 1 y 0.*

Una vez codificados los datos, los símbolos se filtran para conformar la señal que será enviada en la señal estéreo múltiplex. Se trata de un filtro conformador cuya función de transferencia corresponde con un pulso raíz coseno remontado como el de la figura 4.10 [31].

$$H_T = \begin{cases} cos\frac{\pi f t_d}{4} \rightarrow 0 \leq f \leq 2/t_d \\ 0 \rightarrow f > 2/t_d \end{cases} \tag{4.5.1}$$

Donde $t_d = \frac{1}{1187.5}$ segundos.

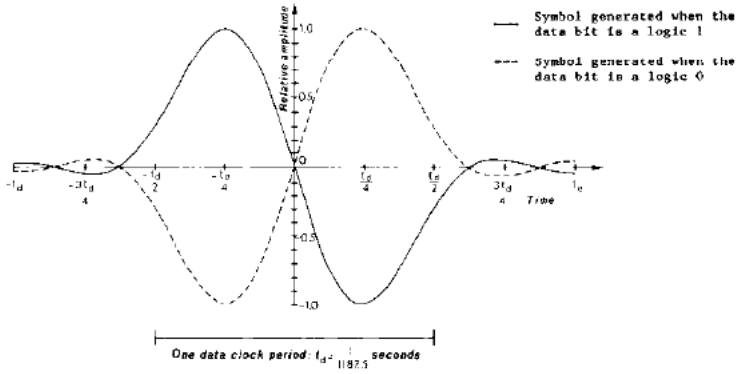


Figura 4.10 – *Pulso conformador: raíz coseno remontado.*

La señal resultante de este filtrado será enviada junto con las señales de audio FM.

4.5.2 Decodificación RDS

La señal RDS va codificada digitalmente y se transmite modulada sobre una senoidal de 57 kHz utilizando un esquema BPSK Diferencial donde cada símbolo corresponde con un bit. Por tanto, para decodificar los servicios RDS, el receptor estéreo de radio genera un tono al triple de la portadora piloto (57 kHz) y realiza un proceso análogo a la recepción de la señal *Diferencia*.

1. *Filtrado paso banda de la señal RDS.* En primer lugar, la señal estéreo múltiplex es filtrado con un filtro paso banda centrado a 57 kHz y con una banda pasante de 6 kHz.

2. *Filtrado de-énfasis.* En el transmisor se aplicó un filtrado de pre-énfasis a la señal para reducir el ruido en la transmisión. La amplitud a frecuencias altas es más sensible al ruido por lo que se aumentó su amplitud en el transmisor para hacer más robusta la transmisión. De este modo, es necesario que en el receptor se reduzca la amplitud de estas componentes en frecuencia altas. Para el diseño de este filtro se emplea una constante de tiempo de 50 μs en Europa.

Una vez que la señal de 57 kHz es demodulada de forma síncrona, se obtiene la señal bifase filtrada. Ya se ha mencionado que la información que se transmite en esta señal es digital por lo que su decodificación es más compleja. Este proceso de decodificación deshace cada uno de las técnicas de codificación descritas en la sección 4.5.1.

1. *Descomposición de la señal en su componente en fase y cuadratura.* Los símbolos de la señal RDS se transmiten en cuadratura de la forma:

$$x_{RDS}(t) = x_I(t) \cdot cos\,(\omega_c t) + x_Q(t) \cdot sen\,(\omega_c t) \tag{4.5.2}$$

Donde $x_I(t)$ y $x_Q(t)$ son los símbolos en fase y cuadratura respectivamente, ω_c es la frecuencia angular de la señal RDS $\omega_c = 2\pi \cdot 57e3$. Los símbolos de la constelación BPSK Diferencial se definen como 4.5.4 y **??**. Para descomponer la señal en cuadratura se utilizan dos mezcladores que combinan la señal bifase filtrada con un coseno y un seno a 57 kHz para obtener la componente en fase y en cuadratura, respectivamente.

$$x_I(t) = a(i)p(t) \tag{4.5.3}$$

$$x_Q(t) = b(i)p(t) \tag{4.5.4}$$

El término $p(t)$ es el pulso conformador y $a(i)$ y $b(i)$ son las secuencia de símbolos transmitidas. Este pulso conformador se trata de un pulso raíz coseno remontado y conforma la secuencia de símbolos.

2. *Obtención de la secuencia de símbolos transmitida.* Dada la señal en cuadratura, se utiliza un filtro cuya función de transferencia sea la inversa de un pulso raíz coseno remontado para deshacer la conformación de los símbolos BPSK Diferencial y obtener la secuencia de símbolos recibida.

3. *Conversión Continua/Discreta.* En este punto se tiene una señal continua con los símbolos transmitidos, que debe convertirse en una secuencia discreta de símbolos que formen la constelación BPSK Diferencial. Para ello se almacena un símbolo cada periodo de muestras y se descartan los demás. Este periodo de muestras se obtiene

como el producto de la frecuencia de muestreo y el periodo de bit 1/1187.5. A partir de esa secuencia discreta de símbolos se genera la constelación de símbolos como $a(i) + jb(i)$.

4. *Decodificación BPSK Diferencial.* Los bits se transmiten con la diferencia de símbolos, cuya duración es de 48 ciclos. Esta modulación digital, descrita en la tabla 4.1, indica que la recepción de dos símbolos iguales corresponde con el bit 0 y la recepción de símbolos distintos con el bit 1. Para decodificar basta con rotar la constelación para pasar de BPSK Diferencial y BPSK.

5. *Decodificación BPSK.* A continuación, el decodificador decida qué símbolo corresponde con el bit 1 y qué símbolo corresponde con el bit 0 para obtener el flujo binario o *bit stream* recibido.

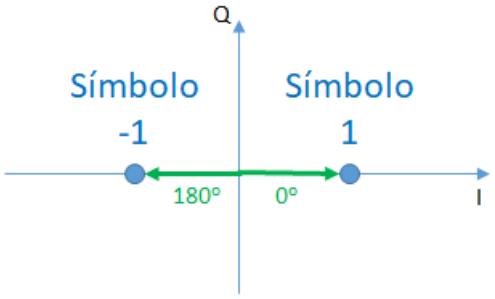


Figura 4.11 – *Constelación BPSK.*

4.5.3 Formato de datos

A partir del flujo binario recibido se obtiene la información codificada. El estándar europeo *"Specification of the radio data system RDS for VHF/FM sound broadcasting in the frequency rango from 87,5 to 108,0 MHz"* [31] especifica el formato de los datos RDS transmitidos.

Los datos son transmitidos en paquetes de 104 bits denominados *grupos*. A su vez, cada grupo está formado por cuatro *bloques* de 26 bits que contienen la palabra de información (16 primeros bits) y la palabra de comprobación (10 bits restantes) para la corrección de errores y la alineación de trama [51] [31].

El sistema RDS divide los grupos en dos versiones (A y B) de 16 tipos, destinados a una aplicación en particular. Esta división, mostrada en la tabla 4.2, permite la flexibilidad en la utilización de este sistema [51]. El tipo y versión de grupo se envía en el segundo bloque de cada grupo, como muestra la figura 4.13.

- El *tipo de grupo* se codifica en los 4 primeros bits.
- La *versión de grupo* se transmite en el quinto bit del bloque. La versión será *A* si este bit vale 0 y *B* si vale 1.

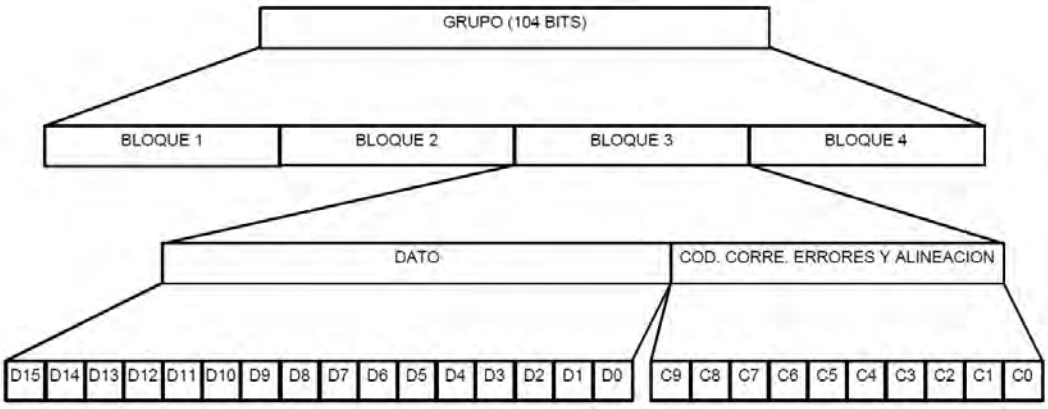


Figura 4.12 – *Paquete de datos del sistema RDS.*

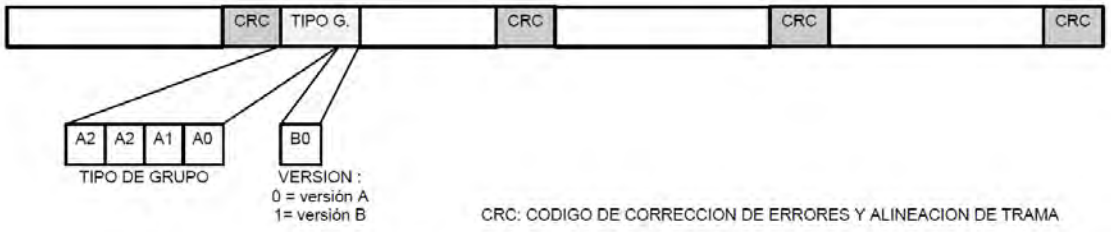


Figura 4.13 – *Tipo de grupo y versión del sistema RDS.*

Grupo	Versión A					Versión B				
ID	**A3**	**A2**	**A1**	**A0**	**Aplicación**	**A3**	**A2**	**A1**	**A0**	**Aplicación**
0	0	0	0	0	Información básica de sintonía	0	0	0	0	Información básica de sintonía
1	0	0	0	1	Información del programa	0	0	0	1	Información del programa
2	0	0	1	0	Radiotexto	0	0	1	0	Radiotexto
3	0	0	1	1	Información de otras redes	0	0	1	1	Información de otras redes
4	0	1	0	0	Hora y Fecha	0	1	0	0	-
5	0	1	0	1	Canales transparentes de datos	0	1	0	1	Canales transparentes de datos
6	0	1	1	0	Aplicaciones de la emisora	0	1	1	0	Aplicaciones de la emisora
7	0	1	1	1	Buscapersonas	0	1	1	1	-
8	1	0	0	0	-	1	0	0	0	-
9	1	0	0	1	-	1	0	0	1	-
10	1	0	1	0	-	1	0	1	0	-
11	1	0	1	1	-	1	0	1	1	-
12	1	1	0	0	-	1	1	0	0	-
13	1	1	0	1	-	1	1	0	1	-
14	1	1	1	0	Soporte ampliado de otras redes	1	1	1	0	Soporte ampliado de otras redes
15	1	1	1	1	-	1	1	1	1	Información de sintonía rápida

Tabla 4.2 – *Tipos de grupo y versión del sistema RDS.*

Además del tipo y versión de grupo se hay otros parámetros fijos que se transmiten siempre como el código identificador de programa PI, la información de tráfico TP o el tipo de programa que se emite PTY [51] [31].

- *Código identificador de programa PI.* El código identificador del programa se transmite en el primer bloque de todos los tipos de grupo de versión A. En los tipos de grupo de versión B además se transmite en el bloque 3. Son 16 bits que contienen el código con el que se identifica la emisora. Está compuesto por el código de país (4 bits), el código de cobertura (4 bits) y el código de identificación de programa (8 bits). La información contenida en dicho código se codifica en hexadecimal.

  – *Código de país.* Diferentes países comparten un mismo código de país por lo que es necesario transmitir el código de país extendido (ECC). Se transmite en el tercer bloque de los grupos de tipo 1A en los 8 últimos bits ($m7...m0$) de los 16 bits de información. Estos bits se agrupan de 4 en 4 obteniendo los dos caracteres que identifican el país.

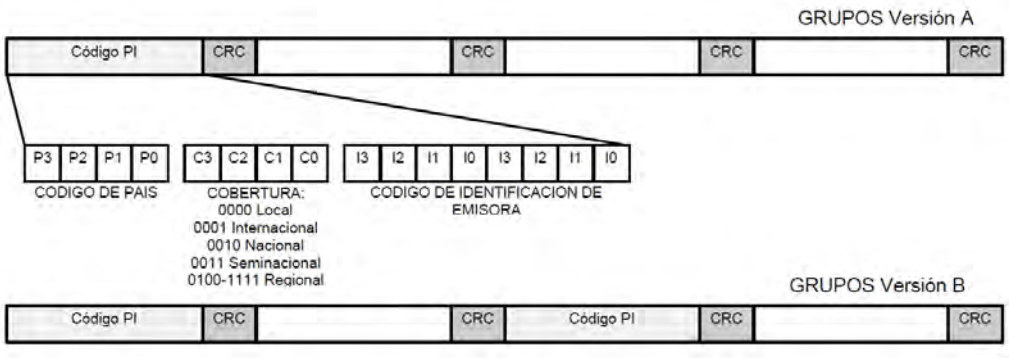


Figura 4.14 – *Código de identificación de emisora PI.*

  – *Código de cobertura*: local, internacional, nacional, seminacional o regional.

  – *Número de referencia del programa.* Está formado por dos valores de 4 bits cada uno y son asignados por en cada país para identificar la emisora..

- *Información de tráfico TP.* El código que identifica si el programa contiene anuncios de tráfico se transmite en el segundo bloque de todos los tipos de grupo, versión A y B. Para saber de qué tipo de anuncios se trata se transmite en el bloque 2 de los grupos 0A, 0B, 14B y 15B un bit denominado TA. Combinando ambos bits se puede saber la aplicación del programa de tráfico. El TP se transmite en el bit 10 del segundo bloque de todos los grupos y el TA en el bit 4 del segundo bloque de los grupos 0A, 0B y 15B y en el bit 3 del segundo bloque del grupo 14B.

- *Tipo de programa que se emite PTY.* El código que identifica el tipo de programa se transmite en el segundo bloque de todos los tipos de grupo, tanto versión A como B. En concreto, se transmite en el rango de bits desde el 9 al 5. Se trata de 5 bits que indican que tipo de programa se transmite en la señal de las 32 posibilidades.

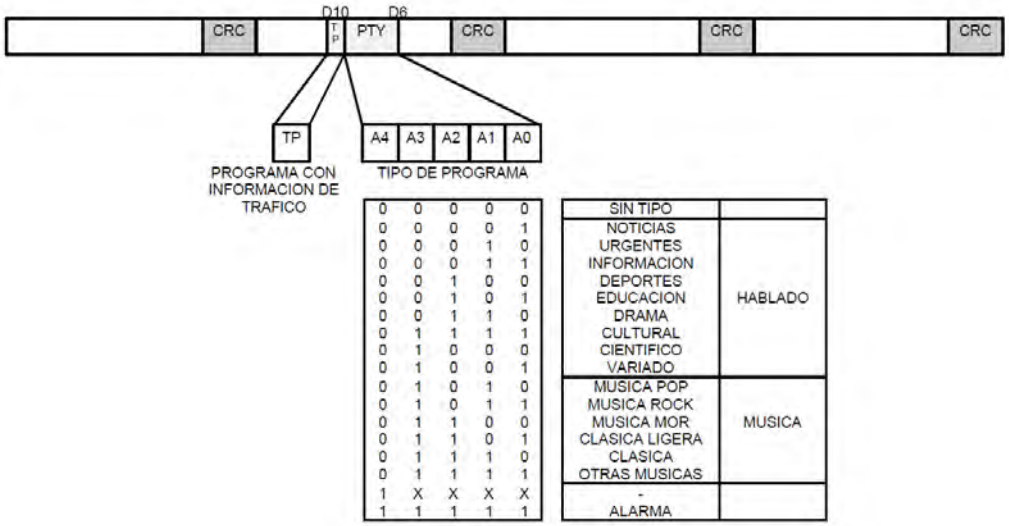


Figura 4.15 – *Tipo de programa PTY e información de tráfico TP.*

El resto de bits contienen otra información que variará en función del tipo de grupo y versión y si la emisora desea transmitir o no información adicional.

CAPÍTULO

5

COMUNICACIONES SATELITATES

Recientemente, el número de aplicaciones que utiliza la comunicación de paquetes de radio ha crecido exponencialmente. Estas aplicaciones incluyen las transmisiones de APRS puesto que permiten la transmisión de datos desde una fuente a varios destinatarios de forma simultánea. Actualmente, se ha estudiado la posibilidad de transmitir estos datos en aplicaciones de *Radio Amateur* mejorando considerablemente la eficiencia de la transmisión puesto que se tienen tasas de transferencia de hasta 9600 Baudios [42].

Una de las aplicaciones más populares es la transmisión de señales satelitales. Hoy en día, hay más de 100 satélites transmitiendo en la banda de frecuencias de RF. La mayoría de ellos transmiten información de telemetría mediante modos digitales y pertenecen a grupos de radioaficionados o entidades educativas y privadas, siendo sus misiones principalmente educativas, científicas y experimentales [40]. A pesar de esto también se transmite voz digital e imágenes tomadas por una cámara a bordo en el satélite.

Las transmisiones de radioaficionado no usan cifrado por lo que cualquier usuario es capaz de decodificar los satélites que transmiten en dicha banda de frecuencias [40]. Estas bandas de frecuencia son:

- *Banda HF*. Rango de frecuencias comprendido entre 3 MHz y 30 MHz también conocido como onda corta. Esta banda se utiliza para transmitir las señales de las emisoras de radio internacionales y radioaficionados ya que por su propagación en línea recta tienen un alcance muy lejano.
- *Banda VHF*. Ocupa el rango de frecuencias desde 30 MHz a 300 MHz y se utiliza en

sistemas de TV, radiodifusión FM, banda aérea y comunicaciones de control de tráfico marítimo.

- *Banda UHF*. Banda del espectro electromagnético que ocupa las frecuencias desde los 300 MHz a los 3 GHz. Esta banda se utiliza par la transmisión de señales de televisión, de radioaficionados y de telefonía móvil.

Gran parte de estas transmisiones se realizan bajo el protocolo AX.25 puesto que se trata de transmisiones de pequeñas cantidades de datos, como es el caso de la telemetría, audios e imágenes. Esta telemetría digital se transmite mediante el Sistema Automático de Informes de Paquetes (APRS) radio que utiliza el protocolo de transmisión AX.25.

En sus inicios, para realizar las comunicaciones satelitales con el sistema APRS era imprescindible disponer de un módem o *TNC* para radiopaquete. Sin embargo, hoy en día existen versiones que permiten incorporar otros módems, tarjetas y cualquier equipo de VHF como es el caso de los dispositivos basados en la tecnología Software-Defined Radio. De este modo, con el receptor SDR y un ordenador basta para recibir APRS puesto que todo el proceso de decodificación lo realiza el Software.

5.1 Sistema APRS

APRS es un Sistema Automático de Informes de Paquetes que permite al usuario conocer la posición en la que está una estación de radioaficionado (móvil o fija), la información meteorológica, señalización de eventos de interés para el radioaficionado o el telemando. Además, su propósito es el intercambio de información entre múltiples estaciones en un área muy amplia de forma simultánea.

El investigador *Bob Bruninga (WB4APR)* registró la marca APRS en los años 90 con una licencia que permite a cualquier radioaficionado utilizar este sistema siempre que no tenga fines comerciales. Se desarrolló con la idea de permitir el seguimiento de equipos a través del sistema GPS y las comunicaciones digitales con *Radio Amateur* permitiendo la monitorización de posiciones en formato de datos digitales. El primer programa que hacía uso de APRS se implementó en 1984 por este ingeniero investigador para el seguimiento de una carrera de caballos de larga distancia. En las comunicaciones satelitales, además de seguir la posición del satélite se utiliza para monitorizar la telemetría que éste transmite.

Este protocolo de comunicaciones se trata de un sistema de red digital de seguimiento de RF con nodos ubicados por todo el mundo diseñado por los operadores de radioaficionados. Su funcionamiento es muy sencillo, una estación transmite mensajes en la banda de RF y los nodos, implementados como repetidores digitales y denominados *digipeaters*, retransmiten esta señal para que sea alcanzable por cualquier receptor de RF [20]. Hoy en día, el número de estaciones APRS o nodos es muy elevado, lo que permite que la transmisión de información sea mucho más completa y precisa.

El sistema APRS utiliza el protocolo AX.25 para realizar la transmisión de los datos. La

entidad encargada de liderar la evolución de este protocolo es *Tucson Amateur Packet Radio Corporation* (TAPR). Se trata de una asociación americana especializada en las comunicaciones digitales que ha desarrollado la especificación *"AX.25 Link Access Protocol for Amateur Packet Radio"* [30], descritos en la sección 5.2.

Los elementos habituales para transmitir APRS operan desde sus inicios a 1200 baudios pero la creciente necesidad de enviar grandes cantidades de datos ha hecho que esta capacidad de transmisión se quede corta. Es por ello que se implementan en la actualidad sistemas APRS que operan a 9600 baudios. Como se ha mencionado, APRS utiliza como medio de transmisión el protocolo AX.25, cuya frecuencia de trabajo típica en Europa es de 144.8 MHz a una velocidad de 1200 baudios.

5.2 Protocolo AX.25

El protocolo AX.25 o *Packet Radio* es un protocolo diseñado por la *Organización Mundial de Estándares* ISO para la transmisión de datos en RF. Se trata de una variante del estándar X.25 enfocada a la radiofrecuencia. Este estándar describe la comunicación entre las estaciones radio mediante la técnica de acceso múltiple al medio CSMA. De este modo, cada estación espera hasta encontrar un canal libre antes de transmitir los datos digitales [42]. La principal diferencia entre estos dos estándares es que el X.25 se utiliza para transmitir datos a través de las líneas de teléfono y AX.25 en RF [54] permitiendo la transmisión de datos simultánea por diferentes estaciones radio . Su principal aplicación es el envío de datos por los canales de audio de los equipos de RF por parte de los radioaficionados en comunicaciones satélite. Como cabe esperar las bandas de frecuencia en las que se puede utilizar este protocolo son las bandas de radioaficionado HF, VHF y UHF.

La ISO define diferentes niveles de protocolos de comunicación en su modelo de capas OSI: de aplicación, de presentación, de sesión, de transporte, de red, de enlace de datos y físico. El protocolo AX.25 forma parte de las dos primeras capas de este modelo, el nivel físico y de enlace de datos, dedicadas a la conexión y verificación de los datos [54].

En la primera capa del modelo OSI se define la técnica de transmisión al medio de la información empleada por el protocolo. Se trata de la modulación digital AFSK, en la que la información se transmite mediante dos tonos de la forma 5.2.1, uno correspondiente al bit 0 a f_i 1200 Hz y otro correspondiente al bit 1 a f_i 2200 Hz. Se utiliza la modulación en frecuencia digital AFSK por su robustez frente a las distorsiones y el ruido en la transmisión radio. A cambio de esto, la señal transmitida se verá afectada por el efecto Doppler. Se describirá en más detalle esta modulación en la sección 5.3 del presente capítulo.

$$x_{tono_i}(t) = Acos\,(2\pi f_i) \tag{5.2.1}$$

A continuación se encuentra la capa de enlace de datos, encargada del direccionamiento físico, del acceso al medio de la estación radio, de la detección de errores y de la formación

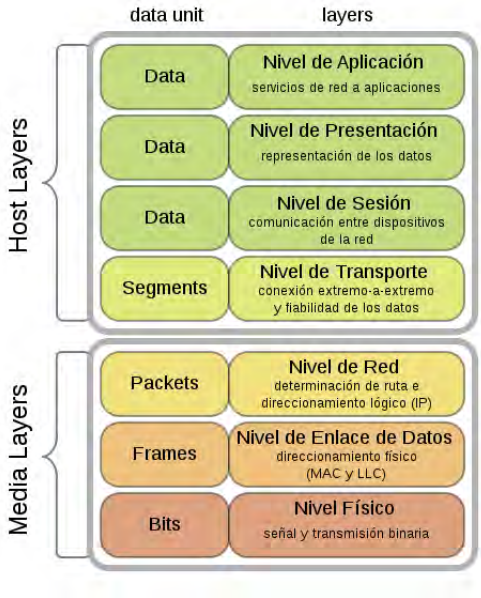


Figura 5.1 – *Modelo de capas OSI.*

de la trama de datos mediante la técnica de codificación necesaria para que la transferencia de información sea segura. Los paquetes radio de transmisión AX.25 están compuestos por bloques de información, denominados tramas, caracterizados por diferentes campos. Todos ellos se describen en la sección 5.6 del capítulo.

5

5.3 Modulación AFSK

La modulación por desplazamiento de frecuencia en audio AFSK es una técnica que modula la señal digital para su transmisión mediante dos tonos de audio, de ahí su nombre. Transforma la señal digital en una señal de audio mediante estos tonos y a continuación la modula en frecuencia (NFM) para obtener la señal de RF que se transmite. De este modo, la señal modulada en AFSK viaja por el espectro radioeléctrico como una señal modulada en FM cualquiera pero en lugar de contener audio de la radio FM comercial contiene información digital [54]. La información digital es transmitida como una señal de audio modulada en frecuencia, los detalles de la modulación FM se describieron en el anterior capítulo. La principal ventaja de trabajar con señales de audio es que se puede utilizar cualquier dispositivo transceptor con una entrada de audio para su transmisión [54].

La señal portadora modulada en FM típicamente se transmite a 144.8 MHz en Europa y transmite un sonido modulado por desplazamiento en frecuencia. Esta señal de información previamente es modulada en AFSK mediante dos tonos que indican el nivel bajo y alto de señal de audio. Se trata de una modulación digital en la que los estados se denominan marca y espacio y corresponden con el tono de 1200 Hz y 2200 Hz, respectivamente. Estos estados serán equivalente a los bit 0 (espacio) y 1 (marca) y operan a una tasa de bits de 1200 bps. De este modo, la forma de onda de la señal modulada en AFSK será como la de la figura 5.2 y se define como:

$$x_{AFSK}(t) = A_c cos\left[2\pi\left(f_c + x_m(t)\Delta f\right)t\right] \tag{5.3.1}$$

Donde A_c y f_c son la amplitud y frecuencia de la señal portadora, x_m es la señal de información binaria y Δf es la desviación máxima en frecuencia definida como:

$$\Delta f = \frac{|f_{marca} - f_{espacio}|}{2} \tag{5.3.2}$$

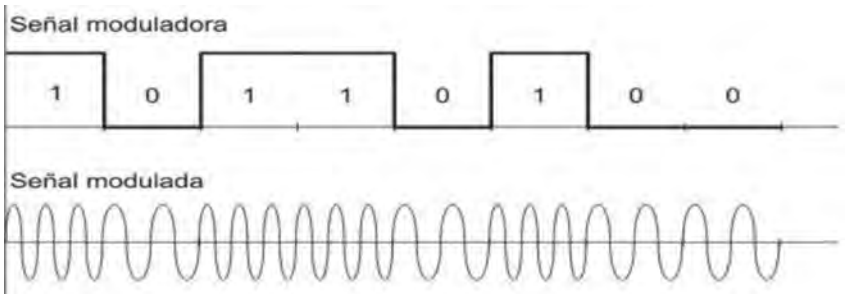


Figura 5.2 – *Señal de información (arriba) y señal modulada en AFSK (abajo).*

De este modo, la salida del modulador AFSK estará relacionada con la señal de información (binaria) a través de la marca y el espacio como puede verse en la componente $x_m(t)\Delta f$ de la expresión 5.3.1 y en 5.3.2. En ella puede verse como un 0 lógico corresponde a la frecuencia del espacio $f_{espacio}$ y un 1 lógico a la frecuencia de marca f_{marca}. En la figura 5.2 además puede verse que el cambio de frecuencia de un tono a otro es continuo para evitar que la señal esté desfasada puesto que los dispositivos receptores no están diseñados para señales desfasadas.

Esta modulación se realiza a diferentes frecuencias, correspondientes con dos señales senoidales pulsadas (expresión 5.2.1). Es por ello que el espectro de la señal modulada en AFSK centra toda la energía en los máximos del espectro de potencia, ubicados a las frecuencias 1200 Hz y 2200 Hz. Este hecho implica que el ancho de banda mínimo de la señal modulada AFSK se aproxime a:

$$B_{AFSK} = 2\left(\Delta f + f_c\right) \tag{5.3.3}$$

5.4 Demodulación AFSK

La demodulación AFSK se encarga de deshacer la modulación de la portadora radio mediante una señal de audio modulada en FSK. Tal y como se ha visto, la señal está compuesta por dos frecuencias correspondientes a los estados marca y espacio. El receptor no conoce la correspondencia entre estos dos estados y los bits 0 y 1 hasta que no demodula la señal.

Debido al filtrado del canal cuando se transmite la la señal AFSK, ésta sufre una variación en la amplitud. Tal y como se observa en la figura 5.3 la señal tendrá una amplitud superior para el tono a 1200 Hz que para el tono a 2200 Hz. Por ello, la demodulación FSK puede aproximarse a una demodulación AM, mucho más fácil de implementar.

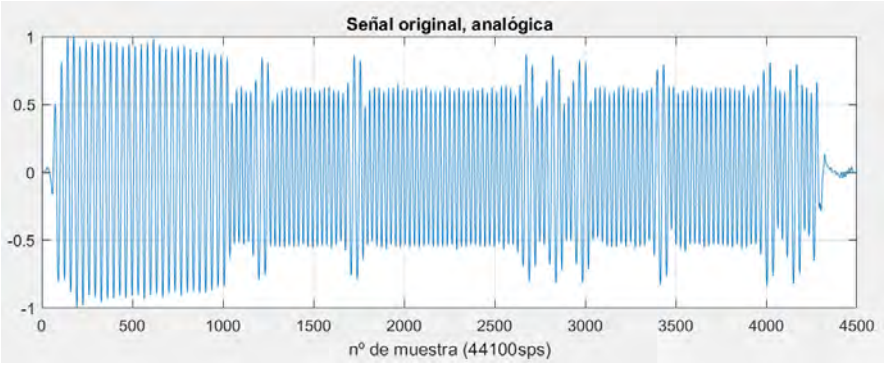


Figura 5.3 – *Señal modulada en AFSK.*

En primer lugar, el receptor demodula en amplitud (AM) la señal recibida mediante un rectificador de señal y un filtro paso baja para obtener la envolvente de la señal. A continuación, el decodificar decide que el nivel alto corresponde con la marca puesto que se trata del tono a 1200 Hz y el nivel bajo con el espacio (2200 Hz). De esta forma la frecuencia más alta será la más atenuada y por ello se le asigna el 0 lógico. Esta decodificación recibe el nombre de decodificación NRZ-I [6].

5

5.5 Codificación NRZ-I

La información de la trama modulada en AFSK previamente es codificada con el código NRZ-I, caracterizada por codificar un cambio de señal cuando el bit de información es 0. La tabla 5.1 muestra un ejemplo de codificación de los datos binarios de información con códigos NRZ-I.

Dado que no existe una señal de reloj para sincronizar el envío y recepción de los bits de información y puesto que el medio de transmisión es el aire existe una posibilidad muy alta de que l señal se vea afectaba por el ruido. Por ello, el protocolo AX.25 emplea la codificación NRZ-I para el envío de las tramas de datos. Al transmitirse la información en los cambios de estado de la señal, la transmisión será mucho más robusta puesto que si varios bits seguidos se ven afectados por el ruido no se sufrirá la información ya que lo importante es el cambio de estado de la señal y no si se trata de un 1 o 0 lógico.

Dato a transmitir	0	0	1	1	0
Dato codificado NRZ-I	1	0	0	0	1
Frecuencia del tono (Hz)	1200	2200	2200	2200	1200

Tabla 5.1 – *Ejemplo de codificación NRZ-I.*

5.6 Tramas de datos

La información que se envía a través del protocolo AX.25 y el sistema APRS se agrupa en paquetes de transmisión radio. Estos bloques de datos se denominan *tramas* y pueden ser de diferente tipo según los datos que contienen: de información, de supervisión o sin numeración. Todos ellos comparten la misma estructura compuesta por diversos grupos de datos denominados *campos* [30].

5.6.1 Trama de información

Las tramas de información (I) contienen los datos que una estación desea transmitir. Su estructura se muestra en la figura 5.4 y se explica en profundidad en la sección 5.6.4.

5.6.2 Trama de supervisión

Las tramas de supervisión (S) también se denominan tramas de control puesto que realizan el control del canal de comunicación por el que se transmitirán las tramas de información. Algunos ejemplo de estas tramas son la notificación de recepción de una trama I o la solicitud de su recepción. Este tipo de tramas se envían debido a que la técnica acceso al medio utilizada por el protocolo AX.25 es CSMA.

En la técnica CSMA se supervisa el canal de transmisión cuando se necesita transmitir para comprobar si está libre o hay alguna estación transmitiendo. Para ello, se utiliza una trama de este tipo. Si el canal está libre la estación que ha supervisado el canal comienza su transmisión y el resto de estaciones escuchan hasta que finalice la transmisión. Si por cualquier motivo, dos estaciones deciden transmitir simultáneamente se produce una colisión y tras un tiempo de espera aleatorio, cada estación retransmite la trama de información.

Por todo ello, las tramas de supervisión se envía cuando se requiere el acuse de recibo por parte del receptor o la retransmisión de la trama de información tras una colisión.

5.6.3 Trama no numeradas

Por último, se denominan las tramas no numeradas (U) a aquellas que mantienen el control sobre la capa de enlace como el establecimiento y la finalización del enlace de conexión de cada estación o la transmisión y recepción de la información, entre otros [42]. El formato de estas tramas es el mismo que en el caso anterior y se muestra en la figura **??**.

5.6.4 Formato de trama

Los tres tipos de tramas de datos AX.25 tienen una estructura de datos similar, compuesta por los siguientes campos:

Flag	Address	Control	Info	FCS	Flag
01111110	112/224 Bits	8/16 Bits	N*8 Bits	16 Bits	01111110

Figura 5.4 – *Formato de la trama de información AX.25 [42].*

- *Bandera o flag (F).* Este primer campo indica el comienzo y final de una trama de datos AX.25 y tiene una longitud de 1 byte. Dos tramas de datos consecutivas pueden compartir una bandera, indicando la finalización de la primera trama y el comienzo de la siguiente.

  Toma un valor fijo: *0111 1110* o *07* en hexadecimal. Este valor es único para identificar con claridad cada trama de datos. De esta forma, en toda la trama, indiferentemente del tipo que sea, no se podrá encontrar una secuencia de 6 bits 1 consecutivos. Para ello se realiza la técnica de *inserción de ceros* o *bit stuffing* en la codificación de la trama. Esta técnica consiste en añadir un 0 lógico cada vez que se encuentren 5 bits 1 consecutivos en cualquier parte de la trama para así evitar que el próximo bit sea un 1 lógico y se confunda con la secuencia de la bandera [30].

MSB	FLAG en binario						LSB
0	1	1	1	1	1	1	0

Figura 5.5 – *Campo flag o bandera de la trama de datos AX.25 [53].*

- *Dirección (A).* El campo de dirección, con una longitud variable entre 14 y 70 bytes, es común a los tres tipos de tramas AX.25 y transmite los identificadores de la estación emisora, receptora y de los repetidores.

  Los primero 8 bytes contienen el identificador del *destinatario* y se compone de dos subcampos: el *indicativo* y el *SSID*. El *indicativo* está formado por los 6 primeros bytes del campo de dirección y por ello está compuesto como máximo de 6 caracteres, cada uno de ellos codificado sobre 7 bits ya que que el octavo bit siempre vale 0. El *SSID* del destino se transmite en el séptimo byte del campo de dirección y tiene la siguiente estructura:

  – El primer bit denominado *H* vale 0 para indicar que se trata del destinatario. En el caso de indicativos de repetición, vale 1 si la trama ha sido repetida por el digipeater y 0 si no se ha repetido.

  – Los siguientes dos bits se denominan *R* y actualmente están reservados para un uso local o futuro por lo que no se presta atención a ellos.

- La secuencia de bits del 4 al 7 se dedican al *SSID*. Estos cuatro bits pueden codificar valores decimales entre 0 y 15.
- El último bit se denomina *bit de extensión* y siempre toma el valor 0 lógico para el destinatario y el 1 para la fuente o si se trata de un indicativo de repetición.

Los bytes del 8 al 14 se dedican al identificador de la *fuente* y su estructura es análoga a la del destinatario. Los bytes restantes (del 15 al 70) pueden transmitirse o no. Si se transmiten, en ellos se codifican los identificadores de los distintos *repetidores digipeaters*. Debido a la variabilidad de la longitud del campo de dirección es necesario tener un bit que indique el fin de dicho campo. Éste será el último bit del SSID y tomará el valor 1 lógico para indicar el fin del campo de direcciones [30].

Destination Address Subfield	Source Address Subfield
A1 A2 A3 A4 A5 A6 A7	A8 A9 A10 A11 A12 A13 A14

Figura 5.6 – *Campo de dirección de la trama de datos AX.25 de longitud mínima [42].*

- *Control (C)*. El campo de control, de 1 byte de longitud, se encarga de identificar el tipo de trama que se transmite y por lo tanto tomará un valor distinto en función del tipo de trama AX.25. A pesar de emplear la técnica de acceso al medio CSMA el sistema APRS no requiere el envío del acuse de recibo por el receptor. Por este motivo, mediante este byte se controla la transmisión de la trama de información. Este campo toma el valor *000P0011*, donde P es el bit que indica si se requiere ($P = 1$) o no ($P = 0$) el acuse de recibo por parte del receptor. Tal y como indica el estándar [30] el bit $P = 1$ puesto que no se envía el acuse de recibo y por tanto, el campo de control tendrá el valor fijo *00000011* [54].

  Para la trama de información (I), se incluye un segundo byte con el identificador de protocolo *(PID)* y especifica el protocolo de la capa de red que se utiliza en la transmisión de lo datos AX.25: TCP/IP, TEXNET, ARPA, ... Sin embargo, el sistema APRS no utiliza un protocolo en la capa de enlace por lo que este identificador indicará *"sin protocolo"*, correspondiente con el valor *11110000*. En definitiva, el campo de control en las tramas de información tendrá el siguiente formato:

Control Field	Protocol ID
00000011	11110000

Figura 5.7 – *Campo de control y PID de la trama de datos AX.25 [54].*

- *Información (I)*. El campo de información contiene los datos transmitidos por la trama codificados en 256 bytes. Se trata de los datos APRS que contienen información de posicionamiento, datos meteorológicos, telemetría o anuncios, entre otros. Además de estos datos APRS puede contener comentarios en forma de mensaje que informan al

usuario sobre la estación emisora o dan datos adicionales APRS como la velocidad o la altitud del satélite [30].

Campo de información tipo				
Campo	ID de tipo de datos	Datos APRS	Extención de los datos APRS	Comentarios
Bytes	1	N	7	N

Figura 5.8 – *Campo de información o I de la trama de datos AX.25 [53].*

- *Verificación de trama (FCS).* El campo FCS se utiliza para la detección de errores en la transmisión y tiene una longitud de 2 bytes, un byte correspondiente con las tramas de la fuente y otro con las del destinatario. Con este campo, el sistema receptor comprueba si el medio de transmisión ha corrompido la trama de datos AX.25.

  La comprobación de errores se realiza según la *Recomendación ISO 3309* y consiste en dividir el número binario formado por los bits enviados bit a bit entre el polinomio de comprobación CRC-CCITT de 16 bits $x^{16}+x^{15}+x^2+1$ (*1100 000 000 0101*). El resto resultante de esta operación será el número utilizado como FCS [30].

  El proceso de comprobación de la trama es el siguiente:

  1. Parte con el campo FCS igual a *11111111 11111111* o *0xFFFF* en hexadecimal.
  2. Cada vez que envía un bit de datos, se realiza un desplazamiento los bits del campo FCS hacía la derecha. Para realizar esta operación se tiene en cuenta el valor del bit enviado.

     Si el bit de datos es distinto al bit desplazado (señalado en negrita) se realiza la operación XOR entre el FCS actual y el valor hexadecimal *0x8408* (*10000100 00001000*). Por el contrario, si los bits son iguales no se realiza ninguna operación.
  3. Cuando se envía todos los bits se calcula el *complemento uno* del valor FCS. Para ello, los bits de valor 1 lógico toman el valor 0 y viceversa.

  El resultado de este proceso es el campo FCS enviado en la trama de datos AX.25 [54].

CAPÍTULO

6

RECEPTOR WFM

Una vez introducidos los conceptos teóricos, el presente capítulo comienza la parte práctica del Trabajo Fin de Máster con el diseño e implementación del algoritmo de demodulación de señales moduladas en frecuencia de la radio FM comercial. Tal y como se ha adelantado, el estudio del demodulador de estas señales es necesario como paso intermedio a la implementación del decodificador de señales satelitales AX.25. Esto se debe a que la radio FM comercial transmite información digital correspondiente a los servicios RDS en la señal de audio transmitida a RF. La codificación digital de esta información se realiza de forma similar a la codificación de las tramas de datos AX.25.

Para el desarrollo del algoritmo se han utilizado dos receptores SDR: RTL-SDR fabricado por *Allwin* y el dispositivo *FUNcube Dongle Pro+*. Estos equipos han sido escogidos teniendo en cuenta la evaluación de su comportamiento realizada en el capítulo 3 del proyecto. De todo los dispositivos de bajo coste se elige el receptor RTL-SDR *Allwin* puesto que presenta una desviación en frecuencia mucho menor que el resto de equipos, de 24 ppm. Por otro lado, se han realizado las pruebas del algoritmo diseñado en el receptor *FUNcube Dongle Pro+* puesto que su tolerancia es mínima ya que está diseñado con un cristal de cuarzo de alta precisión, 1.5 ppm.

Respecto a la herramienta Software utilizada para el diseño del algoritmo de decodificación se ha escogido el Software MatLab puesto que se trata de la herramienta de Software matemático por excelencia para el procesamiento digital de señales. Incorpora gran cantidad de funciones que permiten el análisis, procesado y extracción de características de señales con muy buenas prestaciones.

En primer lugar se describirán las diferentes señales utilizadas en la implementación y testeo del algoritmo así como la técnica empleada para recibirlas con el SDR. A continuación, la sección **??** describe el proceso seguido por la alumna para la implementación del algoritmo de decodificación en el Software MatLab y su evaluación.

6.1 Señales WFM

A lo largo del capítulo se trabajará con los siguientes tipos de señales: señales de entrenamiento, señales de evaluación y señales reales.

Señales de entrenamiento

Se utilizan para el diseño del algoritmo y se trata de señales de radio FM comercial recibidas en condiciones de ruido mínimo y con un dispositivo RTL-SDR de alta precisión. Han sido proporcionadas por el profesor Isaac Manuel Álvarez Ruíz del Departamento de Teoría de la Señal, Telemática y Comunicaciones de la Universidad de Granada tras concertar una visita con la alumna de acuerdo a lo hablado con el tutor del TFM en una de las reuniones.

Esta señal ha sido recibida con ayuda del paquete de soporte de comunicaciones para RTL-SDR Radio del Software MatLab y el complemento *Simulink*. Se utiliza el ejemplo de MatLab denominado *Sepctrum Analysis with RTL-SDR Radio*, que implementa el modelo de un receptor RTL-SDR como el de la figura 6.2. Fijada la frecuencia de sintonización, el bloque *RTL-SDR* recibe datos del dispositivo RTL-SDR conectado al PC y da como salida un vector que contiene las muestras de la señal recibida [25].

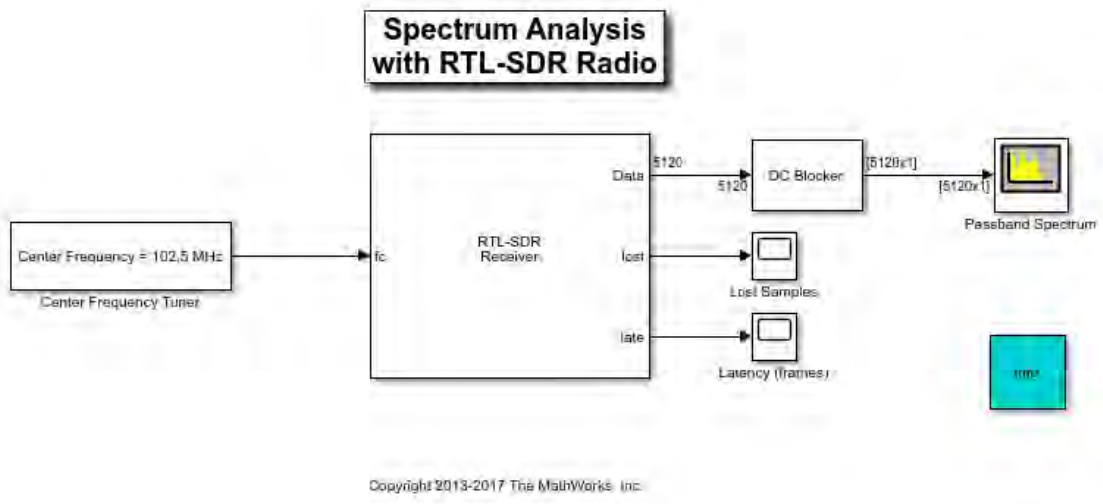


Figura 6.1 – *Modelo de receptor de MatLab.*

La señal resultante se almacena en formato *.mat* puesto que se ha almacenado la variable que da como salida el modelo de *Simulink* de la figura 6.2. En concreto la alumna ha

trabajado con dos señales de entrenamiento, recibidas de las emisoras de radio que emiten más información digital en el RDS: *Cadena 100* a 88.2 MHz y *Ondacero* a 92 MHz. La señal recibida está definida a partir de sus componentes en fase y en cuadratura mediante la expresión compleja 6.1.1 y ha sido grabada con una frecuencia de muestreo de 200 kHz durante 7.38 segundos para la emisora a 88.2 MHz y 3.62 segundos para la emisora a 92 MHz.

$$x(n) = x_I(n) + jx_Q(n) \tag{6.1.1}$$

En la figura 6.2 se puede observar que la señal modulada WFM recibida a 88.2 MHz se recibe en banda base y tiene un ancho de banda de 100 kHz, correspondiente a la mitad de la frecuencia de muestreo.

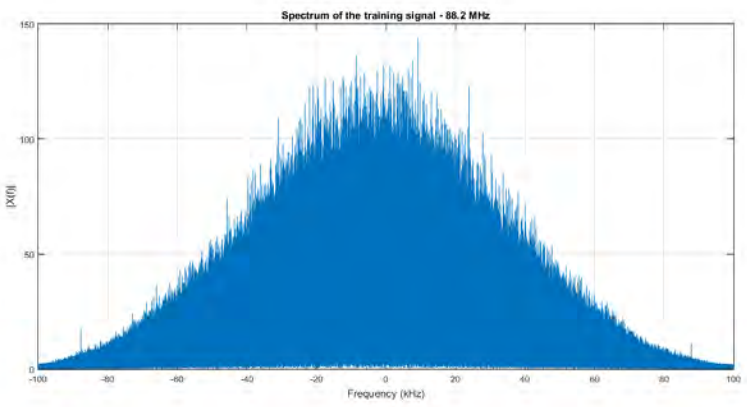


Figura 6.2 – *Espectro de la señal modulada en frecuencia WFM recibida por el modulo de la figura 6.2.*

Señales de evaluación

Las señales de este tipo se utilizan para testear el funcionamiento del demodulador en frecuencia diseñado. Se generan en el Software MatLab de forma teórica atentiendo a las expresiones descritas en el capítulo 4. Las señales portadora y modulante teóricas se modulan con la función de MatLab *fmmod* [10] y se demodulan con el algoritmo implementado por la alumna para su evaluación. Este proceso de generación de la señal modulada en FM de forma teórica es muy sencillo.

En primer lugar, se generan las señales modulante y portadora como se muestra en el código de MatLab de la figura 6.4. La señal que contiene la información es un tono a 1 kHz definido por la expresión 6.1.2 y la señal portadora, dada por 6.1.3, tiene una frecuencia de 90 MHz, ambas señales con una amplitud unitaria. Se ha fijado una duración de estas señales de 5 ms y su frecuencia de muestreo se ha escogido de tal manera que se cumpla el *Teorema de Nyquist*: $f_s \geq 2f_c$ donde f_c es la frecuencia de la portadora de 90 MHz. Se ha escogido una frecuencia de muestreo de 225 MHz, superior al límite frecuencial impuesto por

este Teorema (180 MHz).

$$x_m(t) = A_m cos\left(\omega_m t + \phi_m\right) \tag{6.1.2}$$

$$x_c(t) = A_c cos\left(\omega_c t + \phi_c\right) \tag{6.1.3}$$

```
fprintf('\n- MESSAGE SIGNAL:\n');
Am_char=input('What is the amplitude of the message signal? Am (V) = ','s');
Am=str2num(Am_char);
fm_char=input('What is the frequency of the message signal? fm (Hz) = ','s');
fm=str2num(fm_char);
phim_char=input('What is the phase of the message signal? phi_m (rad) = ','s');
phim=str2num(phim_char);

fprintf('\n- CARRIER SIGNAL:\n');
Ac_char=input('What is the amplitude of the carrier signal? Ac (V) = ','s');
Ac=str2num(Ac_char);
fc_char=input('What is the frequency of the carrier signal? (RF: 88 - 108 MHz) fc (Hz) = ','s');
fc=str2num(fc_char);
phic_char=input('What is the phase of the carrier signal? phi_c (rad) = ','s');
phic=str2num(phic_char);

tmax=5e-3;          % Time required to collect N samples (s)
fs=2.5*fc;          % Sampling frequency (Hz). Nyquist Theorem.
N=tmax*fs;          % Signal length (Number of samples)
f_div=(N-1)/tmax;   % Time division to collect N samples (Hz)
t=0:1/f_div:tmax;   % Time axis (s)

% MESSAGE SIGNAL
wm=2*pi*fm;             % rad/s
x_m=Am*cos(wm*t+phim);  % Message signal

% CARRIER SIGNAL
wc=2*pi*fc;             % rad/s
x_c=Ac*cos(wc*t+phic);  % Carrier signal
```

Figura 6.3 – *Código fuente en MatLab que genera las señales modulante y portadora.*

6

A continuación se modula la señal con ayuda de la función de MatLab *fmmod* y para un índice de modulación $\beta = 5$ puesto que se desea modular en WFM. Esta función toma como entrada la señal de información $x_m(t)$, su frecuencia central f_m, la frecuencia de la portadora f_c y la desviación máxima de la frecuencia instantánea $\triangle f$. Comprueba que se cumple el *Teorema de Nyquist* y modula la señal mensaje con dichos parámetros implementando los conceptos descritos en el capítulo 4 [10].

```
fprintf('\n- FREQUENCY MODULATION PARAMETERS:\n');
beta_char=input('What is the modulation index? (beta<1=NFM / beta>1=WFM) beta = ','s');
beta=str2num(beta_char);

% MODULATED SIGNAL
kw=beta*wm/Am;      % Frequency deviation
delta_f=kw*Am;      % Maximun frequency deviation
B_FM=200e3;         % Band width

x_fm=fmmod(x_m,fc,fs,delta_f);
```

Figura 6.4 – *Código fuente en MatLab que modula en frecuencia la señal portadora.*

Con esta macro de MatLab se han generado diferentes señales moduladas en frecuencia variando únicamente el parámetro f_c, correspondiente a la frecuencia de la señal portadora, dentro del rango de 88 MHz a 108 MHz. La señal resultante se trata de una señal paso banda cuyo espectro se muestra en la figura 6.7. Esta señal tiene un ancho de banda igual a la frecuencia de muestreo puesto que se trata de una señal paso banda.

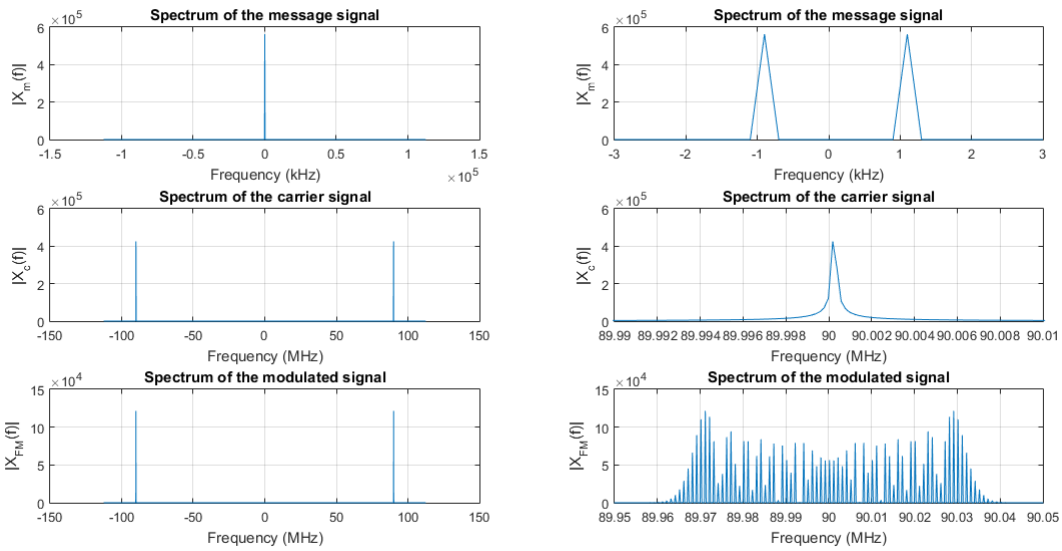


Figura 6.5 – *Representación espectral de la señal modulante (arriba), portadora (centro) y modulada FM (abajo).*

Señales reales

Por último, se utilizan *señales reales* recibidas con los receptores escogidos por la alumna: RTL-SDR de *Allwin* y *FUNcube Dongle Pro+*. Estas señales se pueden capturar a través de distintos métodos: macro del Software MatLab, librería del dispositivo *librtl*, el Software SDRSharp, ... Todas ellas se reciben en las mismas condiciones de trabajo, en el laboratorio del grupo GranaSAT, con una antena terrestre de entrada externa para la recepción en RF como la de la figura 6.6.

Figura 6.6 – *Antena terrestre para la recepción de señales de RF.*

La forma más sencilla de capturar una señal de RF es con el comando *rtl_sdr* de la librería del receptor SDR *librtl*. El comando captura la señal de radiofrecuencia recibida a la frecuencia de sintonización f_c con una frecuencia de muestreo f_s y una ganancia de *"RF gain"* dB y la almacena en un fichero *.bin*.

rtl_sdr -s "fs" -f "fc" -g "RF gain" "captured_signal.bin"

A modo de ejemplo, se muestra la captación de la señal recibida a la frecuencia 90 MHz (emisora *Ondacero*) con una frecuencia de muestreo de 200 kHz y una ganancia de 42 dB para recibir la señal con la suficiente potencia con el dispositivo RTL-SDR de *Allwin*. Tras situarse dentro del directorio de la librería *librtl*, se ejecuta desde la terminal el siguiente comando: *rtl_sdr -s 200000 -f 92.0 -g 42 fm920.bin*.

En la siguiente figura se puede ver como la señal es recibida en banda base y por ello tiene un ancho de banda igual a la mitad de la frecuencia de muestreo de la señal. Si se observa este espectro de potencia comparándolo con el de la figura 6.7 puede verse que la amplitud es mucho más reducida y se la señal es más ruidosa.

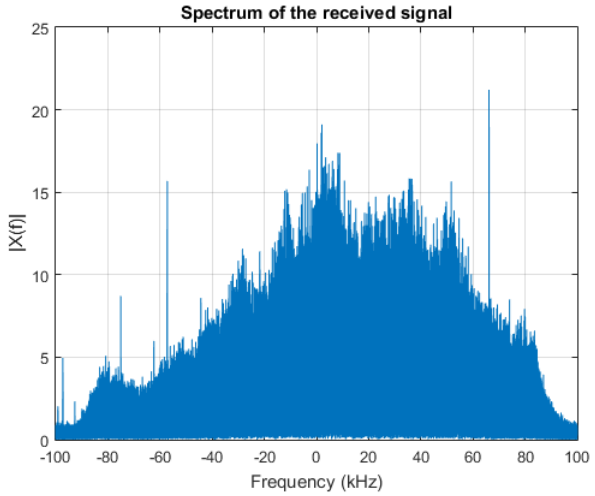


Figura 6.7 – *Espectro de la señal modulada en frecuencia WFM recibida mediante el comando rtl_sdr de la librería librtl.*

Para el caso del receptor *FUNcube Dongle Pro+*, no se puede utiliza esta librería de comando puesto que sólo es compatible con los dispositivos SDR de bajo coste diseñados con el chip demodulador *R820T*. En ese caso, la captación de señales se realiza a través del Software SDRSharp y es tan sencilla como escoger el receptor *FUNcube Dongle Pro+*, fijar la frecuencia de sintonización e irse a la pestaña *Recording* para iniciar la grabación. Transcurrido el tiempo de grabación deseado por el usuario, esta herramienta almacena en su directorio de instalación la señal recibida en cuadratura en formato de señal de audio *.wav*.

6.2 Demodulación de la señal WFM

A lo largo de esta sección se trabaja con la señal de entrenamiento recibida a 88.2 MHz para el diseño e implementación del algoritmo de demodulación en frecuencia y la señal de evaluación generada a 90 MHz para comprobar su correcto funcionamiento.

Previamente a la demodulación en frecuencia de la señal recibida, el algoritmo comprueba que la señal recibida está modulada en FM para continuar o no con el proceso de demodulación. Esta verificación inicial de la señal es necesaria puesto que en algunas ocasiones la señal recibida sufre una atenuación tan elevada que se considera la señal como ruidosa puesto que el demodulador FM no es capaz de decodificar la información transmitid en ella. Esta problemática se da cuando se trabaja en espacios muy cerrados o bajo el nivel del suelo como en sótanos. También puede darse cuando la antena de TV utilizada para la recepción de señales no es de calidad y tiene una ganancia de recepción muy reducida.

La idea de realizar esta breve comprobación inicial surgió debido a los problemas encontrados por la alumna a la hora de demodular las priemras señales reales captadas (sección 6.1). Tras diseñar el algoritmo de demodulación se realizó el testeo con señales reales y se comprobó que la señal demodulada no contenía información.

Las señales recibidas por el dispositivo SDR se obtienen en cuadratura, tal y como muestra la expresión 6.1.1. En primer lugar, se comprueba que la señal recibida está modulada mediante una modulación angular y para ello se obtiene la representación de su *equivalente paso baja*. La equivalente paso baja de una señal no es más que la representación de sus componentes en fase y en cuadratura. De este modo, se puede verificar que la modulación es angular si esta representación es una circunferencia.

```
% Load the received signal
cd C:\PILAR\UNIVERSIDAD\2_MASTER\TFM\Matlab\WFM_Demodulacion\Signals\Entrenamiento
load(signal_name);                          % Struct
signal=reshape(x.Data,numel(x.Data),1); % Vector
fs=200e3;    % Sampling frequency of the received signal (Hz)
Ts=1/fs;     % Sampling period of the received signal (s)
t=0:Ts:(length(signal)-1)*Ts;              % Temporary axis

% Low-pass equivalent: x(n) = x_i(n) + j·x_q(n)
if graphics==1
    figure;
    plot(signal); grid on;
    title('Low-pass equivalent of the received signal');
    xlabel('Real'); ylabel('Imaginary');
end
```

Figura 6.8 – *Código fuente en MatLab que carga la señal recibida y la representa su equivalente paso baja.*

En la figura 6.9 se comprueba que la modulación de la señal es angular y se obtiene a través del código de la figura 6.8. El radio de la circunferencia es aproximadamente la amplitud de la señal modulada y su espesor corresponde con el ruido contenido en esta señal. Idealmente, esta representación será una cirfunferencia.

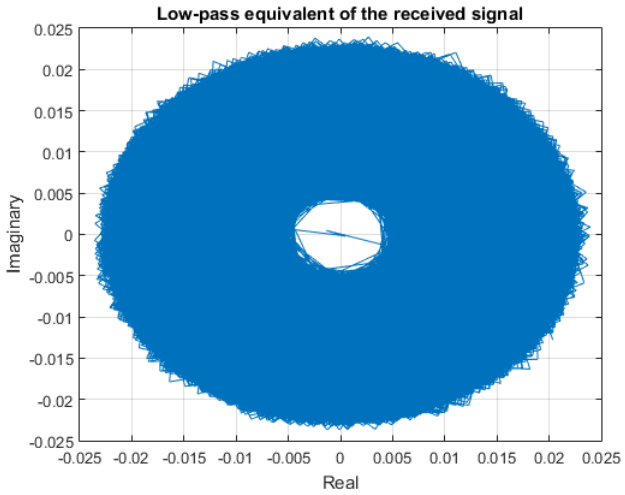


Figura 6.9 – *Equivalente paso baja de la señal recibida.*

Una vez que se ha confirmado que la modulación analógica es angular, el algoritmo asegura que se modula en frecuencia. Para ello se representa el módulo y la fase de la señal recibida. Una señal está modulada en frecuencia si la información se transmite en en la fase y se modula en amplitud si se transmite en el módulo. Observando donde se produce la variación se puede decidir el tipo de modulación empleada.

Se ha representado 50 ms de la señal recibida para observar el comportamiento de su módulo y fase a lo largo del tiempo. En la figura 6.10 se observa que el módulo de la señal

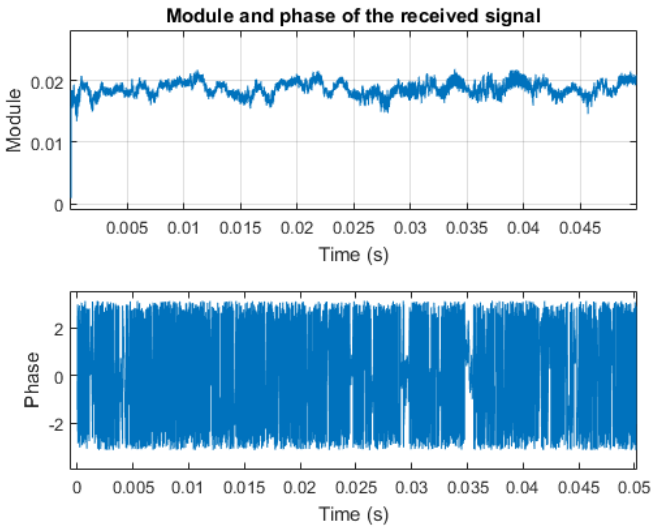


Figura 6.10 – *Equivalente paso baja de la señal recibida.*

permanece prácticamente constante en torno a 0.02 V y la fase oscilan entorno a 0 V dentro del rango [-2, 2] V. La frecuencia instantánea de esta señal corresponde con al derivada de la fase y sus variaciones confirman que la información se transmite en la fase. Por ello, el programa sabe que la señal recibida está modulad en frecuencia y puede continuar con el proceso de demodulación.

Antes de iniciar la demodulación de la señal, se obtiene el espectro de potencia de la señal FM recibida mediante el comando de MatLab *fftshift* (figura 6.11). Tal y como se observa en la figura 6.11 la señal se recibe en banda base y ocupa un rango de frecuencias de $-f_s/2$ a $f_s/2$, lo que implica un ancho de banda de la señal igual a $f_s/2 = 100$ kHz.

```
% Spectrum of the received signal
signal_spectrum=fftshift(fft(signal));
f=linspace(-fs/2,fs/2,length(signal_spectrum)); % Frequency axis
if graphics==1
    figure;
    plot(f/1e3,abs(signal_spectrum)); grid on;
    title('Spectrum of the received signal');
    xlabel('Frequency (kHz)'); ylabel('|X(f)|');
end
```

Figura 6.11 – *Código fuente en MatLab que calcula el espectro de la señal recibida.*

Tras las comprobaciones pertinentes, el algoritmo realiza la demodulación en frecuencia. Puesto que se ha verificado en 6.10 que las variaciones se producen en fase, para demodular la señal basta con quedarse con estas variaciones, dónde se encuentra la señal de información transmitida por la estación. Para ello, se utiliza la función de MatLab *angle* que obtiene la fase de una señal en cuadratura.

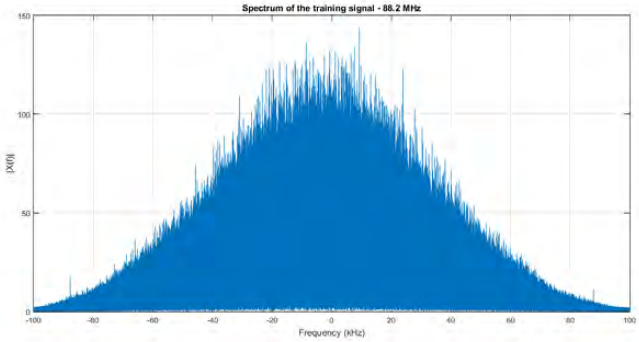


Figura 6.12 – *Espectro de la señal modulada en frecuencia WFM recibida.*

```
% Frequency demodulated signal
signal_demod=angle(signal(2:end).*conj(signal(1:end-1)));
t=0:Ts:(length(signal_demod)-1)*Ts;      % Temporary axis
```

Figura 6.13 – *Código fuente en MatLab que obtiene la señal demodulada en frecuencia.*

A continuación se evalúa el espectro de frecuencia de la señal demodulada, donde se observa que la señal recibida de la radio FM comercial se transmitió en estéreo. Este espectro corresponde con la señal múltiplex (*MPX*) descrita en el capítulo 4 del presente trabajo.

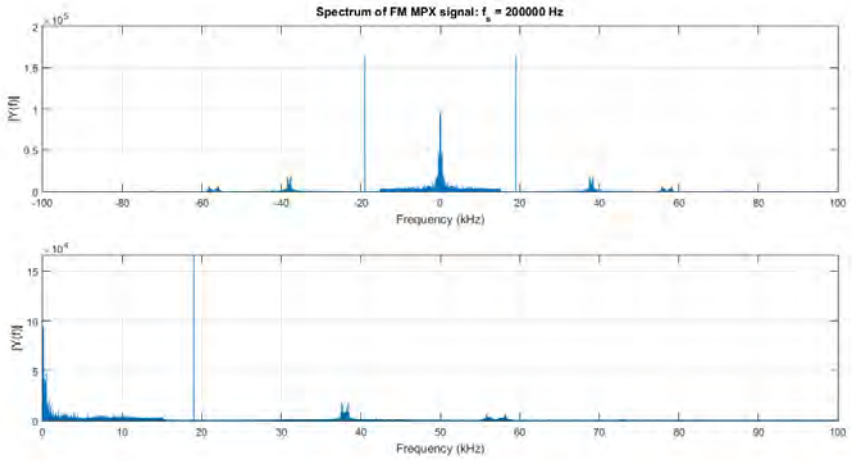


Figura 6.14 – *Espectro de la señal demodulada en frecuencia WFM recibida. Espectro múltiplex o estéreo.*

Tal y como se introdujo en anteriores capítulo, el espectro de la señal FM estéreo mantiene una estructura fija. En al figura 6.15 se puede observar como se transmiten las siguientes señales:

- En el rango de frecuencia de *30 Hz a 15 kHz* se transmite la señal mono, la señal de suma o el canal L+R, única señal capaz de decodificarse en un receptor mono.

- A 19 kHz se transmite la portadora piloto o el estéreo piloto. Como su nombre lo indica, solo se recibe cuando el transmisor es estéreo. Se trata de un tono que tiene la misma fase que la señal diferencia y una amplitud del 10% de la amplitud total de la señal. Este hecho se puede comprobar observando las gráficas de las figuras 6.12 y 6.14.
- Desde los 23 kHz y hasta 53 kHz, se transmite la señal diferencia o el canal L-R. Esta señal se centra en el doble de la portadora piloto (38 kHz), tiene un ancho de banda de 15 kHz y está modulada en DSBSC.
- Los servicios de RDS se transmiten a al triple de la portadora (57 kHz) con un ancho de banda de 3 kHz y modulados con DSBSC.

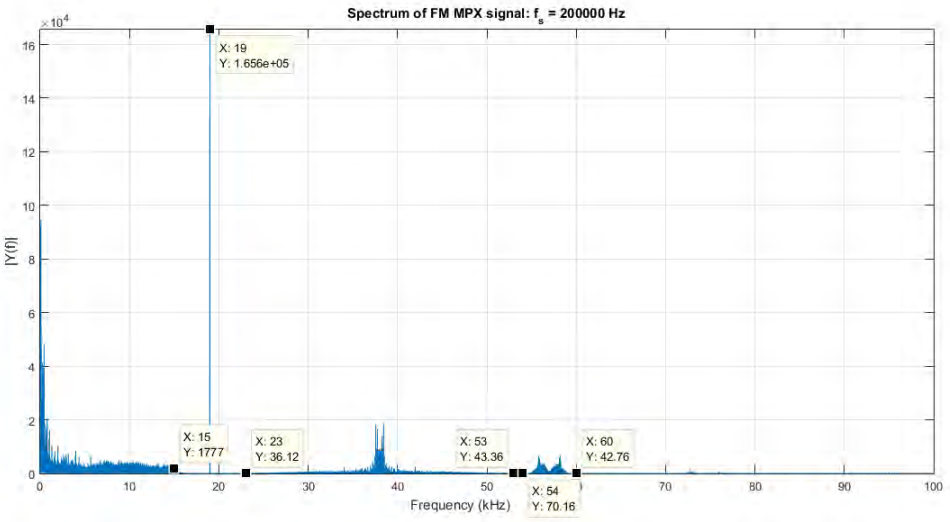


Figura 6.15 – *Espectro múltiplex o estéreo.*

6.2.1 Evaluación

Una vez diseñado el algoritmo de demodulación en frecuencia a partir de la señales de entrenamiento se va a evaluar para la señal de evaluación modulada en frecuencia a 90 MHz, cuyo espectro se muestra en la figura 6.7. En este caso, la señal de información que se desea demodular es conocida y se trata de un tono a 1 kHz. Para ello, basta con pasar esta nueva señal por el algoritmo implementado anteriormente.

Como resultado del demodulador implementado se obtiene la señal de evaluación demodulada de la figura 6.16. En ella puede observarse la señal de información que se transmitía (tono a 1 kHz), la señal modulada en frecuencia a 90 MHz y la señal demodulada en frecuencia por el algoritmo diseñado por la alumna (tono a 1 kHz). De este modo se comprueba que el algoritmo demodula correctamente la señales modulada sen FM y se pueden implementar los receptores de radio FM comercial.

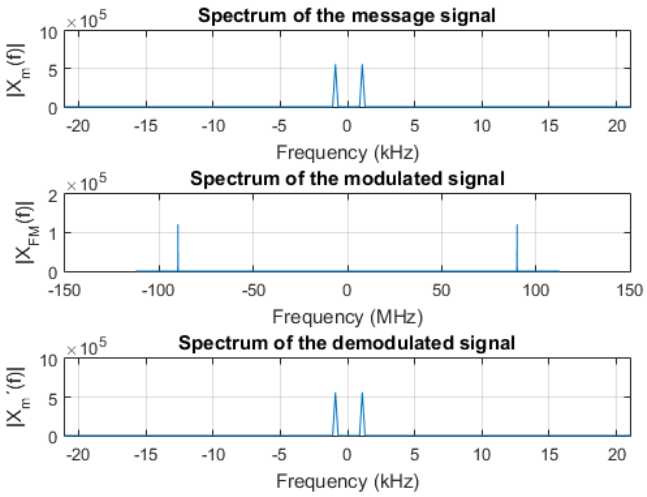


Figura 6.16 – *Espectro de la señal de evaluación demodulada.*

6.3 Receptor mono

El receptor mono únicamente es capaz de decodificar la información transmitida en el canal L+R o Suma de la señal FM. Para la implementación de este receptor se ha utilizado la misma señal de entrenamiento a 88.2 MHz y se han tenido en cuenta los conocimientos expuestos en el capítulo 4 del proyecto. Con esta implementación se pretende demostrar que un receptor monofónico también es capaz decodificar la información de la radio FM comercial a pesar de qu eta se genere y transmita por una estación estereofónica.

Una vez demodulada la señal en frecuencia, la señal múltiplex se pasa por el receptor mono. Este sistema sólo es capaz de decodificar la señal L+R impidiendo así obtener los canales de audio L y R que componen la señal. A pesar de ello, la información de audio analógica contenida en ella es decodificada y el usuario es capaz de conocer su contenido.

El primer bloque del receptor mono es un *filtrado paso baja* con una frecuencia de corte de 15 kHz. Dado que la señal mono, Suma o L+R se transmite en el rango de frecuencia de 30 Hz a 15 kH ésta pasará sin atenuarse y se elimina la información transmitida a frecuencias superiores a dicho canal.

En la figura 6.17 se implementa un filtro paso baja de tipo FIR de fase lineal para que la información de la banda pasante se atenúe lo menos posible. La frecuencia de corte del filtro coincide con la frecuencia a la que termina de transmitirse el canal L+R, esto es, 15 kHz. A partir de dicha frecuencia la atenuación de la señal será cada vez mayor hasta el punto de eliminar la información transmitida. Se ha escogido un orden del filtro lo suficientemente alto para que la pendiente del filtro sea más pronunciada, siendo más restrictivo puesto que la atenuación crece rápidamente. Aumentar el orden del filtro significa que la fase del filtro se vuelve inestable. Por lo tanto, se debe guardar un compromiso entre la estabilidad del filtro (orden bajo) y el filtro restrictivo (orden superior). Se ha decidido fijar el orden a un valor 200.

```
% Linear phase FIR low-pass filter at 15 kHz
fc=15e3;              % Filter cutoff frequency (Hz)
wn=fc/(fs/2);         % Normalized filter cutoff frequency
order=200;            % Filter order
h=fir1(order,wn);     % Filter transfer function h(n)

% Filter transfer function H(w) and angular frequency axis (rad)
[response,w_axis]=freqz(h,1);
f_axis=w_axis.*fs/(2*pi);       % Frequency axis (Hz)

% Graphic representation of the filter
if graphics==1
    figure;
    subplot(2,1,1); plot(f_axis,abs(response)); grid on;
    title('Low Pass Filter: f_c = 15 kHz')
    xlabel('Frequency (Hz)'), ylabel('Module');
    subplot(2,1,2); plot(f_axis,angle(response)); grid on;
    xlabel('Frequency (Hz)'), ylabel('Phase');
end

% Filtering the demodulated signal
LR=filter(h,1,m);     % sum or L+R signal
```

Figura 6.17 – *Código fuente en MatLab que implementa el filtrado del receptor mono.*

En la representación gráfica del filtro se puede comprobar que es lo suficientemente restrictivo puesto que la atenuación decae rápidamente respecto la frecuencia. Además, en la fase del filtro se puede ver que se mantiene la estabilidad puesto que a 15 kHz la fase es prácticamente nula. La señal Suma será filtrada sin apenas atenuaciones puesto que en la gráfica 6.18 se puede ver que la caida de -3 dB corresponde con la frecuencia de corte de 15 kHz.

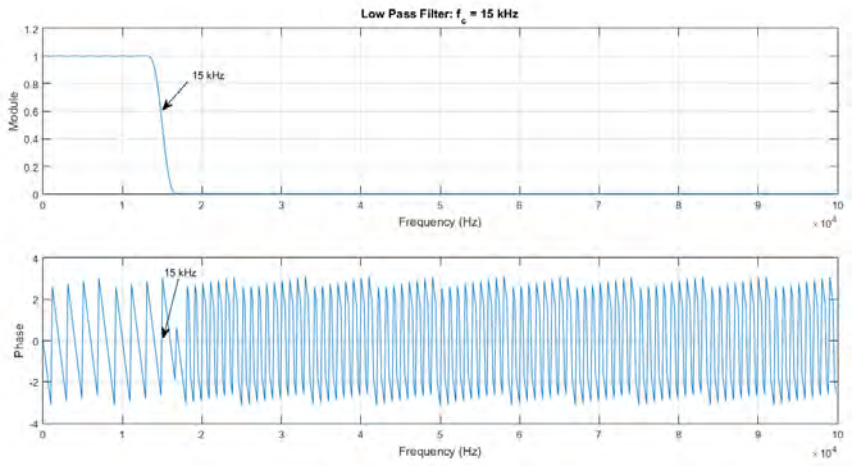


Figura 6.18 – *Módulo y fase del filtro paso baja del receptor mono.*

A continuación es necesario aplicar un filtro de reducción de énfasis para deshacer el filtrado de *pre-énfasis* que se realizó en el transmisor. En el transmisor, la técnica de pre-énfasis se usa para reducir el ruido aplicado al audio de los canales L y R antes de generar

la señal Suma y Diferencia. En el transmisor las componentes a frecuencias altas son más sensibles al ruido y por tanto se incrementa su amplitud con la la constante de tiempo de 50 μs en Europa. De esta manera, la amplitud a frecuencias altas debe reducirse en el receptor para deshacer el pre-énfasis.

```
function [signal_out] = deemphasis_filter(signal_in,fs,graphics)

    N=5; % Orden
    B=1; % Numero de bandas
    F1=[50 100 500 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000 ...
        11000 12000 13000 14000 15000 16000 17000 18000 19000 20000 ...
        97656.25];  %  Vector de frecuencias
    A1=[1 0.998849369936505 0.973867785328633 0.904690437738119 ...
        0.727779804536824 0.577430872713986 0.468813382145265 ...
        0.390840895792402 0.333426412763235 0.290068119869315 ...
        0.256448403651772 0.229614864811236 0.207491351745491 ...
        0.189452351435659 0.174180687339161 0.161064563517827 ...
        0.149795925304488 0.140119958494001 0.131522483219224 ...
        0.123879658653037 0.117084660232024 0.111045253533364 ...
        0.105560150321593 0.0223872113856834]; %  Amplitud del vector


    h=fdesign.arbmag('N,B,F,A',N,B,F1,A1,fs);
    deemphasis_filter=design(h,'iirlpnorm');

    if graphics==1
        fvtool(deemphasis_filter,'Color','White');
        legend('Filtro diseñado','Especificaciones');
        title('Deemphasis Filter');
    end

    signal_out=filter(deemphasis_filter,signal_in);   % Señal filtrada

end
```

Figura 6.19 – *Código fuente en MatLab que implementa el filtrado de-énfasis de la señal Suma.*

El filtro de de-énfasis se implementa mediante la función de la figura 6.19 y en la especificación de la radio FM comercial [31] se define como la respuesta de frecuencia de un sistema RC de primer orden con una constante de tiempo τ 50 μs en Europa y 75 μs en América. Su respuesta impulsiva es:

$$h(t) = \frac{1}{\tau e^{-t/\tau} u(t)} \tag{6.3.1}$$

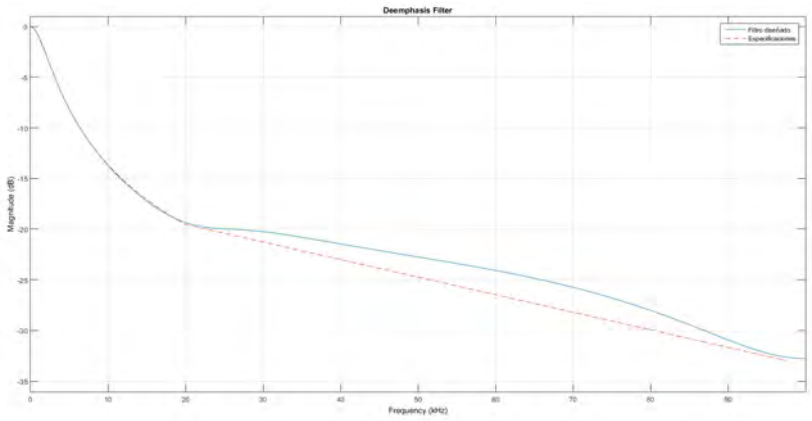


Figura 6.20 – *Módulo y fase del filtro de-énfasis.*

Por último, es necesario reducir la tasa de muestreo de la señal para poder reproducirla. La frecuencia de muestreo de la señal recibida es de 200 kHz pero la frecuencia de muestreo de una señal de audio es mucho menor, de 48 kHz para ser audible por el oído humano. Para ello, se aplica un factor de muestreo:

$$Factor = \frac{48kHz}{200kHz} = \frac{6}{25} \tag{6.3.2}$$

```
fs=48e3;                    % Audio sampling frequency (kHz)
mono=resample(LRd,6,25);    % Mono or sum subsampled Signal
```

Figura 6.21 – *Código fuente en MatLab que remuestrea la señal Suma.*

El receptor mono implementado en MatLab representa gráficamente el procesamiento de señal realizado desde que recibe la señal FM comercial hasta que obtiene la señal de audio L+R.

En la figura 6.22 se puede ver como la señal múltiplez recibida es filtrada para quedarse únicamente con la señal Suma, que posteriormente es filtrad mediante el filtro de-énfasis que reduce la amplitud de las componentes de frecuencia elevada. Por último se observa la señal L+R remuestreada a 48 kHz. Esta señal se reproduce mediante el comando de MatLab *soundsc*.

6.4 Receptor estéreo

Demostrado que el receptor mono es capaz de decodificar la señal de audio transmitida se implementa el receptor estéreo como una mejora del anterior. Es por ello que parte del diseño de este receptor ya se ha visto en la sección anterior y únicamente se mencionará.

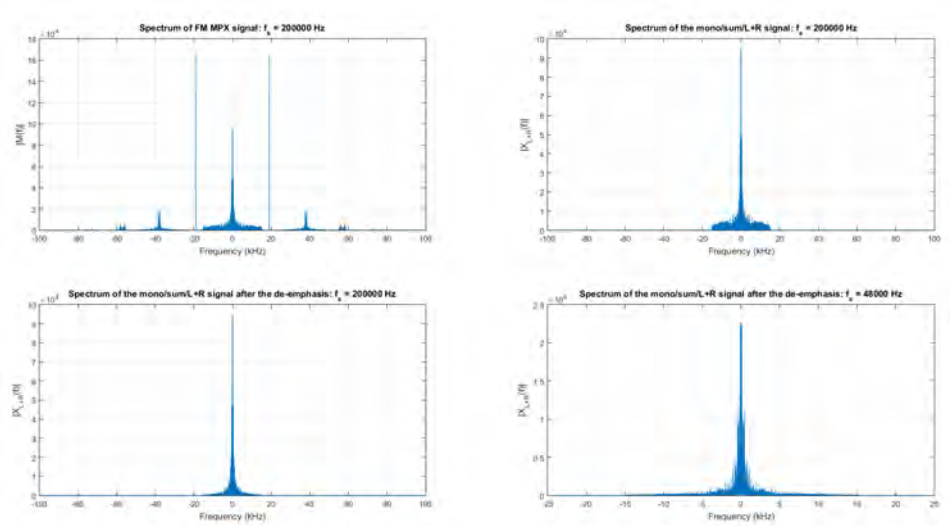


Figura 6.22 – *Procesamiento de señal realizado en el rceptor mono.*

La señal de radio FM comercial recibida a través del dispositivo SDR es demodulada en frecuencia. A diferencia del receptor mono, el sistema estéreo permite decodificar la señal múltiplex al completo (figura 6.14) por lo que se compone de varios subsistemas.

Señal Suma L+R

El proceso para obtener la señal Suma es el mismo que implementa el receptor monofónico, estudiado en la sección 6.3, a excepción del remuestreo final.

Señal piloto

La portadora piloto se trata de un tono transmitido a 19 kHz e indica que la señal transmitida es estéreo. Esta señal es necesaria para decodificar la señal diferencia y el servicio RDS.

Se realiza un filtrado paso banda de la señal con un filtro con banda pasante muy estrecha (NBPF). De forma análoga al filtrado de la señal Suma, se ha implementado un filtro FIR con un orden elevado (200) para que la caída de la banda pasante sea muy rápida. Esto es necesario para que el ancho de banda sea muy estrecho y toda la información transmitida a frecuencias diferentes de 19 kHz sea atenuada.

La función de transferencia de este filtro se muestra en la figura 6.24 y en ella se comprueba que el filtro es lo suficientemente estrecho manteniendo la estabilidad.

```
% FIR Narrow BandPass Filter (NBPF)
fc=19e3;                % Filter center frequency (Hz)
delta=1e-10;            % Frequency deviation from the central
finf=fc-delta;          % Lower cutting frequency
fsup=fc+delta;          % Upper cutting frequency
winf=finf/(fs/2);       % Normalized lower cutoff frequency
wsup=fsup/(fs/2);       % Normalized upper cutoff frequency
order=200;              % Filter order
h_pilot=fir1(order,[winf wsup]);        % Filter transfer function h(n)

% Filter transfer function H(w) and angular frequency axis (rad)
[response,w_axis]=freqz(h_pilot,1);
f_axis=w_axis.*fs/(2*pi);               % Frequency axis (Hz)

% Graphic representation of the filter
if graphics==1
    figure;
    subplot(2,1,1); plot(f_axis,abs(response)); grid on;
    title('Narrow Band Pass Filter at 19 kHz')
    xlabel('Frequency (Hz)'), ylabel('Module');
    subplot(2,1,2); plot(f_axis,angle(response)); grid on;
    xlabel('Frequency (Hz)'), ylabel('Phase');
end

% Filtering the demodulated signal
pilot=filter(h_pilot,1,m);
```

Figura 6.23 – *Código fuente en MatLab que implementa el filtrado de la portadora piloto.*

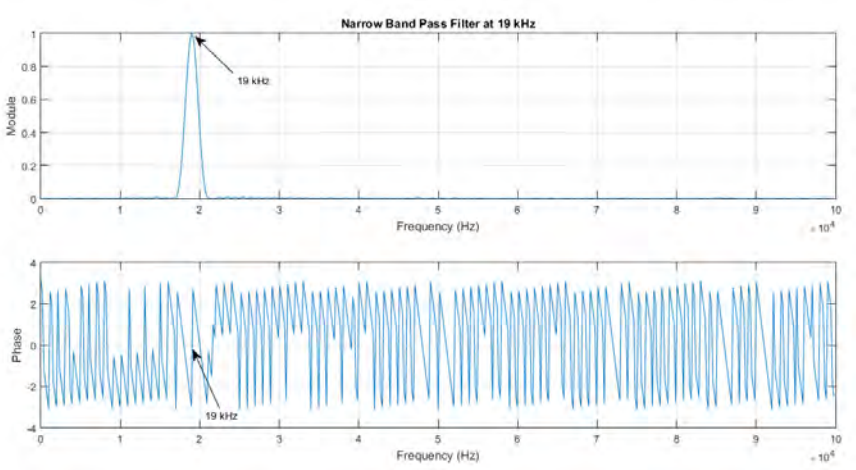


Figura 6.24 – *Módulo y fase del filtro paso banda que filtra la portadora piloto.*

Señal Diferencia L-R

La señal Diferencia o L-R se transmite centrada a 38 kHz, esto es, al doble de la portadora piloto con un ancho de banda de 15 kHz. Para decodificarlo, la señal múltiplex es filtrada por un *filtro paso banda* de estas características: centrado a 38 kHz y con un ancho de banda de 30 kHz. Se implementa un filtro FIR de fase lineal con un orden 200, que guarda un compromiso entre atenuación en la banda de stop y la estabilidad del filtro. La función de transferencia de este filtro se muestra en la figura 6.26 y en ella se comprueba que el filtro es lo suficientemente estrecho manteniendo la estabilidad.

Al filtrar la señal de diferencia, la información se elimina a frecuencias fuera del rango

```
% FIR pass-band filter centered at 38 kHz
fc=38e3;                % Filter center frequency (Hz)
delta=15e3;             % Frequency deviation from the central
finf=fc-delta;          % Lower cutting frequency
fsup=fc+delta;          % Upper cutting frequency
winf=finf/(fs/2);       % Normalized lower cutoff frequency
wsup=fsup/(fs/2);       % Normalized upper cutoff frequency
order=200;              % Filter order
h_L_R=fir1(order,[winf wsup]);      % Filter transfer function h(n)

% Filter transfer function H(w) and angular frequency axis (rad)
[response,w_axis]=freqz(h_L_R,1);
f_axis=w_axis.*fs/(2*pi);              % Frequency axis (Hz)

% Graphic representation of the filter
if graphics==1
    figure;
    subplot(2,1,1); plot(f_axis,abs(response)); grid on;
    title('Band Pass Filter at 38 kHz')
    xlabel('Frequency (Hz)'), ylabel('Module');
    subplot(2,1,2); plot(f_axis,angle(response)); grid on;
    xlabel('Frequency (Hz)'), ylabel('Phase');
end

% Filtering the demodulated signal
L_R=filter(h_L_R,1,m); % This signal is centered at twice the carrier
```

Figura 6.25 – *Código fuente en MatLab que implementa el filtrado de la señal diferencia L-R.*

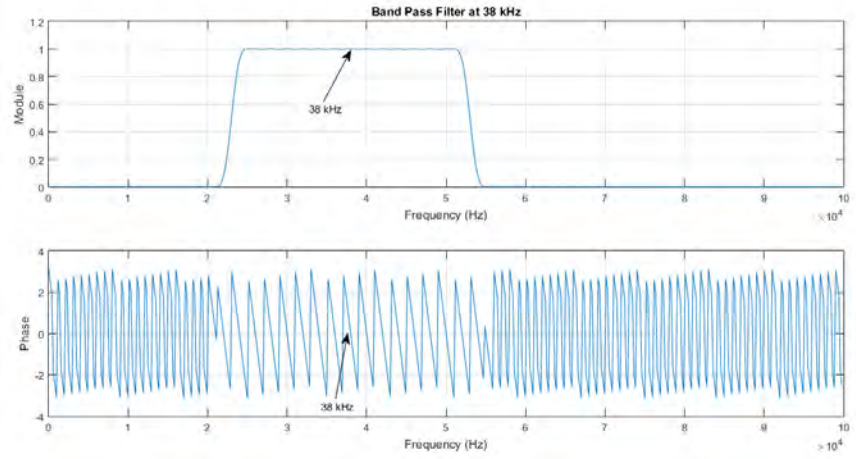


Figura 6.26 – *Módulo y fase del filtro paso banda que filtra la señal diferencia.*

de 23-53 kHz. Sin embargo, esta señal L-R paso banda está centrada al doble de la portadora piloto. Para poder decodificarla es necesario trasladarla a banda base. Esto se hace generando un tono al doble de la portadora piloto para multiplicarlo por la señal de paso de banda implementando así un *mezclador*.

El tono a 38 kHz se genera al duplicar la portadora piloto a 19 kHz. Sin embargo, al realizar esto se generan réplicas a frecuencias múltiplos de 19 kHz: $n \cdot 19$ kHz con $n = 0, \pm 1, \pm 2, \pm 3, ...$ De todas estas réplicas únicamente es de interés la componente a ± 38 kHz por lo que es necesario filtrar la señal para obtener el tono al doble de la portadora con el filtro paso banda implementado en la figura 6.25. Una vez que se tiene este tono, se

multiplica con un mezclador implementado en Software que traslada la señal a banda base. Al igual que ocurría antes, hay réplicas de la señal L-R centrado en $n \cdot 38$ kHz de forma que se filtra nuevamente la señal. Para eliminar las frecuencia imagen y quedarse únicamente con la señal en banda base se utiliza el filtro paso baja de la figura 6.19.

```
% Generation of the tone at 38 kHz
tono38=pilot.*pilot;     % Mixer result

fc=38e3;                 % Filter center frequency (Hz)
delta=1e-10;             % Frequency deviation from the central
finf=fc-delta;           % Lower cutting frequency
fsup=fc+delta;           % Upper cutting frequency
winf=finf/(fs/2);        % Normalized lower cutoff frequency
wsup=fsup/(fs/2);        % Normalized upper cutoff frequency
order=200;               % Filter order
h=fir1(order,[winf wsup]); % Filter transfer function h(n)

tono38=filter(h,1,tono38); % 38 kHz tone

% Conversion of the bandpass to baseband signal
L_R=filter(h_LR,1,L_R.*tono38);
```

Figura 6.27 – *Código fuente en MatLab que implementa el oscilador y mezclador a 38 kHz.*

A continuación, la señal diferencia banda base se filtra con el filtro de-énfasis al igual que se hacía con la señal Suma.

El algoritmo representa el espectro de las señales filtradas hasta ahora para permitir al usuario que analice y estudie el procesamiento de señal realizado. En la figura 6.28 se observa el resultado del filtrado de cada una de las señales que componen la transmisión analógica de la señal múltiplex FM. En ella se puede observar como la potencia de la señal no ha variado debido a que el filtrado se ha implementado cuidadosamente para no atenuar la amplitud en las bandas de frecuencia de interés.

Canales de audio L y R

Los canales de audio L y R transmitidos por la emisora se pueden obtener a partir de las señales Suma L+R y Diferencia L-R. Para ello, es suficiente con combinar ambas señales de la siguiente manera:

$$Canal_L = \frac{(L+R)+(L-R)}{2} \tag{6.4.1}$$

$$Canal_R = \frac{(L+R)-(L-R)}{2} \tag{6.4.2}$$

Obtenidos los canales de audio L y R es necesario reducir la frecuencia de muestreo de las señales a 48 kHz para ser audible por el oído humano y para ello se aplica el factor de

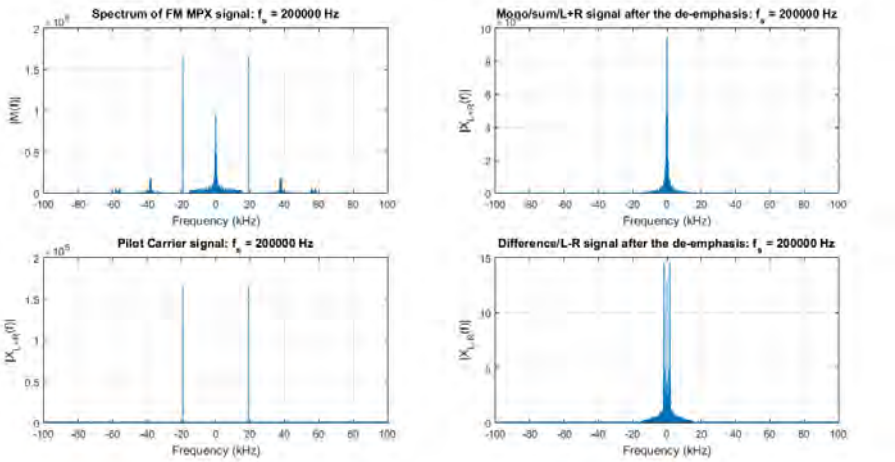


Figura 6.28 – *Espectro de la señal múltiplex FM, la señal L+R, L-R y la portadora piloto.*

remuestreo de la expresión 6.3.2.

```
L_channel=(mono+difference)/2;
R_channel=(mono-difference)/2;

fs=48e3;                                % Audio sampling frequency (kHz)
L_channel=resample(L_channel,6,25);
R_channel=resample(R_channel,6,25);

% Reproduction of the signal
soundsc([L_channel R_channel],fs);
```

Figura 6.29 – *Código fuente en MatLab que obtiene los canales de audio L y R.*

Señal RDS

Los servicios RDS se transmiten al triple de la portadora piloto (57 kHz) con un ancho de banda de 3 kHz. Es por ello que el primer bloque implementado para la decodificación de esta señal es un filtro paso banda muy similar al diseñado para filtrar la señal Diferencia. Este filtro FIR, mostrado en la figura 6.31, tiene un ancho de banda de 6 kHz y un orden de 200 para que sea restrictivo manteniendo la estabilidad de la fase.

Una vez filtrada la señal múltiplex, se tiene un espectro paso banda con la señal RDS. Dado que no se trata de una señal de audio no es necesario trasladarla a banda base. Por ello, a continuación se le aplica el filtro de de-énfasis de igual forma que se ha venido realizando hasta ahora. En la figura 6.33 se puede ver la señal RDS antes y después del filtrado de de-énfasis, donde se comprueba que la amplitud de la señal completa se ve reducida puesto que se transmite a frecuencias elevadas.

```
% FIR filter bandpass at 57 kHz
fc=57e3;            % Filter center frequency (Hz)
delta=3e3;          % Frequency deviation from the central
finf=fc-delta;      % Lower cutting frequency
fsup=fc+delta;      % Upper cutting frequency
winf=2*finf/fs;     % Normalized lower cutoff frequency
wsup=2*fsup/fs;     % Normalized upper cutoff frequency
order=200;          % Filter order
h_rds=fir1(order,[winf wsup]);      % Filter transfer function h(n)

% Filter transfer function H(w) and angular frequency axis (rad)
[response,w_axis]=freqz(h_rds,1);
f_axis=w_axis.*fs/(2*pi);               % Frequency axis (Hz

% Graphic representation of the filter
if graphics==1
    figure;
    subplot(2,1,1); plot(f_axis,abs(response)); grid on;
    title('Band Pass Filter at 57 kHz')
    xlabel('Frequency (Hz)'), ylabel('Module');
    subplot(2,1,2); plot(f_axis,angle(response)); grid on;
    xlabel('Frequency (Hz)'), ylabel('Phase');
end

% FM MPX signal bandpass filteredat 57 kHz
rds_signal=filter(h_rds,1,m);
```

Figura 6.30 – *Código fuente en MatLab que implementa el filtrado de la señal RDS.*

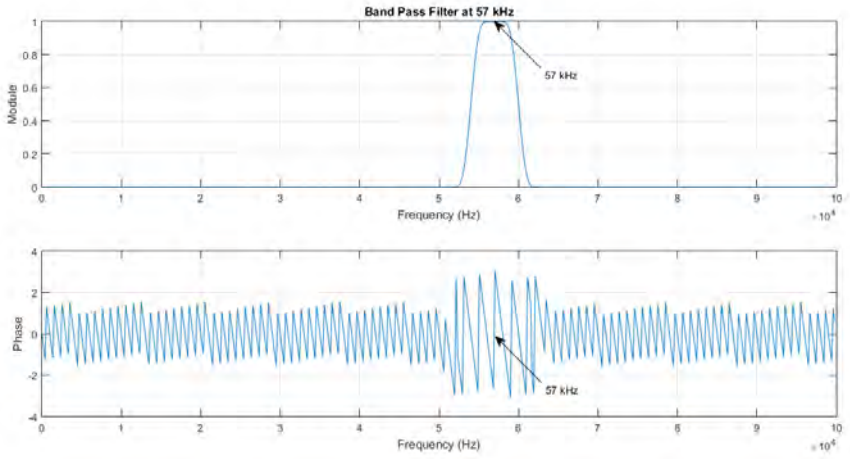


Figura 6.31 – *Módulo y fase del filtro paso banda centrado a 57 kHz.*

6.5 Decodificador de los servicios RDS

En esta señal se transmiten los servicios de datos RDS, que se trata de un sistema de transmisión de información digital a una tasa de 1187.5 bps. Por este motivo, la señal RDS está codificada digitalmente y los datos binarios se transmiten modulado en una onda senoidal de 57 kHz usando un esquema BPSK Diferencial. Par ala decodificación de estas tramas de datos se consulta el estándar *Especificación del sistema de datos de radio (RDS) para el sonido VHF/FM de transmisión en el rango de frecuencia de 87.5 a 108.0 MHz* [31].

El producto de la frecuencia de muestreo y el período del símbolo debe ser un número entero para que la información digital pueda decodificarse. La frecuencia de muestreo de la señal recibida típicamente es 200 kHz por lo que el producto de 200 kHz y 1/1187.5 no es un número entero (168.4211). Por este motivo, es necesario remuestrear la señal a 152 kHz de modo que el período de cada muestra sea 128. Para esto, se usa un factor de remuestreo:

$$Factor_{RDS} = \frac{152kHz}{200kHz} = \frac{19}{25} \tag{6.5.1}$$

```
fs=152e3;                          % Sampling frequency
Ts=1/fs;                           % Sampling period
rds=resample(rds_signal,19,25);    % Resampled RDS signal
trds=0:Ts:(length(rds)-1)*Ts;      % Temporary axis
```

Figura 6.32 – *Código fuente en MatLab que remuestrea la señal RDS.*

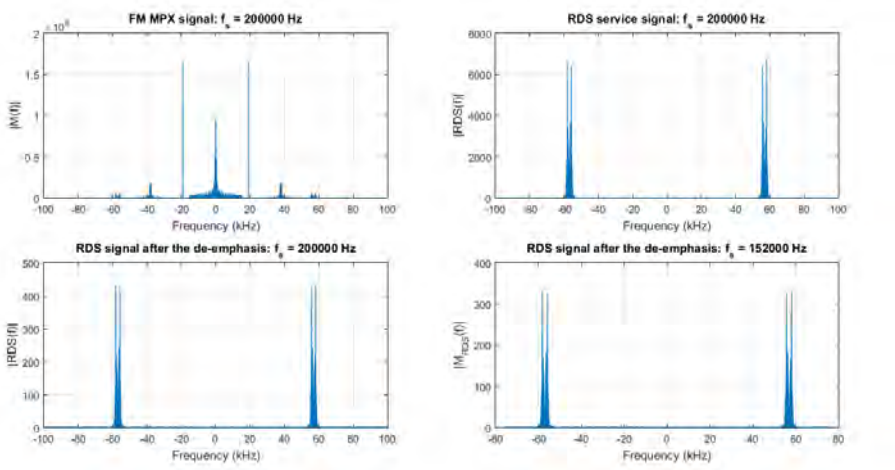


Figura 6.33 – *Espectro de la señal múltiplex, RDS ante y después del de-énfasis y tras el remuestreo.*

La señal RDS contiene información digital modulada en BPSK Diferencial, es decir, transmite dos símbolos de un bit en cuadratura. Por lo tanto, la señal remuestreada se descompondrá en su componente en fase y en cuadratura mediante las expresiones 4.5.2 y 4.5.4 del capítulo 4 del presente trabajo. Para ello, se implementan mediante Software dos mezcladores que multipliquen la señal RDS por un coseno y un seno, tal y como muestra la figura 6.34.

A continuación, se genera un pulso raíz coseno remontado que actuará como conformador de los símbolos transmitidos. Este pulso, definido por el estándar [31] se genera con el código de la figura 6.35 y sirve para obtener así la secuencia de símbolos recibidos.

```
% Components in phase and quadrature
frds=3*19e3; wrds=2*pi*frds;     % Central frequency of the RDS signal
rds_xi=rds.*cos(wrds.*trds');    % Phase component
rds_xq=rds.*sin(wrds.*trds');    % Quadrature component
```

Figura 6.34 – *Código fuente en MatLab que descompone la señal RDS en sus componentes en fase y cuadratura.*

```
bitrate=1187.5;          % Value of the standard
Tbit=1/bitrate;          % Bit period
Tsimb=Tbit;              % Period of symbol
tpulse=-Tsimb:Ts:Tsimb; % Temporary duration of the shaping pulse

Tsamples=fs*Tsimb; % Testing
% fprintf('\nfs*Tsimb is a whole number equal to %d\n',Tsamples);

% Creation of the pulse vector of zeros
pulse=zeros(1,length(tpulse));

% Definition of the value of the delta in the desired positions
pulse(find(tpulse>=-Tsimb/4,1))=1;  % t = -Tsimb / 4
pulse(find(tpulse>=Tsimb/4,1))=-1;  % t = Tsimb / 4

% Definition of the cosine root traced
beta=1;           % Rolloff factor (control the lobes)
span=8;           % No. of symbols
sps=Tsimb*fs/2;   % Samples per symbol period
reassembled_cos_root=rcosdesign(beta,span,sps,'sqrt');

% Convolution of the deltas and the root pulse cosine traced to obtain the
% conformation pulse
conforming_pulse=conv(pulse,reassembled_cos_root,'same');

% Graphic representation of the shaping pulse
if graphics==1
    figure;
    plot(tpulse,conforming_pulse); grid on;
    title('Conforming pulse');
    xlabel('Time (s)'); legend('Time limit: [-Tsymbol, Tsymbol]');
end
```

Figura 6.35 – *Código fuente en MatLab que genera el pulso conformador raíz coseno remontado.*

Este pulso conformará las componentes en fase y cuadratura de la señal RDS mediante un filtrado cuya función de transferencia es la inversa del pulso raíz coseno remontado de la figura 6.36. Como resultado de este filtrado se obtiene la secuencia de símbolos recibidos.

La secuencia de símbolos recibidos se ve afectada por el ruido causado por los símbolos adyacentes. Por lo tanto, los primeros símbolos de esta secuencia se verán afectados por un ruido de símbolos desconocidos de forma que se deberán eliminar de la secuencia. Para decidir qué primeros símbolos se deben descartar, se utiliza el *diagrama de ojo* de la señal, que muestra el comportamiento de cada símbolo respecto al ruido. Cada ojo del diagrama corresponde al símbolo presente y sus extremos corresponden al tiempo en que el presente símbolo se mezcla con el anterior (extremo derecho) o posterior (extremo izquierdo). Además, con esta representación, el algoritmo comprueba si se ha recibido información digital en la

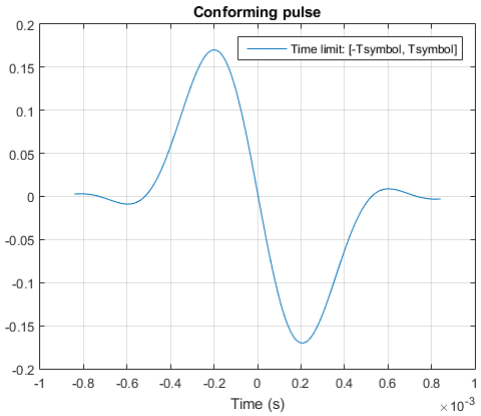


Figura 6.36 – *Pulso raíz coseno remontado.*

```
% Filtering the components in phase and quadrature with the shaping pulse
% so that the signal is continuous
symbol_sequence_i=filter(fliplr(conforming_pulse),1,rds_xi);  % Phase
symbol_sequence_q=filter(fliplr(conforming_pulse),1,rds_xq);  % Quadrature
```

Figura 6.37 – *Código fuente en MatLab que implementa el filtro conformador de las componentes en fase y cuadratura de la señal RDS.*

señal. Debido a que la señal suele tiene una gran cantidad de muestras, se representan las primeras 15000 muestras.

```
rds=[symbol_sequence_i(1:15000) symbol_sequence_q(1:15000)];
if graphics==1
    eyediagram(rds,256);
end
```

Figura 6.38 – *Código fuente en MatLab que obtiene el diagrama de ojo de los símbolos recibidos.*

6

Se observa en la figura 6.39 que las primeras muestras contienen mucho ruido ya que el ojo no está muy abierto. Estas muestras son descartadas y se define que se comience en la mitad de un ojo que esté limpio, libre de ruido.

En el momento de generar la constelación de símbolos se descartan aquellos que sufren ruido, tal y como se ha decidido a partir del diagrama de ojo de la secuencia recibida.Una vez hecho esto, se implementa el *conversor discreto continuo* para generar la constelación de símbolos almacenando un símbolo cada periodo de muestra:

$$T_s = f_s \cdot T_{bit} = 152kHz \cdot \frac{1}{1187.5bps} = 128s \tag{6.5.2}$$

La representación gráfica de la constelación de símbolos $a_i + jb_i$ de la figura 6.42 se

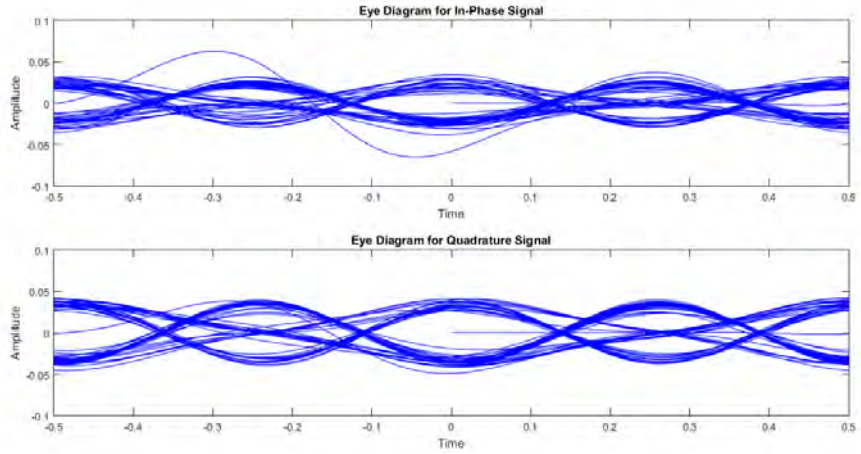


Figura 6.39 – *Diagrama de ojo de los símbolos recibidos.*

```
ai_symbols=[];
bi_symbols=[];
for sample=128:Tsamples:length(symbol_sequence_i)
    ai_symbols=[ai_symbols symbol_sequence_i(sample)];  % a(i) symbols
    bi_symbols=[bi_symbols symbol_sequence_q(sample)];  % b(i) symbols
end

% Symbol constellation
constellation_dif=ai_symbols+bi_symbols.*j;
if graphics==1
    figure;
    subplot(1,2,1); plot(constellation_dif,'o'); grid on;
    title('DBPSK Constellation');
end
```

Figura 6.40 – *Código fuente en MatLab que obtiene la constelación de símbolos recibidos mediante un conversor D/C.*

verifica que la constelación es BPSK Diferencial. Esto implica que los bits se transmiten con la diferencia de símbolo, de duración 48 ciclos. Para decodificar la codificación BPSK Diferencial el primer paso es convertirla a BPSK. Para ello, basta con girar la constelación y decidir que símbolo corresponde al bit 1 y qué símbolo corresponde al bit 0 para así obtener el flujo de bits recibido.

```
% We rotate the constellation: DBPSK -> BPSK
constellation=constellation_dif(2:end).*conj(constellation_dif(1:end-1));
if graphics==1
    subplot(1,2,2); plot(constellation,'o'); grid on;
    title('BPSK Constellation');
end

% Bit stream
for k=1:length(constellation)
    if constellation(k)<0
        bit_stream(k)=1;
    else
        bit_stream(k)=0;
    end
end
```

Figura 6.41 – *Código fuente en MatLab que decodifica la constelación de símbolos recibidos BPSK Diferencial.*

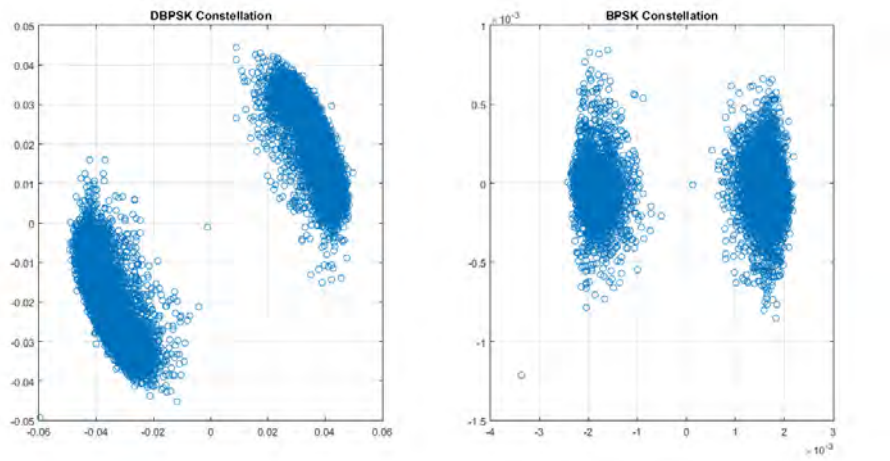


Figura 6.42 – *Constelación de símbolos recibidos: DBPSK y BPSK.*

6

6.5.1 Sincronización del sistema

A partir del flujo binario, se aplica la especificación [31] para obtener la información contenida en las tramas de datos RDS. La norma se describe en el capítulo 4 del proyecto y a continuación se describe cómo se obtienen los principales parámetros. Antes, es necesario *sincronizar la secuencia de bits recibidos con el receptor* través del síndrome para poder aplicar el estándar correctamente. El síndrome s es una palabra obtenida como el producto del flujo binario recibido y y la matriz de verificación de paridad H definida en la norma:

$$s = yH \tag{6.5.3}$$

De esta manera, el vector z que calcula el producto de la secuencia recibida y por la

transmitida x valdrá 1 en las posiciones en las que la secuencia recibida no coincida con la transmitida. Conocido el error introducido por el canal, se conoce el síndrome y se realizará la sincronización del sistema. Esta secuencia z se llama *palabra offset* y existen 5 palabras distintas: A, B, C, C 'y D.

```
% Parity check matrix (26x10 bits)
H=[1 0 0 0 0 0 0 0 0 0;
   0 1 0 0 0 0 0 0 0 0;
   0 0 1 0 0 0 0 0 0 0;
   0 0 0 1 0 0 0 0 0 0;
   0 0 0 0 1 0 0 0 0 0;
   0 0 0 0 0 1 0 0 0 0;
   0 0 0 0 0 0 1 0 0 0;
   0 0 0 0 0 0 0 1 0 0;
   0 0 0 0 0 0 0 0 1 0;
   0 0 0 0 0 0 0 0 0 1;
   1 0 1 1 0 1 1 1 0 0;
   0 1 0 1 1 0 1 1 1 0;
   0 0 1 0 1 1 0 1 1 1;
   1 0 1 0 0 0 0 1 1 1;
   1 1 1 0 0 1 1 1 1 1;
   1 1 0 0 0 1 0 0 1 1;
   1 1 0 1 0 1 0 1 0 1;
   1 1 0 1 1 1 0 1 1 0;
   0 1 1 0 1 1 1 0 1 1;
   1 0 0 0 0 0 0 0 0 1;
   1 1 1 1 0 1 1 1 0 0;
   0 1 1 1 1 0 1 1 1 0;
   0 0 1 1 1 1 0 1 1 1;
   1 0 1 0 1 0 0 1 1 1;
   1 1 1 0 0 0 1 1 1 1;
   1 1 0 0 0 1 1 0 1 1];
```

Figura 6.43 – *Código fuente en MatLab que define la matriz de paridad H.*

```
% Syndrome words
word_A   = [1 1 1 1 0 1 1 0 0 0];  % Word A (block 1)
word_B   = [1 1 1 1 0 1 0 1 0 0];  % Word B (block 2)
word_C   = [1 0 0 1 0 1 1 1 0 0];  % Word C (block 3)
word_Cp = [1 1 1 1 0 0 1 1 0 0];   % Word C' (block 3)
word_D   = [1 0 0 1 0 1 1 0 0 0];  % Word D (block 4)
```

Figura 6.44 – *Código fuente en MatLab que define las palabras offset A, B, C, C' y D.*

La sincronización de los datos recibidos se realiza buscando en el flujo binario las palabras offset anteriores. Para ello, tal y como se ve en el código de la figura 6.45, se divide el flujo binario en bloques de 26 bits y y se multiplican por la matriz de paridad H obteniendo así el síndrome (expresión 6.5.3). A continuación el algoritmo compara el síndrome obtenido con la palabra offset definida en el estándar: A, B, C, C 'y D.

El receptor estará sincronizado cuando se obtenga la secuencia de palabras offset A, B, C y D o A, B, C' y D.

```
bits_group=104;
bits_block=26;
bits_CRC=10;

n=1;
for i=1:length(bit_stream)-bits_block
    % Division in blocks of 26 bits
    block_received = bit_stream(i:i+bits_block-1);

    % Syndrome received
    syndrome_received = block_received*H;
    for j=1:length(syndrome_received)    % 10 bits
        if rem(syndrome_received(j),2)==0    % It's par
            word_received(j) = 0;
        else    % It's odd
            word_received(j) = 1;
        end
    end

    % Comparison of the received syndrome with those associated with the
    % offset words of the standard. Each word contains different information.
    if word_received==word_A
            check(i)='A';
    else if word_received==word_B
            check(i)='B';
        else if word_received==word_C
            check(i)='C';
            else if word_received==word_Cp
                check(i)='P';
                else if word_received==word_D
                    check(i)='D';
                    else
                        check(i)='X';
                    end
                end
            end
        end
    end

    % Stores the word offset corresponding to the current syndrome to have
    % a vector with the sequence of words corresponding to the received
    % binary flow.
    if check(i)~='X'
        word_sequence(n)=check(i);
        word_sequence_position(n)=i;
        n=n+1;
    end
end
```

Figura 6.45 – *Código fuente en MatLab que obtiene el síndrome.*

6.5.2 Decodificación de la información

Una vez que se ha sincronizado el sistema, el algoritmo decodifica la información digital transmitida en el servicio RDS. Para ello, se tienen el cuenta el estándar [31] y los conceptos del capítulo 4. Para que esta parte de la sección no se haga muy tediosa se adjuntará en el anexo **??** el código que implementa la decodificación de cada parámetro y a continuación se recoge la información recibida más relevante: tipo y versión de cada grupo, identificador de programa de tráfico, identificador de programa, identificador de país y nombre de la emisora.

Para la señal recibida se han obtenido los siguientes resultados y se han comprobado que son correctos accediendo a la información proporcionada por la emisora en Internet. Para la señal procedente de la emisora *Cadena 100*, transmitida a 88.2 MHz y con una longitud

```
% Search for the word sequence A B C D or A B C 'D to synchronize the
% receiver.
blocks_group=4;  % In each group of 104 bits there are 4 blocks of 26 bits.
n=0;
groups=[];
for i=1:length(word_sequence)-blocks_group
    if ((word_sequence(i)=='A' & word_sequence(i+1)=='B' & word_sequence(i+2)=='C' &
word_sequence(i+3)=='D') | (word_sequence(i)=='A' & word_sequence(i+1)=='B' & word_sequence
(i+2)=='P' & word_sequence(i+3)=='D') )
        n=n+1;
        groups(n,:) = bit_stream(word_sequence_position(i):word_sequence_position(i)
+bits_group-1);
    end
end
```

Figura 6.46 – *Código fuente en MatLab que sincroniza el receptor.*

inferior a 10 ms se han transmitido 53 grupos de bits con los siguientes parámetros:

```
Group 53
     - 0A: Basic tuning and switching information.
     - A traffic announcement is being broadcast on this programme at present.
     - Programme type: Pop Music.
     - PI: Romania, Spain or Sweden.
     - No ECC: please record at least one minute of signal.
     - National programme: transmitted throughout the country.
     - Programme reference number assigned: CE (Hex-coding).
```

Figura 6.47 – *Información digital recibida en el grupo 53 de la secuencia binaria del servicio RDS.*

```
NAME OF THE BROADCASTING STATION: CAD-100
```

Figura 6.48 – *Nombre de la emisora de la que procede la señal WFM recibida.*

El algoritmo que implementa mediante Software el decodificador WFM tiene como resultado cierta información de interés acerca de la señal recibida. Resulta muy útil para descubrir a qué emisora pertenece dicha señal, en este caso CAD-100, el tipo de aplicación al que está dedicada, dónde opera, si transmite anuncios de tráfico sobre el estado de las carreteras, el tipo de programa que emite y otros identificadores de interés técnico. Es de gran interés que toda esta información se obtenga tras un procesado de señal digital mediante Software que da flexibilidad al sistema y durabilidad en el tiempo.

6.6 Pruebas del receptor FM

Por último se evalúa el algoritmo diseñado con señales reales. Por ejemplo, se ha decidido decodificar la señal modulada en frecuencia recibida a 92.0 MHz con el dispositivo RTL-SDR de *Allwin*. Se ha escogido este receptor y no el dispositivo de alta precisión *FUNcube Dongle Pro+* para demostrar la robustez del algoritmo frente al ruido ya que el RTL-SDR recibe señales de menor calidad espectral y menor potencia además de sufrir la desviación en

frecuencia de 24 ppm provocada por el aumento de la temperatura del cristal de cuarzo. De este modo, se evalúa el decodificador implementado para el peor caso de estudio ya que si los resultados obtenidos son buenos para el receptor RTL-SDR, lo serán mucho mejor para el *FUNcube Dongle Pro+*. Tras lanzar el script se obtienen los siguientes resultados.

En primer lugar se obtiene el espectro de la señal múltiplex en el que se puede ver que la señal recibida es estéreo y por tanto se puede trabajar con un receptor estéreo. Además, se puede ver que la señal no está afectada por mucho ruido pero tiene una potencia reducida. Como ya se comentó en la sección 6.1 esto se debe a las condiciones del medio en el que se trabaja así como a la calidad del dispositivo SDR empleado. A pesar de ello, se ha podido comprobar mediante audición que la señal recibida y demodulada tiene una buena calidad subjetiva cuando se ha escuchado.

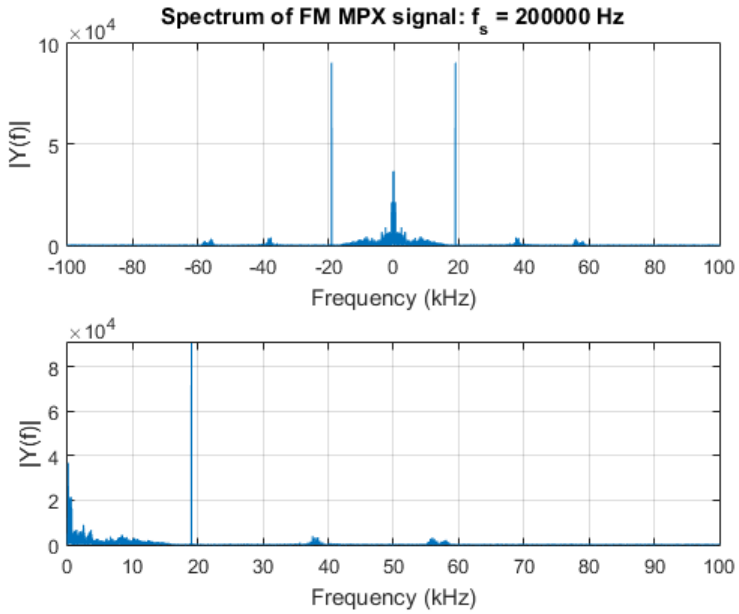


Figura 6.49 – *Espectro múltiplex la señal WFM recibida.*

Para comprobar que se transmite información digital, se ha representado la constelación de símbolos recibidos. En al figura 6.50 se observa que la constelación BPSK Diferencial recibida se ve más afectada por el ruido que para el caso de entrenamiento estudiado previamente. Sin embargo, el decodificador BPSK implementado ha logrado obtener el flujo binario correctamente puesto que se identifican con claridad las dos nubes de puntos correspondientes a los bits 1 y 0.

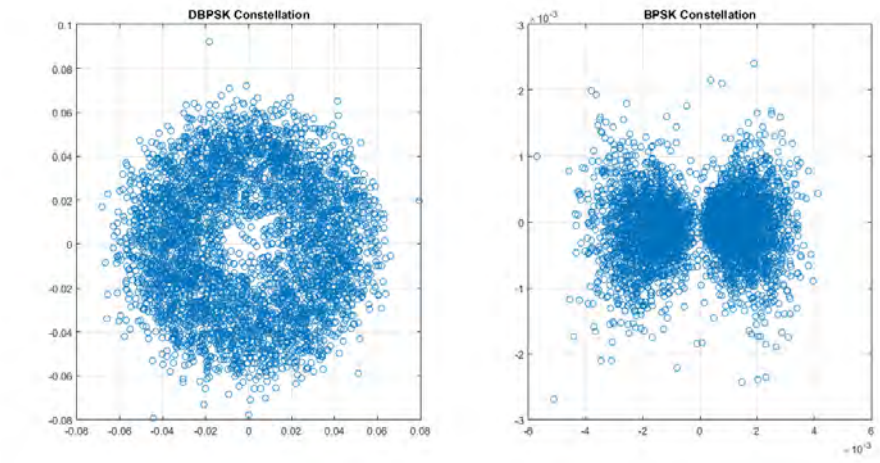


Figura 6.50 – *Constelación de símbolos recibida.*

Finalmente, se adjunta parte de los resultados obtenidos tras la decodificación de la información del servicio RDS. En primer lugar se identifica el nombre de la emisora, que en esta recepción se trata de ONDA CERO. En este grupo número 11 se puede ver que el grupo es de tipo 0 y versión A, lo que significa que la aplicación a la que está destinada a transmitir información básica de sintonización. Además, indica que se envían anuncios de tráfico sobre el estado de las carreteras y que el tipo de programa es informativo. Por otra parte, se sabe por el código de país que la señal se ha recibido en Rumanía, España o Suecia. El algoritmo no es capaz de determinar qué país es de los tres puesto que se utiliza el mismo identificador de país ya que su localización geográfica es tan alejada que resultaría imposible tener interferencias entre las señales. Existe un parámetro que indica cuál de los países se trata sin embargo en el tipo de tramas analizadas no se transmitía dicho parámetro. El ámbito en el que transmite la emisora también se recibe y es a nivel nacional.

```
--------------------------------------------------------------------------
Group 11
     - 0A: Basic tuning and switching information.
     - A traffic announcement is being broadcast on this programme at present.
     - Programme type: Information.
     - PI: Romania, Spain or Sweden.
     - No ECC: please record at least one minute of signal.
     - National programme: transmitted throughout the country.
     - Programme reference number assigned: EE (Hex-coding).
--------------------------------------------------------------------------
NAME OF THE BROADCASTING STATION: ONDA CERO
```

Figura 6.51 – *Información digital recibida en la secuencia binaria del servicio RDS.*

6

CAPÍTULO

7

RECEPTOR AX.25

El último capítulo práctico del Trabajo Fin de Máster se dedica a la implementación Software de un receptor de señales satelitales moduladas bajo el protocolo AX.25. Los conceptos necesarios para el desarrollo y diseño de este algoritmo se describen en el capítulo 5 del presente trabajo. Además, resulta muy útil el hecho de haber desarrollado el receptor WFM comercial en el capítulo anterior puesto que la codificación digital de las señales AX.25 es muy similar a la empleada en el servicio RDS.

De forma análoga al diseño del receptor de radio FM comercial, se han utilizado los dispositivos SDR RTL-SDR de *Allwin* y *FUNcube Dongle Pro+* tras el estudio de sus comportamiento bajo diferentes condiciones externas como el tiempo de trabajo continuado o la temperatura del dispositivo (capítulo 3). La herramienta Software de diseño seguirá siendo MatLab por los motivos descritos en 6.

A continuación se describirán las señales empleadas en la implementación y evaluación del receptor AX.25 así como el proceso seguido por la alumna para su diseño.

7.1 Señales APRS AX.25

El desarrollo del decodificador de señales satelitales transmitidas en el sistema APRS bajo el protocolo AX.25 se realiza a partir de una una señal real recibida a 1200 baudios a la frecuencia 144.7 MHz, parámetros típicos en Europa para la recepción de APRS.

Debido a la existencia de numerosas herramientas de decodificación de tramas AX.25 no

es necesario trabajar con una señal de evaluación teórica que compruebe el funcionamiento del demodulador AX.25. La evaluación del receptor Software implementado se realiza comparando los resultados obtenidos con los que proporciona la herramienta *SoundModem*, descrita en la sección 7.4.

La señal que servirá de ejemplo para el desarrollo del decodificador AX.25 se obtiene a través de la herramienta SDRSharp sintonizando el dispositivo *FUNcube Dongle Pro+* a 144.7 MHz. Tras ver el espectro de esta señal en el Software, se capta y almacena la señal recibida en una señal de audio con formato *.wav* del mismo modo que se hizo con la señal WFM en el anterior capítulo.

Se ha recibido APRS durante 25.7183 minutos con una frecuencia de muestreo de 44.1 kHz, obteniendo la señal de la figura 7.1. En ella se puede ver como la señal se recibe en banda base, realizando la demodulación en frecuencia el propio dispositivo SDR y entregando las muestras banda base en cuadratura.

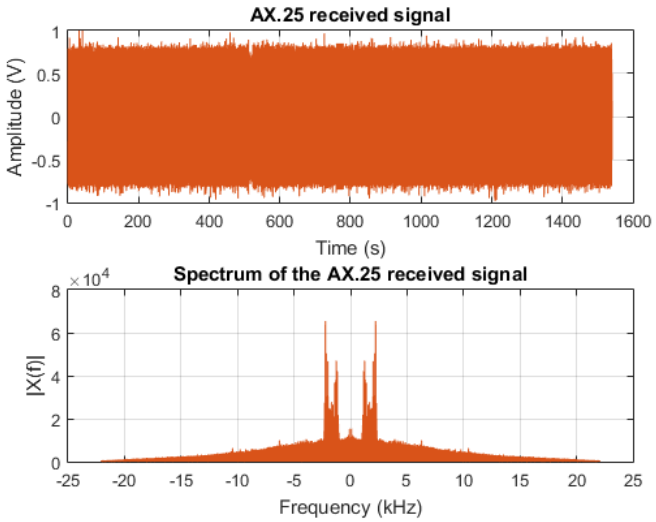


Figura 7.1 – *Señal APRS recibida por el receptor SDR.*

7.2 Demodulación de la señal AX.25

La señal recibida banda base es una señal de audio modulada digitalmente en frecuencia (AFSK). La figura 7.2 muestra un segmento de esta señal recibida, y en ella se puede ver que la señal modulada AFSK está compuesta por dos tonos a frecuencias diferentes. Debido la transmisión, esta señal se ve afectada en amplitud, por lo que se tiene una señal compuesta por dos tonos de diferente amplitud y frecuencia. Por tanto, la señal cambiará de estado cada vez que cambie el tono modulado por la información.

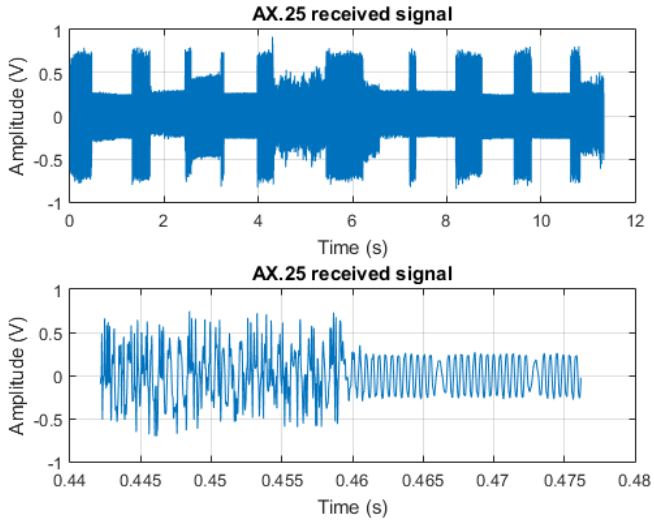


Figura 7.2 – *Segmento de la señal APRS recibida por el receptor SDR.*

Teóricamente estas frecuencias son 1200 Hz, denominada *marca*, y 2200 Hz, denominada *espacio*. Sin embargo cuando se recibe la señal se puede ver que ambos tonos varían levemente su frecuencia debido al ruido en la transmisión de la señal. El algoritmo calcula la frecuencia de cada tono extrayendo un segmento de señal de cada amplitud encontrada. Como resultado se obtiene la siguiente gráfica, en la que puede verse que el tono a menor frecuencia (1252 Hz) tiene una amplitud mayor, de 0.5 V y el tono de mayor frecuencia (2223 Hz) tiene una amplitud ligeramente más reducida de 0.2 V. La diferencia entre la frecuencia teórica y práctica es apenas de 52 Hz y 23 Hz para la marca y el espacio, respectivamente.

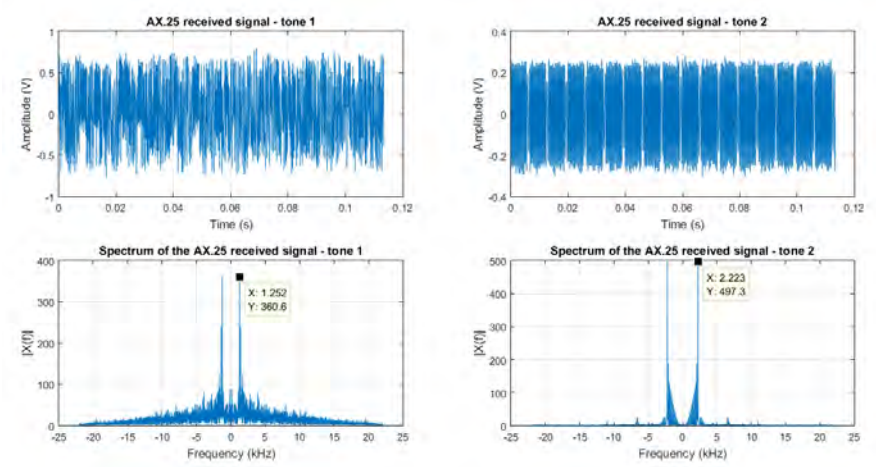


Figura 7.3 – *Segmento de los tonos que componen la señal APRS recibida.*

7.2.1 Demodulación AFSK

Tras comprobar que la señal se recibe modulada en AFSK, el decodificador se encarga de demodularla. Para ello, ya se vio en el capítulo 5 que se podía aproximar la demodulación AFSK a una demodulación AM considerando que la señal recibida está compuesta por una sola frecuencia, correspondiente a la marca (1200 Hz) y sufre variaciones en amplitud debido a la transmisión de la información.

El proceso de demodulación en amplitud es mucho más simple por lo que su implementación favorece el ahorro del coste computacional del receptor Software diseñado. Basta con rectificar la señal y filtrarla paso baja para obtener la envolvente de la señal y para ello, la alumna ha utilizado el comando de MatLab *amdemod*. Esta función es capaz de demodular la señal en amplitud tomando como parámetros de entrada la señal a demodular y su frecuencia (1200 Hz), la frecuencia de muestreo de la señal f_s y la función de transferencia del filtro paso bajo que rectifica la señal.

```
fc = 1200;
[num,den] = butter(10,fc*2/fs); % filtro paso bajo
signal_demod = amdemod(signal,fc,fs,0,0,num,den); % Demodulacion.

t_demod=0:Ts:(length(signal_demod)-1)*Ts;           % Temporary axis
signal_demod_spectrum=fftshift(fft(signal_demod));
f_demod=linspace(-fs/2,fs/2,length(signal_demod_spectrum)); % Frequency axis

if graphics==1
    figure;
    subplot(2,1,1)
    plot(t(1:5e5),signal(1:5e5)); grid on;
    hold on;
    stem(t_demod(1:5e5),signal_demod(1:5e5)); grid on;
    ylim([0 1])
    title('AFSK demodulated signal');
    xlabel('Time (s)'); ylabel('Amplitude (V)');

    subplot(2,1,2)
    plot(t(1.95e4:2.1e4),signal(1.95e4:2.1e4)); grid on;
    hold on;
    stem(t_demod(1.95e4:2.1e4),signal_demod(1.95e4:2.1e4)); grid on;
    ylim([0 1])
    title('AFSK demodulated signal');
    xlabel('Time (s)'); ylabel('Amplitude (V)');
    legend('Received signal','Demodulated signal')
end
```

Figura 7.4 – *Código fuente que implementa la demodulación AM.*

7

En la figura 7.5 se observa un segmento de las 500000 primeras muestras de la señal demodulada AFSK y de la señal recibida AX.25 para así observar mejor su comportamiento. Con esta comparativa, se puede ver fácilmente como se ha obtenido la envolvente de la señal, que indica los niveles altos y bajos de la señal correspondientes con la marca y el espacio respectivamente.

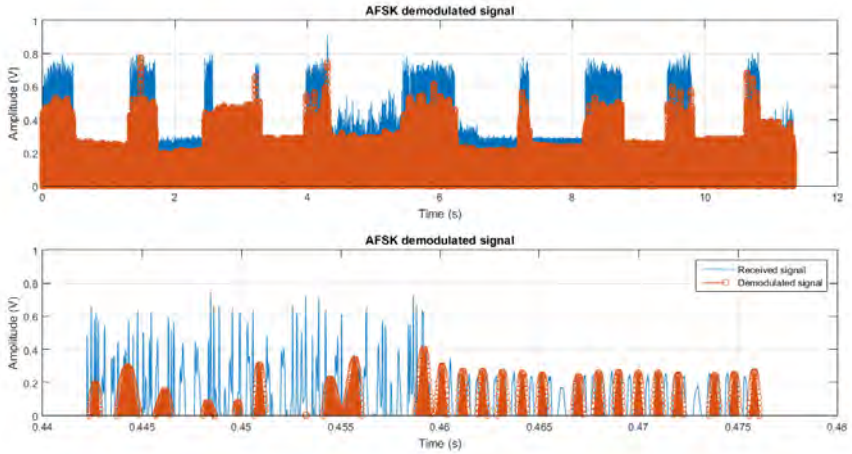


Figura 7.5 – *Señal demodulada AFSK.*

7.2.2 Decodificación NRZ-I

Demodulada la señal, el algoritmo obtiene la secuencia binaria decidiendo que nivel de amplitud asigna al bit 0 y qué nivel de amplitud asigna al bit 1. Para ello, implementa un decodificar NRZ-I que, como se describió en el capítulo 5, asigna el 1 lógico al nivel alto de señal correspondiente con la marca (1200 Hz) y el bit 0 al nivel bajo de señal correspondiente con el espacio (2200 Hz).

Discretiza la señal cada 75 muestras y decide el bit correspondiente al nivel de señal de cada periodo de muestra. Este flujo binario está codificado con el código NRZ-I por lo que será necesario decodificarlo. El código codifica los cambios de señal de forma que se cambia el estado o nivel de señal cuando el bit de datos a transmitir es un 0 y se mantiene cuando se transmite un 1 lógico. El código de la figura 7.6 realiza este proceso inverso:

En la figura 7.7 se representan los primeros 500000 bits de l secuencia recibida para demostrar que se ha rcibido información digital en al señal de audio transmitida mediante el sistema APRS bajo el protocolo AX.25.

```
toneH=signal(1e4:1.5e4);
toneL=signal(2.5e4:3e4);

% Genera el flujo binario.
signal_demod=signal_demod(:,1);
bits_nrzi=[];
for i=1:75:length(signal_demod)
    if (signal_demod(i)>max(toneL))      % Nivel alto (1200 Hz)
        bits_nrzi(i)=1;
    else                                 % Nivel bajo (2200 Hz)
        bits_nrzi(i)=0;
    end
end

% Se decodifica el código NRZ-I.
bitstream=[];
for i=length(bits_nrzi):-1:2
    bitstream(i)=bits_nrzi(i)==bits_nrzi(i-1);
end
if bits_nrzi(1)==1
    bitstream(1)=0;
else
    bitstream(1)=1;
end
```

Figura 7.6 – *Código fuente que implementa el decodificador NRZ-I.*

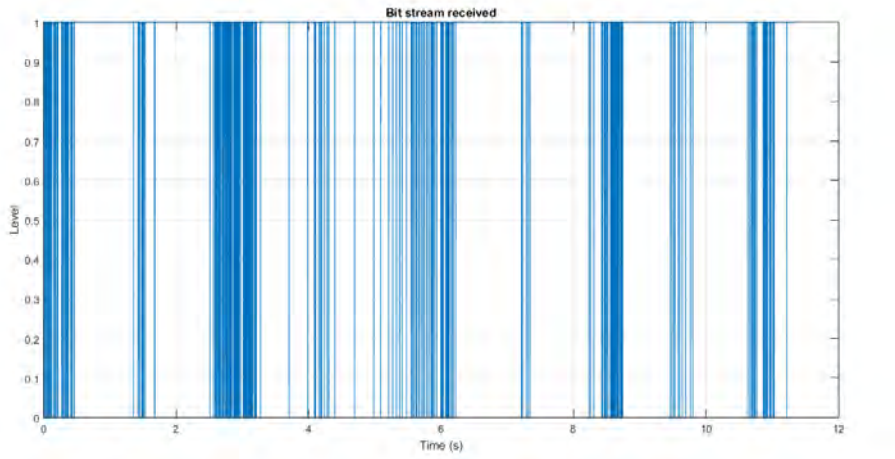


Figura 7.7 – *Segmento de la secuencia binaria recibida.*

A continuación, el decodificador implementado deshace la técnica *bit stuffing* realizada en el transmisor para evitar que el campo *flag* pueda confundirse con los propios bits de datos de la trama. Esta técnica inversa consiste en localizar grupos de bits formados por 5 1's lógicos seguidos y comprobar el bit que se recibe a continuación. Si se trata de un 0 lógico se debe descartar este bit puesto que se incluyó en el transmisor mediante esta técnica.

```
% Deshacemos el bit stuffing
% Buscamos secuencias de 5 bits 1 seguidos para eliminar el 0 que se
% transmite después. Recorro el flujo de datos y cuando me encuentro 5 bits
% 1 seguidos miro el siguiente bit, si se trata de un cero lo elimino y si
% se trata de un 1 significa que es la bandera.
bitstream_sincr=[];
if (bitstream(1:6))==[1 1 1 1 1 0]  % Tenemos 5 bits seguidos de un 0
    bitstream_sincr=[bitstream_sincr bitstream(1:5)];
else
    bitstream_sincr=[bitstream_sincr bitstream(1:6)];
end
for i=2:length(bitstream)-5
    if sum((bitstream(i:i+5))~=[1 1 1 1 1 0])==1  % Tenemos 5 bits seguidos de un 0
        bitstream_sincr=[bitstream_sincr bitstream(i+5)];
    end
end
```

Figura 7.8 – *Código fuente que deshace el bit stuffing.*

Por último y antes de pasar a interpretar la información contenida en el flujo de bits, el estándar [30] indica que las tramas AX.25 están formadas por grupos de bytes por lo que se agrupa el flujo de bits recibido de 8 en 8. Además, especifica que los bits se han enviado en orden *LSB*, es decir, transmitiendo primero el bit menos significativo de cada grupo de 8 bits. Por este motivo, el algoritmo agrupa los bits en bytes y los invierte para que la recepción se realice en el mismo orden que la transmisión. En este punto del algoritmo, comprueba que la decodificación es correcta puesto que se obtienen grupos de 8 bits y no sobra ni falta ninguno.

```
% Agrupamos el flujo binario en bytes y reordenamos los bits ya que la
% transmisión se hace en orden LSB, transmitiendo primer oe lbit menos
%significativo de cada grupo.
bytestream=[];
n=1;
for i=1:8:length(bitstream_sincr)-8
    bytestream(n,:)=fliplr(bitstream_sincr(i:i+7));
    n=n+1;
end
```

Figura 7.9 – *Código fuente que obtiene el flujo de bytes recibido.*

7.3 Decodificador de la información de la señal AX.25

Una vez que se tiene el flujo de bytes, el decodificador lo divide en tramas localizando el primer campo de cada trama, denominado *bandera* o *flag*, puesto que tiene un valor fijo *0111 1110*. Una vez que se ha separado por tramas la secuencia de bytes recibida, el algoritmo localiza cada uno de los campos definidos en el capítulo 5 mediante el código de la figura 7.10

```
% División en tramas.
[Nbytes, bits_byte]=size(bytestream);
% Localizamos la bandera
n=1;
trama=[];
for i=1:Nbytes
    if sum(bytestream(i,:)==flag)==bits_byte    % Es la bandera (INICIO)
        trama(n,:)=[trama(n,:) bytestream(i,:)];
        for j=i:Nbytes
            trama(n,:)=[trama(n,:) bytestream(j,:)];
            if sum(bytestream(j,:)==flag)==bits_byte % Es la bandera (FIN)
                j=Nbytes;   % sale del bucle ya que tengo una trama completa
            end
        end

        % Campo de dirección - Del byte 2 al 8 para el destinatario y del 9
        % al 14 para la fuente
        address_dest(n,:)=trama(n,9:72);        % 8 bytes
        address_orig(n,:)=trama(n,73:120);      % 8 bytes
        Nrepetidores=0;
        address_rep=[];
        for k=121:8:560     % Bytes del 15 al 70 opcionales
            % Si el SSID vale 1 no se transmiten los identificadores de los
            % repetidores
            if trama(n,k-1)==1
                % Campo de control
                control(n,:)=trama(n,k:k+7);    % 1 byte de control
                lastbit_control=k+7;
                k=560;
            else
                address_rept(n,:)={address_rept trama(n,k:k+7)};
                Nrepetidores=Nrepetidores+1;
            end
        end

        % Campo de información
        info(n,:)=trama(n,lastbit_control:lastbit_control+(256*8)-1);   % 256 bytes

        % Campo FCS - 2 bytes
        FCS(n,:)=trama(lastbit_control+(256*8):lastbit_control+(256*8)+16);

    end
    n=n+1;
end
```

Figura 7.10 – *Código fuente que estructura el flujo de bytes recibido en tramas y campos.*

Hecho esto, comprueba el campo de verificación de errores para determinar si la recepción es errónea y se necesita la retransmisión del paquete de datos AX.25 mediante el proceso descrito en el capítulo 5 y mostrado en 7.11.

Por último, el algoritmo interpreta cada byte de información de los campos estructurados previamente para así obtener la información recibida en la trama y poder mostrarla en texto plano por pantalla. En la siguiente figura se muestran las dos primeras tramas decodificadas por el receptor implementado en Software.

```
[Ntramas,bits_trama]=size(trama);

% Verificación de trama (FCS)
polinomy='0x8408';
CRCini='0xFFFF';
FCSini=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
% Función XOR binaria
xorini=xor(CRC,polinomy);
for i=1:Ntramas
   for j=1:7
       % Desplaza una posición a la derecha el CRC
           FCSini=[ 0 FCSini(j+1:end)];
           CRC=num2hex(FCSini);
       if sum(nxorbin==[0 0 0 0 0 1])==0
           % XOR con el polinomio
           xordesp=xor(CRC(1),polinomy);
       end
   end
   % XOR con el polinomio
   CRC=xor(CRC(1),CRCini);
end
```

Figura 7.11 – *Código fuente que verifica si hay errores de transmisión.*

```
------------------------------------------------------------
AE6MP SS5PPQ-2 R F0 UI
------------------------------------------------------------
KB6CYS BEACON R F0 UI
------------------------------------------------------------
```

Figura 7.12 – *Ejemplo de las dos primeras tramas decodificadas en MatLab por el algoritmo implementado.*

La figura 7.12 muestra dos tramas de información de tipo *UI* (trama de información no numerada) con un valor hexadecimal de PID F0, lo que indica que no se ha implementado el protocolo de capa de enlace (capa 3). También puede observarse que el campo de control toma el valor R lo que indica que el sistema está listo para recibir y las direcciones del destinatario y la fuente.

Para realizar las pruebas del algoritmo, debido a que la señal recibida tiene un elevado número de muestras se ha recortado para recibir menos tramas de datos y así facilitar el manejo de los datos a la hora de realizar las pruebas pertinentes durante el desarrollo e implementación del algoritmo anterior.

7.4 Evaluación del receptor AX.25

Finalmente, se evalúa el funcionamiento del algoritmo implementado mediante la herramienta *SoundModem*. Este Software es un módem de baja velocidad (1200 baudios) que se encarga de decodificar las tramas de datos AX.25 moduladas en AFSK. Con él, la alumna ha decodificado la señal recibida para comparar los resultados obtenidos con los ofrecidos por esta herramienta.

El programa *SoundModem*, figura 7.13, recibe la señal de audio *.wav* con información APRS a través del asistente virtual de audio *Virtual Audio Cable*, que se encarga de establecer un enlace virtual entre la salida de los altavoces y la entrada del micrófono del PC.

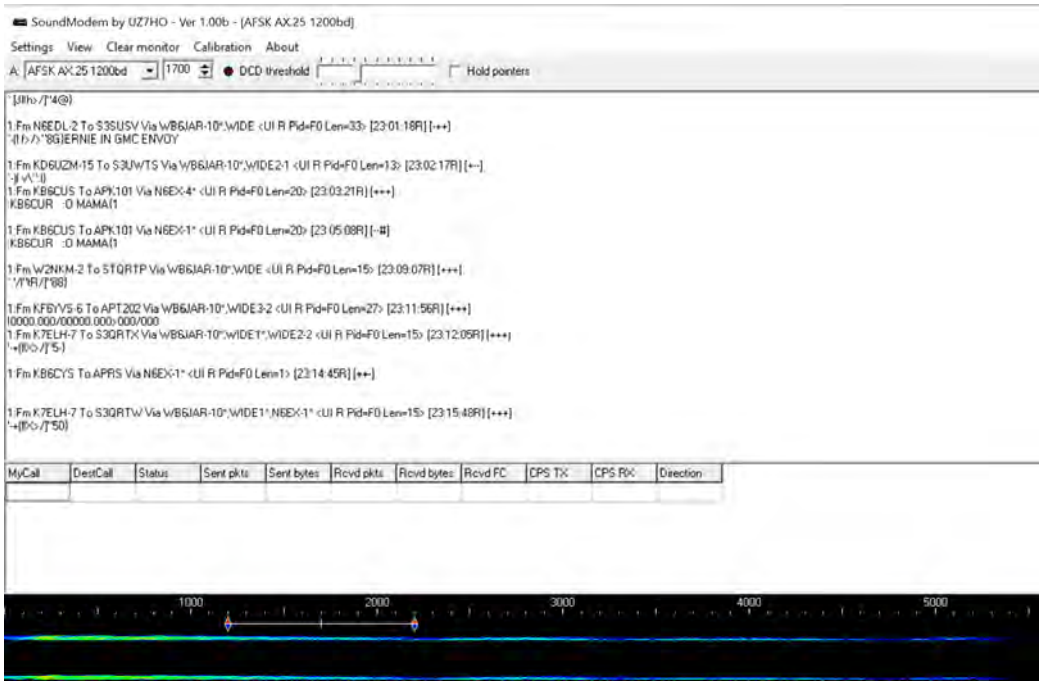


Figura 7.13 – *Interfaz gráfica de la herramienta SoundModem durante la decodificación de tramas AX.25.*

En la figura se muestra la decodificación de las tramas de información, dónde se comprueba que se reciben los mismos datos que se obtenían a través del decodificador AX.25 implementado en Software.

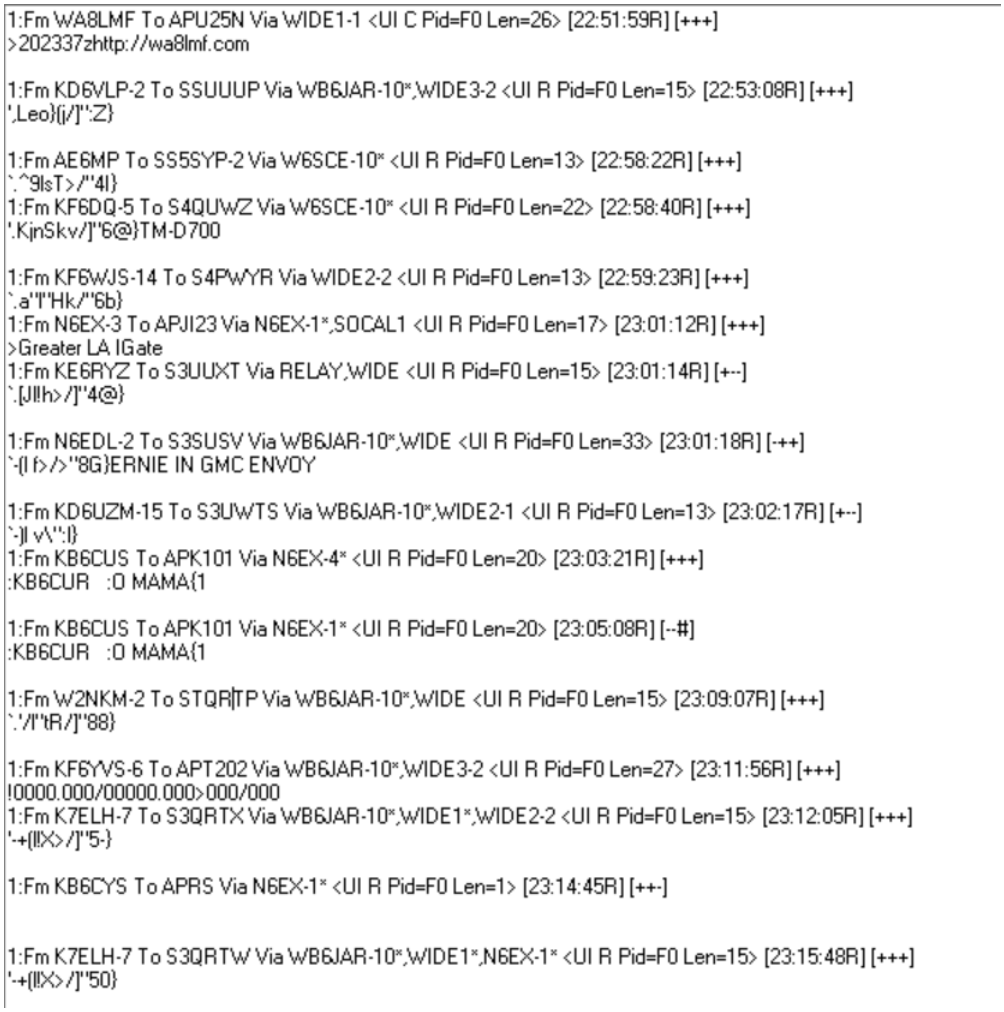


Figura 7.14 – *Información decodificada por el Software SoundModem.*

7

CAPÍTULO

8

CONCLUSIONES Y TRABAJO FUTURO

En este proyecto se ha demostrado las bondades de la tecnología Software-Defined Radio a través de la implementación de dos decodificadores de señales analógicas y digitales. Principalmente, el hecho de implementar en Software un demodulador de señales satelitales y decodificador de las tramas AX.25 contenida en ellas permite acercar la tecnología aeroespacial, excesivamente costosa en la mayoría de los casos, a los usuarios de a pie e incluso a los proyectos académicos. Este hecho ha sido posible gracias a la integración de dos filosofías: el procesamiento clásico de señal y la tecnología emergente de la Radio Definida por Software. Con el presente Trabajo Fin de Máster ha quedado demostrado la posibilidad de implementar prácticamente cualquier sistema de radiofrecuencia a partir de un dispositivo SDR y unos conocimientos de procesado de señal y de programación previos.

En lo que respecta al trabajo técnico del proyecto, puro de un ingeniero de las telecomunicaciones, se pueden realizar diferentes afirmaciones. La primera de ellas es el hecho de poder implementar un sistema de RF, hasta los últimos tiempo muy costoso en cuanto a Hardware, a través de herramientas al alcance de cualquiera. Este hecho resulta increíble y es la base que sustenta la exponencial evolución de la tecnología Software-Defined Radio puesto que no se requieren grandes inversiones económicas iniciales para desarrollar un sistema de este tipo. Como se ha visto a lo largo del trabajo, existen dispositivos SDR de bajo coste con grandes prestaciones a pesar de su reducido precio. En los primeros capítulos del documento se pudo ver cómo un receptor RTL-SDR de apenas 12 € era capaz de recibir señales de RF con un error mínimo. Este error, debido a la

desviación en frecuencia provocada por la poca precisión de alguno de sus componentes (cristal de cuarzo), se puede mitigar fácilmente tras un estudio del dispositivo y su calibración en función de parámetros críticos como el tiempo de trabajo continuado o la temperatura del dispositivo. Además, se ha evaluado un dispositivo de altas prestaciones y un coste algo mayor que el anterior pero igualmente asequible por un usuario aficionado. Este receptor *FUNcube Dongle Pro+* permite la recepción de señales mediante una sintonización de señal prácticamente exacta, como ha quedado demostrado en el capítulo 3.

Ambos dispositivos permiten la recepción de señales con buenas prestaciones. Sin embargo, ha quedado demostrado que el receptor *FUNcube Dongle Pro+* ofrece características que el receptor RTL-SDR no alcanza, como la prácticamente nula desviación en frecuencia o la recepción de señales minimizando el ruido y las interferencias del medio de transmisión con ganancias muy elevadas.

Por otra parte, ha quedado claro en los últimos capítulos la flexibilidad ofrecida por la tecnología Software-Defined Radio a la hora de implementar cualquier tipo de sistema. Partiendo de unos conocimientos previos de procesamiento de señales digitales y analógicas cualquier persona puede montar su propio equipo de radioaficionado para la recepción de señales, tanto de la radio FM comercial como telemetrías y otras informaciones de los satélites, en este caso de estudio. Esta idea de desarrollar un Software que decodifique las señales WFM y AX.25 es extrapolable a la recepción de cualquier tipo de señal de RF recibida en el dispositivo SDR.

Los algoritmos implementados en el presente trabajo suplen la necesidad del proyecto en el que colabora la alumna de realizar el procesamiento de las señales recibidas. Relacionado con esto, se sugiere como líneas futuras e incluso como trabajo para finalizar el Máster en Ingeniería Informática, debido a las competencias en lenguajes de programación de un graduado en esta ingeniería, lo siguiente. Se propone la migración de los algoritmos desarrollados en este proyecto a la aplicación web *GranaSDR* para su posterior entrada al mercado como uno de los primeros Software con licencia libre que permite la recepción y decodificación de todo tipo de señales de RF.

A lo largo del trabajo, se han encontrado también algunas dificultades. Principalmente en los recursos Software utilizados por la alumna al inicio del proyecto. Los receptores RTL-SDR con los que se comenzó a trabajar recibían señales muy ruidosas debida a que la potencia de recepción era escasa. Tras diferentes validaciones de estos dispositivos se apartaron y se compraron otros modelos de receptores, también de bajo coste, pero de fabricación más actual. El principal motivo de fallo de los receptores descartados fue el hecho de haber trabajado con ellos durante demasiado tiempo, lo que provocó ciertos daños irreparables en algunos de sus componentes electrónicos. Debido al frecuente uso de estos receptores en el laboratorio de electrónica, llegó un momento en el que el dispositivo aumentaba tanto su temperatura que los componentes encargados de la sintonización de señal se volvían inestables y era prácticamente imposible mantener estos parámetros fijos.

Respecto al Software se encontraron ciertos problemas a la hora de trabajar con señales con un número de muestras muy elevado. A priori, se puede pensar que esta problemática se

soluciona captando señales durante menor tiempo. Pero esto no es viable ya que la mayoría de información digital transmitida en las tramas, tanto de datos AX.25 como en el RDS de la FM comercial, requiere una recepción mínima de bits para poder ser decodificada.

Como conclusión final, la alumna ha adquirido conocimientos en una tecnología desconocida para ella como es la Radio Definida por Software y ha profundizado en los conocimientos de procesamiento de señal y programación adquiridos durante su etapa formativa. Además, ha sido capaz de comprender el funcionamiento de un dispositivo Hardware a través del estudio y evaluación de sus componentes. De este modo, se han cumplido satisfactoriamente los objetivos marcados, tanto en el terreno personal como profesional de la alumna, en cuanto a la realización de un trabajo experimental con productos reales y el aprendizaje de conceptos que abarcan las principales ramas de las Telecomunciaciones.

8

REFERENCIAS

[1] Acerca de e4000. Página Web. https://www.passion-radio.com/sdr-receivers/rtlsdr-e4000-360.html.

[2] Allwin usb dvb-t rtl-sdr realtek rtl2832u y r820t tuner receiver dongle pal iec input. Página Web. https://www.lazada.com.ph/products/allwin-usb-dvb-t-rtl-sdr-realtek-rtl2832u-r820t-tuner-receiver-dongle-pal-iec-in
html.

[3] Andoer® mini antena de tv sintonizador receptor soporte a dvb-t + dab + fm + sdr rtl2832u + r820t digital portátil usb 2.0. Página Web. https://www.amazon.es/Andoer%C2%AE-Sintonizador-Receptor-RTL2832U-Port%C3%A1til/dp/B013Q93X2I/ref=sr_1_10/259-6684206-2633449?ie=UTF8&qid=1535794193&sr=8-10&keywords=rtl-sdr&th=1.

[4] Biografía de dr. joe mitola iii. Página Web. https://www.wirelessinnovation.org/assets/documents/hall_mitola.html.

[5] Comunidad de satélites aficionados de reino unido. Página Web. https://amsat-uk.org/about/.

[6] Demodular afsk, desde cero. Página Web. http://electronicayciencia.blogspot.com/2017/10/demodular-afsk-desde-cero.html.

[7] Especificaciones dongle funcube. Página Web. http://www.funcubedongle.com/?page_id=1201.

[8] Flexradio systems. Página Web. https://www.flexradio.com/.

[9] Fm estéreo. Página web. https://es.wikipedia.org/wiki/FM_est%C3%A9reo#La_se%C3%B1al_m%C3%BAltiplex_(MPX).

[10] fmmod matlab. Página Web. https://es.mathworks.com/help/comm/ref/fmmod.html.

[11] Frecuencias de radio am y fm. Página web. http://hyperphysics.phy-astr.gsu.edu/hbasees/Audio/radio.html.

[12] The funcube dongle. Página web. http://www.funcubedongle.com/MyImages/FCDAnIntroduction.pdf.

[13] Funcube dongle pro + (a20). Página Web. http://funcubedongle.3dcartstores.com/FUNcube-Dongle-Pro-A20_p_27.html.

[14] Grupo de electrónica aeroespacial granasat. Página Web. https://granasat.ugr.es/.

[15] Guía de inicio rápido del rtl-sdr. Página Web. https://www.rtl-sdr.com/rtl-sdr-quick-start-guide/.

[16] kalibrate-rtl. Página Web. http://rtl-sdr.sceners.org/?p=193.

[17] ModulaciÓn de frecuencia - fm. Página web. https://www.textoscientificos.com/redes/modulacion/frecuencia.

[18] Per vices | high performance software defined radio products. Página Web. https://www.pervices.com/.

[19] Proyecto software openbts. Página Web. http://openbts.org/.

[20] ¿que es aprs? Página Web. http://www.radioaficioncr.net/2013/02/normal-0-21-false-false-false.html.

[21] Radio fm. Página web. https://es.wikipedia.org/wiki/Radio_FM.

[22] Receptor flex-5000 de flexradio systems. Página Web. https://www.astroradio.com/271500.html.

[23] rtl-sdr. Página Web. https://osmocom.org/projects/rtl-sdr/wiki/Rtl-sdr.

[24] Rtl2832u. Página Web. http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257.

[25] Spectral analysis with rtl-sdr radio. Página Web. https://es.mathworks.com/help/supportpkg/rtlsdrradio/examples/spectral-analysis-with-rtl-sdr-radio.html.

[26] Tema 3: espectro y modulación desarrollo en serie de fourier. Página web. http://bazica.org/tema-3-espectro-y-modulacin-desarrollo-en-serie-de-fourier.html?page=3.

[27] Transceptores. Página Web. https://granasat.ugr.es/2017/10/21/transceivers/.

[28] Institute of electrical and electronics engineers. Página Web, 1963. https://www.ieee.org/.

[29] Wireless innovation forum. Página Web, 1996. http://www.wirelessinnovation.org.

[30] Ax.25 link access protocol for amateur packet radio. Estándar, Julio 1998.

[31] Specification of the radio data system rds for vhf/fm sound broadcasting in the frequency rango from 87,5 to 108,0 mhz. Estándar Europeo, Abril 1998.

[32] Sitio web oficial de autosar. Página Web, Abril 2011. http://www.autosar.org/.

[33] Gnu radio companion. Página Web, 2013. https://wiki.gnuradio.org/index.php/Main_Page.

[34] R820t. Datasheet, 2013. https://www.rtl-sdr.com/wp-content/uploads/2013/04/R820T_datasheet-Non_R-20111130_unlocked1.pdf.

[35] Ettus research. Página Web, 2015. http://www.ettus.com/.

[36] Hackrf one. Página Web, 2015. https://greatscottgadgets.com/hackrf/.

[37] Ieee spectrum. samsung, nokia show 5g tech at ni week. Página Web, 2015. https://spectrum.ieee.org/tech-talk/at-work/test-and-measurement/samsung-nokia-show-5g-tech-at-ni-week.

[38] Realtek rtl2832u. Datasheet, 2015. http://datasheetcafe.databank.netdna-cdn.com/wp-content/uploads/2015/09/RTL2832U.pdf.

[39] Rtl-sdr blog. Página Web, 2015. http://www.rtl-sdr.com/.

[40] Decodificación de satélites digitales. Temario de asignatura, Septiembre 2017.

[41] Algora, C. M. G. Radio definido por software usando matlab. Trabajo de diploma, Universidad Central "Marta Abreu" de Las Villas, Santa Clara, 2011. Ingeniería de Telecomunicación.

[42] Ariel Ricardo Moncaleano Ospina, J. D. J. P. A. y. E. E. G. G. DiseÑo e implementaciÓn de un analizador para el protocolo ax.25. *Universidad Distrital Francisco José de Caldas 4*, 2 (Diciembre 2013), 70–81.

[43] Consultancy, J. G. Sdr market study. task 1: Market segmentation and sizing. *The SDR Forum* (Marzo 2005).

[44] Cook, P. G., and Bonser, W. Architectural overview of the speakeasy system. *Revista IEEE en áreas seleccionadas en comunicaciones 17*, 4 (Abril 1999), 650–661.

[45] Cruz, O. M. S. *Desviación de fase, el índice de modulación y la desviación de frecuencia*. U.T.N. Facultad Regional Córdoba - Depto. Electrónica, 2010.

[46] Domínguez, I. P. Software defined radio: Usrp y plataformas de desarrollo. Proyecto fin de grado, Universidad de Sevilla, Sevilla. Ingeniería de Telecomunicación.

[47] e Iván Pinar Dominguez, J. J. M. F. Laboratorio de comunicaciones digitales radio definidas por software.

[48] Esquivel, A. P. Diseño de sistema de comunicaciones usando software defined radio. Proyecto fin de grado, Universidad de Sevilla, Sevilla, 2015. Ingeniería de Telecomunicación.

[49] et al., C. P. V. *Sistemas de Telecomunicación*. Universidad de Cantabria, 2007.

[50] González, M. P. Estudio piloto de los demoduladores de la serie rtl de realtek para la radio definida por software. Proyecto fin de carrera, Universidad de Sevilla, Sevilla, 2014. Ingeniería de Telecomunicación.

[51] Izaguirre, A. Z. Radio data system rds. *EUSKAL HERRIKO UNIBERTSITATEA UNIVERSIDAD DEL PAIS VASCO* (Julio 1996).

[52] Lackey, R. I., and Upmal, D. W. Speakeasy: the military software radio. *Revista IEEE en áreas seleccionadas en comunicaciones 33*, 5 (Mayo 1995), 56–61.

[53] Menna, B. Aprs-tramas ax.25. Página Web. https://es.slideshare.net/bmenna/aprs-y-tramas-ax25.

[54] Mesa, I. A. A. DiseÑo, construcciÓn e integraciÓn de un mÓdulo de interfaz digital analogo para sistemas repetidores de globo sonda usados en situaciones de emergencia. Trabajo fin de carrera, Universidad de Chile, Santiago de Chile, 2011.

[55] Morillas, J. L. Servicio web para satélites meteorológicos noaa basado en receptor sdr. Trabajo fin de máster, Universidad de Granada, Granada, 2017. Ingeniería de Telecomunicación.

[56] Reed, J. Software radio: A modern approach to radio engineering.

[57] Rubenstein, R. Software-defined radio: Softly does it. *Technology Trends* (Marzo 2011).

[58] Zevallos, G. Fm modulation and demodulation of three simultaneous musical tones. Curso: Ce 0601 telecomunicaciones i, Universidad Ricardo Palma, Lima, Perú, 2011.

ANEXO

A

CALIBRACIÓN DEL *RTL-SDR*

El calibrado de un dispositivo SDR consiste en calcular la desviación en frecuencia medida en ppm de forma que se ajuste el oscilador para la sintonización a la frecuencia RF deseada. El receptor SDR se calienta a medida que aumenta el tiempo de trabajo por lo que es recomendable dejar en funcionamiento el dispositivo cierto tiempo antes de calibrarlo, por ejemplo unos 20 minutos.

Las técnicas de calibración del receptor RDS son muy variadas y a continuación se describen las utilizadas por la alumna.

A.1 Software *kalibrate*

El Software Open-Source *kalibrate* es una aplicación de consola disponible en los sistemas operativos GNU/Linux y Windows y descargable en su web oficial [16].

Debido a que se trata de una aplicación de consola se trabaja desde la terminal o ventana de comandos del sistema operativo utilizado, en el caso de la alumna Windows [55]. Los pasos a seguir para calibrar un dispositivo SDR son los siguientes:

1. *Directorio de instalación.* En primer lugar el usuario debe acceder desde la terminal al directorio donde se ha guardado el Software Open-Source *kalibrate*.

2. *Conexión del dispositivo SDR.* ES importante conectar el dispositivo al ordenador a través de su interfaz USB y a la antena receptora de TV mediante la interfaz restante.

3. *Instalación del driver correspondiente.* Antes de realizar la calibración del dispositivo, se debe comprobar que el PC reconoce el dispositivo. El sistema operativo Windows no reconoce los receptores de bajo coste por lo que hay que instalar sus drivers manualmente. A pesar de que esto solo ocurre con el es necesario comprobarlo con todos los dispositivos SDR que se utilicen.

   Para que Windows reconozca el dispositivo se utiliza el Software *Zadig* que se encarga de instalar los drivers necesarios para ello. Esta herramienta lista los dispositivos USB conectados al PC, donde se seleccionará el receptor denominado como *Bulk-In, Interface (Interface 0)*, *RTL2832UHIDIR* o *RTL2832U*. Es importante que se verifique que el *ID* del USB es *0BDA 2838 00*, correspondiente con el sistema [15].

   Por otra parte se selecciona el controlador que hay que instalar, denominado *WinUSB*, tal y como puede verse en la figura A.1. Esta figura muestra una captura de la herramienta *Zadig*, en la que se ve que el controlador actualmente instalado es *RTL2832UUSB* y el que se desea instalar es *WinUSB* [15].

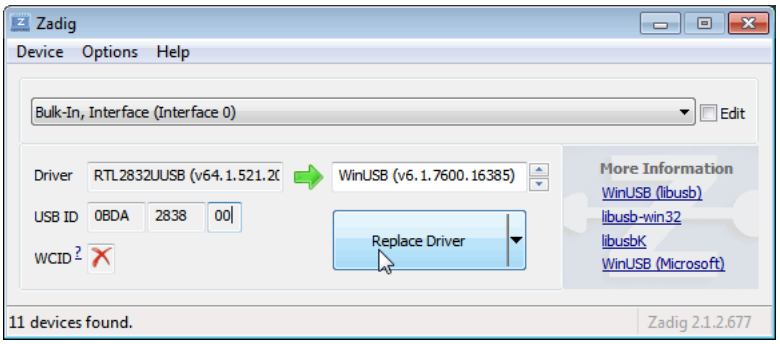


Figura A.1 – *Instalador de controladores Zadig.*

4. *Búsqueda de la banda GSM.* El Software Open-Source *kalibrate* utiliza la señal de una estación móvil GSM para calibrar el receptor [55]. Para ello, se utiliza el comando *kal*, que escanea la banda GSM definida. En Europa se escanea la banda GSM900. Este comando se puede acompañar de los parámetros de la tabla A.1. De todos ellos, es imprescindible especificar la banda de escaneo *GSM900* y de forma opcional se puede indicar la ganancia de la antena de recepción o el error en frecuencia inicial del dispositivo. Algunos ejemplos de uso de este comando son:

   kal -s GSM900 escanea las frecuencias de dicha banda

   kal -g 42 -e 22 -s GSM900 escanea la banda con un error incial de 22 ppm una ganancia de recepción de 42 dB

5. *Selección del canal GSM de mayor potencia.* Tras escanear la banda de frecuencias aparece un listado de los canales recibidos con la potencia de recepción. Es importante escoger el canal recibido con mayor potencia y que ésta sea elevada (al menos 50 kW) para que la recepción sea robusta [55].

A

Parámetro	Descripción
-s	Banda de escaneo: GSM850, GSM900, EGSM, DCS o PCS
-f	Frecuencia cercana a la banda
-c	Canal de la estación base GSM
-g	Ganancia en dB
-b	Band indicator: GSM850, GSM900, EGSM, DCS o PCS
-d	Indicador de dispositivo RTLSDR
-e	Error de frecuencia inicial en ppm
-v	Verbose
-D	Debug activo
-h	Ayuda sobre el comando

Tabla A.1 – *Parámetros del comando kal del Software kalibrate.*

Una vez localizado el canal con mayor potencia se obtiene la desviación en frecuencia del receptor SDR con el comando *kal* y el parámetro de la tabla A.1 *"-c"* para escoger el canal. Al igual que antes, opcionalmente se puede incluir el error inicial y el *flag* para que muestre por pantalla las operaciones realizadas [55].

kal -c 23 escanea el canal 23 de GSM900 para obtener la desviación en frecuencia

kal -e 41 -c 23 -v escanea el canal 23 de GSM900 con un error inicial de 42 ppm y muestra en pantalla las operaciones realizadas para obtener la desviación en frecuencia

A continuación se muestra un ejemplo de calibración para un receptor RTL-SDR. Tras 20 minutos de trabajo del dispositivo se ha procedido a su calibración con el Software. En primer lugar, se accede al directorio donde se encuentra el ejecutable de la herramienta y se escanea la banda de frecuencia GSM900 mediante el comando *kal -g 42 -e 22 -s GSM900*. Tras el escaneo de canales, se observa en la figura A.2 que el canal recibido con mayor potencia es el canal 109, con 96903.36 W por lo que se recibirá la señal de dicho canal para obtener la desviación en frecuencia con el comando *kal -e 41 -c 109 -v* (Figura A.3).

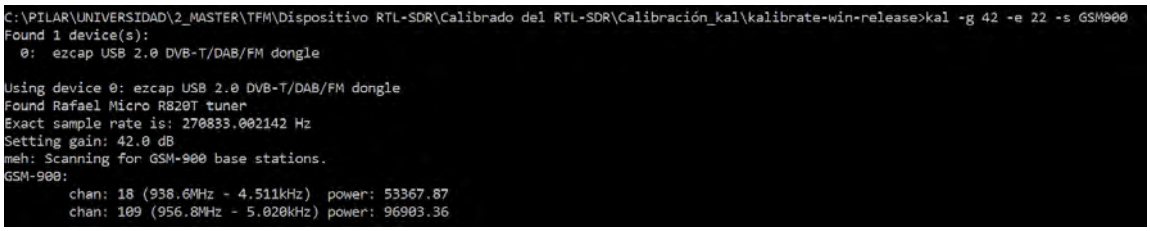
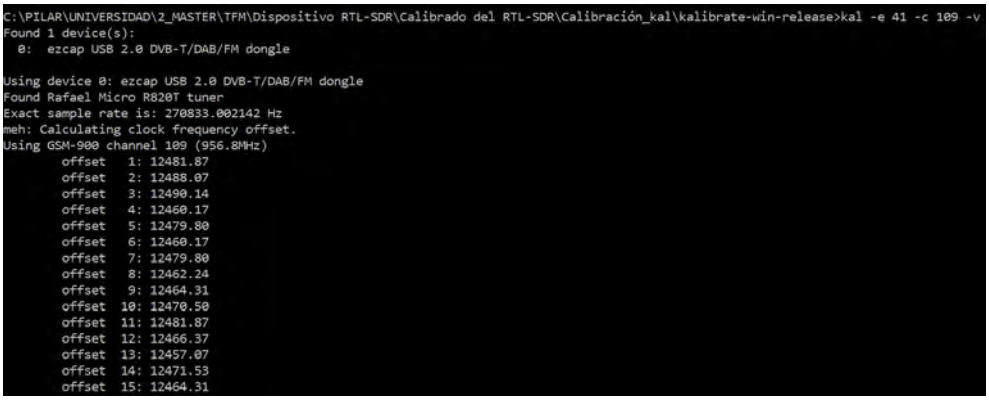
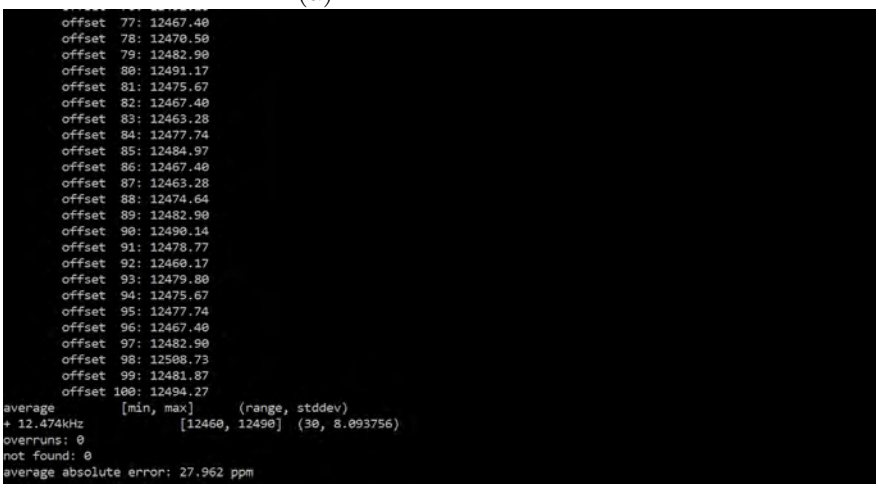


Figura A.2 – *Escaneo de señales con el Software kalibrate.*



(a)



(b)

Figura A.3 – *Calibrado del receptor a partir de la recepción del canal 109 de GSM900.*

De este modo, puede observarse que el receptor RTL-SDR tiene una desviación en frecuencia de 27.962 ppm. Con este ejemplo puede verse lo sencillo que es la calibración del dispositivo con esta herramienta obteniendo resultados exactos. Sin embargo, este Software solo está operativo para el modelo de SDR RTL-SDR.

A.2 Software *SDRSharp*

Otra técnica de calibración del dispositivo SDR es utilizar el popular Software SDRSharp. Para ello es necesario, además del receptor SDR, un generador de señal. En concreto, se ha utilizado el generador de señal del laboratorio del grupo GranaSAT, *Marconi Instrument R2955A* como el de la figura A.4.

En primer lugar se conectará el receptor SDR al PC a través de su interfaz USB y al generador de señal mediante un cable coaxial. En este equipo se genera un tono a una frecuencia de RF de 300 MHz y a un nivel de amplitud de -100 dBm, como se muestra en la figura A.5 [55].

Figura A.4 – *Generador de señal Marconi Instrument R2955A.*

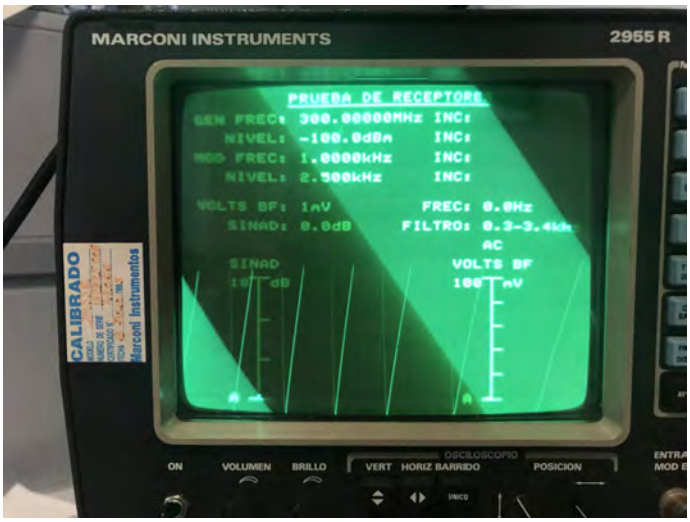


Figura A.5 – *Señal generada en el equipo Marconi Instrument R2955A.*

Hecho esto, se abre el Software SDRSharp y se prepara para la recepción de señal indicando el tipo de receptor utilizado. Además, se define la frecuencia de recepción a 300 MHz y se fija un error de frecuencia inicial *Tuner correction (ppm)* nulo. De este modo, en la representación gráfica del espectro de la señal recibida se observará el tono transmitido desplazado cierta frecuencia de los 300 MHz, como muestra la figura A.6. Esta desviación en frecuencia será la que nos dé la tolerancia del dispositivo SDR. Basta con modificar el parámetro de configuración correspondiente al error de frecuencia (en ppm) hasta observar que el tono está centrado a 300 MHz (figura A.7).

A

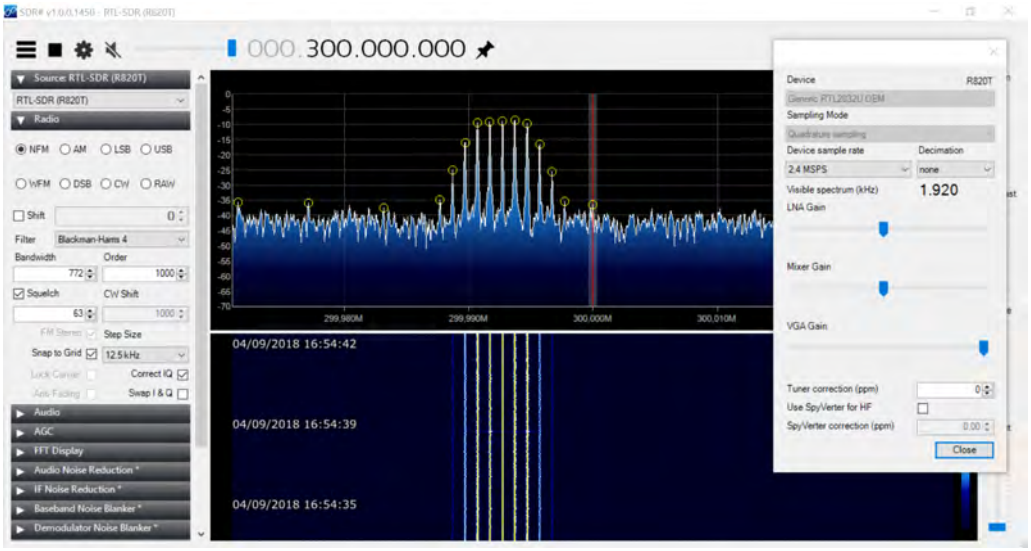


Figura A.6 – *Espectro del tono a 300 Hz antes de corregir la desviación en frecuencia.*

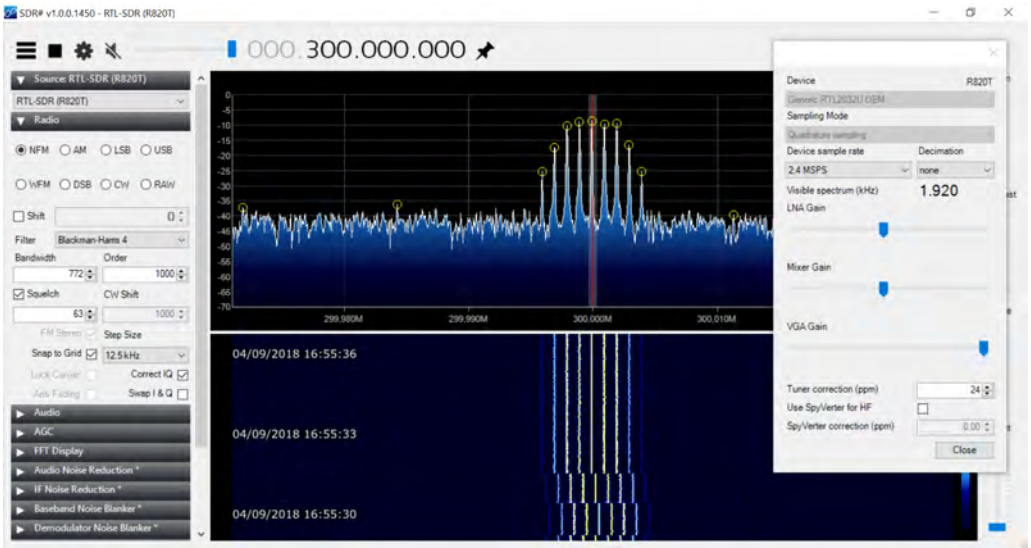


Figura A.7 – *Espectro del tono a 300 Hz después de corregir la desviación en frecuencia.*

A

En las figuras anteriores se ha calibrado un receptor RTL-SDR cuya desviación en frecuencia obtenida es 24 ppm. Esta técnica es un poco mas costosa en Hardware ya que se requiere tener un generador de señal pero es la más precisa ya que la señal generada se transmite sin interferencias por transmisión en el aire ya que viaja directamente al receptor a través del cable coaxial. Únicamente se verá afectada por las atenuaciones del cable, que son mucho menores que las interferencias de RF en el espacio.

A.3 Macro *sdrrFrequencyCalibrationReceiver* de MatLab

Mediante la macro de MatLab *sdrrFrequencyCalibrationReceiver.m* se comprueba el funcionamiento del dispositivo SDR dada una frecuencia de recepción sintonizada por el usuario y devuelve el offset en frecuencia (en ppm).

Esta última técnica de calibración es la menos precisa ya que el resultado oscila entorno al offset real y como resultado no ofrece el valor promedio de desviación en frecuencia sino que da el último valor obtenido. Al igual que ocurría con el Software *kalibrate* (sección A.1), para que la calibración se realice adecuadamente se debe sintonizar el canal de la banda GSM900 recibido con mayor potencia.

Tal y como se ha visto en el ejemplo de la sección A.1 el canal 109 se recibe con una potencia suficientemente elevada. Por ello, se fija como frecuencia de recepción la correspondiente a dicho canal de GSM900, 956.8 MHz, y se lanza la macro.

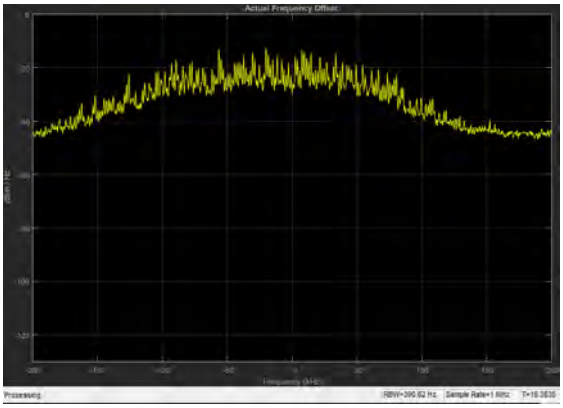


Figura A.8 – *Señal recibida por el dispositivo en MatLab.*

```
>> sdrrFrequencyCalibrationReceiver

hSDRrRx =

  System: comm.SDRRTLReceiver

  Properties:
             RadioAddress: '0'
        CenterFrequency: 956800000
         EnableTunerAGC: true
             SampleRate: 1000000
         OutputDataType: 'double'
        SamplesPerFrame: 4096
     FrequencyCorrection: 0


Frequency offset = -57128.9063 Hz,
FrequencyCorrection value = 59.7083
```

Figura A.9 – *Datos de recepción de la señal sintonizada en MatLab.*

Como resultado, el Software ofrece al usuario la gráfica de la figura A.8 que muestra la

señal recibida en dicha frecuencia a tiempo real y los datos de recepción de la figura A.9. Muestra los datos de entrada como la frecuencia de recepción de la señal o la desviación en frecuencia inicial fijada a cero puesto que se quiere calcular con esta macro y como salida devuelve el último valor de desviación en frecuencia calculado. Es por ello que el resultado obtenido no se acerca al obtenido a través de las otras técnicas puesto que devolvían el valor promedio del conjunto medido y esta macro el último valor medido.

ANEXO

B

CÓDIGO DEL ALGORITMO DE DECODIFICACIÓN DEL SERVICIO RDS

En el presente apéndice se incluye el código de MatLab implementado para la decodificación de los principales parámetros transmitidos en el servicio RDS de la señal de radio FM comercial.

```
%% ----------------- DECODING OF INFORMATION ----------------- %%
% Once the bitstream is synchronized in the receiver, the transmitted
% digital information can be decoded. The following information is decoded:
%       · Type and group version => Type of application
%       · Traffic program identifier (TP and TA)
%       · Program type (PTY)
%       · Program identifier (PI): country code, coverage code and program
%         reference number
%       · Name of the station

[Ngroups,bits_group]=size(groups);
bits_type=4;
bits_PTY=5;
group_type=[];
group_version=[];
group_app=[];

%% GROUP TYPE CODES AND VERSION CODES
% The groups are divided into two versions of 16 types each. Each type of
% group is intended for one type of application. The type and version of
% the group is in block 2 of the group.
%
%      TYPE OF GROUP: 4 first bits identify the 16 group tips
%      GROUP VERSION: 5th bit identifies version A (0) or B (1)
%
%        VERSION A                                          VERSION B
%      A3  A2  A1  A0                                  A3  A2  A1  A0
%      0   0   0   0    Basic tuning info              0   0   0   0
%      0   0   0   1    Program info                   0   0   0   1
%      0   0   1   0    Radiotext                      0   0   1   0
%      0   0   1   1    Other network info             0   0   1   1
%      0   1   0   0    Time and date / -              0   1   0   0
%      0   1   0   1    Transparent data channels      0   1   0   1
%      0   1   1   0    Applications of the station    0   1   1   0
%      0   1   1   1    Pager / -                      0   1   1   1
%      1   0   0   0    - / -                          1   0   0   0
%      1   0   0   1    - / -                          1   0   0   1
%      1   0   1   0    - / -                          1   0   1   0
%      1   0   1   1    - / -                          1   0   1   1
%      1   1   0   0    - / -                          1   1   0   0
%      1   1   0   1    - / -                          1   1   0   1
%      1   1   1   0    Support extended other networks 1   1   1   0
%      1   1   1   1    - / Quick tuning info          1   1   1   1

for i=1:Ngroups
    % Block 2
    block2(i,:)=groups(i,bits_block+1:bits_block*2);
    group_type(i,:)=block2(i,1:bits_type);
    group_version(i,1)=block2(i,bits_type+1);

    fprintf('Group %d\n',i);

    % Group type code and version
    switch mat2str(group_type(i,:))
        case '[0 0 0 0]'
            if group_version(i,1)==0
                fprintf('     - 0A: ');
            else
                fprintf('     - 0B: ');
            end
            fprintf('Basic tuning and switching information.\n');
        case '[0 0 0 1]'
            if group_version(i,1)==0
                fprintf('     - 1A: ');
                fprintf('Programme Item Number and slow labelling codes.\n');
```

```
    else
        fprintf('     - 1B: ');
        fprintf('Programme Item Number.\n');
    end
case '[0 0 1 0]'
    if group_version(i,1)==0
        fprintf('     - 2A: ');
    else
        fprintf('     - 2B: ');
    end
    fprintf('Radiotext.\n');
case '[0 0 1 1]'
    if group_version(i,1)==0
        fprintf('     - 3A: ');
        fprintf('Applications Identification for ODA.\n');
    else
        fprintf('     - 3B: ');
        fprintf('Open Data Applictions.\n');
    end
case '[0 1 0 0]'
    if group_version(i,1)==0
        fprintf('     - 4A: ');
        fprintf('Clock-time and date.\n');
    else
        fprintf('     - 4 B: ');
        fprintf('Open Data Applications.\n');
    end
case '[0 1 0 1]'
    if group_version(i,1)==0
        fprintf('     - 5A: ');
    else
        fprintf('     - 5B: ');
    end
    fprintf('Transparent Data Channels (32 channels) or ODA.\n');
case '[0 1 1 0]'
    if group_version(i,1)==0
        fprintf('     - 6A: ');
    else
        fprintf('     - 6B: ');
    end
    fprintf('In House applications or ODA.\n');
case '[0 1 1 1]'
    if group_version(i,1)==0
        fprintf('     - 7A: ');
        fprintf('Radio Paging or ODA.\n');
    else
        fprintf('     - 7B: ');
        fprintf('Open Data Applications.\n');
    end
case '[1 0 0 0]'
    if group_version(i,1)==0
        fprintf('     - 8A: ');
        fprintf('Traffic Message Channel or ODA.\n');
    else
        fprintf('     - 8B: ');
        fprintf('Open Data Applications.\n');
    end
case '[1 0 0 1]'
    if group_version(i,1)==0
        fprintf('     - 9A: ');
        fprintf('Emergency Warning System or ODA.\n');
    else
        fprintf('     - 9B: ');
        fprintf('Open Data Applications.\n');
    end
```

```
        case '[1 0 1 0]'
            if group_version(i,1)==0
                fprintf('     - 10A: ');
                fprintf('Programme Type Name.\n');
            else
                fprintf('     - 10B: ');
                fprintf('Open Data Applications\n');
            end
        case '[1 0 1 1]'
            if group_version(i,1)==0
                fprintf('     - 11A: ');
            else
                fprintf('     - 11B: ');
            end
            fprintf('Open Data Applications.\n');
        case '[1 1 0 0]'
            if group_version(i,1)==0
                fprintf('     - 12A: ');
            else
                fprintf('     - 12B: ');
            end
            fprintf('Open Data Applications.\n');
        case '[1 1 0 1]'
            if group_version(i,1)==0
                fprintf('     - 13A: ');
                fprintf('Enhanced Radio Paging or ODA.\n');
            else
                fprintf('     - 13B: ');
                fprintf('Open Data Applications.\n');
            end
        case '[1 1 1 0]'
            if group_version(i,1)==0
                fprintf('     - 14A: ');
            else
                fprintf('     - 14B:');
            end
            fprintf('Enhanced Other Networks informtion.\n');
        otherwise
            if group_version(i,1)==0
                fprintf('     - 15A: ');
                fprintf('Defined in RBDS.\n');
            else
                fprintf('     - 15B: ');
                fprintf('Fast switching information.\n');
            end
    end

%% TRAFFIC PROGRAMME (TP) AND TRAFFIC ANNOUNCEMENT (TA) CODES
% The code that identifies if the program contains traffic announcements
% (TP) is transmitted in the second block of all group types, version A and
% B. To know what type of ads it is transmitted in block 2 of the groups
% 0A, 0B, 14B and 15B a bit called TA. Combining both bits we can know the
% application of the traffic program, if it is transmitted.
%
% The TP is transmitted in bit 10 of block 2 of all the groups. The TA is
% transmitted in bit 4 of block 2 of group 0A, 0B and 15B and in bit 3 of
% group 14B.

    PT(i,:)=block2(i,bits_type+2);
    TA(i,:)=block2(i,bits_type+2+bits_PTY+1);
    switch mat2str(PT(i,:))
        case '0'
            if TA(i,:)==0
                fprintf('     - This programme does not carry traffic announcements nor does↙
it refer to a programme that does.\n');
```

```
            else
                fprintf('     - This programme carries information about another programme↙
which gives traffic information.\n');
            end
        case '1'
            if TA(i,:)==1
                fprintf('     - This programme carries traffic announcements but none are↙
being broadcast at present and may also carry information about other traffic announcements.↙
\n');
            else
                fprintf('     - A traffic announcement is being broadcast on this programme at↙
present.\n');
            end
        otherwise
            fprintf('     - Group received with errors.\n');
    end

%% PROGRAMME TYPE CODES (PTY)
% The code that identifies the type of program is transmitted in the second
% block of all group types, both version A and B. Specifically, it is
% transmitted in bits 9 through 5. It is 5 bits that tells us what type of
% program is transmitted in the signal of the 32 possibilities.

    PTY(i,:)=block2(i,bits_type+3:bits_type+3+bits_PTY-1);
    switch mat2str(PTY(i,:))
        case '[0 0 0 0 0]'
            fprintf('     - No programme type or undefined programme type.\n');
        case '[0 0 0 0 1]'
            fprintf('     - Programme type: News.\n');
        case '[0 0 0 1 0]'
            fprintf('     - Programme type: Current Affairs.\n');
        case '[0 0 0 1 1]'
            fprintf('     - Programme type: Information.\n');
        case '[0 0 1 0 0]'
            fprintf('     - Programme type: Sport.\n');
        case '[0 0 1 0 1]'
            fprintf('     - Programme type: Education.\n');
        case '[0 0 1 1 0]'
            fprintf('     - Programme type: Drama.\n');
        case '[0 0 1 1 1]'
            fprintf('     - Programme type: Culture.\n');
        case '[0 1 0 0 0]'
            fprintf('     - Programme type: Science.\n');
        case '[0 1 0 0 1]'
            fprintf('     - Programme type: Varied.\n');
        case '[0 1 0 1 0]'
            fprintf('     - Programme type: Pop Music.\n');
        case '[0 1 0 1 1]'
            fprintf('     - Programme type: Rock Music.\n');
        case '[0 1 1 0 0]'
            fprintf('     - Programme type: Easy Listening Music.\n');
        case '[0 1 1 0 1]'
            fprintf('     - Programme type: Light classical.\n');
        case '[0 1 1 1 0]'
            fprintf('     - Programme type: Serious classical.\n');
        case '[0 1 1 1 1]'
            fprintf('     - Programme type: Other Music.\n');
        case '[1 0 0 0 0]'
            fprintf('     - Programme type: Weather.\n');
        case '[1 0 0 0 1]'
            fprintf('     - Programme type: Finance.\n');
        case '[1 0 0 1 0]'
            fprintf('     - Programme type: Children´s programmes.\n');
        case '[1 0 0 1 1]'
            fprintf('     - Programme type: Social Affairs.\n');
```

```
        case '[1 0 1 0 0]'
            fprintf('     - Programme type: Religion.\n');
        case '[1 0 1 0 1]'
            fprintf('     - Programme type: Phone In.\n');
        case '[1 0 1 1 0]'
            fprintf('     - Programme type: Travel.\n');
        case '[1 0 1 1 1]'
            fprintf('     - Programme type: Leisure.\n');
        case '[1 1 0 0 0]'
            fprintf('     - Programme type: Jazz Music.\n');
        case '[1 1 0 0 1]'
            fprintf('     - Programme type: Country Music.\n');
        case '[1 1 0 1 0]'
            fprintf('     - Programme type: National Music.\n');
        case '[1 1 0 1 1]'
            fprintf('     - Programme type: Oldies Music.\n');
        case '[1 1 1 0 0]'
            fprintf('     - Programme type: Folk Music.\n');
        case '[1 1 1 0 1]'
            fprintf('     - Programme type: Documentary.\n');
        case '[1 1 1 1 0]'
            fprintf('     - Programme type: Alarm Test.\n');
        case '[1 1 1 1 1]'
            fprintf('     - Programme type: Alarm.\n');
        otherwise
            fprintf('     - Group received with errors.\n');
    end

%% PROGRAMME IDENTIFICATION CODES (PI)
% The identifier code of the program is transmitted in the first block of
% all group types of version A. In group types of version B it is also
% transmitted in block 3.
%
% It consists of the country code (4 bits), the coverage code (4 bits) and
% the program identification code (8 bits). To know the information
% contained in said code, the bits must be converted to hexadecimal.

% --- Country code ---
%
% Different countries share the same country code. To differentiate them,
% the extended country code (ECC) is transmitted in block 3 of type 1A
% groups. It is transmitted in the last 8 bits of block 3 (m7 ... m0) of
% the 16 bits of information. These bits are grouped from 4 to 4, obtaining
% two characters.
%
%   0000    0   -
%   0001    1   - Germany (E0), Greece (E1), Moldova (E4), Morocco (E2)
%   0010    2   - Algeria (E0), Cyprus (E1), Czech Republic (E2), Estonia
%                 (E4), Ireland (E3)
%   0011    3   - Andorra (E0), Poland (E2), San Marino (E1), Turkey (E3)
%   0100    4   - Israel (E0), Macedonia (E3), Switzerland (E1), Vatican
%                 City State (E2)
%   0101    5   - Italy (E0), Jordan (E1), Slovakia (E2)
%   0110    6   - Belgium (E0), Finland (E1), Syrian Arab Republic (E2),
%                 Ukraine (E4)
%   0111    7   - Luxembourg (E1), Russian Federation (E0), Tunisia (E2)
%   1000    8   - Portugal (E4), Bulgaria (E1), Netherlands (E3), Palestine
%                 (E0)
%   1001    9   - Albania (E0), Denmark (E1), Latvia (E3), Liechtenstein
%                 (E2), Slovenia (E4)
%   1010    A   - Austria (E0), Gibraltar (United Kingdom) (E1),Iceland (E2
%                 ), Lebanon (E3)
%   1011    B   - Hungary (E0), Iraq (E1), Monaco (E2)
%   1100    C   - Croatia (E3), Lithuania (E2), Malta (E0), United Kingdom
%                 (E1)
```

```
%   1101    D   - Germany (E0), Libya (E1), Yugoslavia (E2)
%   1110    E   - Romania (E1), Spain (E2), Sweden (E3)
%   0111    F   - Belarus (E3), Bosnia Herzegovina (E4), Egypt (E0), France
%                 (E1), Norway (E2)

% --- Coverage code ---
%
%   0000    0   L       Local
%   0001    1   I       International
%   0010    2   N       National
%   0011    3   S       Supra-regional
%   0100    4   R1      Regional
%   0101    5   R2      Regional
%   0110    6   R3      Regional
%   0111    7   R4      Regional
%   1000    8   R5      Regional
%   1001    9   R6      Regional
%   1010    A   R7      Regional
%   1011    B   R8      Regional
%   1100    C   R9      Regional
%   1101    D   R10     Regional
%   1110    E   R11     Regional
%   0111    F   R12     Regional

% --- Program reference number ---
%
% It consists of two values of 4 bits each.
%
% 00            - Not assigned
% 02 to FF      -

    % Programme identification code (PI) => Block 1
    group_PI(i,:)=groups(i,1:bits_block);
    group_country(i,:)=binaryVectorToHex(group_PI(i,1:4));
    group_area(i,:)=binaryVectorToHex(group_PI(i,5:8));
    group_reference(i,:)=binaryVectorToHex(group_PI(i,9:16));

    % Extended Country Codes
    type1A=[0 0 0 0];
    if (group_type(i,:)==type1A & group_version(i)==0)
        ECC(i,:)=binaryVectorToHex(groups(i,bits_block*3-bits_CRC-8:bits_block*3-bits_CRC-1));
    else
        ECC(i,:)=[0 0];
    end

    % Country and Extended Country Codes
    switch group_country(i)
        case '1'
            fprintf('     - PI: Germany, Greece, Moldova or Morocco.\n');
            if ECC(i,1)=='E'    % Contains ECC
                switch ECC(i,2)
                    case '0'
                        fprintf('     - ECC E0: Germany.\n');
                    case '1'
                        fprintf('     - ECC E1: Greece.\n');
                    case '2'
                        fprintf('     - ECC E2: Morocco.\n');
                    case '4'
                        fprintf('     - ECC E4: Moldova.\n');
                    otherwise
                        fprintf('     - ECC received with errors.\n');
                end
            else                % Does not contain ECC
                fprintf('     - No ECC: please record at least one minute of signal.\n');
            end
```

```
case '2'
    fprintf('     - PI: Algeria, Cyprus, Czech Republic, Estonia or Ireland.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Algeria.\n');
            case '1'
                fprintf('     - ECC E1: Cyprus.\n');
            case '2'
                fprintf('     - ECC E2: Czech Republic.\n');
            case '3'
                fprintf('     - ECC E3: Ireland.\n');
            case '4'
                fprintf('     - ECC E4: Estonia.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '3'
    fprintf('     - PI: Andorra, Poland, San Marino or Turkey.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Andorra.\n');
            case '1'
                fprintf('     - ECC E1: San Marino.\n');
            case '2'
                fprintf('     - ECC E2: Poland.\n');
            case '3'
                fprintf('     - ECC E3: Turkey.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '4'
    fprintf('     - PI: Israel, Macedonia, Switzerland or Vatican City State.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Israel.\n');
            case '1'
                fprintf('     - ECC E1: Switzerland.\n');
            case '2'
                fprintf('     - ECC E2: Vatican City State.\n');
            case '3'
                fprintf('     - ECC E3: Macedonia.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '5'
    fprintf('     - PI: Italy, Jordan or Slovakia.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Italy.\n');
            case '1'
                fprintf('     - ECC E1: Jordan.\n');
            case '2'
```

```
                fprintf('     - ECC E2: Slovakia.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '6'
    fprintf('     - PI: Belgium, Finland, Syrian Arab Republic or Ukraine.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Belgium.\n');
            case '1'
                fprintf('     - ECC E1: Finland.\n');
            case '2'
                fprintf('     - ECC E2: Syrian Arab Republic.\n');
            case '4'
                fprintf('     - ECC E4: Ukraine.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '7'
    fprintf('     - PI: Luxembourg, Russian Federation or Tunisia.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Russian Federation.\n');
            case '1'
                fprintf('     - ECC E1: Luxembourg.\n');
            case '2'
                fprintf('     - ECC E2: Tunisia.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '8'
    fprintf('     - PI: Portugal, Bulgaria, Netherlands or Palestine.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Palestine.\n');
            case '1'
                fprintf('     - ECC E1: Bulgaria.\n');
            case '3'
                fprintf('     - ECC E3: Netherlands.\n');
            case '4'
                fprintf('     - ECC E4: Portugal.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case '9'
    fprintf('     - PI: Albania, Denmark, Latvia, Liechtenstein or Slovenia.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Albania.\n');
```

```
                case '1'
                    fprintf('     - ECC E1: Denmark.\n');
                case '2'
                    fprintf('     - ECC E2: Liechtenstein.\n');
                case '3'
                    fprintf('     - ECC E3: Latvia.\n');
                case '4'
                    fprintf('     - ECC E4: Slovenia.\n');
                otherwise
                    fprintf('     - ECC received with errors.\n');
            end
        else                % Does not contain ECC
            fprintf('     - No ECC: please record at least one minute of signal.\n');
        end
    case 'A'
        fprintf('     - PI: Austria, Gibraltar (United Kingdom),Iceland or Lebanon.\n');
        if ECC(i,1)=='E'    % Contains ECC
            switch ECC(i,2)
                case '0'
                    fprintf('     - ECC E0: Austria.\n');
                case '1'
                    fprintf('     - ECC E1: Gibraltar (United Kingdom).\n');
                case '2'
                    fprintf('     - ECC E2: Iceland.\n');
                case '3'
                    fprintf('     - ECC E3: Lebanon.\n');
                otherwise
                    fprintf('     - ECC received with errors.\n');
            end
        else                % Does not contain ECC
            fprintf('     - No ECC: please record at least one minute of signal.\n');
        end
    case 'B'
        fprintf('     - PI: Hungary, Iraq or Monaco.\n');
        if ECC(i,1)=='E'    % Contains ECC
            switch ECC(i,2)
                case '0'
                    fprintf('     - ECC E0: Hungary.\n');
                case '1'
                    fprintf('     - ECC E1: Iraq.\n');
                case '2'
                    fprintf('     - ECC E2: Monaco.\n');
                otherwise
                    fprintf('     - ECC received with errors.\n');
            end
        else                % Does not contain ECC
            fprintf('     - No ECC: please record at least one minute of signal.\n');
        end
    case 'C'
        fprintf('     - PI: Croatia, Lithuania, Malta or United Kingdom.\n');
        if ECC(i,1)=='E'    % Contains ECC
            switch ECC(i,2)
                case '0'
                    fprintf('     - ECC E0: Malta.\n');
                case '1'
                    fprintf('     - ECC E1: United Kingdom.\n');
                case '2'
                    fprintf('     - ECC E2: Lithuania.\n');
                case '3'
                    fprintf('     - ECC E3: Croatia.\n');
                otherwise
                    fprintf('     - ECC received with errors.\n');
            end
        else                % Does not contain ECC
            fprintf('     - No ECC: please record at least one minute of signal.\n');
```

```
    end
case 'D'
    fprintf('     - PI: Germany, Libya or Yugoslavia.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Germany.\n');
            case '1'
                fprintf('     - ECC E1: Libya.\n');
            case '2'
                fprintf('     - ECC E2: Yugoslavia.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case 'E'
    fprintf('     - PI: Romania, Spain or Sweden.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '1'
                fprintf('     - ECC E1: Romania.\n');
            case '2'
                fprintf('     - ECC E2: Spain.\n');
            case '3'
                fprintf('     - ECC E3: Sweden.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
case 'F'
    fprintf('     - PI: Belarus, Bosnia Herzegovina, Egypt, France or Norway.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0: Egypt.\n');
            case '1'
                fprintf('     - ECC E1: France.\n');
            case '2'
                fprintf('     - ECC E2: Norway.\n');
            case '3'
                fprintf('     - ECC E3: Belarus.\n');
            case '4'
                fprintf('     - ECC E4: Bosnia Herzegovina.\n');
            otherwise
                fprintf('     - ECC received with errors.\n');
        end
    else                % Does not contain ECC
        fprintf('     - No ECC: please record at least one minute of signal.\n');
    end
otherwise
    fprintf('     - No PI: The group has been received with errors.\n');
    if ECC(i,1)=='E'    % Contains ECC
        switch ECC(i,2)
            case '0'
                fprintf('     - ECC E0.\n');
            case '1'
                fprintf('     - ECC E1.\n');
            case '2'
                fprintf('     - ECC E2.\n');
            case '3'
                fprintf('     - ECC E3.\n');
```

```
                    case '4'
                        fprintf('     - ECC E4.\n');
                    otherwise
                        fprintf('     - ECC received with errors.\n');
                end
            else                % Does not contain ECC
                fprintf('     - No ECC: please record at least one minute of signal.\n');
            end
    end

    % Programme type in term of area coverage
    switch group_area(i,:)
        case '0'
            fprintf('     - Local programme: transmitted via a single transmitter only during↙
the whole transmitting time.\n');
        case '1'
            fprintf('     - International programme: transmitted in other countries.\n');
        case '2'
            fprintf('     - National programme: transmitted throughout the country.\n');
        case '3'
            fprintf('     - Supra-regional programme: transmitted throughout a large part of↙
country.\n');
        case '4'
            fprintf('     - Regional R1 programme: available in one region over one or more↙
frequencies.\n');
        case '5'
            fprintf('     - Regional R2 programme: available in one region over one or more↙
frequencies.\n');
        case '6'
            fprintf('     - Regional R3 programme: available in one region over one or more↙
frequencies.\n');
        case '7'
            fprintf('     - Regional R4 programme: available in one region over one or more↙
frequencies.\n');
        case '8'
            fprintf('     - Regional R5 programme: available in one region over one or more↙
frequencies.\n');
        case '9'
            fprintf('     - Regional R6 programme: available in one region over one or more↙
frequencies.\n');
        case 'A'
            fprintf('     - Regional R7 programme: available in one region over one or more↙
frequencies.\n');
        case 'B'
            fprintf('     - Regional R8 programme: available in one region over one or more↙
frequencies.\n');
        case 'C'
            fprintf('     - Regional R9 programme: available in one region over one or more↙
frequencies.\n');
        case 'D'
            fprintf('     - Regional R10 programme: available in one region over one or more↙
frequencies.\n');
        case 'E'
            fprintf('     - Regional R11 programme: available in one region over one or more↙
frequencies.\n');
        case 'F'
            fprintf('     - Regional R12 programme: available in one region over one or more↙
frequencies.\n');
        otherwise
            fprintf('     - Signal received with errors.\n');
    end

    % Programme reference number
    switch group_reference(i,:)
        case '[0 0]'
```

```
            fprintf('     - Programme reference number not assigned.\n');
        otherwise
            fprintf('     - Programme reference number assigned: %s (Hex-coding).\n',↙
group_reference(i,:));
    end

    fprintf('--------------------------------------------------------------------------------\n');
end

%% PROGRAMME SERVICE NAME (PS)
% The name of the station is transmitted in the 4th block of all groups of
% type 0A. It is about 8 characters so it is enough with 4 groups to get
% the name of the program. These characters are not transmitted in order,
% but are transmitted in block 2 of group 0A to the position they will
% occupy (d3, d2, d1 or d0).
%
%                   PS  =  _ _    _ _     _ _    _ _
%                            d3     d2      d1     d0

fprintf('NAME OF THE BROADCASTING STATION: ');

for i=1:Ngroups
    block2(i,:)=groups(i,bits_block+1:bits_block*2);
    block4(i,:)=groups(i,bits_block*3+1:end);

    position(i,:)=block2(i,bits_block-11:bits_block-10);
    pos=binaryVectorToDecimal(position(i,:),'MSBFirst')*2+1;

    character1=char(binaryVectorToDecimal(block4(i,1:(bits_block-bits_CRC)/2),'MSBFirst'));
    character2=char(binaryVectorToDecimal(block4(i,(bits_block-bits_CRC)/2+1:(bits_block-↙
bits_CRC)),'MSBFirst'));
    PS_name(pos)=character1;
    PS_name(pos+1)=character2;
end

fprintf('%s\n',PS_name);
```

ANEXO

C

PRESUPUESTO

El presente apéndice detalla los costes relacionados con la implementación Software de los receptores radio FM comercial y AX.25. Éstos se clasifican en tres tipos:

C.1 Costes electrónicos

Los costes electrónicos son aquellos derivados de los recursos Hardware utilizados en el presente trabajo. Todos ellos han sido cedidos por el grupo GranaSAT y se recogen en la siguiente tabla.

Hardware	**Coste (€)**
RTL-SDR Andoer® blanco	12.99
RTL-SDR Andoer® negro	12.99
RTL-SDR Allwin	10.53
SDR FUNcube Dongle Pro+	107
Charger Doctor	6
Marconi Instrument R2955A	900
TOTAL	**1047.51** €

Tabla C.1 – *Costes Hardware.*

C.2 Software

A continuación, se especifica el coste de los recursos Software necesarios par el correcto desarrollo del TFM.

Software	Licencia	Coste (€)
MatLab	Estudiante	69
SDRSharp	Open-Source	0
LaTeX	Open-Source	0
TOTAL		**69** €

Tabla C.2 – *Costes Software.*

C.3 Recursos humanos

Finalmente, en la tabla C.3 se recogen los costes asociados al tiempo y trabajo dedicado por la alumna y el tutor para el desarrollo del presente TFM.

Dado que el proyecto tiene una ponderación de 12 créditos del Máster en Ingeniería de Telecomunicaciones, el tiempo mínimo dedicado por la alumna para su desarrollo es de 300 horas lo que supone 2 meses de trabajo a jornada completa. Sin embargo, tal y como se aprecia en el Diagrama de Gantt del capítulo primero del trabajo, la alumna ha trabajado durante 12 meses a tiempo parcial empleando finalmente 500 horas de trabajo. La alumna, como ingeniera junior, supone un coste de 10 €/h.

Por otra parte, el tutor del proyecto se contrata como ingeniero senior con un coste por hora de 50 € y una disponibilidad de 2 horas semanales durante el curso académico de 9 meses.

Tipo de contrato	Horas	Coste (€/h)	Coste (€)
Ingeniero junior	500	10	5000.00
Ingeniero senior	72	50	3600.00
TOTAL			**8600** €

Tabla C.3 – *Costes en recursos humanos*

.