GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Implementación y evaluación de algoritmos de sincronismo de símbolo en un receptor digital

Autor: Juan Orlando Cives Espejo

Director: Lucas Francisco Novales Peleato

Co-Director: Javier Matanza Domingo

Madrid

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Implementación y evaluación de algoritmos de sincronismo de símbolo en un receptor digital

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2024/25 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: Juan Orlando Cives Espejo Fecha: 05/ 06/ 2025

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Fecha: 5 / 6 / 2025

Fdo.: Fecha: ……/ ……/ ……

Firmado por MATANZA DOMINGO JAVIER - ***4397** el día 06/06/2025 con un certificado emitido

GRADO EN INGENIERÍA EN TECNOLOGÍAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

Implementación y evaluación de algoritmos de sincronismo de símbolo en un receptor digital

Autor: Juan Orlando Cives Espejo

Director: Lucas Francisco Novales Peleato

Co-Director: Javier Matanza Domingo

Madrid

Agradecimientos

A mis padres, por su sacrificio constante y por brindarme, con esfuerzo y generosidad, la oportunidad de estudiar en esta universidad.

A mis hermanas, por su cariño, por confiar siempre en mí y por darme fuerzas en los momentos más difíciles de la carrera.

A mis directores, Lucas y Javier, por inspirarme con su entusiasmo y por guiarme y apoyarme con paciencia en cada etapa.

A todos ellos, gracias de corazón.

IMPLEMENTACIÓN Y EVALUACIÓN DE ALGORITMOS DE SINCRONISMO DE SÍMBOLO EN UN RECEPTOR DIGITAL

Autor: Cives, Espejo, Juan Orlando.
Director: Novales, Peleato, Lucas Francisco.
Co-Director: Matanza, Domingo, Javier.
Entidad Colaboradora: ICAI – Universidad Pontificia Comillas.

RESUMEN DEL PROYECTO

Este proyecto pretende analizar el potencial docente de la SDR *ADALM-Pluto* en el ámbito de asignaturas de señales y sistemas, teoría de la comunicación y procesamiento digital de señales. Para ello, se han realizado distintos experimentos de transmisión y recepción de señales con este dispositivo. Estos se han centrado en la implementación de un sistema de comunicaciones analógico basado en un módem AM y un sistema de comunicaciones digital basado en un modulador BPSK.

En concreto, en el sistema digital, se ha diseñado también un receptor que incorpora funcionalidades de sincronismo de portadora, de trama y de símbolo. Este último ha sido el foco central del proyecto. Los algoritmos de sincronismo de símbolo evaluados han sido el de cruce por cero y los planteados por Gardner y "Müller y Mueller". Los resultados de la transmisión y recepción de señales utilizando la SDR se han analizado mediante la SNR para el caso analógico y la BER para el caso digital. Asimismo, para evaluar cómo los algoritmos de sincronismo de símbolo mejoran la recepción de la señal, se han utilizado medidas estadísticas como el error cuadrático medio.

En concreto se ha demostrado la mejora de la SNR en el sistema analógico empleando ganancias de -10 dB en el transmisor y 10 dB receptor. Además, se ha comprobado que es preferible el uso de factores de *roll-off* pequeños en algoritmos como el de Müller y Mueller, mientras que un factor de *roll-off* alto disminuye el error de temporización en el caso de algoritmos como el de cruce por cero y el de Gardner.

Palabras clave: SDR, *ADALM-Pluto*, módem AM, receptor digital, sincronismo de símbolo.

1. Introducción

En el ámbito de la Ingeniería en Telecomunicación, las radios definidas por software (SDR, por sus siglas en inglés) representan una de las tecnologías más novedosas y punteras. Esto se debe a que reemplazan el hardware tradicional de las radios por uno más flexible que puede ser definido por software. Esto las hace mucho más versátiles para operar en diferentes entornos para los que, tradicionalmente, se requerían equipos específicos.

Las SDR presentan un gran potencial para ser empleadas como herramientas docentes. En concreto, esto resulta especialmente útil en la parte práctica de asignaturas relacionadas con el análisis y procesamiento de señales.

Un ejemplo destacable es la SDR ADALM-Pluto desarrollada por la empresa *Analog Devices*. Está especialmente diseñada para ser utilizada en el ámbito educativo. Cuenta

con un sistema de comunicaciones completo integrado en un único chip basado en una arquitectura Cero-IF y con un canal de transmisión y otro de recepción.

2. Definición del proyecto

Este proyecto propone diferentes pruebas de transmisión y recepción de señales utilizando el lenguaje de programación MATLAB. Estas pruebas están basadas en sistemas analógicos y sistemas digitales. En concreto, se hace especial énfasis en el análisis de algoritmos de sincronismo de símbolo en un receptor digital. El propósito de estas pruebas es evaluar el potencial de la ADALM-Pluto como herramienta educativa.

En primer lugar, se utiliza un sistema de comunicaciones analógico basado en un módem AM. Este se utiliza para transmitir audio utilizando la ADALM-Pluto. En segundo lugar, se implementa un sistema de comunicaciones digital basado en una modulación BPSK con un receptor digital completo. Seguidamente, se realiza la transmisión de una secuencia de caracteres utilizando este sistema digital.

Por último, se modifica el receptor digital, utilizando diferentes algoritmos de sincronismo de símbolo. Se plantean diferentes escenarios con parámetros que permitan analizar cuáles son los mejores algoritmos en distintas situaciones. Con ello, se evalúa cómo mejora la recepción de la señal.

3. Descripción del sistema

La conexión entre la ADALM-Pluto y MATLAB se logra empleando la *Communication Toolbox Support Package for Analog Devices* [1]. Esta contiene los objetos y métodos necesarios para definir por software los parámetros hardware de la ADALM-Pluto y para habilitar el canal de transmisión y recepción, tal y como muestra la Figura 1.

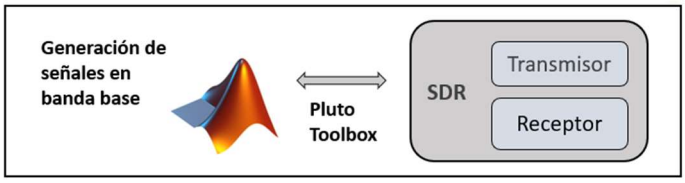

Figura 1. Esquema de la interacción entre MATLAB y la ADALM-Pluto.

Por otro lado, el sistema analógico se basa en un módem AM. Este emplea una exponencial compleja como portadora y un detector de envolvente en su demodulador. La Figura 2 muestra el diagrama de bloques que utiliza el módem AM, incluyendo el modulador y el demodulador.

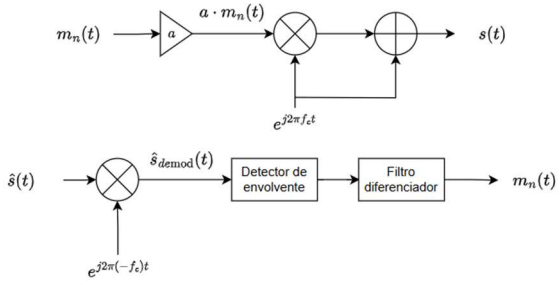

Figura 2. Diagrama de bloques del módem AM.

Finalmente, el sistema digital se basa en un modulador BPSK y en un receptor digital. Este incluye algoritmos de sincronismo de portadora, sincronismo de símbolo y sincronismo de trama. La Figura 3 muestra el esquema del modulador y del receptor.

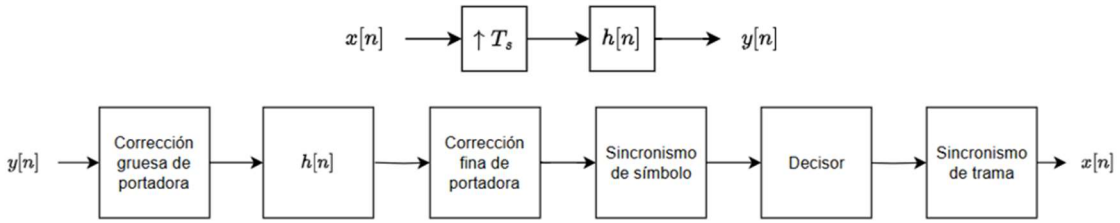

Figura 3. Diagrama de bloques del modulador BPSK y del receptor digital.

4. Resultados

- La señal de audio transmitida y recibida a través de la ADALM-Pluto y demodulada mediante el módem AM de MATLAB se muestra en la Figura 4. En ella, se pueden observar espurios a 10 kHz, 15 kHz y 20 kHz.

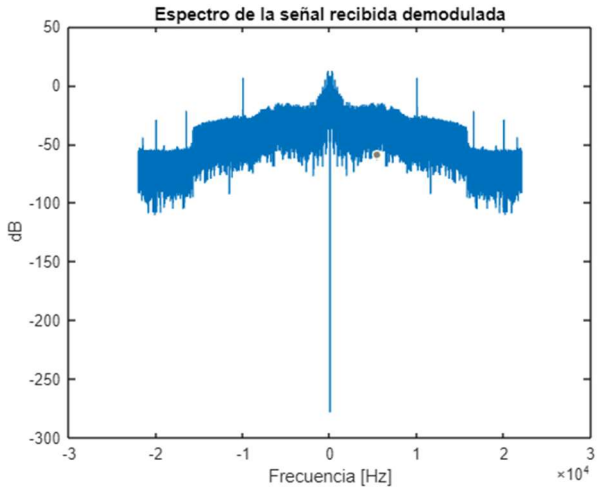

Figura 4. Resultado de la señal de audio procesada mediante el módem AM.

- La secuencia de caracteres se ha convertido en una secuencia de bits mediante codificadores de canal y de fuente. A continuación, se le ha aplicado una modulación BPSK y se ha transmitido utilizando la ADALM-Pluto. La constelación resultante tras aplicar técnicas de sincronismo de portadora, de símbolo y de trama en el receptor digital implementado en MATLAB se muestra en la Figura 5.

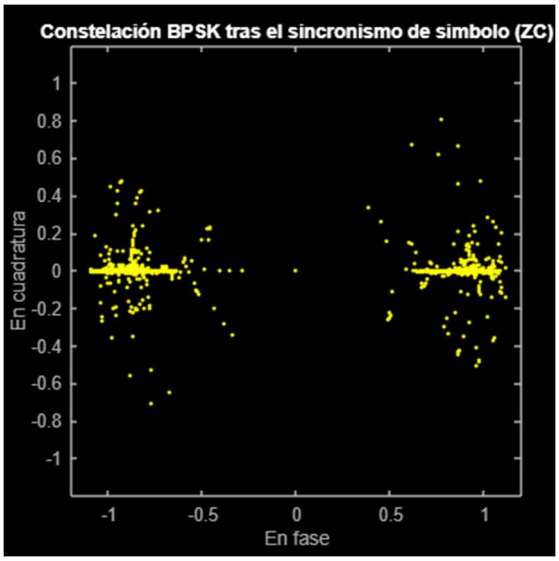

Figura 5. Constelación BPSK procesada por un receptor digital.

5. Conclusiones

A pesar de la presencia de distorsiones no lineales en la señal recibida, el módem AM implementado en MATLAB recupera exitosamente la señal de audio que se ha transmitido utilizando la ADALM-Pluto. La relación señal a ruido mejora notablemente utilizando ganancias en transmisión y recepción de -10 dB y 10 dB respectivamente. El uso de filtros paso bajo con frecuencias de corte $f_c = 8\,kHz$ permite eliminar estos espurios. Se concluye de este modo que es viable utilizar esta SDR para trabajar con señales analógicas.

Asimismo, tras el procesamiento del receptor digital, también es posible recuperar correctamente la cadena de caracteres transmitida usando la SDR. La tasa de error de bit (BER, por sus siglas en inglés, *Bit Error Rate*) obtenida es igual a 0 tras aplicar el decodificador de canal.

El rendimiento de los algoritmos de sincronismo de símbolo estudiados mejora para valores grandes de muestras por símbolo como $sps = 10$. Por otro lado, en los experimentos realizados se tiene que un valor pequeño de ancho de banda de bucle normalizado como $b_n = 0{,}001$ y un valor grande de factor de amortiguamiento $\zeta = 2$ son convenientes para el PLL. El factor de *roll-off* (β) óptimo depende del tipo de TED. Algoritmos como el de Gardner o el de cruce por cero mejoran con un valor de $\beta \in (0{,}5;\ 1)$, mientras que el algoritmo MM tiene un mejor rendimiento para valores de β pequeños.

Por tanto, estos resultados permiten concluir que la ADALM-Pluto es una SDR adecuada para ser introducida en prácticas relacionadas con el diseño de sistemas de comunicación y procesado digital. Estas prácticas estarían basadas en las pruebas realizadas en este proyecto.

6. Referencias

[1] Mathworks, "Get Started with Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio," [Online]. Available: https://es.mathworks.com/help/comm/getting-started-with-communications-system-toolbox-support-package-for-pluto-radio.html?s_tid=CRUX_lftnav. [Accessed 14 5 2025].

[2] T. F. Collins, R. Getz, D. Pu and A. M. Wyglinski, Software-Defined Radio for Engineers, Artech House, 2018.

[3] "IEEE Approved Draft Standard for Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management Amendment: Addition of New Terms and Associated Definitions," *IEEE P1900.1a/D2.0,* pp. 1-18, 7 Dec 2012.

[4] MathWorks, "Communications Toolbox," [Online]. Available: https://es.mathworks.com/help/comm/index.html. [Accessed 14 5 2025].

[5] L. F. Novales Peleato, *Implementación de técnicas mejoradas de comunicaciones digitales aplicadas a docencia,* Trabajo de Fin de Grado, Universidad Pontificia Comillas, 2021.

[6] Teleco Renta, «Recursos para Escuelas de Teleco,» [En línea]. Available: https://github.com/Telecorenta?tab=repositories. [Último acceso: 26 05 2025].

[7] M. Lázaro, Teoría de la Comunicación, Universidad Carlos III de Madrid Open Course Ware, 2014.

[8] A. Artés Rodrígez, F. Pérez González, J. Cid Sueiro, R. López Valcarce, C. Mosquera Nartallo y F. Pérez Cruz, Comunicaciones Digitales, Prentice Hall, 2012.

[9] M. Lichtman, «PySDR: Guía de uso para SDR/DSP con Python,» [En línea]. Available: https://pysdr.org/es/content-es/digital_modulation.html. [Último acceso: 23 5 2025].

[10] A. V. Oppenheim y R. W. Schafer, Tratamiento de señales en tiempo discreto, Tercera edición, Pearson Educación, 2011.

[11] R. W. Stewart, K. W. Barlee, D. S. W. Atkinson y L. H. Crockett, Software Defined Radio using MATLAB & Simulink and the RTL-SDR, Strathclyde Academic Media, 2015.

[12] MathWorks, «Coarse Frequency Compensator,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/coarsefrequencycompensator.html. [Último acceso: 27 05 2025].

[13] Y. Wang, K. Shi y E. Serpedin, «Non-Data-Aided Feedforward Carrier Frequency Offset Estimators for QAM Constellations: A Nonlinear Least-Squares Approach,» *EURASIP Journal on Advances in Signal Processing,* nº 13, 2004.

[14] M. Luise y R. R., «Carrier Frequency Recovery in All-Digital Modems for Burst-Mode Transmission,» *IEEE Transactions on Communications 43,* nº 2/3/4, 1995.

[15] MathWorks, «Carrier Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/carriersynchronizer.html. [Último acceso: 27 05 2025].

[16] MathWorks, «Symbol Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/symbolsynchronizer.html. [Último acceso: 27 05 2025].

[17] F. Gardner, «A BPSK/QPSK Timing-Error Detector for Sampled Receivers,» *IEEE Transactions on Communications,* vol. 34, nº 5, pp. 423-429, Mayo 1986.

[18] K. Mueller y M. Muller, «Timing Recovery in Digital Synchronous Data Receivers,» *IEEE Transactions on Communications,* vol. 24, nº 5, pp. 516-531, Mayo 1976.

IMPLEMENTATION AND EVALUATION OF TIMING SYNCHRONIZATION ALGORITHMS IN A DIGITAL RECEIVER

Author: Cives, Espejo, Juan Orlando.
Supervisor: Novales, Peleato, Lucas Francisco.
Co-Supervisor: Matanza, Domingo, Javier.
Collaborating Entity: ICAI – Universidad Pontificia Comillas.

ABSTRACT

This project aims to analyze the teaching potential of the ADALM-Pluto SDR in the context of courses on signals and systems, communication theory, and digital signal processing. To this end, various signal transmission and reception experiments have been carried out using this device. These experiments focused on the implementation of an analog communication system based on an AM modem and a digital communication system based on a BPSK modulator.

Specifically, in the digital system, a receiver was also designed, which incorporates carrier, frame, and symbol synchronization functionalities. The latter has been the main focus of the project. The evaluated symbol synchronization algorithms include the zero-crossing method and those proposed by Gardner and "Müller and Mueller." The results of the signal transmission and reception using the SDR were analyzed using the SNR in the analog case and the BER in the digital case. Furthermore, to assess how the timing synchronization algorithms improve signal reception, statistical measures such as the mean squared error were used.

In particular, the improvement of the SNR in the analog system was demonstrated by using gains of -10 dB at the transmitter and 10 dB at the receiver. Additionally, it was found that the use of small roll-off factors is preferable for algorithms such as the Müller and Mueller method, whereas a high roll-off factor reduces timing error in algorithms such as the zero-crossing and Gardner methods.

Keywords: SDR, ADALM-Pluto, AM modem, digital receiver, timing synchronization.

1. Introduction

In the field of Telecommunication Engineering, software-defined radios (SDR) represent one of the most innovative and cutting-edge technologies. Their main advantage lies in replacing traditional radio hardware with a more flexible solution that can be defined via software. This makes them more versatile to operate in a variety of environments that traditionally require specific hardware.

SDRs have great potential as educational tools. This is especially useful in practical courses related to signal analysis and processing.

A notable example is the ADALM-Pluto SDR, developed by Analog Devices. It is specifically designed for educational purposes and integrates a complete communication system integrated into a single chip. It is based on Zero-IF architecture and includes one transmit, and one receive channel.

2. Project definition

This project proposes several signal transmission and reception tests using MATLAB. These tests are based on both analog and digital communication systems. In particular, special emphasis is placed on the analysis of timing synchronization algorithms in a digital receiver. The purpose of these tests is to evaluate the potential of the ADALM-Pluto as an educational tool.

In the first place, an analog communication system based on an AM modem is used to transmit audio using the ADALM-Pluto. Then, a digital communication system is implemented using BPSK modulation and a complete digital receiver. A sequence of characters is then transmitted using this digital system.

Finally, the digital receiver is modified by applying different timing synchronization algorithms. Several scenarios are considered, with varying parameters to analyze which algorithms perform best under different conditions. The goal is to evaluate how signal reception improves with each approach.

3. System description

The connection between the ADALM-Pluto and MATLAB is established using the *Communication Toolbox Support Package for Analog Devices* [1]. This toolbox contains the objects and methods required to configure the hardware parameters of the ADALM-Pluto via software and to enable the transmission and reception channels, as illustrated in Figure 1.

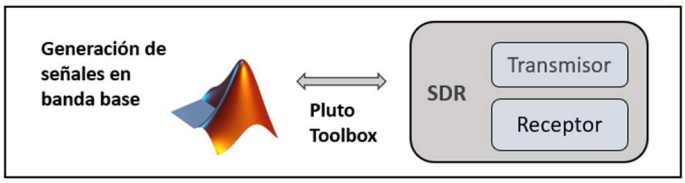

Figure 1. Interaction between MATLAB and the ADALM-Pluto.

The analog system is based on an AM modem, which uses a complex exponential as the carrier and an envelope detector in the demodulator. Figure 2 shows the block diagram of the AM modem, including both the modulator and the demodulator.

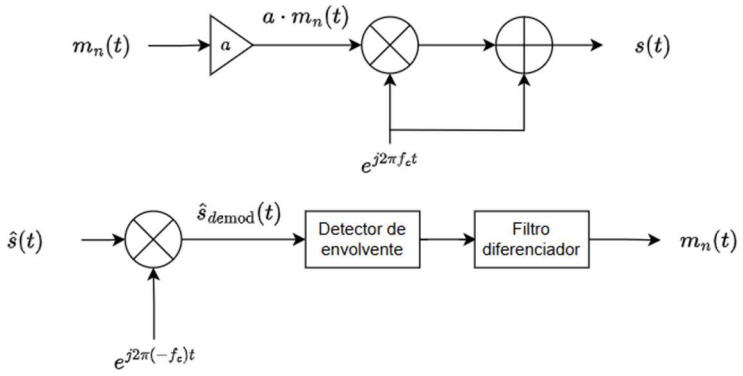

Figure 2. Block diagram of the AM modem.

The digital system is based on a BPSK modulator and a digital receiver. The receiver includes carrier synchronization, timing synchronization, and frame synchronization algorithms. Figure 3 presents the schematic of the BPSK modulator and the digital receiver.

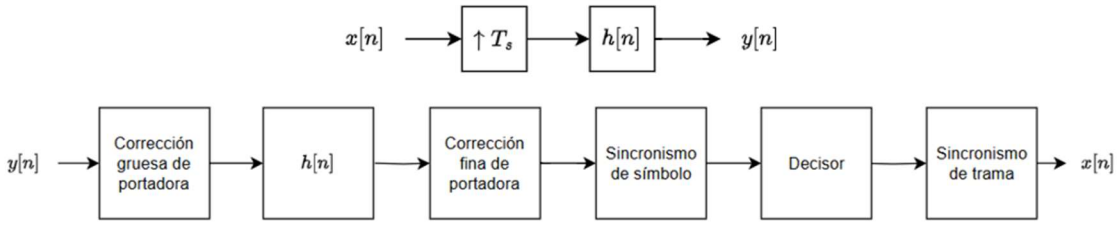

Figure 3. Block diagram of the BPSK modulator and digital receiver.

4. Results

- The transmitted and received audio signal using the ADALM-Pluto, demodulated through the MATLAB-based AM modem is shown in Figure 4. The presence of spurious components at 10 kHz, 15 kHz, and 20 kHz can be observed.

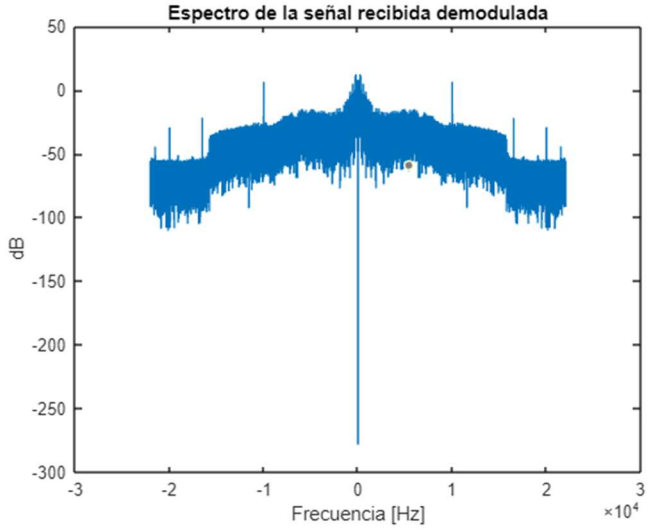

Figure 4. Audio signal processed using the AM modem.

- The character sequence was converted into a bitstream using source and channel encoders. This bit stream was then modulated using BPSK and transmitted with the ADALM-Pluto. The resulting constellation, after applying carrier, symbol, and frame synchronization techniques in the MATLAB-implemented digital receiver, is shown in Figure 5.

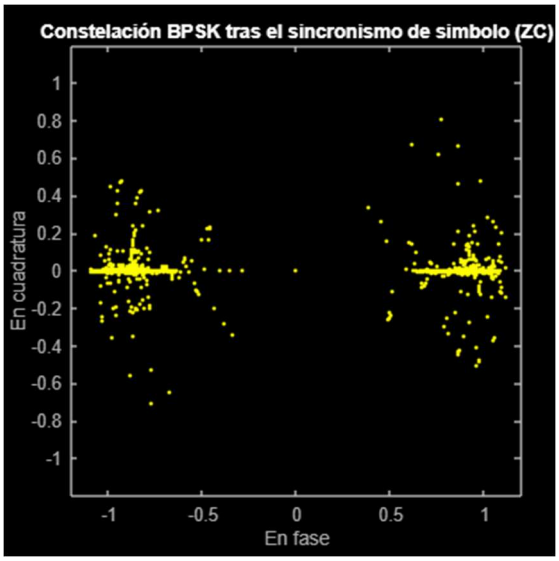

Figure 5. BPSK constellation processed by the digital receiver.

5. Conclusions

Despite the presence of nonlinear distortions in the received signal, the AM modem implemented in MATLAB successfully recovers the transmitted audio signal using the ADALM-Pluto. The signal-to-noise ratio improves significantly when using transmission and reception gains of -10 dB and 10 dB, respectively. The use of low-pass filters with cutoff frequencies of $f_c = 8\ kHz$ allows for the elimination of these spurious components. This confirms that SDR ADALM-Pluto is feasible for audio recovery.

Likewise, after digital processing, the character sequence transmitted using the SDR is correctly recovered. The bit error rate (BER) obtained is zero after applying the channel decoder.

The performance of the studied symbol synchronization algorithms improves for high values of samples per symbol, such as $sps = 10$. On the other hand, the experiments show that a small, normalized loop bandwidth value, such as $b_n = 0.001$, and a large damping factor, $\zeta = 2$ is beneficial for the PLL. The optimal roll-off factor (β) depends on the type of TED. Algorithms like Gardner or Zero-Crossing perform better with $\beta \in (0.5, 1)$, while the MM algorithm shows better performance with lower β values.

Therefore, these results support the conclusion that ADALM-Pluto is a suitable SDR platform for use in laboratory sessions related to communication system design and digital signal processing. These sessions would be based on the real-world communication tests carried out in this project.

1. References

[1] Mathworks, "Get Started with Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio," [Online]. Available: https://es.mathworks.com/help/comm/getting-started-with-communications-system-

toolbox-support-package-for-pluto-radio.html?s_tid=CRUX_lftnav. [Accessed 14 5 2025].

[2] T. F. Collins, R. Getz, D. Pu and A. M. Wyglinski, Software-Defined Radio for Engineers, Artech House, 2018.

[3] "IEEE Approved Draft Standard for Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management Amendment: Addition of New Terms and Associated Definitions," *IEEE P1900.1a/D2.0,* pp. 1-18, 7 Dec 2012.

[4] MathWorks, "Communications Toolbox," [Online]. Available: https://es.mathworks.com/help/comm/index.html. [Accessed 14 5 2025].

[5] L. F. Novales Peleato, *Implementación de técnicas mejoradas de comunicaciones digitales aplicadas a docencia,* Trabajo de Fin de Grado, Universidad Pontificia Comillas, 2021.

[6] Teleco Renta, «Recursos para Escuelas de Teleco,» [En línea]. Available: https://github.com/Telecorenta?tab=repositories. [Último acceso: 26 05 2025].

[7] M. Lázaro, Teoría de la Comunicación, Universidad Carlos III de Madrid Open Course Ware, 2014.

[8] A. Artés Rodrígez, F. Pérez González, J. Cid Sueiro, R. López Valcarce, C. Mosquera Nartallo y F. Pérez Cruz, Comunicaciones Digitales, Prentice Hall, 2012.

[9] M. Lichtman, «PySDR: Guía de uso para SDR/DSP con Python,» [En línea]. Available: https://pysdr.org/es/content-es/digital_modulation.html. [Último acceso: 23 5 2025].

[10] A. V. Oppenheim y R. W. Schafer, Tratamiento de señales en tiempo discreto, Tercera edición, Pearson Educación, 2011.

[11] R. W. Stewart, K. W. Barlee, D. S. W. Atkinson y L. H. Crockett, Software Defined Radio using MATLAB & Simulink and the RTL-SDR, Strathclyde Academic Media, 2015.

[12] MathWorks, «Coarse Frequency Compensator,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/coarsefrequencycompensator.html. [Último acceso: 27 05 2025].

[13] Y. Wang, K. Shi y E. Serpedin, «Non-Data-Aided Feedforward Carrier Frequency Offset Estimators for QAM Constellations: A Nonlinear Least-Squares Approach,» *EURASIP Journal on Advances in Signal Processing,* nº 13, 2004.

[14] M. Luise y R. R., «Carrier Frequency Recovery in All-Digital Modems for Burst-Mode Transmission,» *IEEE Transactions on Communications 43,* nº 2/3/4, 1995.

[15] MathWorks, «Carrier Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/carriersynchronizer.html. [Último acceso: 27 05 2025].

[16] MathWorks, «Symbol Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/symbolsynchronizer.html. [Último acceso: 27 05 2025].

[17] F. Gardner, «A BPSK/QPSK Timing-Error Detector for Sampled Receivers,» *IEEE Transactions on Communications,* vol. 34, nº 5, pp. 423-429, Mayo 1986.

[18] K. Mueller y M. Muller, «Timing Recovery in Digital Synchronous Data Receivers,» *IEEE Transactions on Communications,* vol. 24, nº 5, pp. 516-531, Mayo 1976.

Índice de la memoria

Índice de figuras

Índice de tablas

Índice de códigos

Capítulo 1. INTRODUCCIÓN

Como ingenieros en Tecnologías de la Telecomunicación, no basta con conocer los elementos de un sistema de comunicaciones a nivel teórico; es fundamental complementar su estudio con experimentos que refuercen su comprensión. Este es el enfoque que sigue la parte práctica de asignaturas como *Señales y Sistemas*, *Teoría de la Comunicación* o *Procesamiento Digital de Señales*. En este escenario, tiene un gran interés utilizar dispositivos innovadores que faciliten la asimilación de los conceptos teóricos.

Por otro lado, en el ámbito social, la conectividad es un elemento esencial y esto motiva una creciente importancia del sector de las Telecomunicaciones. Una de las tecnologías más punteras en este campo son las radios definidas por software (en adelante, SDR, por sus siglas en inglés, *Software Defined Radio*).

Las SDR tienen un enorme potencial como herramienta docente, pues integran en un único chip un sistema de comunicaciones completo.

1.1 MOTIVACIÓN DEL PROYECTO

Este proyecto busca explorar el potencial de las SDR como herramientas educativas. Para ello, se llevarán a cabo diferentes experimentos de transmisión y recepción de señales que podrían servir de base para futuras prácticas en el Grado en Ingeniería en Tecnologías de Telecomunicación. La incorporación de estas prácticas no solo ayudaría a asimilar conceptos clave, sino que introduciría tecnologías emergentes en la formación de los futuros ingenieros en Telecomunicación.

Los experimentos realizados se centran en dos tipos de sistemas: analógicos y digitales. Por un lado, se diseñará e implementará un módem AM. Por otro lado, se desarrollará un transmisor y receptor digital completo basado en una modulación BPSK. Finalmente, dentro

de este receptor digital, se implementarán y evaluarán diferentes algoritmos de sincronismo de símbolo.

1.2 ARQUITECTURA DEL SISTEMA

Una vez presentada la motivación del proyecto y los experimentos que se van a abordar en él, se presenta la arquitectura del sistema que utiliza la SDR escogida para este proyecto. Se trata de la *Adalm-Pluto* diseñada por la compañía *Analog Devices*. En concreto, la Figura 1.1 muestra la arquitectura del chip AD9361.

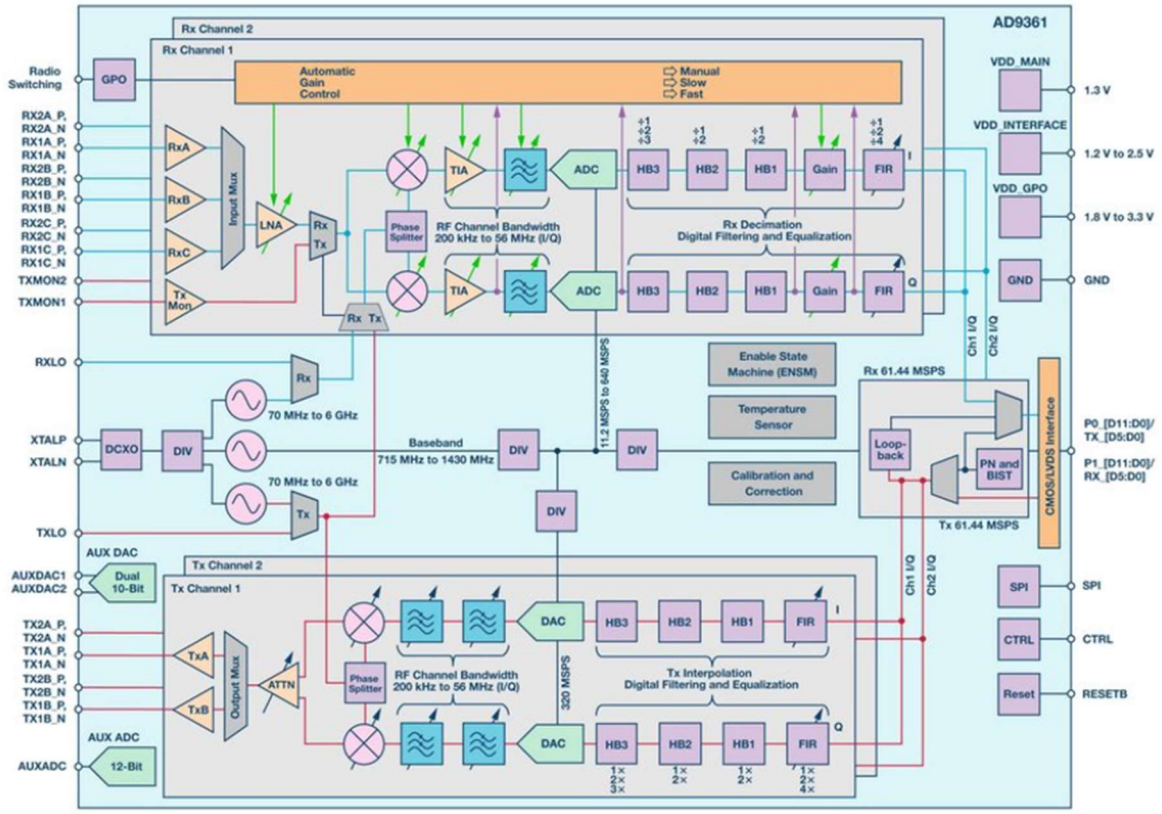

Figura 1.1. Diagrama de bloques del chip AD9361 [2].

La Figura 1.1 muestra dos partes claramente diferenciadas: en la parte superior se encuentra el receptor, mientras que la parte inferior muestra el transmisor. El transmisor presenta dos grandes bloques: el dominio digital en la zona derecha y el dominio analógico a la izquierda,

separados por el bloque DAC, por sus siglas en inglés, *Digital-to-Analog Converter*. En el dominio digital, el diagrama presenta un filtro FIR configurable y los interpoladores que adecuan la frecuencia de muestreo. En el dominio analógico, se muestran unos filtros responsables de seleccionar la primera banda de Nyquist, seguido de un modulador IQ.

El receptor funciona de forma análoga. En este caso, el dominio analógico presenta un amplificador de bajo ruido, seguido de un demodulador IQ y con filtros que previenen problemas de *aliasing*. En el dominio digital, ahora se tendrán diezmadores para adecuar la frecuencia de muestreo. Ambos dominios estarán unidos por medio de un ADC, por sus siglas en inglés, *Analog-to-Digital Converter*.

1.3 OBJETIVOS

A continuación, se describen los objetivos que abarca este proyecto:

- Estudiar cómo interactuar con los recursos *hardware* del dispositivo de la *Adalm-Pluto* mediante el lenguaje de programación MATLAB y programar la transmisión y recepción de un tono.
- Programar en MATLAB un módem AM y emplearlo para transmitir audio por medio de la SDR.
- Simular en MATLAB un transmisor y receptor digital basado en una modulación BPSK. Evaluar la *Bit Error Rate* (en adelante, BER) de la simulación.
- Llevar a cabo una transmisión y recepción digital mediante la SDR *Adalm-Pluto*. Evaluar la nueva BER experimental.
- Estudiar e implementar diferentes algoritmos de sincronismo de símbolo en el receptor digital mediante el lenguaje de programación MATLAB.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

Como ya se ha explicado, a lo largo de este proyecto se empleará la SDR *ADALM-Pluto*. Su configuración se consigue mediante el lenguaje de programación MATLAB, tal y como se describe a lo largo de este capítulo.

2.1 DEFINICIÓN DE SDR

Según el IEEE P1900.1 se define la SDR como una radio en la que algunas o todas las funciones de la capa física están definidas por software [3]. En esta definición, caben destacar los siguientes conceptos:

- **Radio**: tecnología empleada para la transmisión y recepción inalámbrica de radiación electromagnética para facilitar la transferencia de información.
- **Software**: instrucciones modificables ejecutadas por un dispositivo de procesamiento programable.
- **Capa física**: capa del protocolo inalámbrico en la que tiene lugar el procesado de radio frecuencias (RF en adelante), frecuencias intermedias (IF en adelante) o señales en banda base (incluyendo la codificación de canal). Es la capa más baja de la arquitectura ISO 7 adaptada a transmisiones y recepciones inalámbricas.
- **Definida por software**: se refiere al uso del procesamiento software en un sistema o dispositivo radio para implementar funciones operativas.

Los sistemas modernos de procesado de señal han progresado hasta tal punto que la mayoría de las funcionalidades en banda base están siendo implementadas mediante software [2].

2.2 CONFIGURACIÓN DE LA SDR MEDIANTE MATLAB

El lenguaje de programación en el que se desarrolla el presente proyecto es MATLAB. En concreto, para proyectos relacionados con sistemas de comunicación en general, y con este

proyecto en particular, se utilizan los entornos *Communications Toolbox* [4] y *Communications Toolbox Support Package for Analog Devices, Adalm Pluto* [1]. Por medio de este último entorno, es posible definir por software funciones de la capa física como la frecuencia de muestreo, la frecuencia de la portadora, el ancho de banda, la ganancia que se debe aplicar en transmisión y recepción…

Algunas funciones de la *Communications Toolbox Support Package for Analog Devices* que tienen especial interés en la configuración de la SDR se indican a continuación:

- Para crear el objeto "sistema transmisor" de la *ADALM-Pluto*, se utiliza "**sdrtx**".
- Para crear el objeto "sistema receptor" de la *ADALM-Pluto*, se usa "**sdrrx**".
- Para efectuar una transmisión, se tiene el método "**transmitRepeat**" del objeto sdrtx.
- Para habilitar la recepción, se emplea la función "**rx**".
- Para dejar de transmitir y recibir, se emplea la función "**release**".

En el Código 1 del ANEXO II se muestra un ejemplo de cómo configurar la *ADALM-Pluto* para habilitar la transmisión y recepción de una señal en banda base mediante el código de la función *plutoTransmitter*.

La Figura 2.1 muestra un esquema que ilustra la interacción de las diferentes tecnologías descritas en este capítulo. Por un lado, MATLAB es el lenguaje de programación que se utiliza para la creación de las señalas en banda base que se transmiten utilizando la SDR. A continuación, la *toolbox* de la ADALM-Pluto es la responsable de definir por software las funcionalidades de la capa física. Finalmente, mediante el chip AD9361 se logra transmitir y recibir las señales banda base definidas previamente.

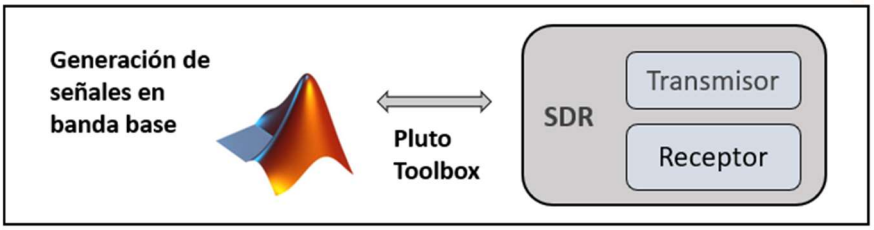

Figura 2.1. Esquema de la interacción entre MATLAB y la ADALM-Pluto.

Esto es posible gracias al firmware de la ADALM-Pluto. Entre sus funcionalidades, se encuentra definir el comportamiento software a través de librerías como *libiio* (*Industrial I/O library*), la mencionada *Communications Toolbox Support Package for Analog Devices, Adalm Pluto* de MATLAB y *pyadi-iio* de GNU Radio y Python. También fija el rango de frecuencias, presenta una interfaz web accesible a través de la IP 192.168.2.1 y permite activar modos de *loopback* para pruebas de transmisión y recepción sin salir del dispositivo.

Capítulo 3. ESTADO DE LA CUESTIÓN

A lo largo de este capítulo, se presenta la evolución de los sistemas de comunicación hacia las arquitecturas Cero-IF típicas en las SDR. Además, se muestran las distintas alternativas más populares de SDR en el mercado. Finalmente, se exponen distintos contextos docentes donde ya se están utilizando las SDR.

3.1 ARQUITECTURAS RF PARA SDR

La nueva generación de sistemas de comunicación presenta desafíos que requieren filtros configurables o diseños RF con un ancho de banda más flexible para facilitar la interoperabilidad de diferentes aplicaciones. El núcleo del diseño de los sistemas de comunicación radio ha sido la arquitectura superheterodina que se muestra en la Figura 3.1.

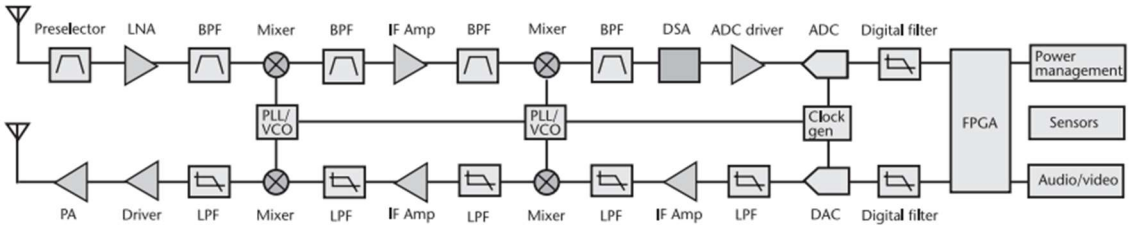

Figura 3.1. Diagrama de bloques basado en una arquitectura superheterodina multietapa [2].

Las principales ventajas de este tipo de arquitecturas son la reducción de emisiones espurias, el hecho de que se fija el ancho de banda del canal mediante filtros IF y la distribución de la ganancia a lo largo de las diferentes etapas [2].

Sin embargo, los filtros de alto rendimiento son físicamente grandes, lo cual no acompaña a la miniaturización de componentes electrónicos. Además, los filtros IF fijan el ancho de banda, dando lugar a diseños que no son reutilizables en otros sistemas. Estos inconvenientes motivan el diseño de arquitecturas alternativas, como la Cero-IF. En ella, se utiliza una única etapa mezcladora con un oscilador local sintonizado a la frecuencia de la banda de interés.

De este modo, se desplaza la señal recibida a banda base tanto en fase (I) como en cuadratura (Q). Por ello, utilizan moduladores y demoduladores IQ [2].

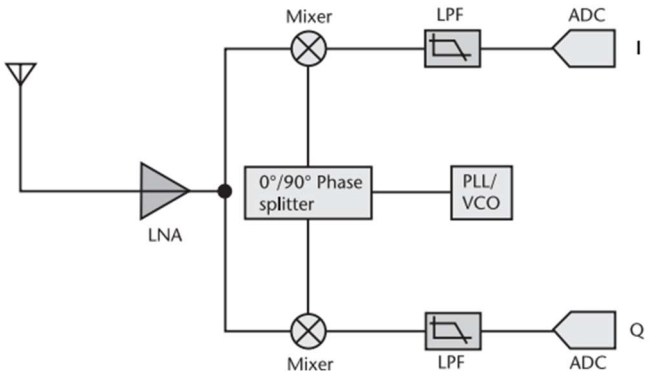

Figura 3.2. Diagrama de bloques basado en una arquitectura Cero-IF [2].

3.2 DISPOSITIVOS SDR EN EL MERCADO

En este apartado, se analizan algunas de las alternativas comerciales más populares de SDR del mercado. Para ello, la Tabla 1 muestra una comparativa de las principales características técnicas y económicas de las 3 SDR más populares en el contexto didáctico.

Características	**RTL-SDR**	**ADALM-Pluto**	**HackRF One**
Precio	30-50 €	150-200€	300-350€
Rango de Frecuencias	De 500 kHz hasta 1,75 GHz	De 325 MHz hasta 3,8 GHz	De 1 MHz hasta 6 GHz
Resolución del ADC	8 bits	12 bits	8 bits
Ancho de banda	3,2 MHz	20 MHz	20 MHz
Diseño *Open Source*	Sí	Sí	Sí
Interfaz	USB 2.0	USB 2.0 (Ethernet opcional)	USB 2.0
Incluye transmisión y recepción	Solo recepción	Sí	Sí

Tabla 1. Comparativa de dispositivos SDR en el mercado.

Para este proyecto, se ha escogido la ADALM-Pluto. Esto se debe a que fue desarrollada como parte del programa *Active Learning Modules* y está pensada para facilitar el aprendizaje de conceptos de sistemas de comunicación y procesado digital de señales. Cabe destacar que, a pesar de que la ADALM-Pluto está diseñada para transmitir hasta los 3,8 GHz, esto se puede modificar. Utilizando una comunicación serial con la SDR mediante la aplicación PuTTY, es posible utilizar portadoras de hasta 6 GHz mediante los siguientes comandos[1]:

```
fw_setenv attr_name compatible

fw_setenv attr_val ad9364

fw_setenv maxcpus

reboot
```

También, la ADALM-Pluto incluye una capa de abstracción que hace más sencilla la configuración de los parámetros hardware involucrados como la frecuencia de muestreo o el ancho de banda [5]. Esta SDR es compatible con software educativo como MATLAB/Simulink o GNU Radio y cuenta con una comunidad académica que ofrece una buena documentación con ejemplos y laboratorios didácticos.

3.3 EJEMPLOS DE USO DE SDR EN EL ÁMBITO DOCENTE

Las SDR se están introduciendo progresivamente en el ámbito docente. Un claro ejemplo de esto son los talleres de *Teleco Renta* [6]. Estos talleres son desarrollados por diferentes grupos de investigación. Su objetivo es poner a disposición de las diferentes Escuelas de Ingenieros en Telecomunicación prácticas para ofrecer a sus estudiantes, incorporando el uso de las SDR. Algunos de los talleres más representativos abordan las siguientes temáticas:

[1] Para acceder a la configuración de la pluto hay que usar el usuario *root* y la contraseña *analog*, configurados por defecto.

- **Sincronización en comunicaciones móviles**: este taller está pensado para introducir a los estudiantes al concepto de sincronización en frecuencia de sistemas multiplexados en frecuencia (OFDM).
- **Comunicaciones móviles con SDR**: este taller familiariza al estudiante con la tecnología SDR. En concreto, se centra en comunicaciones inalámbricas en redes celulares.
- **Comunicación MIMO con SDR**: este taller explora la comunicación multiusuario con sistemas basados en antenas MIMO.

Capítulo 4. COMUNICACIONES ANALÓGICAS

En este capítulo se presenta una introducción a las comunicaciones analógicas y sus principales características. En concreto, se centrará en la modulación convencional AM, presentando un esquema de su transmisor y receptor.

4.1 INTRODUCCIÓN A LOS SISTEMAS DE COMUNICACIÓN ANALÓGICOS

Un sistema de comunicación analógico envía la información impresa en una señal continua y en tiempo continuo. En este sentido, la forma de onda es la clave. Por tanto, lo más importante en un sistema de comunicación analógico es transmitir la forma de onda de un extremo a otro de la manera más fiel posible. Generalmente, una señal analógica es muestreada y cuantificada, para después transmitirse mediante un sistema de comunicación digital. No obstante, aún existen sistemas de radiodifusión de voz y vídeo basados en sistemas analógicos [7].

En concreto, para este proyecto se emplea una transmisión paso-banda, empleando una modulación AM convencional. En ella, el espectro de la señal de información se traslada y pasa a estar centrada en torno a una cierta frecuencia de portadora.

Las modulaciones AM convencionales son modulaciones lineales. Se caracterizan porque la amplitud de la portadora varía en el tiempo. Otras alternativas más sofisticadas son las modulaciones angulares de fase (PM) o de frecuencia (FM). En ellas, el valor angular de la portadora varía en función del tiempo.

4.2 MODULACIÓN AM CONVENCIONAL

La expresión analítica de una señal moduladora es [7]:

$$s(t) = c(t) + m_a(t) \cdot c(t) = A_c \cdot [1 + a \cdot m_n(t)] \cdot \cos(2\pi f_c t + \phi_c)$$

Ecuación 4.1. Expresión analítica de una modulación AM.

donde A_c, f_c y ϕ_c son la amplitud, frecuencia y fase de la señal portadora respectivamente. Asimismo, $m_n(t)$ es la señal moduladora normalizada según el desarrollo mostrado en el ANEXO III. Para prevenir el problema de la *sobremodulación*, se introduce un índice de modulación a tal que:

$$0 < a \leq 1$$

La Figura 4.1 muestra un ejemplo de señal moduladora en MATLAB. En este caso, se ha escogido un tono puro a 10 kHz.

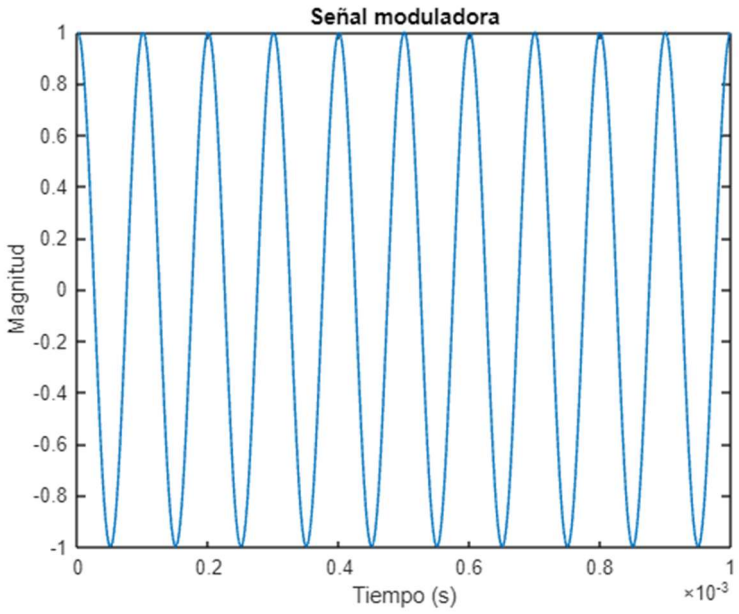

Figura 4.1. Señal moduladora siguiendo un esquema AM.

Por otro lado, la Figura 4.2 y la Figura 4.3 muestran un ejemplo de señal portadora y de señal modulada respectivamente. La señal portadora es una exponencial compleja a 200 kHz. El hecho de emplear una exponencial compleja como portadora en lugar de un coseno se explica más adelante. Por otro lado, la señal modulada de la Figura 4.3 tiene un índice de modulación $a = 1$ y una envolvente roja que señala la forma de onda de la señal moduladora.

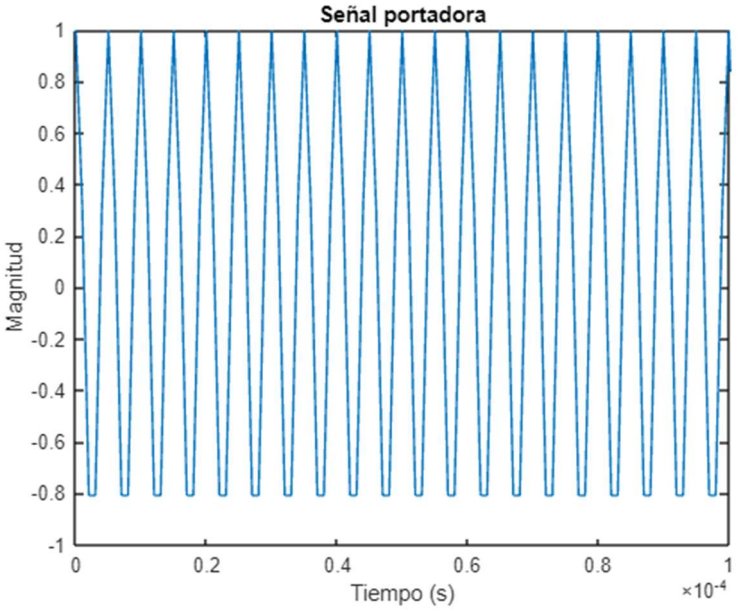

Figura 4.2. Señal portadora siguiendo un esquema AM.

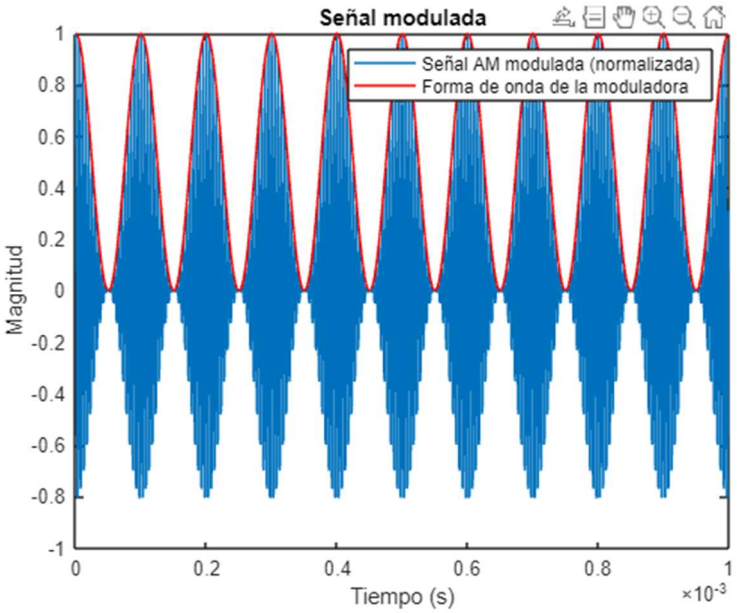

Figura 4.3. Señal modulada siguiendo un esquema AM.

En este caso, se emplea el máximo índice de modulación posible. Utilizar un índice mayor provocaría la *sobremodulación* de la señal y el esquema de demodulación sería más complejo.

4.3 MODULADOR AM

El diagrama de bloques que se ha empleado para crear el módem AM se muestra en la Figura 4.4. La señal $m_n(t)$ está normalizada y se multiplica por el índice de modulación a. Seguidamente, se utiliza una exponencial compleja como señal portadora. Cabe señalar que, como es característico en este tipo de modulaciones, la señal modulada $s(t)$ incluye la información de la señal $m_n(t)$ y de la portadora $c(t)$.

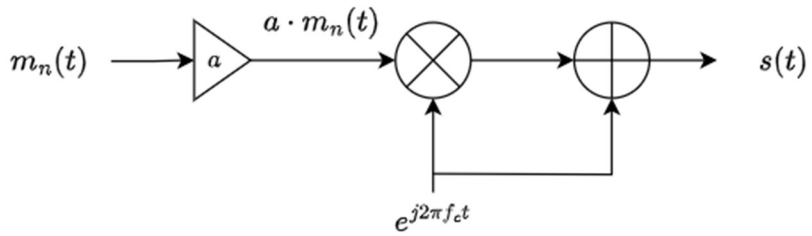

Figura 4.4. Diagrama de bloques de un módem AM.

El motivo de emplear una exponencial compleja como portadora en lugar de un coseno es evitar la réplica de la señal $m_n(t)$ en frecuencias negativas. La Ecuación 4.1 mostraba la expresión analítica de una modulación AM. Analizando su espectro, se tiene que la transformada de Fourier de la señal $s(t)$ indicada en la expresión anterior es:

$$S(f) = \frac{A_c}{2}[\delta(f - f_c) + \delta(f + f_c)] + \frac{A_c \cdot a}{2}[M_n(f - f_c) + M_n(f + f_c)]$$

Ecuación 4.2. Transformada de Fourier de una modulación AM convencional.

Donde, por simplicidad, se ha asumido que:

$$\phi_c = 0$$

En la Ecuación 4.2, $\delta(f)$ denota la delta de Dirac y $M_n(f)$ es el espectro de la señal $m_n(t)$.

En este caso, el espectro de la señal $s(t)$ sería el que se muestra en la Figura 4.5. En este caso, se ha utilizado como señal $m_n(t)$ un tono puro a 10 kHz.

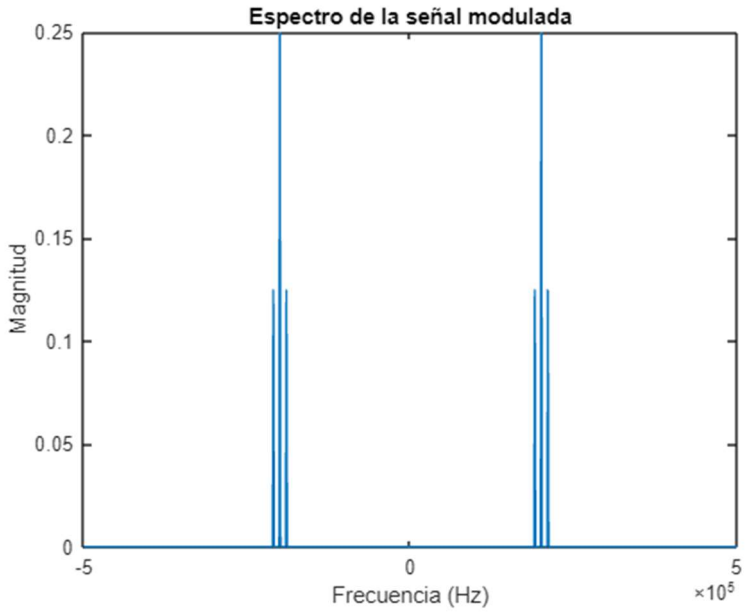

Figura 4.5. Espectro de una modulación AM utilizando un coseno como portadora.

No obstante, si se utiliza como portadora una exponencial compleja, entonces, la nueva expresión analítica $s(t)$ es:

$$s(t) = A_c \cdot [1 + a \cdot m_n(t)] \cdot \mathrm{e}^{\mathrm{j}2\pi f_c t + \phi_c}$$

Ecuación 4.3. Expresión analítica de una modulación AM con una exponencial compleja.

Donde el nuevo espectro de la señal $s(t)$ es:

$$S(f) = \frac{A_c}{2} \cdot \delta(f - f_c) + \frac{A_c \cdot a}{2} \cdot M_n(f - f_c)$$

Ecuación 4.4. Espectro de una modulación AM con una exponencial compleja.

Con esta nueva portadora, el espectro de la señal $m_n(t)$ queda tal y como se muestra en la Figura 4.6. En esta imagen, se observa claramente que ya no existe una réplica de la señal $m_n(t)$ en torno a la frecuencia negativa de portadora $-f_c$.

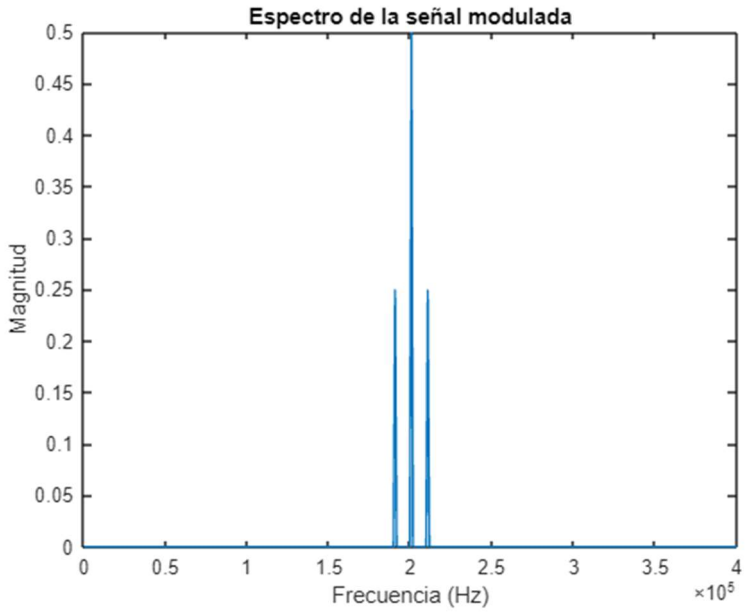

Figura 4.6. Espectro de una modulación AM usando una exponencial compleja como portadora.

En el Código 2 del ANEXO II se encuentra el desarrollo en MATLAB del módem AM.

4.4 DEMODULADOR AM

El receptor AM construido está basado en un demodulador no coherente [7] que utiliza un detector de envolvente. Esta es la alternativa más simple cuando se utiliza una modulación AM convencional. Este esquema se muestra en la Figura 4.7, donde el procesado se realiza sobre la señal $\hat{s}(t)$. Esta señal representa la señal modulada $s(t)$ obtenida en la sección 4.3 afectada por las perturbaciones y distorsiones [7] propias del canal de comunicación.

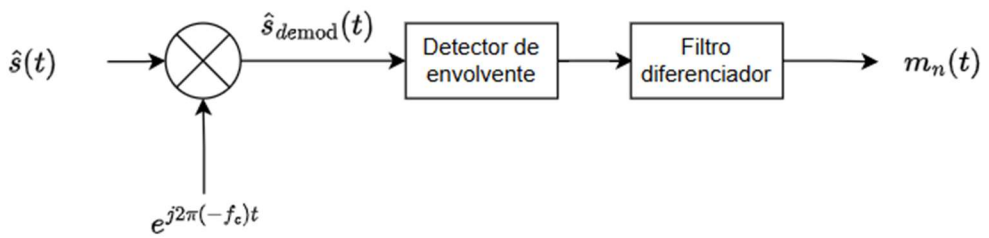

Figura 4.7. Esquema del demodulador AM no coherente mediante detección de envolvente.

Por simplicidad, en la simulación de la transmisión implementada, se asume que

$$s(t) = \hat{s}(t)$$

Seguidamente, se utiliza una exponencial compleja con frecuencia $-f_c$, donde f_c es la frecuencia de portadora definida en la sección 4.3. El espectro resultante se muestra en la Figura 4.8, donde se comprueba que el espectro está ahora centrado en torno al 0.

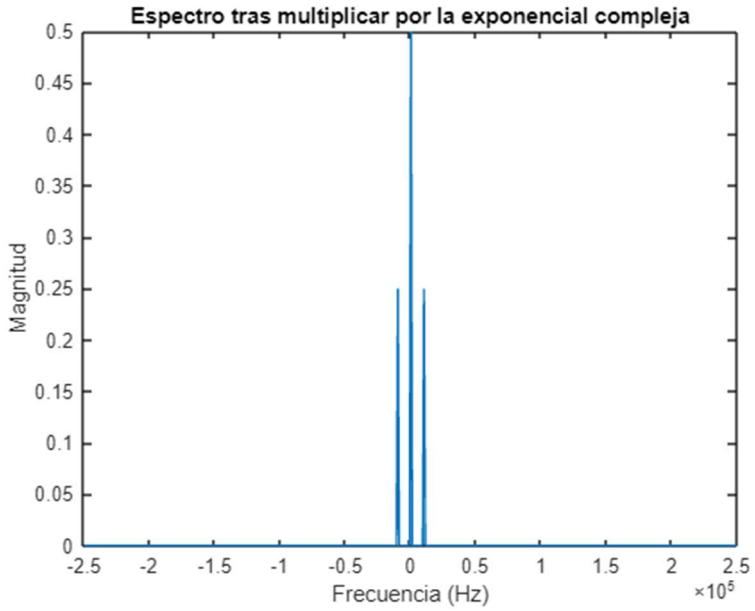

Figura 4.8. Espectro de la señal AM en la recepción tras multiplicar por la portadora.

A continuación, se toma la envolvente de la señal, la cual se muestra en la Figura 4.9. Tal y como se adelantó en la sección 4.2, la forma de onda de la señal recibida es la clave en los sistemas analógicos basados en modulaciones AM. Concretamente, esta simulación muestra cómo es posible recuperar la señal transmitida a partir de este detector de envolvente.

El filtro diferenciador [5] es un filtro cuya respuesta en frecuencia es tal que elimina la componente continua de la señal. Dicho de otro modo, solo deja pasar aquellas frecuencias distintas de 0 Hz. La razón de esto es eliminar la portadora que se ha transmitido en la señal modulada $s(t)$. De este modo, se obtendría finalmente la señal $m_n(t)$ transmitida, cuyo espectro se muestra en la Figura 4.10.

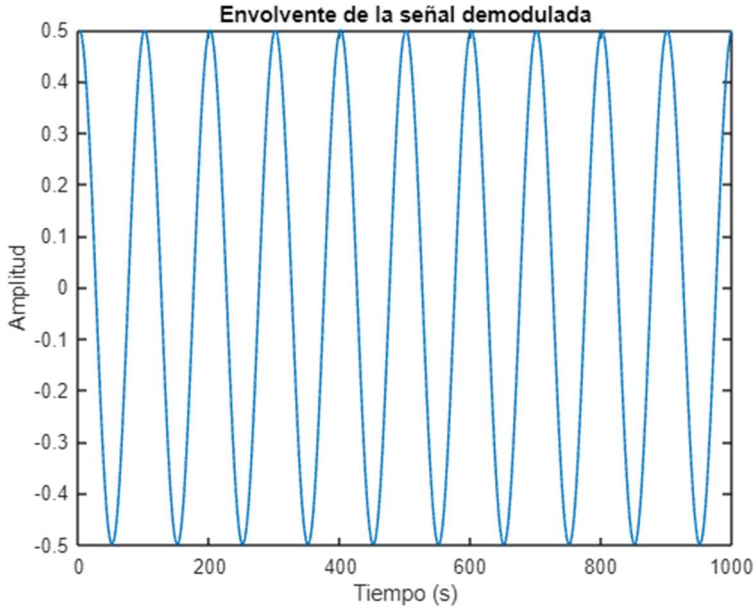

Figura 4.9. Envolvente de la señal AM recibida.

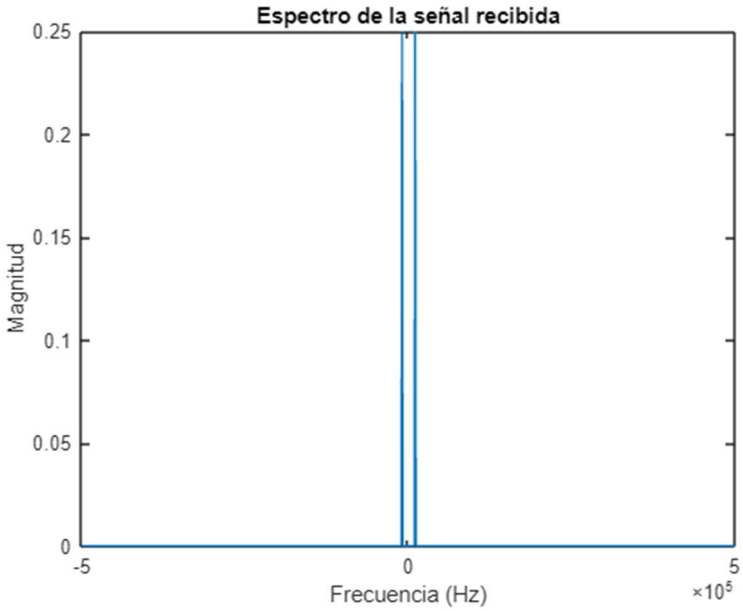

Figura 4.10. Efecto del filtro diferenciador sobre la señal AM recibida.

El código que se ha empleado para el desarrollo de este demodulador se puede encontrar en el Código 3 del ANEXO II. Cabe señalar nuevamente en este punto las ventajas de utilizar

una exponencial compleja como portadora. Utilizando un coseno, la señal demodulada habría generado réplicas en $2f_c$ y $-2f_c$, tal y como se muestra en la Figura 4.11.

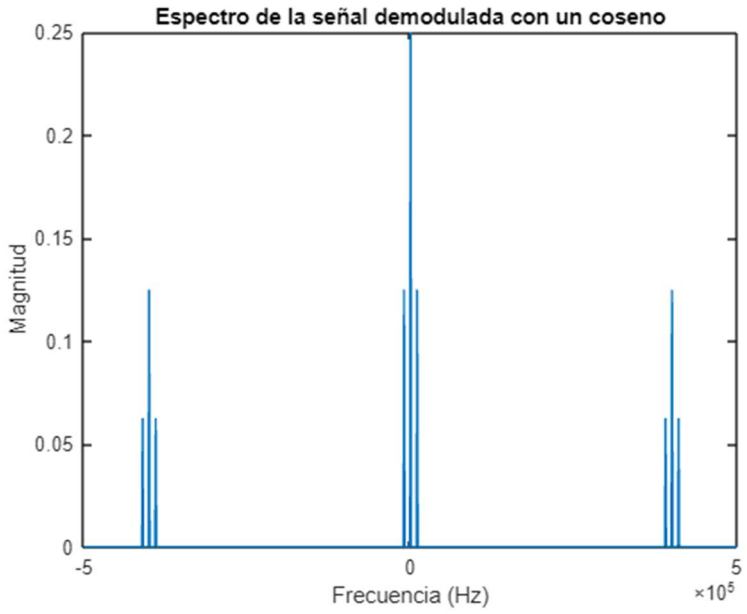

Figura 4.11. Espectro de la señal AM demodulada con un coseno.

Esto habría requerido el diseño de un filtro paso bajo para eliminar las réplicas del espectro en altas frecuencias. Además, el espectro de la señal centrado en 0 Hz está solapado, lo cual provoca problemas en osciladores reales que no están perfectamente sincronizados. En su lugar, usar una exponencial compleja como portadora evita tener que introducir estos filtros, al provocar únicamente un desplazamiento en frecuencia.

Capítulo 5. COMUNICACIONES DIGITALES

La diferencia entre un sistema de comunicaciones analógico y un sistema de comunicaciones digital [8] es que en el segundo el número de formas de onda que se pueden transmitir es finito, lo que no ocurre en el primer caso. Estas formas de onda se corresponden con los bits. La información digital se transmite mediante estas formas de onda o señales $s_k(t)$ a las que se denominan símbolos [8]. Cada símbolo agrupa N bits, por lo que en total se necesitan $M = 2^N$ combinaciones para cubrir todas las combinaciones posibles.

Algunas de las ventajas de utilizar la transmisión digital son la capacidad de abstracción de los bits, la regeneración de la señal, la integración de diferentes servicios o la facilidad de multiplexación [8], entre otras. No obstante, también requiere un mayor ancho de banda. Además, para fuentes de información analógicas, al introducir un ADC en el transmisor y un DAC en el receptor se añade un ruido irreversible de cuantificación. Por otro lado, uno de los principales inconvenientes es la complejidad del receptor digital, que necesita sincronismo de portadora, de trama y de símbolo.

La Figura 5.1 muestra los diferentes elementos que conforman un sistema de comunicaciones digital.

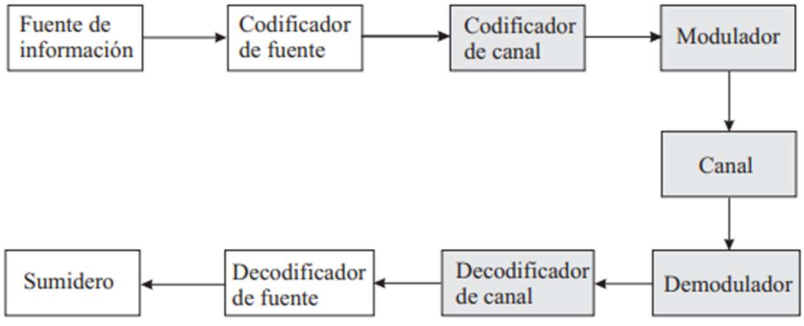

Figura 5.1. Elementos de un sistema de comunicaciones digitales [8].

Cabe señalar que, si bien la fuente de información típicamente es de naturaleza digital, como un fichero, también se pueden tener fuentes de información analógicas. En este caso, es necesario agregar al esquema un conversor A/D previo al codificador de fuente y un conversor D/A posterior al decodificador de fuente. Generalmente, esto se incluye en el bloque que se ha denominado sumidero [8] en la Figura 5.1.

A lo largo de los siguientes puntos y capítulos, se analizan en detalle los demás bloques que conforman un sistema de comunicaciones digitales. Para ello, se crean estos bloques mediante el lenguaje MATLAB y se simula una transmisión y recepción de datos mediante un sistema de comunicaciones digital.

5.1 CODIFICADOR DE FUENTE Y DECODIFICADOR DE FUENTE

El primer bloque del sistema de la Figura 5.1 es el codificador de fuente [8]. Su objetivo es eliminar la redundancia del mensaje que emite la fuente. Para ello, es necesario expresar esta información de la manera más compacta posible, es decir, con el menor número de bits posible.

Esto se debe al hecho de que enviar información por medio de un canal implica un consumo de ancho de banda. Por lo tanto, solo se debe transmitir aquella información que sea estrictamente necesaria. La entropía es una medida que permite saber si el mensaje se ha codificado adecuadamente para que sea lo más compacto posible. Esto se consigue gracias al Primer Teorema de Shannon.

El Primer Teorema de Shannon sostiene que la longitud media L de todo código prefijo para una variable aleatoria discreta X es mayor o igual que su entropía $H(X)$:

$$L \geq H(X)$$

Para el cálculo de la entropía, se considera una variable aleatoria discreta X que puede tomar uno de entre M símbolos posibles, denotados como x_i de acuerdo con la distribución de probabilidades de aparición $p_x(x_i)$. El cálculo de la entropía $H(X)$ [8] se define como

$$H(X) = \sum_{i=1}^{M} p_x(x_i) \log_2 \frac{1}{p_x(x_i)} = -\sum_{i=1}^{M} p_x(x_i) \log_2 p_x(x_i)$$

y se expresa en bits.

En este proyecto, para simular una comunicación, se crea una secuencia de caracteres que incluye 4 letras: "A", "B", "C" y "D". Cada una de ellas presenta una frecuencia de aparición diferente, que se calcula de forma dinámica en función de cada secuencia. El objetivo es recuperarlas de forma exacta después de realizar la transmisión.

En este contexto, la función del codificador de fuente es la de asignar una secuencia de bits a cada letra. Para minimizar la longitud media del símbolo, a las letras con un ratio de frecuencia mayor se les asigna un menor número de bits. Para las letras con un ratio de frecuencia menor, se asigna un mayor número de bits. Para ello, se utilizarán códigos instantáneos.

Por ejemplo, si se tiene que la letra "A" aparece en un 60%, la letra "B" en un 25% y la letra "C" en un 15%, la asignación de bits sería:

A 0

B 10

C 110

Para implementar un codificador de fuente en Matlab, se ha desarrollado una función auxiliar que se presenta en el Código 4 del ANEXO II. Esta función recibe como parámetro una cadena de caracteres y devuelve dos parámetros:

- El texto convertido a bits según el criterio expuesto.
- La relación entre cada letra y su símbolo asociado.

Por otro lado, el decodificador de fuente es uno de los elementos que conforman el receptor digital. Realiza la función inversa del codificador de fuente. Es decir, recibiendo como

parámetros una cadena de bits y la relación entre cada letra y un símbolo, el decodificador identifica qué bits conforman un símbolo en la cadena recibida y lo convierte en la letra correspondiente según el mapeo realizado en el codificador. De este modo, se recupera la cadena de caracteres original. La función auxiliar que implementa este decodificador se muestra en el Código 5 del ANEXO II. Esta función recibe como parámetros una cadena de bits y la relación entre cada letra y su símbolo asociado, y devuelve una cadena de caracteres.

5.2 CODIFICADOR DE CANAL Y DECODIFICADOR DE CANAL

Para proteger los bits durante la transmisión, se utiliza el bloque denominado codificador de canal mostrado en la Figura 5.1. Para ello, este elemento introduce redundancia a propósito y de forma inteligente para contrarrestar los efectos del canal.

Los códigos Hamming [8] son una familia de códigos que permiten detectar errores dobles y corregir errores simples. Se representan mediante la notación $C(n,k)$ con bloques de k bits a la entrada y n bits a la salida. Un ejemplo de código Hamming $C(3,1)$ provocaría la siguiente transformación sobre los bits:

$$0 \rightarrow 000$$

$$1 \rightarrow 111$$

La implementación del codificador se consigue mediante el Código 6 del ANEXO II.

Para observar más claramente el efecto de este elemento, la Figura 5.2 muestra una secuencia de bits[2] obtenida al aplicar un codificador de fuente a una cadena de caracteres. Seguidamente se aplica a esta misma cadena un codificador de canal que emplea un código Hamming $C(3,1)$. Asimismo, se muestra el espectro de ambas secuencias.

[2] La frecuencia de muestreo usada en la generación de esta secuencia es $f_s = 1\ MHz$.

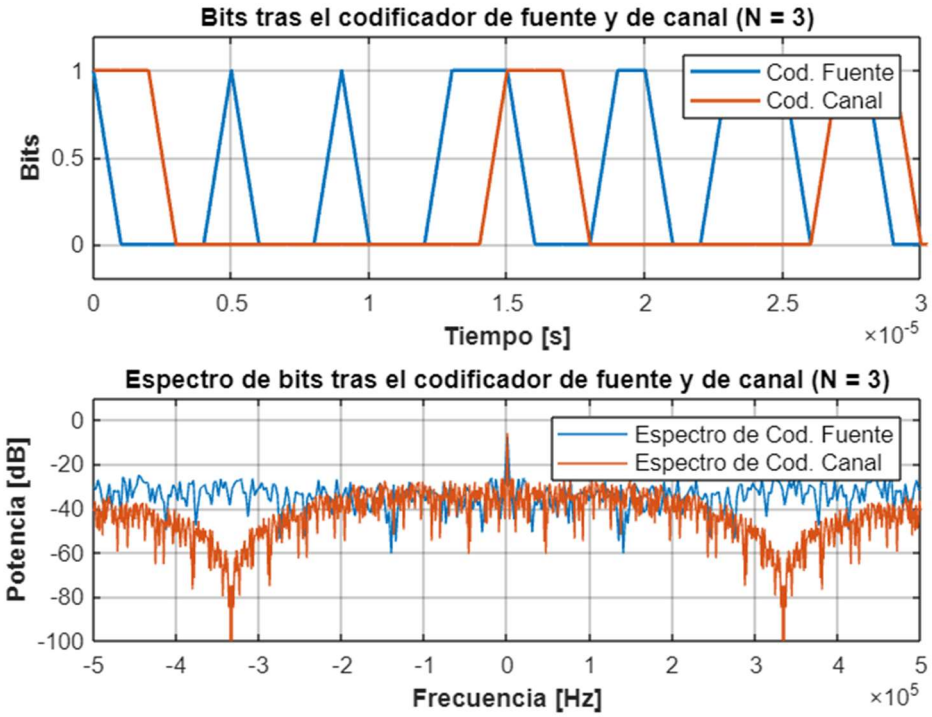

Figura 5.2. Forma de onda y espectro de una secuencia de bits tras el codificador de fuente y canal.

El decodificador de canal no solo realiza la función inversa del codificador de canal. En caso de detectar un error en la transmisión, lo corrige. Por ejemplo, si llegara una secuencia de bits que tuviera la forma 110, sería claro que se ha producido un error durante la transmisión, ya que las únicas cadenas válidas que se podrían recibir serían 000 o 111. Lo más probable es que el bit que ha llegado como 0 en realidad fuera un 1. Entonces, el decodificador de canal es capaz de corregir este error, tomando la cadena 110, corrigiéndola a la cadena 111 y devolviendo finalmente el bit 1. La implementación del decodificador se muestra en el Código 7 del ANEXO II.

5.3 MODULADOR BPSK

El modulador [8] de la Figura 5.1 es el elemento que transforma los bits que salen del codificador en formas de onda que, en el caso de una modulación digital, pertenecen a un conjunto finito. Estas formas de onda, $s_i(t)$ se adecúan a las características del medio de transmisión. Adecuar al medio es necesario ya que, a menor frecuencia, mayor es el tamaño

de la antena requerida, y ya que los pulsos cuadrados ocupan un espectro mayor [9]. El número de formas de onda depende de cada modulación. Modulaciones que agrupen dos bits, con $N = 2$ generan $M = 2^2$ símbolos, por ejemplo. Estas señales de energía finita admiten una representación como vectores dentro de un espacio vectorial [8]. En consecuencia, se tiene que la señal $s(t)$ se puede expresar como

$$s(t) = \sum_{n} \sum_{j=0}^{N-1} A_j[n] \phi_j(t - nT)$$

donde ϕ_j es el elemento j de la base ortonormal y $A_j[n]$ es la coordenada j del símbolo n.

En el contexto de las comunicaciones digitales, se distinguen dos grandes bloques de modulaciones: las modulaciones de amplitud, PAM o ASK, y las modulaciones de frecuencia y fase FSK y PSK.

Las modulaciones de amplitud modifican la amplitud de señales de acuerdo con los símbolos a transmitir. Se dice que la señal $s(t)$ está en banda base cuando no se ha trasladado a otras bandas de frecuencia. En caso de que sí se haya trasladado a otras bandas de frecuencia, se habla de señales paso banda. Por su parte, las modulaciones FSK y PSK construyen la señal a transmitir modulando la frecuencia y la fase de la portadora respectivamente.

Para el caso particular de una modulación BPSK, se tiene un bit por símbolo. Es decir, según la notación seguida en este proyecto, $N = 1$ y por extensión hay un total de $M = 2^1$ símbolos. Un ejemplo gráfico de una modulación BPSK se muestra en la Figura 5.3.

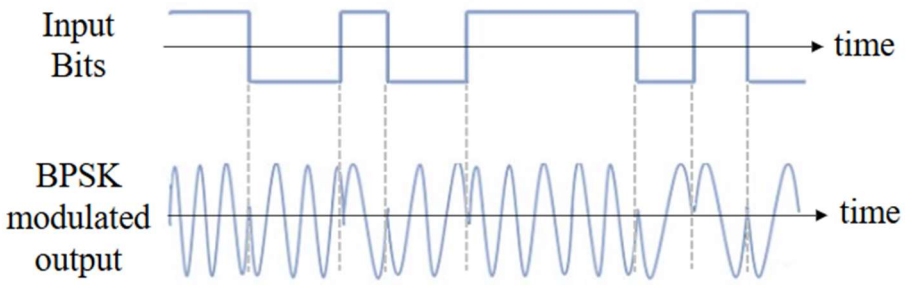

Figura 5.3. Ejemplo de una señal $s(t)$ a la salida de un modulador BPSK [9].

No obstante, para una representación que permita mayor interpretabilidad se utiliza el plano complejo. A estas representaciones se les denomina constelaciones [9]. La Figura 5.4 muestra un ejemplo de una constelación BPSK.

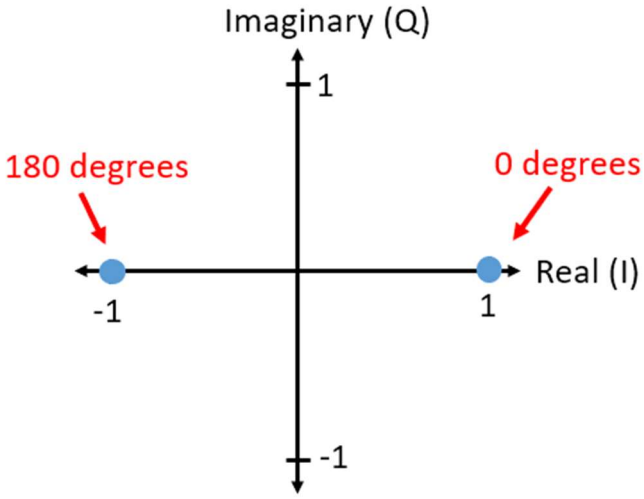

Figura 5.4. Ejemplo de una constelación BPSK [9].

Una base ortonormal habitual para modulaciones BPSK es aquella formada por las señales:

$$\phi_0 = A \cdot sen(2\pi f_c t)$$

$$\phi_1 = A \cdot \cos(2\pi f_c t)$$

donde A es la amplitud de la señal tal que la base esté normalizada, ya que el seno y el coseno son ortonormales.

Este proyecto contempla dos formas de implementar un modulador BPSK. Por un lado, en MATLAB, se puede encontrar la función *pskmod* de la *Communications Toolbox* [4]. Esta función recibe como parámetros el orden de la modulación, M, un vector $x(t)$ con valores enteros comprendidos entre $[0, M-1]$, un desplazamiento de fase para la constelación expresado en radianes y el orden en el que se asignan los símbolos. Empleando esta primera alternativa, la función se puede invocar como:

```
samplesTx = pskmod(bitsTx, M);
```

donde `bitsTx` es la cadena binaria que se obtiene a la salida del codificador de canal y `M` es el orden de la modulación[3].

La otra alternativa para implementar un modulador BPSK es utilizar un interpolador junto a un filtro adaptado. La Figura 5.5 muestra el diagrama de bloques que permite esta implementación. En ella, cabe señalar que la secuencia $x[n]$ ya no está formada por una secuencia de 0 y 1, sino que todos los bits 0 se han transformado en -1.

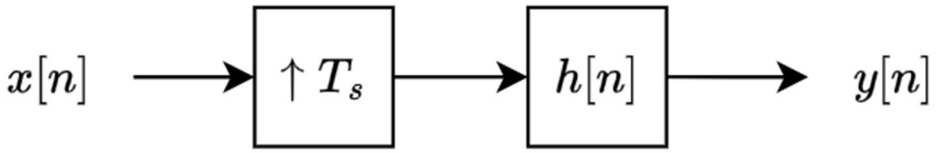

Figura 5.5. Diagrama de bloques de un modulador BPSK [5].

5.3.1 INTERPOLADOR DEL MODULADOR BPSK

Un interpolador [10] permite incrementar la frecuencia de muestreo por un factor entero y su salida viene definida por la expresión

$$x_e[n] = \begin{cases} x[n/T_s], & n = 0, \pm T_s, \pm 2T_s, \dots \\ 0, & en\ el\ resto \end{cases}$$

donde T_s es el tiempo de símbolo. De este modo, la entrada del interpolador $x[n]$ es una secuencia de bits con valores 0 y 1. Lo que el interpolador hace internamente es, entre dos bits consecutivos, insertar ceros. El número de ceros que introduce el interpolador depende del factor de interpolación, que en este caso es T_s. En definitiva, el interpolador inserta entre dos bits consecutivos $T_s - 1$ ceros.

El resultado que se obtiene con este algoritmo es que a la salida del interpolador ya no haya bits, sino símbolos. La duración de este símbolo viene definida por T_s. Por ejemplo, en el

[3] Para una modulación BPSK, el valor de *M* es 2.

caso de que $T_s = 10$, un símbolo dura 10 bits. Por lo que, en el receptor, se deberá realizar el proceso inverso, y tener en cuenta que 10 bits recibidos, en realidad, corresponden a un símbolo, y este, a su vez, a 1 bit, ya que el orden de esta modulación es $M = 2$. La Figura 5.6[4] muestra la secuencia de bits original y el efecto del interpolador tras haber convertido los bits 0 en -1. Además, también muestra el efecto que tiene el filtro adaptado sobre la señal, del que se discutirá en la sección 5.3.2.

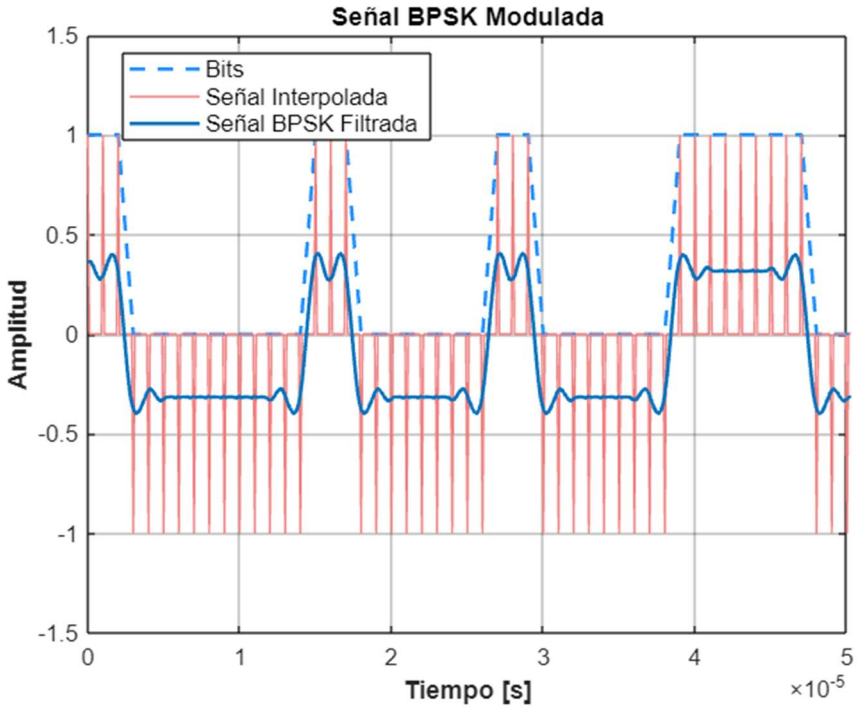

Figura 5.6. Señal BPSK modulada mediante un interpolador y un filtro adaptado.

5.3.2 FILTRO ADAPTADO DEL MODULADOR BPSK

Típicamente, el filtro adaptado recibe el nombre de conformador de pulsos en el transmisor. En general, se dice que un filtro adaptado [8] es aquel filtro cuya respuesta al impulso cumple que $h(t) = x(-t)$.

[4] El T_s que se ha empleado en esta figura es 10.

Utilizar filtros adaptados ofrece tres grandes ventajas. En primer lugar, limita el ancho de banda efectivo de la señal que se transmite. En segundo lugar, maximiza la SNR de la forma de onda recibida. Por último, reduce la interferencia entre símbolos [8] (en adelante, ISI, por sus siglas en inglés *Intersymbol Interference*). Esto se debe a que reducen las transiciones bruscas en la forma de onda de la señal, lo que hace que su espectro sea más eficiente.

Existen dos estrategias a la hora de diseñar filtros adaptados basados en filtros de coseno alzado (en adelante RC, por sus siglas en inglés *raised cosine*) [2]. Se puede utilizar o bien un filtro RC en el transmisor, o bien una solución basada en un filtro raíz cuadrada de coseno alzado (en adelante, SRRC, por sus siglas en inglés *Squared Root Raised Cosine*). Este último caso incluye un filtro SRRC en el transmisor y otro en el receptor. Esto se muestra en la Figura 5.7.

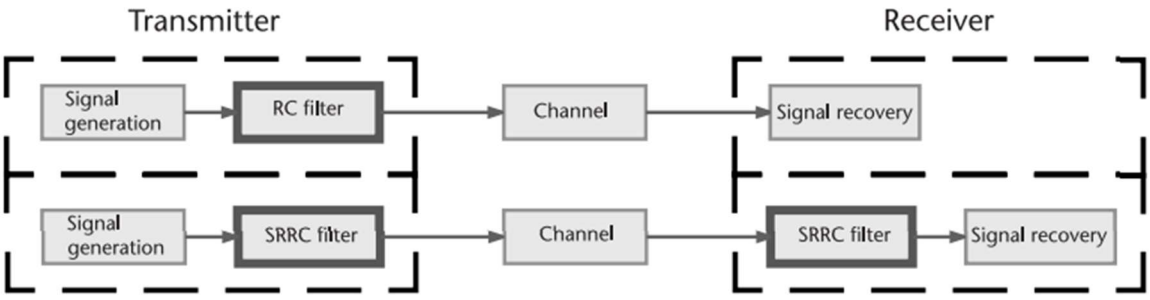

Figura 5.7. Diagrama de bloques de un transmisor y receptor usando filtros RC y SRRC.

En este proyecto, se decide emplear un filtro SRRC en el transmisor y su gemelo en el receptor. La respuesta al impulso de este filtro SRRC [2] es:

$$h(t) = \begin{cases} \dfrac{1}{\sqrt{T_s}}\left(1-\beta+4\dfrac{\beta}{\pi}\right), & t=0 \\ \dfrac{\beta}{\sqrt{2T_s}}\left[\left(1+\dfrac{2}{\pi}\right)sen\left(\dfrac{\pi}{4\beta}\right)+\left(1-\dfrac{2}{\pi}\right)\cos\left(\dfrac{\pi}{4\beta}\right)\right], & t=\pm\dfrac{T_s}{4\beta} \\ \dfrac{1}{\sqrt{T_s}}\dfrac{sen\left[\pi\frac{t}{T_s}(1-\beta)\right]+4\beta\frac{t}{T_s}\cos\left[\pi\frac{t}{T_s}(1+\beta)\right]}{\pi\frac{t}{T_s}\left[1-\left(4\beta\frac{t}{T_s}\right)^2\right]}, & en\ otro\ caso \end{cases}$$

donde T_s es el tiempo de símbolo que se ha definido en el interpolador previo al filtro SRRC de transmisión y $\beta \in [0,1]$ es el factor de caída o factor de *roll-off* [8]. Este parámetro controla la anchura del lóbulo principal del filtro SRRC. Cuanto menor es β, se mejora la eficiencia espectral a costa de empeorar la sincronización. La Figura 5.8 muestra cómo afecta a la respuesta al impulso del filtro SRRC diferentes valores de β tomando un T_s constante de 10 y una longitud del filtro de 60 coeficientes. Además, la Figura 5.6 también ilustra cómo afecta el filtro SRRC a la secuencia de bits modulada utilizando un $\beta = 0{,}35$.

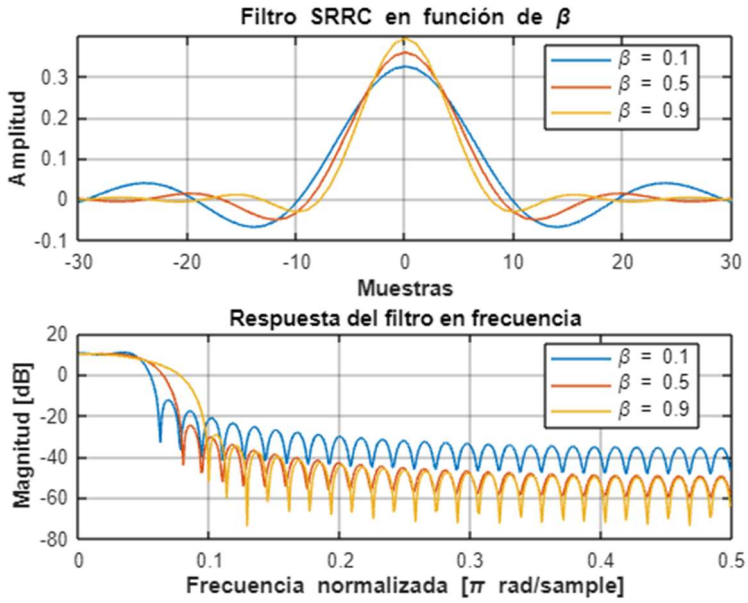

Figura 5.8. Efecto del factor de caída sobre la respuesta al impulso de un filtro SRRC.

El Código 8 del ANEXO II muestra la función que permite generar estos filtros en función de los parámetros β, T_s y la longitud del filtro en múltiplos de T_s. Del mismo modo, la implementación del modulador BPSK se muestra en el Código 9.

5.4 CANAL DE COMUNICACIONES

El canal mostrado en la Figura 5.1 se corresponde con el medio físico sobre el que se transmite la señal. En este proyecto, el medio físico es el cable coaxial o el par de antenas propias de la ADALM-Pluto. Todos los medios deterioran la señal o bien porque introducen

ruido en la señal o bien porque distorsionan la forma de onda. Son precisamente estos efectos los que motivan la construcción de un receptor digital que sea capaz de recuperar la señal que se ha transmitido originalmente.

Antes de utilizar la ADALM-Pluto como medio de transmisión, se van a simular los efectos típicos de un canal de comunicaciones introduciendo ruido blanco gaussiano aditivo, una desviación frecuencial y un retardo temporal.

5.4.1 RUIDO BLANCO GAUSSIANO ADITIVO

El ruido blanco gaussiano aditivo [8] (AWGN, por sus siglas en inglés *Additive White Gaussian Noise*) se trata de un ruido impulsivo cuyo espectro en frecuencia es plano. Por ello, afecta a todas las frecuencias por igual. Se utiliza para simular el ruido térmico generado por los componentes electrónicos del receptor. Matemáticamente, se modela como una señal aleatoria que sigue una distribución gaussiana y que se suma directamente a la señal recibida. Utilizando la función de MATLAB *pskmod* indicada en la sección 5.3 para realizar una modulación BPSK de una secuencia de bits, se ha generado la Figura 5.9. En ella se muestra el efecto del ruido blanco sobre una modulación BPSK para valores de $SNR = 10\ dB$ y $SNR = 30\ dB$.

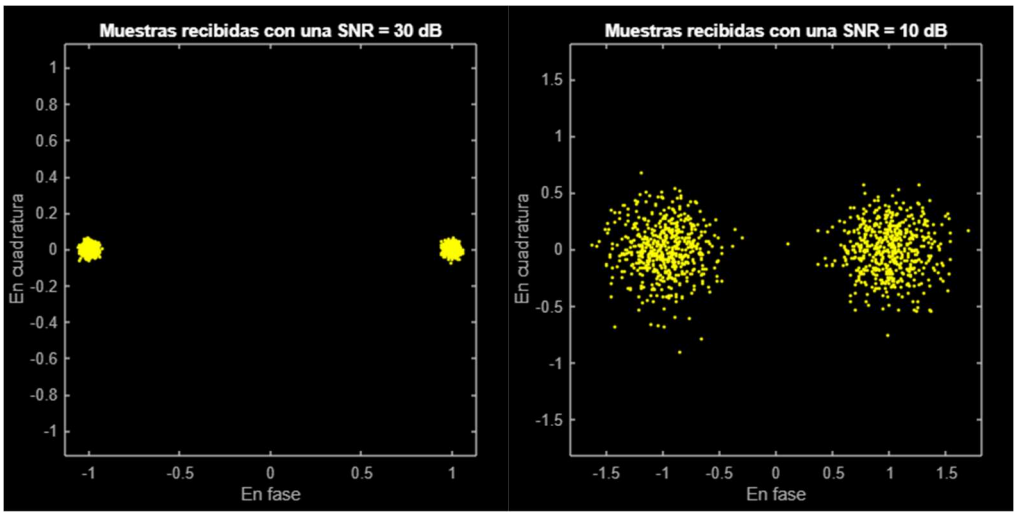

Figura 5.9. Efecto del ruido blanco gaussiano aditivo sobre una señal modulada en BPSK.

5.4.2 DESVIACIÓN FRECUENCIAL

Los osciladores locales del transmisor y del receptor no están perfectamente sincronizados. La diferencia entre estas frecuencias provoca una desviación frecuencial. Esto desplaza la portadora de la señal en frecuencia [2]. Visualmente, se puede observar cómo se genera una rotación de la constelación BPSK.

Para simular este efecto, se multiplica la señal recibida por una exponencial compleja $e^{j2\pi(f_{offset})t}$, donde f_{offset} representa la diferencia entre las frecuencias del oscilador del transmisor y del receptor. La Figura 5.10 muestra la rotación de la constelación BPSK provocada por la desviación frecuencial. El valor de desviación frecuencial escogido ha sido $f_{offset} = 0.2f_s$, donde $f_s = 1MHz$. Para esta figura[5], se ha utilizado la función *pskmod* de MATLAB.

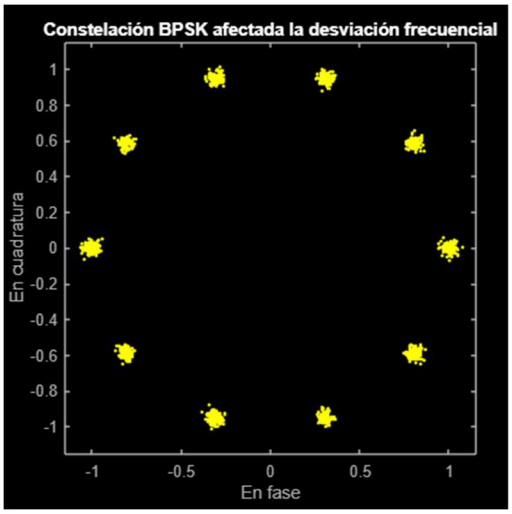

Figura 5.10. Efecto de la desviación frecuencial sobre la constelación de una modulación BPSK.

Asimismo, la Figura 5.11 muestra el efecto de la desviación frecuencial, pero esta vez sobre el espectro de la señal BPSK. Este espectro se ha representado con la frecuencia digital normalizada y utilizando la función *BPSKmodulator* del Código 9, ya que el filtro adaptado

[5] También se ha introducido un ruido AWGN con $SNR = 30dB$.

limita el ancho de banda del espectro de la señal BPSK. Por ello, se observa que la señal desplazada se centra en torno a los $0{,}2f_s$.

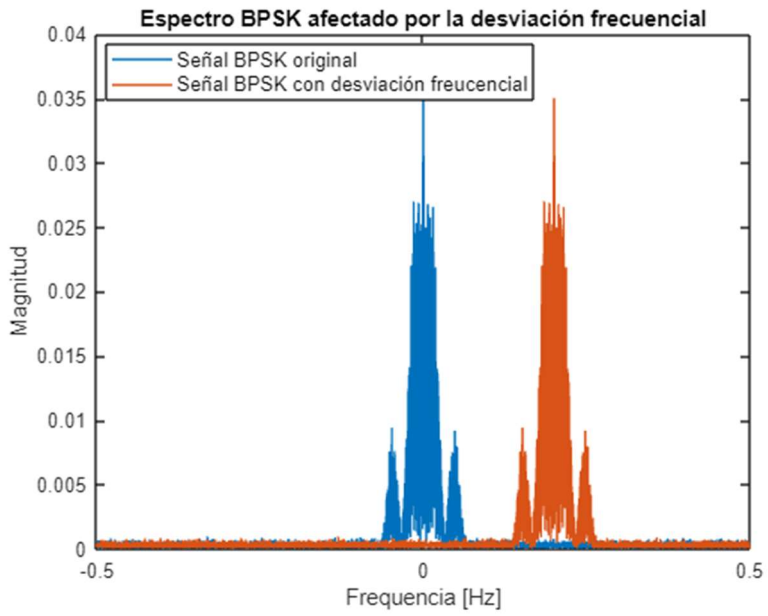

Figura 5.11. Efecto de la desviación frecuencial sobre el espectro de una señal modulada en BPSK.

5.4.3 RETARDO TEMPORAL

La llegada de la señal al receptor presenta un desfase temporal producido por la distancia física, por trayectorias múltiples que puede seguir la señal o incluso por el procesamiento interno del sistema. Este retardo afecta a la sincronización temporal, provocando que el receptor muestree los símbolos en momentos incorrectos. Es precisamente este problema el que motiva la introducción del sincronismo de símbolo, del que se habla en el Capítulo 7.

Si la señal transmitida es $s[n]$, ya que se trabaja en tiempo discreto, entonces la señal recibida $r[n]$ se puede expresar como:

$$r[n] = s[n - d]$$

donde $d \in \mathbb{Z}$[6] es el número de muestras de retardo [2]. En MATLAB, se puede simular este retardo mediante una convolución con una delta desplazada en el tiempo como:

$$r[n] = (s * h)[n]$$

con $h[n] = \delta[n - d]$. Esto es equivalente a insertar d ceros al principio de la señal. En la Figura 5.12 se muestra el efecto del retardo temporal en el dominio del tiempo. Para generar esta figura se ha tomado un retardo temporal, d, de 20 bits[7].

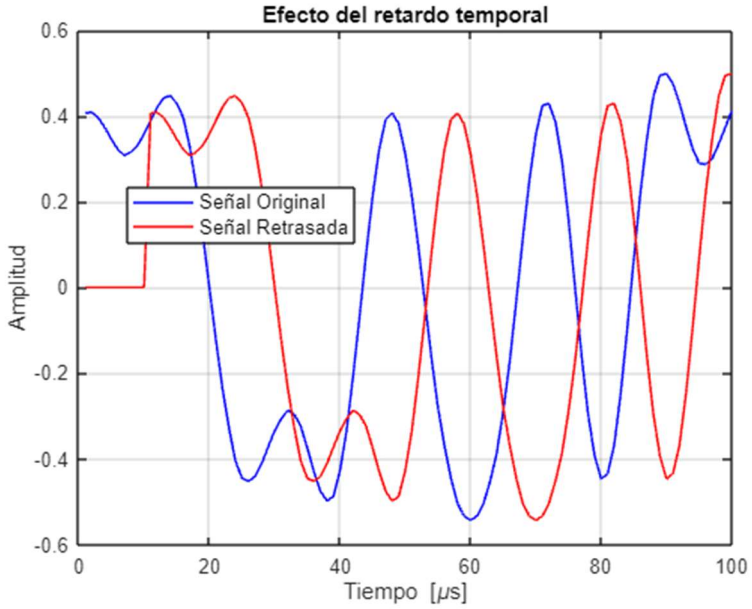

Figura 5.12. Efecto del retardo temporal sobre una modulación BPSK en el dominio del tiempo.

Desde otra perspectiva, la Figura 5.13 muestra el efecto del retardo temporal sobre una constelación BPSK. En ella, se observa que aparecen ciertas muestras en torno al 0. En concreto, se ven 2 muestras. El motivo es que el retardo temporal introducido es de 20 bits. Sin embargo, las constelaciones muestran símbolos, no bits. Por tanto, como el factor de interpolación es $Ts = 10$, los 20 bits se convierten en 2 muestras en torno al 0. Además,

[6] En este caso, se considera un retardo de muestras enteras. Para introducir un retardo de muestras fraccionario, sería necesario utilizar una función sinc [9].
[7] 20 bits equivalen a dos muestras ya que $T_s = 10$.

también se puede observar el efecto del ruido con una $SNR = 30dB$ y el efecto de la desviación frecuencial con un $f_{offset} = 0{,}2f_s$.

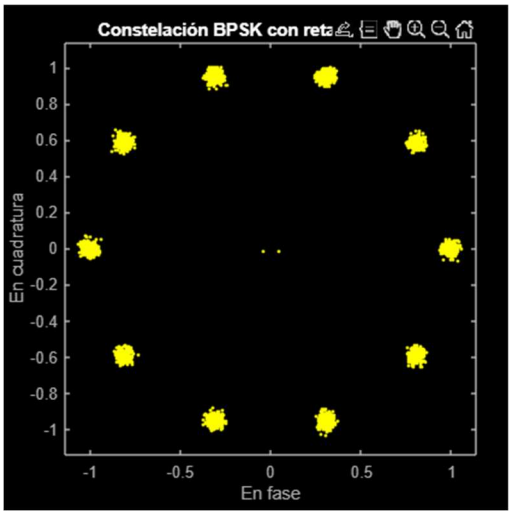

Figura 5.13. Constelación de una modulación BPSK afectada por el retardo temporal.

El Código 10 del ANEXO II permite simular el ruido impulsivo, la desviación frecuencial y el retardo temporal del canal de comunicaciones.

Capítulo 6. RECEPTOR DIGITAL

En este capítulo se aborda el desarrollo de un receptor digital basado en un demodulador BPSK. Este bloque se corresponde con el demodulador mostrado en la Figura 5.1. Una primera aproximación para implementar el receptor digital podría ser introducir un simple demodulador BPSK mediante la función *pskdemod* de la *Communications Toolbox* [4]. No obstante, esta función no incluye las funcionalidades de sincronismo necesarias para combatir los efectos del retardo temporal y de la desviación frecuencial.

Por ello, el receptor digital [2] que se desarrolla a lo largo de este capítulo se basa en el diagrama de bloques mostrado en la Figura 6.1. En este diagrama, los bloques de corrección gruesa y fina de portadora son los responsables de implementar el sincronismo de portadora, mientras que el bloque de sincronismo de símbolo corrige el retardo temporal. Además, el sincronismo de trama introduce unos bits de preámbulo que permiten detectar cuando comienza la trama.

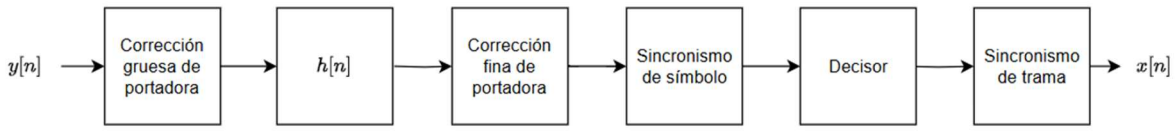

Figura 6.1. Diagrama de bloques de un receptor digital.

6.1 CORRECCIÓN GRUESA DE PORTADORA

Tal y como se comentó en la sección 5.4.2, la diferencia entre los osciladores locales del transmisor y del receptor provoca una desviación frecuencial. Típicamente, esta desviación frecuencial se proporciona en partes por millón (PPM). En concreto, los osciladores de la ADALM-Pluto provocan una desviación frecuencial de 25 PPM [2]. Esto significa que, si la portadora tiene una frecuencia de 1 GHz, la desviación frecuencial producida sería de 25 kHz, tal y como se expone a continuación:

$$\Delta f = 25 \cdot \frac{1\ GHz}{10^6} = 25\ kHz$$

Matemáticamente, se puede modelar la señal corrupta [2] por la desviación frecuencial como:

$$r(k) = s(k)e^{(j2\pi f_0 k T_s + \theta)} + n(k)$$

Ecuación 6.1. Expresión analítica de la señal afectada por la desviación frecuencial.

donde f_0 es la desviación frecuencial introducida, T_s el periodo del símbolo, θ la fase de la portadora y $n(k)$ un ruido blanco gaussiano. Una desviación frecuencial de 25 kHz es bastante grande. Por ello, el sincronismo de portadora se implementa en dos etapas: la corrección gruesa y la corrección fina.

Para la corrección gruesa de portadora, se utiliza el algoritmo basado en el diagrama de bloques [11] mostrado en la Figura 6.2.

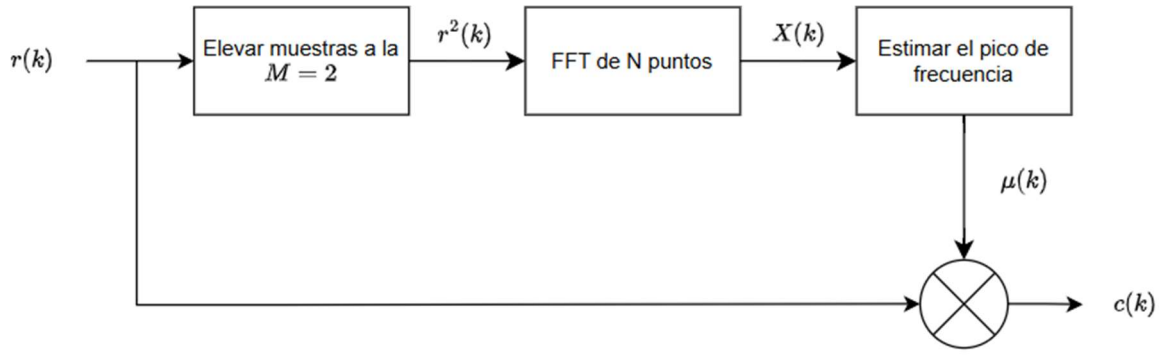

Figura 6.2. Algoritmo de corrección gruesa de portadora.

En primer lugar, el algoritmo toma la señal $r(k)$ de la Ecuación 6.1, de la que, por simplicidad, se ignora el ruido $n(k)$. A continuación, se elevan las muestras al orden de la modulación que, en el caso de la BPSK, es $M = 2$. De esta forma, se elimina el efecto de la modulación, dejando como resultado una única sinusoide con frecuencia f_0, que es la desviación frecuencial [9]. La expresión analítica de $r(k)$ queda como:

$$r^2(k) = s^2(k)e^{2\cdot(j2\pi f_0 kT_s+\theta)}$$

En este punto, $s^2(k)$ es una constante completamente real o imaginaria. Por ello, para el análisis de la desviación frecuencial, puede ser ignorado, quedando únicamente el tono de la señal f_0.

Seguidamente, se calcula la FFT de N puntos de la señal $r^2(k)$ y se estima el pico de frecuencia, f_0. Sin embargo, este aparece desplazado M veces la frecuencia f_0 esperada [9]. Por lo tanto, la estimación de f_0 obtenida debe ser dividida por M. La señal $c(k)$ corregida será el resultado de multiplicar $r(k)$ por una exponencial compleja de frecuencia f_0.

Es importante señalar que si se utiliza una frecuencia de muestreo f_s, este algoritmo solo es válido para estimar frecuencias comprendidas en el rango:

$$f_0 \in \left[-\frac{f_s}{2M}, \frac{f_s}{2M}\right]$$

Para este proyecto, tomando una $f_s = 1MHz$ y una modulación BPSK de $M = 2$, es posible estimar frecuencias entre -250 kHz y 250 kHz. Esto es notablemente superior a los 25 kHz que puede provocar el oscilador local de la ADALM-Pluto con una portadora de 1 GHz. Fuera de estos rangos, la estimación obtendría los picos de frecuencia propios de la banda imagen.

Existen dos modos de implementar este algoritmo en MATLAB. Por un lado, se puede utilizar el objeto *Coarse Frequency Compensator* [12]. El código que permite crear este objeto es el siguiente:

```
coarseFreqComp = comm.CoarseFrequencyCompensator( ...
  'Modulation', 'BPSK', ...
  'Algorithm','FFT-based', ...
  'SamplesPerSymbol', sps, ...
  'MaximumFrequencyOffset', rb/(2*M), ...
  'SampleRate', rb, ...
  'FrequencyResolution', 100);
```

Este código recibe como parámetros:

- **Tipo de modulación**: por defecto, utiliza una QAM. Permite otras modulaciones como la BPSK, QPSK, 8PSK…
- **Algoritmo**: por defecto utiliza el algoritmo de estimación basado en la FFT [13] descrito en esta sección. Permite otras técnicas que utilizan métodos de estimación basados en correlación [14].
- **Muestras por símbolo**: en este caso, $T_s = 10$.
- **Desviación frecuencial máxima**: viene dada por $f_s/2M$.
- **Frecuencia de muestreo**: en este caso $f_s = 1\ MHz$.
- **Resolución de la frecuencia**: el algoritmo es capaz de estimar la desviación frecuencial con una resolución de 100 Hz.

La segunda manera de implementar este algoritmo es mediante el Código 11 incluido en el ANEXO II. Empleando este segundo método, se puede observar en la Figura 6.3 el efecto que tiene la corrección gruesa de portadora sobre la constelación BPSK recibida. En concreto, se ha introducido un retardo temporal de 20 muestras, un ruido caracterizado por una SNR de 30 dB y una desviación frecuencial de 25 kHz. Además, se ha usado una frecuencia de muestreo de 1 MHz, un codificador de canal con un código Hamming $C(3,1)$, un filtro SRRC con una β de 0.35 y 60 coeficientes y un interpolador que introduce 10 muestras por símbolo[8].

Por otro lado, el espectro de la señal BPSK que resulta tras aplicar la corrección gruesa se muestra en la Figura 6.4. Aunque pueda parecer que este espectro está perfectamente centrado en torno al 0, esto no es así, tal y como se puede comprobar en la constelación corregida de la Figura 6.3, que no muestra las dos nubes de puntos compactas propias de la constelación BPSK. Realmente, la estimación de frecuencia que se obtiene con este

[8] Esta configuración se mantiene constante a lo largo del capítulo.

algoritmo es de 25,008 kHz, lo cual resulta en una corrección ligeramente desfasada. Para resolverlo, se emplea la corrección fina de frecuencia.

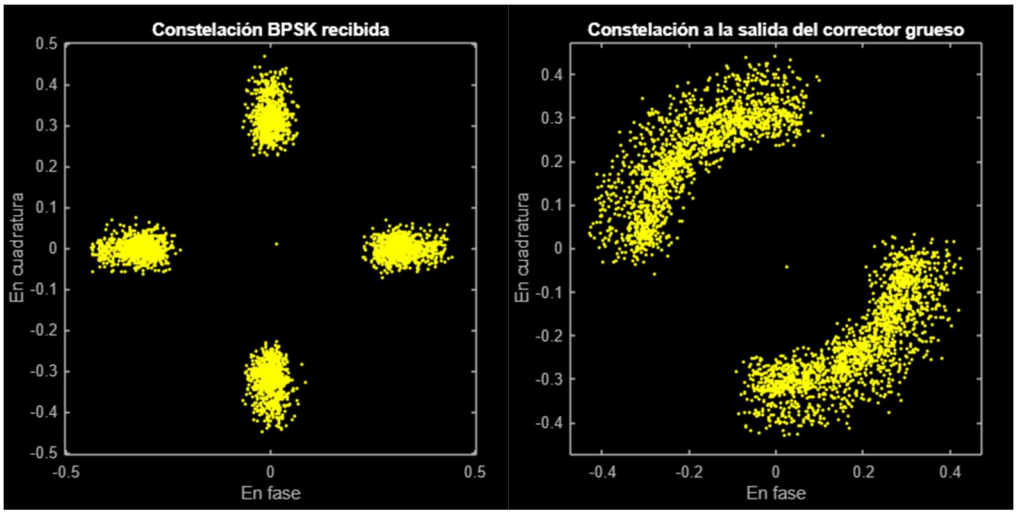

Figura 6.3. Constelación BPSK tras la corrección gruesa de portadora.

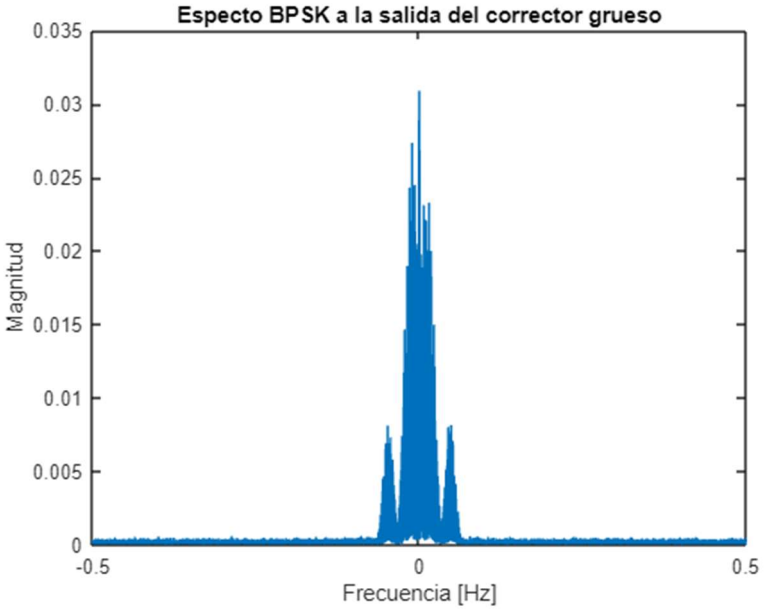

Figura 6.4. Espectro BPSK tras aplicar la corrección gruesa.

6.2 FILTRO ADAPTADO DEL RECEPTOR DIGITAL

Tras la corrección gruesa, es importante recordar que este proyecto utiliza dos filtros SRRC. Ya se introdujo en la sección 5.3.2 el concepto de filtro adaptado y la necesidad de introducirlo. Por ello, esta sección únicamente muestra el efecto que provoca el filtro SRRC sobre la constelación BPSK en el receptor digital. Esto se observa en la Figura 6.5. Comparándolo con la Figura 6.4, el cambio más notable se observa en los ejes. La Figura 6.4 muestra unos ejes acotados entre -0,5 y 0,5. Por otro lado, la Figura 6.5 presenta unos ejes acotados entre -1 y 1, que son los dos niveles originales en los que deben encontrarse las nubes de puntos propias de una modulación BPSK.

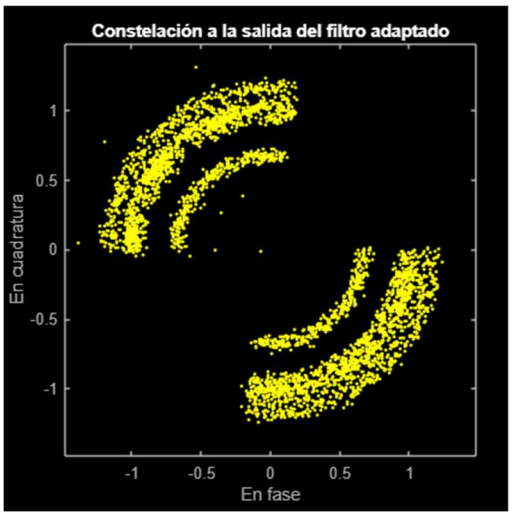

Figura 6.5. Constelación BPSK a la salida del filtro SRRC del receptor digital.

6.3 CORRECCIÓN FINA DE PORTADORA

Tal y como ya se ha adelantado en la anterior sección, la corrección gruesa corrige grandes desplazamientos del orden de kilohercios, y la sincronización fina es la responsable de corregir la desviación frecuencial residual. En esencia, esto dejaría perfectamente centrado en 0 el espectro de la señal mostrado en la Figura 6.4, produciendo una constelación BPSK estable.

Para realizar una sincronización precisa, es necesario utilizar un bucle cerrado realimentado basado en un PLL [2].

Este algoritmo utiliza un primer bloque, el detector de errores de fase (en adelante, PED, por sus siglas en inglés *Phase Error Detector*) en el que se mide el error de fase de la muestra recibida, generando una señal $e(n)$. A continuación, el siguiente bloque es un filtro de bucle que controla el tiempo de enganche del PLL y cómo de estable es la corrección. Estos típicamente se implementan como un filtro proporcional-integral [2] (en adelante, PI). Un filtro PI calcula la señal de control $f(n)$ mediante una ganancia proporcional al error actual y una ganancia integral que acumula los errores anteriores. El último bloque es el sintetizador digital discreto (en adelante DDS, por sus siglas en inglés *Direct Digital Synthesizer*). Este genera la señal de corrección $\phi(n)$ que se aplica a las muestras de entrada. Esta señal $\phi(n)$ se calcula a partir de los errores corregidos, $f(n)$. La Figura 6.6 ilustra el diagrama de bloques descrito.

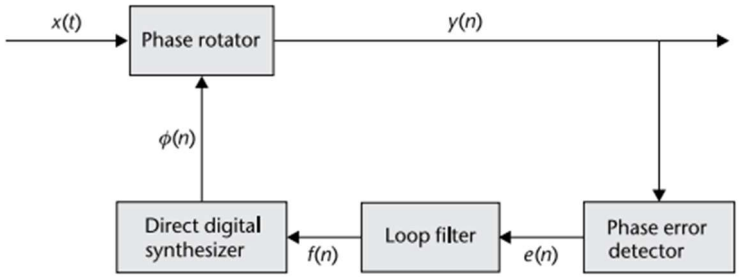

Figura 6.6. Diagrama de bloques del algoritmo de la corrección fina de frecuencia [2].

La implementación de la corrección fina de portadora que se plantea en este proyecto puede seguir dos alternativas. En primer lugar, puede utilizarse la técnica *Costas Loop* [9]. Esta técnica se aplica en modulaciones digitales como la BPSK y la QPSK. La segunda opción es emplear el objeto *Carrier Synchronizer* de MATLAB [15]. Este objeto sigue el diagrama de bloques ilustrado en la Figura 6.6 y el código que permite crearlo es:

```
fineSync = comm.CarrierSynchronizer( ...
        DampingFactor           = 0.707, ...
        NormalizedLoopBandwidth = 0.01, ...
        SamplesPerSymbol        = sps, ...
        Modulation              = 'BPSK');
```

Los parámetros indicados son:

- **Tipo de modulación**: por defecto, utiliza una QAM. Permite otras modulaciones como la BPSK, QPSK, 8PSK…
- **Muestras por símbolo**: en este caso, $T_s = 10$.
- **Factor de amortiguamiento**: para un coeficiente de amortiguamiento muy alto, el sistema es más lento, pero es más estable. Para un coeficiente pequeño, la respuesta es rápida, pero presenta oscilaciones. En general, un valor de 0,707 representa un compromiso óptimo.
- **Ancho de banda normalizado del lazo**: cuanto mayor es este parámetro, más rápido corrige los errores, pero mayor es su sensibilidad al ruido. Si toma un valor muy grande, responde lentamente, pero es más robusto ante señales aleatorias. Para este caso, se utiliza el valor por defecto de 0,01.

La constelación BPSK resultante tras la corrección fina de portadora se muestra en la Figura 6.7. Esto ilustra cómo el sincronismo de portadora implementado da como resultado una constelación BPSK estable. En este punto, los efectos de la desviación frecuencial han sido corregidos.

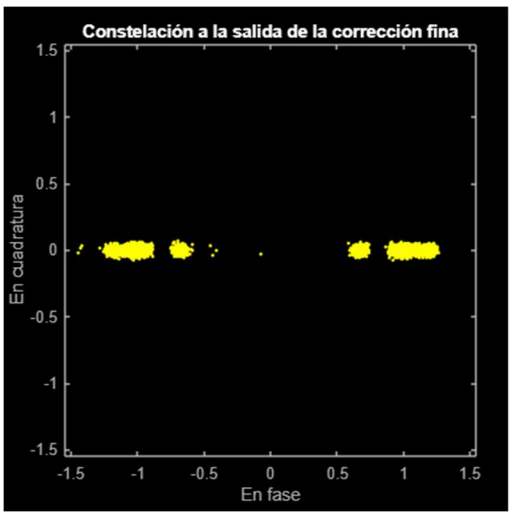

Figura 6.7. Constelación BPSK tras la corrección fina de portadora.

6.4 INTRODUCCIÓN AL SINCRONISMO DE SÍMBOLO

Al transmitir una señal, esta llega con un cambio de fase y con un retardo temporal que provoca que no se pueda muestrear al tiempo de símbolo T_s con el que se ha enviado originalmente. Esto puede derivar en dos problemas: un error de fase de temporización y un error de frecuencia de temporización. El error de fase de temporización [11] implica que las muestras se toman al ritmo adecuado, pero con una fase incorrecta. El error de frecuencia de temporización [11] provoca que las muestras no se tomen al ritmo adecuado, sino que se muestree más rápido o más lento. Esto provoca que la señal no se muestree en los puntos óptimos (Figura 6.8), sino en puntos en los que la muestra tomada es deficiente (Figura 6.9). De hecho, las 4 nubes de puntos que se observan en la Figura 6.7 son similares a los que se observan en la Figura 6.9 debido al retardo temporal que se ha introducido en la simulación y todavía no se ha corregido.

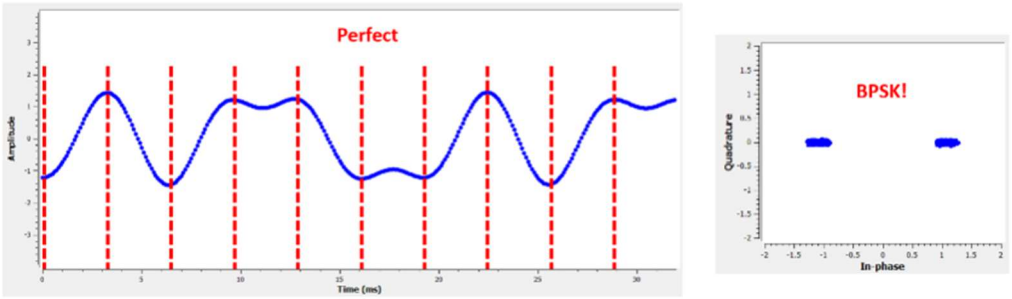

Figura 6.8. Resultado de un muestreo óptimo sin retardo temporal [9].

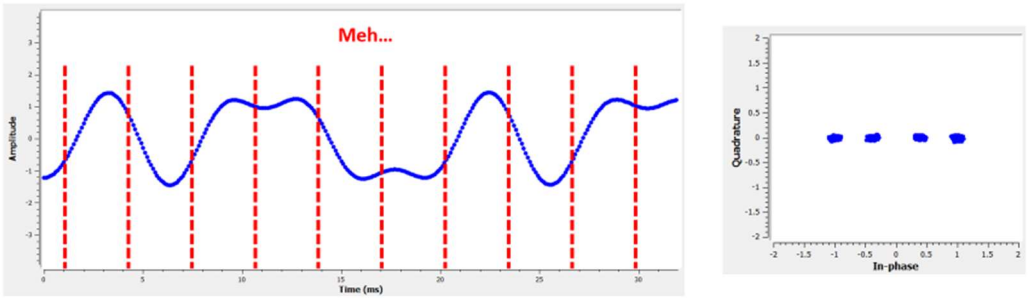

Figura 6.9. Resultado de un muestreo deficiente [9].

Matemáticamente, se puede modelar este error temporal como [2]:

$$r(t) = \sum_{n} x(n)h(t - \tau(t) - nT_s) + v(t)$$

donde $x(n)$ es el símbolo transmitido, h es el filtro SRRC conformador de pulsos, τ es el retardo temporal, T_s es el periodo de símbolo, n el índice de muestreo y $v(t)$ es un ruido blanco gaussiano.

Este bloque también actúa como un diezmador, de tal forma que la entrada del bloque son símbolos, mientras que su salida son los bits transmitidos originalmente preparados para ser introducidos al decisor. La mayoría de las técnicas de sincronismo de símbolo se basan en bucles de bloqueo de fase (PLL), tal y como se explica más adelante en el Capítulo 7.

Para su implementación en MATLAB se ha utilizado el objeto *Symbol Synchronizer* [16]. El código que permite crearlo se muestra a continuación:

```
symbolSync = comm.SymbolSynchronizer( ...
        SamplesPerSymbol        = sps, ...
        NormalizedLoopBandwidth = 0.01, ...
        DampingFactor           = 1.0, ...
        TimingErrorDetector     = "Zero-Crossing (decision-directed)", ...
        TimingErrorOutputPort   = true);
```

Los parámetros que aparecen en este objeto son:

- **Muestras por símbolo**: en este caso, $T_s = 10$.
- **Factor de amortiguamiento**: detallado en la sección 6.3. En este caso, el valor por defecto que toma es 1.
- **Ancho de banda normalizado del lazo**: detallado en la sección 6.3. El valor por defecto que toma también es 0,01.
- **Detector de error de temporización**: más detalles sobre este bloque y el algoritmo *Zero-Crossing* se explican en el Capítulo 7.
- **Puerto para visualizar el error de temporización**: permite almacenar en una variable cómo ha ido evolucionando el error de temporización.

La constelación BPSK que se obtiene tras aplicar el sincronismo de símbolo se ilustra en la Figura 6.10. Comparándola con la salida del bloque de la corrección fina (Figura 6.7) se observa que ya no existen cuatro nubes de puntos, sino solo dos nubes de puntos, ya que ahora se ha muestreado en los instantes óptimos y se ha corregido el error temporal.

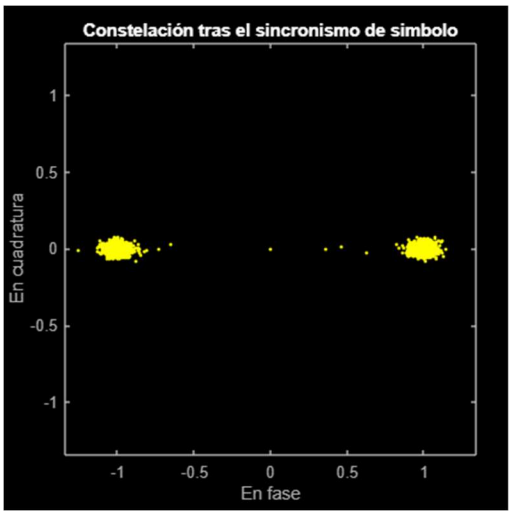

Figura 6.10. Constelación BPSK tras el bloque de sincronismo de símbolo.

6.5 DECISOR

El decisor es el bloque responsable de asignar un símbolo a cada vector de entrada procurando minimizar la probabilidad de error de símbolo [8]. Esta probabilidad de error se basa en una distribución normal, ya que la simulación ha utilizado un ruido gaussiano.

El error de símbolo en una modulación binaria como la BPSK ocurre cuando el decisor estima que el símbolo recibido es a_0 cuando, en realidad, se transmitió b_0. Esto se ilustra mejor en la Figura 6.11. El decisor determina cuál es el símbolo recibido en función de un umbral. Para este proyecto, el umbral decidido es el 0, ya que la modulación utiliza valores que oscilan entre -1 y 1. Gráficamente, observando la Figura 6.10 fijar este umbral es equivalente a que todos aquellos símbolos que se encuentran en el primer o cuarto cuadrante se transformen en el símbolo 1, mientras que todos aquellos que estén en el segundo o tercer cuadrante se conviertan en el símbolo -1. El resultado a la salida del decisor se observa en la

Figura 6.12. En ella, se muestra una constelación BPSK que ya no presenta ningún problema de desviación frecuencial, retardo temporal ni ruido blanco.

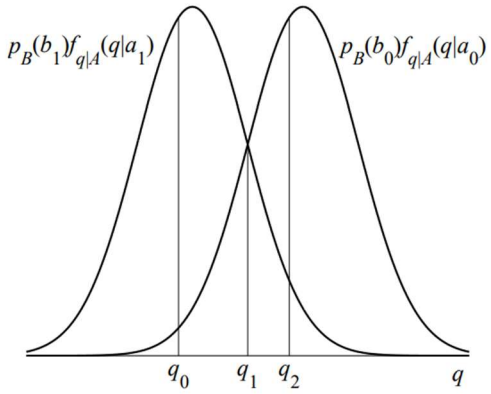

Figura 6.11. Probabilidad de error de símbolo en un receptor digital [8].

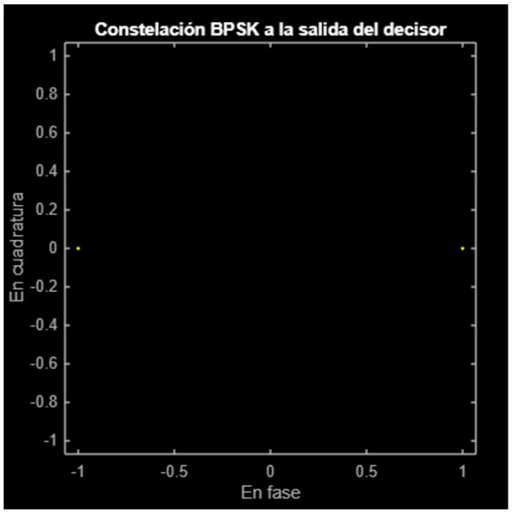

Figura 6.12. Constelación BPSK a la salida del decisor.

El decisor se puede implementar en MATLAB utilizando la función *pskdemod* [4], que además también demodula la señal. No obstante, para este proyecto, se ha utilizado el siguiente código:

```
% Decisor
phases = angle(samplesRx);
bitsRx = 2 * (phases <= pi/2 & phases > -pi/2) - 1;
```

Capítulo 7. SINCRONISMO DE SÍMBOLO

La sección 6.4 introdujo la problemática del error temporal y la necesidad de aplicar algoritmos de sincronismo de símbolo. Existen muchas alternativas para arreglar los problemas de sincronismo de símbolo en un receptor digital. Sin embargo, este proyecto se centra en estrategias basadas en bucles PLL como el explicado brevemente en la sección 6.3.

7.1 ESTRUCTURA DEL BLOQUE PLL

El diagrama de bloques de un PLL aplicado al sincronismo de símbolo se muestra en la Figura 7.1. En ella se observan 4 elementos. El detector de error de temporización (en adelante TED, por sus siglas en inglés *Timing Error Detector*) mide el error de temporización de la señal recibida y genera una señal de error $e(n)$. El filtro de bucle gobierna la dinámica del PLL mediante parámetros como el factor de amortiguamiento o el ancho de banda normalizado del lazo. El interpolador controlado genera la señal de corrección. Por último, el interpolador ajusta la señal de entrada aplicando la corrección del interpolador controlado. A continuación, se estudia en detalle cada uno de estos bloques.

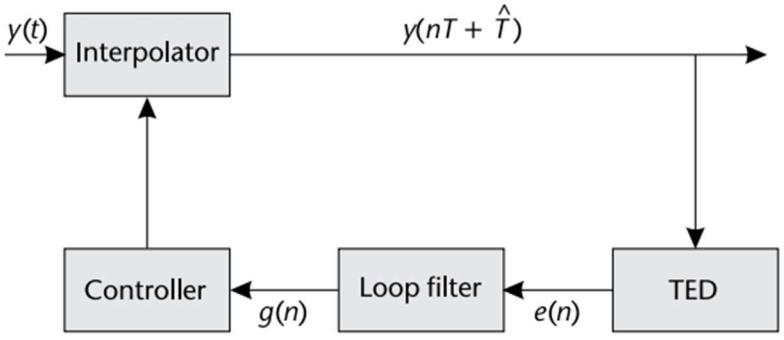

Figura 7.1. Diagrama de bloques de un PLL aplicado al sincronismo de símbolo [2].

7.1.1 DETECTOR DE ERROR DE TEMPORIZACIÓN

Para estudiar el funcionamiento estos bloques, se analiza el algoritmo de cruce por cero. Este algoritmo utiliza, al menos, dos muestras por símbolo[9]. El detector de error de temporización en este caso es [2]:

$$e(n) = Re\left(y\left(\left(n-\frac{1}{2}\right)T_s+\tau\right)\right)\left[sgn\left\{Re\left(y((n-1)T_s+\tau)\right)\right\} - sgn\{Re(y(nT_s+\tau))\}\right] + Im\left(y\left(\left(n-\frac{1}{2}\right)T_s+\tau\right)\right)\left[sgn\left\{Im\left(y((n-1)T_s+\tau)\right)\right\} - sgn\{Im(y(nT_s+\tau))\}\right]$$

Ecuación 7.1. Ecuación del detector de error de temporización para el algoritmo de cruce por cero [2].

Este algoritmo se basa en la idea de que, si entre dos muestras consecutivas hay un cambio de signo, el cruce por cero debió ocurrir en la muestra intermedia. Si el valor de esa muestra intermedia no es cero, entonces, se ha producido un desfase que se utiliza como señal de error para ajustar la temporización.

De este modo, el término $y\big((n-1/2)T_s+\tau\big)$ representa la muestra intermedia $y_{1/2}$, el término $y(nT_s+\tau)$ hace referencia al símbolo actual y_n y el término $y\big((n-1)T_s+\tau\big)$ indica la muestra anterior y_{n-1}. La Ecuación 7.1 se puede escribir, entonces, de una forma mucho más clara de este modo:

$$e(n) = Re\big(y_{1/2}\big)[sgn\{Re(y_{n-1})\} - sgn\{Re(y_n)\}] + Im\big(y_{1/2}\big)[sgn\{Im(y_{n-1})\} - sgn\{Im(y_n)\}]$$

y como el análisis para la parte real y para la parte imaginaria es la misma, tomando solo la parte real queda que:

$$Re\big(e(n)\big) = Re\big(y_{1/2}\big)[sgn\{Re(y_{n-1})\} - sgn\{Re(y_n)\}]$$

Cuando $sgn\{Re(y_{n-1})\} - sgn\{Re(y_n)\}$ sea 0, la señal no ha cruzado por 0. Entonces, la señal de error $e(n) = 0$. Si $sgn\{Re(y_{n-1})\} - sgn\{Re(y_n)\}$ es distinto de 0, la señal ha

[9] En este proyecto $T_s = 10$, por lo que este algoritmo es válido.

cambiado de signo. En ese caso, $y_{1/2}$ debería ser 0 si la temporización es correcta, en cuyo caso $e(n) = 0$. Pero si hay un error de temporización, $e(n) \neq 0$, por lo que $y_{1/2}$ indica cuánto error hay.

7.1.2 FILTRO DE BUCLE

El filtro de bucle que se propone en este proyecto se basa en el filtro PI introducido en la sección Corrección fina de portadora6.3. Su función de transferencia expresada en el dominio de Z es [2]:

$$F(z) = G_1 + \frac{G_2}{1 - z^{-1}}$$

donde G_1 es la ganancia proporcional que se aplica directamente al error $e(n)$ y G_2 es la ganancia integradora que se aplica a la suma acumulada de errores anteriores. Para su cálculo se utilizan el factor de amortiguamiento ζ y el ancho de banda de bucle B_{Loop} [2]:

$$G_1 = \frac{4\zeta\theta/\Delta}{T_s} \qquad\qquad G_2 = \frac{4\theta^2/\Delta}{T_s}$$

donde T_s es el número de muestras por símbolo, Δ es un factor que normaliza las ganancias del sistema y θ es la frecuencia normalizada del PLL. Se calculan como [2]:

$$\theta = \frac{B_{Loop}}{T_s(\zeta + 0{,}25/\zeta)}$$

$$\Delta = 1 + 2\zeta\theta + \theta^2$$

El factor de amortiguamiento ζ controla cómo responde el sistema ante la señal de error $e(n)$. Si $\zeta < 1$, el error se corrige en menos iteraciones, pero el PLL puede ser inestable, es decir, generar una señal $g(n)$ demasiado grande ante errores pequeños. Si $\zeta = 1$ se tiene un equilibrio entre rapidez y estabilidad que es óptimo. Si $\zeta > 1$ el sistema corrige los errores en más iteraciones, pero de forma estable.

El diagrama de bloques que permite representar este tipo de filtros de bucle se muestra en la Figura 7.2.

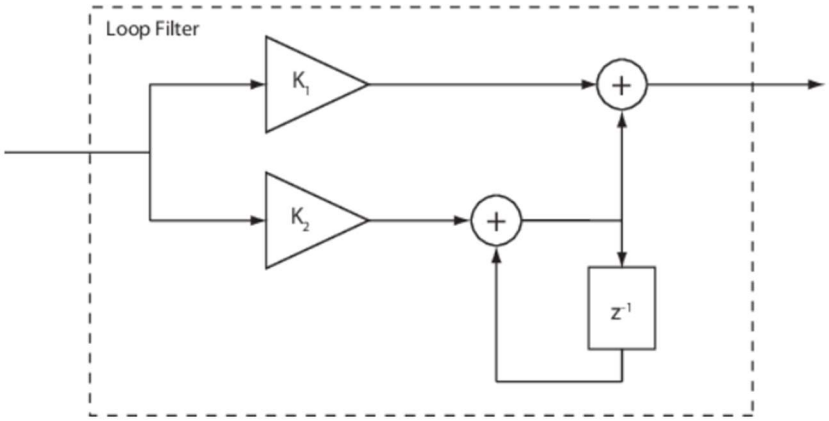

Figura 7.2. Diagrama de bloques del filtro de bucle del sincronismo de símbolo [16].

7.1.3 INTERPOLADOR CONTROLADO

Como el interpolador es el elemento encargado de tomar la muestra en el instante óptimo, el interpolador controlado debe enviarle la información necesaria para aplicar esta corrección. Este sistema se basa en un contador de módulo 1. La lógica de este algoritmo se muestra en la Figura 7.3.

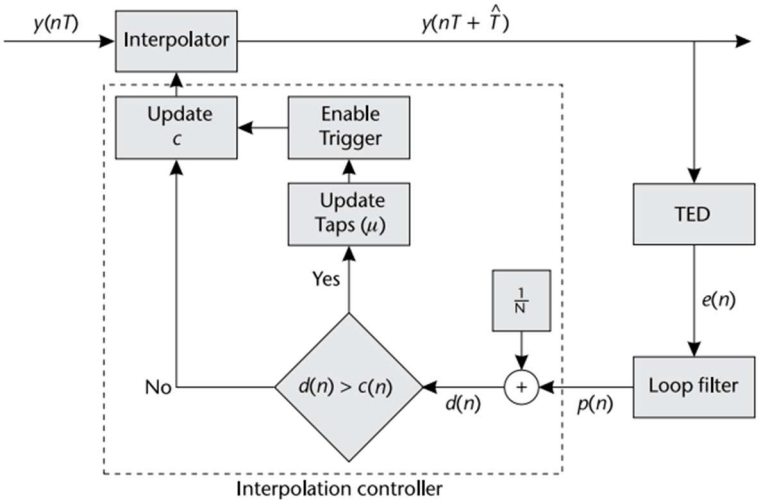

Figura 7.3. Diagrama de bloques del interpolador controlado del sincronismo de símbolo [2].

El propósito del algoritmo es generar una señal que indique al interpolador cuándo se debe tomar una muestra basándose en un contador. Este contador $c(n)$ comienza con un valor de 1 y se irá decrementando en $d(n)$ unidades. Por ejemplo, si se considera un valor de 10 muestras por símbolo, entonces, en una situación ideal, $d(n)$ toma el valor de

$$d(n) = \frac{1}{T_s} = \frac{1}{10} = 0{,}1$$

es decir, el contador se decrementa en 0,1 unidades. Tras 10 bits, el contador $c(n)$ llega a 0, y el decremento $d(n)$ es 0,1. Por ello, el interpolador controlado enviaría una señal de activación al interpolador para que tomara una muestra y $c(n)$ tomaría nuevamente el valor de 1.

No obstante, cuando el sistema detecta un error, el decremento que se aplica al contador depende de la señal recibida por el bucle de filtro $g(n)$:

$$d(n) = g(n) + \frac{1}{T_s}$$

La condición para tomar una muestra es que el decremento sea mayor que el contador, es decir, $d(n) > c(n)$. En ese caso, el interpolador controlado envía una señal de activación indicando que el factor de interpolación fraccionario que se debe aplicar es $\mu(n)$. Este valor se calcula como:

$$\mu(n) = \frac{d(n)}{c(n)}$$

y representa una proporción de cuánto quedaría para tomar una nueva muestra. De este modo, el interpolador retrasará la toma de la muestra según $\mu(n)$. Seguidamente, el contador se actualiza, siguiendo la operación módulo 1. Además, el TED solo calcula el error de temporización cuando se envía esta señal de activación.

7.1.4 INTERPOLADOR DEL SINCRONISMO DE SÍMBOLO

El interpolador [2] es el bloque final del sistema. En este proyecto, este interpolador se construye utilizando un filtro FIR adaptativo (en adelante, PPF, por sus siglas en inglés *Piecewise Polynomial Filter*).

El PPF proporciona un error de temporización con una precisión que depende del orden del filtro. En este caso, se utiliza una interpolación de segundo orden basada en el PPF mostrado en la Figura 7.4.

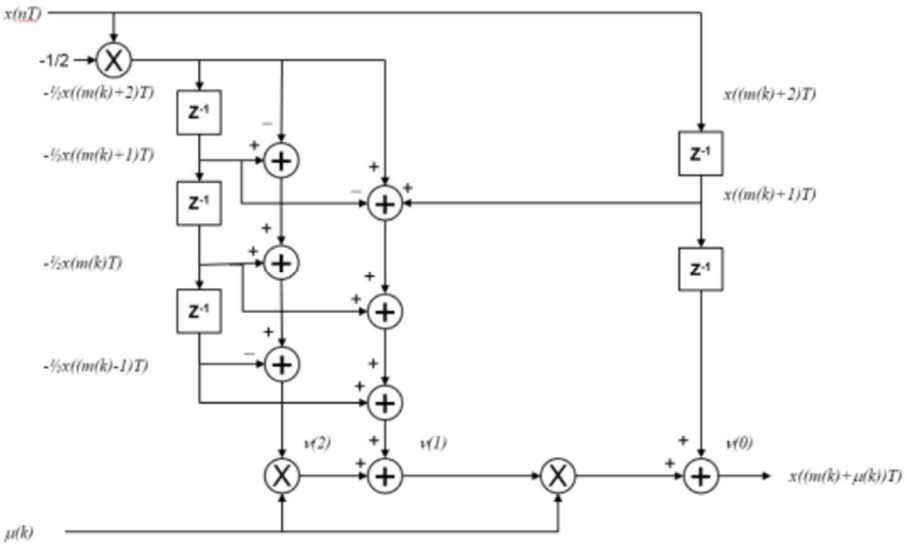

Figura 7.4. Diagrama de bloques del filtro PPF utilizado como interpolador del sincronismo de símbolo [16].

Los coeficientes h_k del PPF se calculan como:

$$h = [\alpha\mu(k)(\mu(k) - 1),$$

$$-\alpha\mu(k)^2 - (1 - \alpha)\mu(k) + 1,$$

$$-\alpha\mu(k)^2 + (1 + \alpha)\mu(k),$$

$$\alpha\mu(k)(\mu(k) - 1)]$$

donde $\alpha = 0{,}5$ y $\mu(k)$ es el retardo temporal obtenido por el interpolador controlado. Con este filtro, la salida general del interpolador es:

$$y(kT_s + \mu(k)T_s) = \sum_{n=1}^{2} h(n)y\big((k-n)T_s\big)$$

donde el retardo temporal estimado τ es:

$$\hat{\tau} \sim \mu(k)T_s$$

Gráficamente, se puede observar el efecto corrector del interpolador en la Figura 7.5.

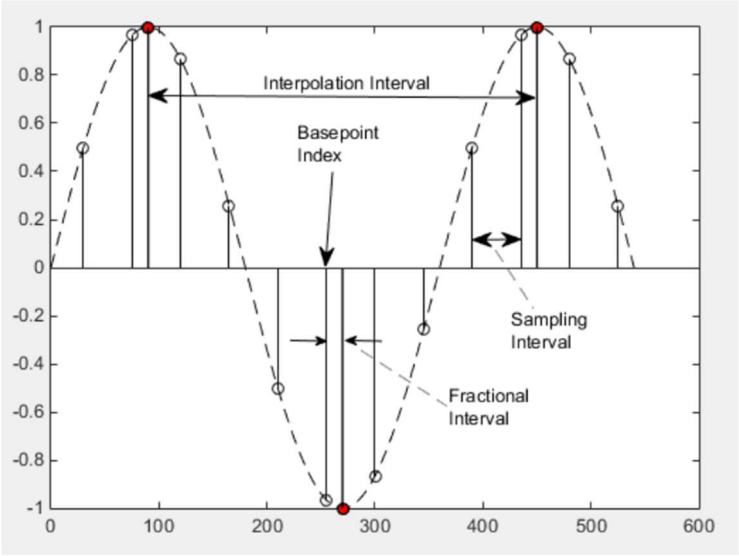

Figura 7.5. Efecto del interpolador del sincronismo de símbolo sobre la toma de muestras [16].

7.2 ALGORITMOS DE SINCRONISMO DE SÍMBOLO

En el diagrama de bloques mostrado en la Figura 7.1, se pueden utilizar diferentes TED según las características del sistema. Por ejemplo, el algoritmo de cruce por cero discutido en la sección 7.1 necesita al menos dos muestras por símbolo para funcionar adecuadamente. Además, presenta un peor rendimiento en escenarios con desviación frecuencial. Este es

precisamente el motivo por el que el receptor digital mostrado en la Figura 6.1 aplica primero la corrección gruesa y fina de portadora y, tras ello, el sincronismo de símbolo.

A continuación, se discuten alternativas al algoritmo de cruce por cero en función del modo de calcular la señal de error $e(n)$.

7.2.1 ALGORITMO DE GARDNER

La primera alternativa considerada es el algoritmo de Gardner [17]. Este es un algoritmo que no está dirigido por decisión. Se dice que un algoritmo de sincronismo de símbolo está dirigido por decisión cuando utiliza decisiones sobre los símbolos recibidos para estimar el error de temporización. Típicamente, esto se hace utilizando la función sgn en la expresión de la señal de error $e(n)$. El algoritmo de cruce por cero o el de Müller y Mueller (ver sección 7.2.2) es un claro ejemplo de esto. En cambio, el error de temporización en el caso de Gardner se calcula como:

$$e(n) = Re\left(y\left(\left(n-\frac{1}{2}\right)T_s+\tau\right)\right)\left[Re\left(y\big((n-1)T_s+\tau\big)\right)-Re\big(y(nT_s+\tau)\big)\right] + Im\left(y\big((n-1/2)T_s+\tau\big)\right)\left[Im\left(y\big((n-1)T_s+\tau\big)\right)-Im\big(y(nT_s+\tau)\big)\right]$$

la cual, siguiendo el desarrollo expuesto en la sección 7.1.1, se puede reescribir como:

$$Re(e(n)\,) = Re\big(y_{1/2}\big)\cdot[Re(y_{n-1})-Re(y_n)]$$

Este algoritmo se basa en la idea de la simetría de los pulsos digitales. Por ello, funciona especialmente bien con señales BPSK y QPSK. Si la muestra se toma en el centro del pulso, la forma del pulso antes y después de ese centro es la misma. Por lo tanto, al calcular la diferencia entre la muestra actual y la anterior, $Re(y_{n-1})-Re(y_n)$, y multiplicarlo por el valor de la muestra intermedia, $y_{1/2}$, el resultado será muy cercano a cero.

En cambio, si el muestreo está desfasado, este producto será distinto de cero, con signo positivo o negativo en función de si el muestreo está adelantado o atrasado respectivamente. Es por ello que este algoritmo utiliza también dos muestras por símbolo.

7.2.2 ALGORITMO DE MÜLLER Y MUELLER

El algoritmo de Müller y Mueller [18] (en adelante, MM) puede funcionar con una muestra por símbolo, es decir, no requiere de ningún proceso de interpolación. No obstante, aunque es técnicamente posible operar con $T_s = 1$, el rendimiento mejora aplicando una interpolación.

La señal de error $e(k)$ se calcula como:

$$e(k) = Re\left(y\left((k)T_s + \tau\right)\right) \cdot sgn\left\{Re\left(y\left((k-1)T_s + \tau\right)\right)\right\}$$

$$-Re\left(y\left((k-1)T_s + \tau\right)\right) \cdot sgn\left\{Re\left(y\left((k)T_s + \tau\right)\right)\right\}$$

$$+Im\left(y\left((k)T_s + \tau\right)\right) \cdot sgn\left\{Im\left(y\left((k-1)T_s + \tau\right)\right)\right\}$$

$$-Im\left(y\left((k-1)T_s + \tau\right)\right) \cdot sgn\left\{Re\left(y\left((k)T_s + \tau\right)\right)\right\}$$

Este algoritmo se basa en productos y restas cruzadas. Si las muestras se están tomando en el momento correcto, los valores de las muestras anterior y posterior deberían ser los mismos. No obstante, si existe un desfase temporal, esta simetría se rompe y se genera una señal de error proporcional.

Capítulo 8. OBTENCIÓN Y ANÁLISIS DE RESULTADOS CON LA ADALM-PLUTO

A lo largo de este proyecto se han presentado las simulaciones de un módem AM y de una modulación BPSK con un receptor digital. De este modo, se puede entender la diferencia entre los sistemas analógicos y digitales y la importancia del sincronismo de portadora y de símbolo. Sin embargo, los resultados finales que se presentan en este proyecto son los que se obtienen a través de transmisiones reales con la ADALM-Pluto.

8.1 MÓDEM AM CON LA ADALM-PLUTO

Para este primer experimento se va a utilizar el módem AM desarrollado en el Capítulo 4. No obstante, ahora se transmite una señal[10] de audio en lugar de un coseno. La Figura 8.1 muestra el espectro de la señal de audio, mientras que en la Figura 8.2 se ilustra esta señal de audio modulada en AM según el esquema descrito en la sección 4.3 con una portadora de 5 kHz.

Esta señal de audio es transmitida a continuación utilizando la ADALM-Pluto. Para su configuración, se ha utilizado una frecuencia de muestreo de 500 kHz ya que, para una señal de audio muestreada a 44,1 kHz, utilizar frecuencias de muestreo del orden de megahercios solo añade carga computacional y reduce la SNR efectiva. La frecuencia de portadora usada en la SDR es de 1 GHz y las ganancias en transmisión y recepción son -10 dB y 10 dB respectivamente.

[10] La señal de audio ha sido muestreada a 44,1 kHz.

El espectro recibido tras la transmisión con la ADALM-Pluto se observa en la Figura 8.3. Asimismo, tras ser demodulada siguiendo el esquema propuesto en la sección 4.4, el espectro de la señal queda tal y como se muestra en la Figura 8.4.

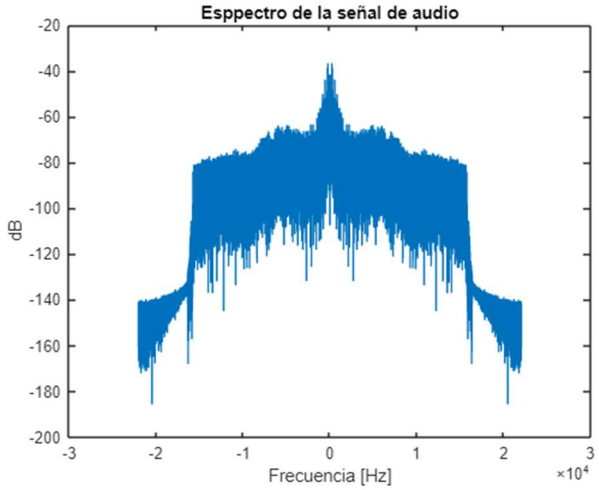

Figura 8.1. Espectro de la señal de audio muestreada a 44,1 kHz.

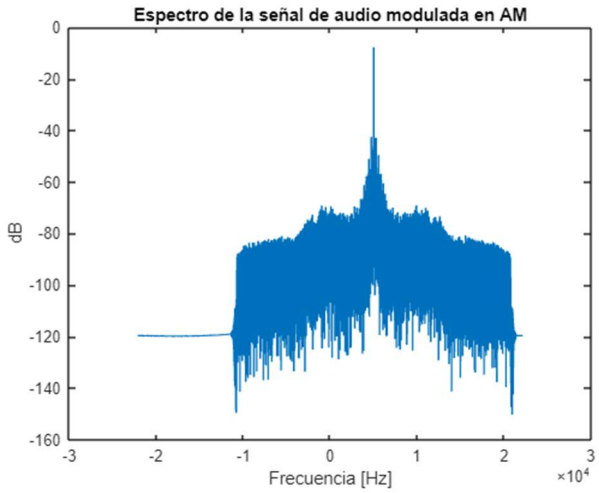

Figura 8.2. Espectro de la señal de audio modulada en AM con una portadora a 5 kHz.

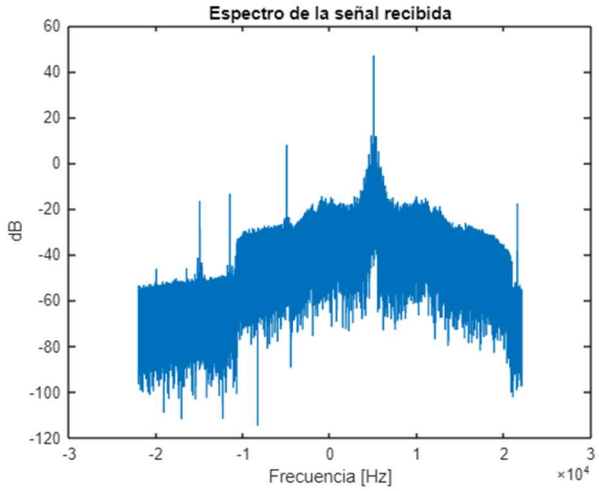

Figura 8.3. Espectro de la señal de audio AM recibida usando la ADALM-Pluto.

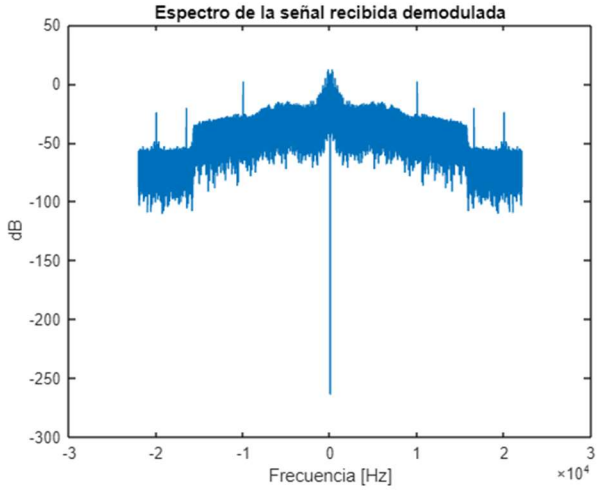

Figura 8.4. Espectro de la señal de audio recibida usando la ADALM-Pluto tras ser demodulada.

8.1.1 REORDENACIÓN DE LOS DATOS RECIBIDOS

Escuchando el audio recibido, se puede comprobar que este no comienza necesariamente en el mismo punto en el que comienza el audio transmitido. Esto es, el principio del audio recibido es el final del audio transmitido, por lo que la señal está desordenada.

Para corregirlo, se utiliza un preámbulo basado en los códigos de Barker. A través de la correlación cruzada entre la señal recibida y el preámbulo, es posible detectar el comienzo

de la señal mediante el valor máximo de correlación. Esta es una técnica de sincronismo de símbolo de la que se habla más en detalle en la sección 8.2.2.

El valor máximo de correlación se ilustra en la Figura 8.5. De ella se obtiene que el inicio real del audio recibido comienza en la muestra número 285.262 de las 500.000 enviadas.

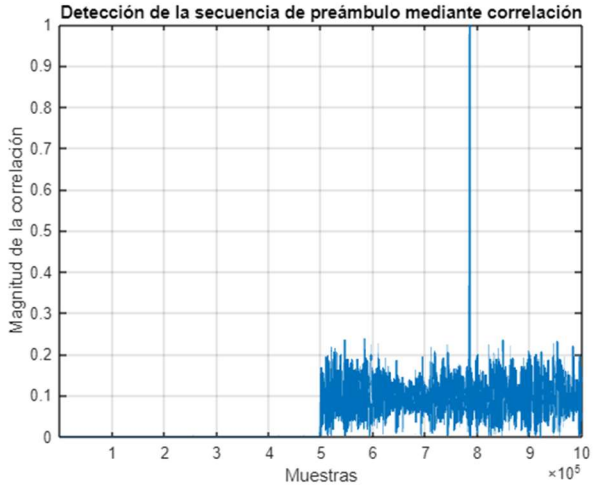

Figura 8.5. Función de autocorrelación para la detección del inicio del audio modulado en AM.

Para reordenar la trama, se toman las 214.738 muestras que hay entre el pico de correlación y el final de la señal y se ponen al principio de la trama, eliminando el preámbulo. De esta forma las 285.262 muestras restantes se ubican al final. De este modo, el orden de la señal recibida y de la transmitida es el mismo. El código empleado en MATLAB para este propósito ha sido el siguiente:

```
% Cálculo de la correlación
correlation = abs(xcorr(dataRx, preamble));
correlation = correlation/max(abs(correlation));
[~, idx] = max(correlation);

% Reordenación de la trama
startIndex = idx - length(dataRx)
songBegin = dataRx(startIndex + length(preamble) + 1: end);
endSong = dataRx(1 : startIndex);
dataRx = [songBegin, endSong]
```

8.1.2 TRATAMIENTO DE ESPURIOS. PARAMETRIZACIÓN DE GANANCIAS

Los espurios son componentes espectrales no deseados que aparecen fuera de las bandas útiles de la señal transmitida. Estos pueden aparecer por diferentes motivos. Por ejemplo, si las ganancias en transmisión o en la recepción son muy elevadas, esto puede saturar los componentes electrónicos de la ADALM-Pluto y generar distorsiones. Si hay señales cercanas a la frecuencia de 1 GHz utilizada para la transmisión, estas también pueden generar emisiones espurias. Además, desajustes en la frecuencia de los osciladores locales del transmisor y del receptor (como los expuestos en la sección 5.4.2) también pueden introducir espurios.

En la Figura 8.4 se observan claramente varios picos espurios en torno a los 10 kHz, 15.5 kHz y 20 kHz. Esto degrada la calidad de la señal en la recepción y motiva el uso de técnicas para eliminar estos espurios.

En primer lugar, es importante escoger ganancias adecuadas para no saturar los componentes electrónicos de la ADALM-Pluto. La ganancia de transmisión se puede configurar entre los -89,75 dB y 0 dB, mientras que la ganancia de recepción oscila entre los -10 dB y los 73 dB. Para determinar qué ganancia de transmisión y recepción escoger, se realizan diferentes transmisiones con diferentes valores de ganancias y se evalúa la relación señal a ruido (en adelante, SNR, por sus siglas en inglés *Signal to Noise Ratio*).

La SNR [7] es la forma habitual de cuantificar que la forma de onda de la señal recibida se parece lo máximo posible a la transmitida. Para ello mide el cociente entre la potencia de ruido (que es el resultado de la diferencia de la señal transmitida y recibida) y la potencia de la señal transmitida:

$$SNR_{dB} = 10 \cdot \log_{10}\left(\frac{P_{Señal}}{P_{ruido}}\right)$$

Una forma de calcular la potencia de la señal es utilizando el valor cuadrático medio, mientras que para la potencia del ruido se utiliza la varianza. El cálculo de la potencia de ruido se basa en la suposición de que:

$$n(t) = r(t) - s(t)$$

donde $n(t)$ representa el ruido, $r(t)$ es la señal recibida y $s(t)$ es la señal original. Por ello, para que los resultados sean válidos, ha sido clave la reordenación de la trama recibida realizada en el punto anterior y normalizar ambas señales.

Inicialmente, se fija una ganancia de transmisión de -10 dB para no saturar la salida del canal y se varía la ganancia de recepción entre 10 dB y 40 dB, ya que valores más altos podrían amplificar demasiado el ruido y los espurios. Además, se realizan 20 transmisiones por configuración para reducir el impacto del ruido aleatorio y variabilidad inherente a los componentes analógicos de la ADALM-Pluto. Los resultados medios obtenidos en estas 20 transmisiones han sido:

Ganancia en la recepción (dB)	Potencia de ruido (mW)	SNR (dB)
10	2.77	10.06
20	2.78	10.05
30	2.97	9.87
40	3.17	9.49
50	3.19	9.46

Tabla 2. Resultados de potencia de ruido y SNR tras realizar 20 transmisiones con la ADALM-Pluto fijando la ganancia en transmisión a -10 dB.

Por otro lado, si se fija una ganancia de transmisión de -5 dB, superior a los -10 dB anteriores, los resultados medios obtenidos en estas 20 nuevas transmisiones son:

Ganancia en la recepción (dB)	Potencia de ruido (mW)	SNR (dB)
10	3.92	8.60
20	3.92	8.58
30	3.81	8.68
40	3.46	9.12
50	3.37	9.21

Tabla 3. Resultados de potencia de ruido y SNR tras realizar 20 transmisiones con la ADALM-Pluto fijando la ganancia en transmisión a -5 dB.

La conclusión obtenida es clara: a medida que se aumenta la ganancia de transmisión, se distorsiona la señal a la salida de la ADALM-Pluto, mientras que aumentar la ganancia en la recepción amplifica el ruido recibido. Por ello, la configuración que se utiliza definitivamente es una ganancia de -10 dB en el transmisor y una ganancia de 10 dB en el receptor, según los resultados mostrados.

En MATLAB, se puede calcular la SNR haciendo uso de la función *snr*, que se basa en el siguiente código:

```
signalPower = rms(messageSignal)^2;
noisePower = var(demodulatedSignal - messageSignal);
SNR_dB = 10 * log10(signalPower / noisePower)
```

8.1.3 TRATAMIENTO DE ESPURIOS CON FILTRADO PASO BAJO

Configurar adecuadamente la ganancia en transmisión y recepción mejora la SNR, pero no elimina los espurios. Por lo tanto, en este proyecto, se plantea el uso de un filtro paso bajo para eliminar estos picos de frecuencia no deseados.

El filtro paso bajo utiliza una frecuencia de corte de 8 kHz. Esto se debe a que la mayor parte de la energía de la señal de audio se encuentra concentrada por debajo de esta frecuencia tal y como se puede comprobar en la Figura 8.1. Además, los espurios aparecen en torno a los 10 kHz (Figura 8.4). Por ello, este filtro a partir de los 10 kHz atenuará fuertemente la señal. En concreto, la atenuación aplicada es de 60 dB. Por otro lado, el rizado de la banda de paso se configura con un valor de 0,01, que equivale a $\pm 0{,}086$ dB. Así se garantiza que la señal no se distorsione en su rango útil.

El código utilizado en MATLAB para implementar este filtro paso bajo es el siguiente:

```
lpFilt = designfilt('lowpassfir', ...
    'PassbandFrequency', 8000, ...
    'StopbandFrequency', 10000, ...
    'SampleRate', fs, ...
    'PassbandRipple', 0.01, ...
    'StopbandAttenuation', 60);
```

Tras aplicar este filtrado, los espurios quedan completamente eliminados, tal y como se muestra en la Figura 8.6, donde, además, se comprueba que la atenuación aplicada por encima de los 10 kHz es de -60 dB.

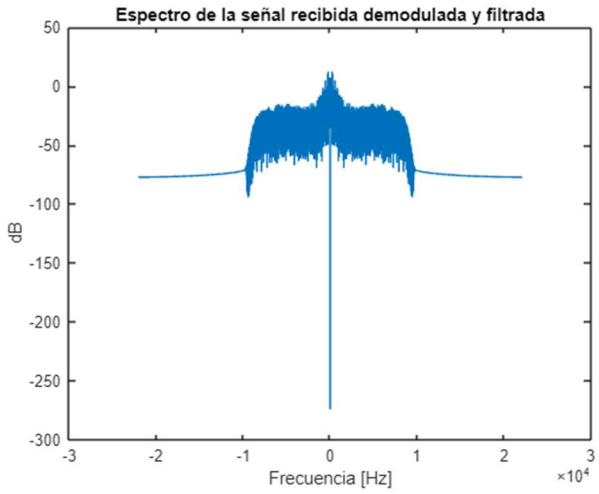

Figura 8.6. Espectro de la señal de audio recibida usando la ADALM-Pluto tras ser demodulada y filtrada.

8.2 RECEPTOR DIGITAL CON LA ADALM-PLUTO

Para el segundo experimento, se utiliza el modelo transmisor BPSK desarrollado en el Capítulo 5. Asimismo, se emplea el receptor digital expuesto en el Capítulo 6. La frecuencia de muestreo utilizada para la generación de bits y para la transmisión con la ADALM-Pluto es de $f_s = 1MHz$. Se utiliza el codificador de fuente y el codificador de canal con el código Hamming $C(3,1)$. El modulador BPSK utiliza 10 muestras por símbolo y el filtro adaptado utiliza un valor de $\beta = 0{,}35$ con 60 coeficientes.

Además, para evaluar si el sincronismo de frecuencia es capaz de corregir la desviación frecuencial, se utiliza una portadora en la ADALM-Pluto de $f_c = 3\ GHz$ que permite ver esta desviación. La constelación que se recibe utilizando el cable coaxial de la ADALM-Pluto como canal de transmisión se muestra en la Figura 8.7. En ella, se observa claramente el efecto de la desviación frecuencial ya que la constelación ha rotado y el efecto del error de temporización ya que hay nubes de puntos cercanas al cero.

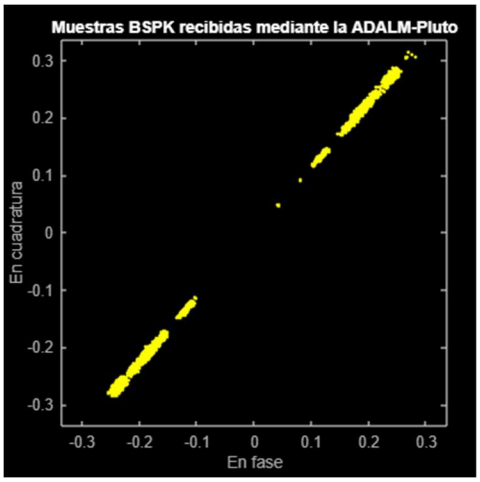

Figura 8.7. Constelación BPSK de los símbolos recibidos utilizando la ADALM-Pluto.

Como el formato de la transmisión es digital, se utiliza la BER para evaluar el número de errores que se han producido. El cálculo de la BER se basa en la operación lógica XOR y para medirla se utiliza el Código 12 del ANEXO II.

8.2.1 SINCRONISMO DE FRECUENCIA Y DE SÍMBOLO CON LA ADALM-PLUTO

Siguiendo el esquema de la Figura 6.1, se utiliza la corrección gruesa y fina de portadora para implementar el sincronismo de frecuencia[11]. La Figura 8.8 muestra el resultado de la corrección gruesa. La estimación de la desviación frecuencial es realmente pequeña, en torno a los 7,5 Hz. Esto motiva el uso de la corrección fina para corregir esta desviación frecuencial.

Tras utilizar el filtro SRRC, se utiliza la corrección fina de portadora, utilizando como factor de amortiguamiento 0,707 y como ancho de banda de bucle normalizado 0,01. El resultado se observa en la Figura 8.9. En ella, se ha conseguido reducir significativamente el efecto de la desviación frecuencial con respecto a la salida de la corrección gruesa. Ya se observan las dos nubes de puntos propias de la constelación BPSK, no obstante, se puede comprobar

[11] Se utiliza el filtro SRRC entre el corrector grueso y el fino.

como todavía quedan algunas muestras afectadas por la desviación frecuencial y el retardo temporal.

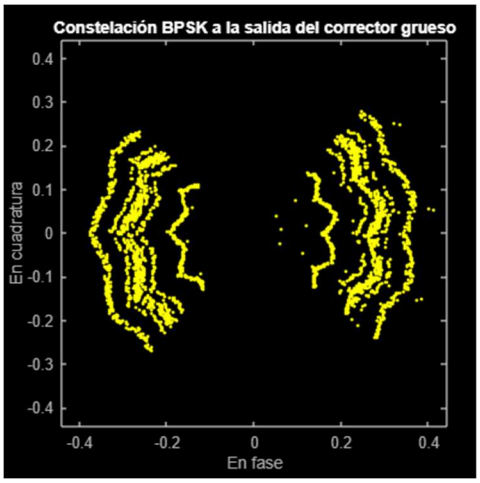

Figura 8.8. Corrección gruesa de portadora utilizando como canal la ADALM-Pluto.

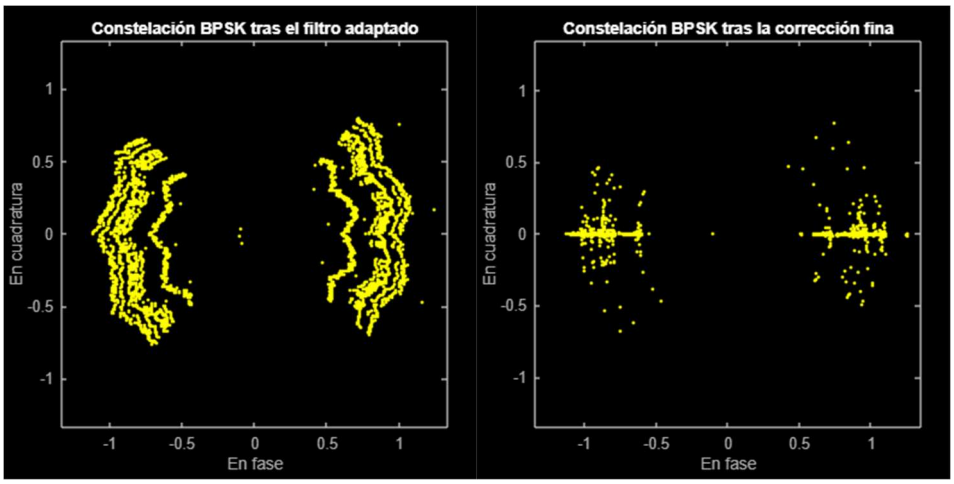

Figura 8.9. Filtro SRRC y corrección fina de portadora utilizando como canal la ADALM-Pluto.

Cuantitativamente, se puede determinar el rendimiento de este bloque utilizando el error de fase instantáneo entre la fase de las muestras corregidas y las referencias ideales, que en este caso son 0 y π. A continuación, se utiliza el error cuadrático medio (en adelante, MSE, por sus siglas en inglés *Mean Squared* Error) para comprobar cuánto varía el error de fase. Cuanto menor sea este estadístico mejor ha funcionado el algoritmo de sincronismo de símbolo. Para calcular el MSE, se utiliza la expresión:

$$MSE = \frac{1}{N}\sum_{k=1}^{N}\left(\phi(k)\right)^2$$

donde $\phi(k)$ representa el error de fase en la muestra k. La evolución de este error se muestra en la Figura 8.10 y el MSE obtenido es de $5{,}52 \cdot 10^{-3}\ rad^2$. En esta figura se observa que el error converge rápidamente a cero, lo cual indica que el PLL funciona adecuadamente. No obstante, los picos esporádicos se deben a ruido o símbolos mal alineados. En general, el MSE bajo indica que, en promedio, la fase está bien corregida, y no existen grandes desviaciones sistemáticas.

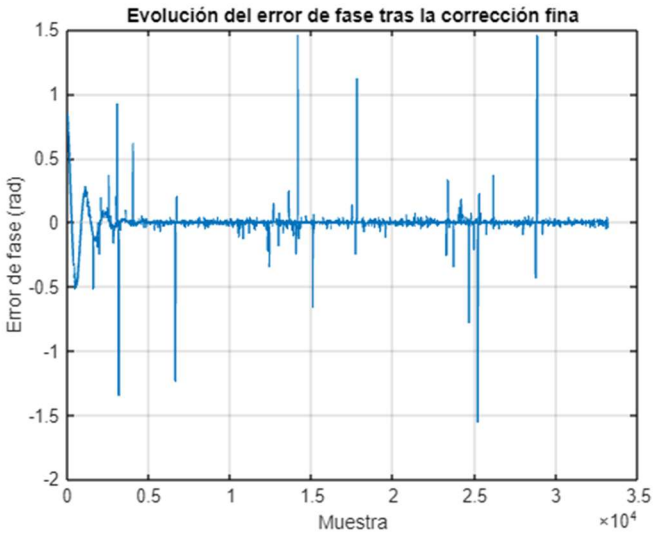

Figura 8.10. Evolución del error de fase en la corrección fina tras la transmisión con la ADALM-Pluto.

El código de MATLAB para el cálculo del error de fase y del MSE es:

```
% Normalizar los símbolos y calcular la fase residual
samplesRxNorm = samplesRx ./ abs(samplesRx);
residualPhase = unwrap(angle(samplesRxNorm));

% En BPSK, los ángulos ideales son 0 (en el cuadrante 1,4) y π (en 2,3).
expectedPhases = pi * (real(samplesRxNorm) < 0);
phaseError = residualPhase - expectedPhases;

% Casos de ambigüedad de errores de +2π y cálculo del MSE
phaseError = mod(phaseError + pi, 2*pi) - pi;
mseFinePhaseError = mean(phaseError.^2);
```

Seguidamente, se añade el bloque de sincronismo de símbolo. En él, se utiliza el algoritmo de cruce por cero. El resultado obtenido se muestra en la Figura 8.11. Cualitativamente, se puede comprobar que este algoritmo de sincronismo de símbolo mejora ligeramente el procesado con respecto a la corrección fina, haciendo las nubes de puntos en torno al -1 y 1 aún más compactas. Cuantitativamente, se puede utilizar de nuevo el MSE.

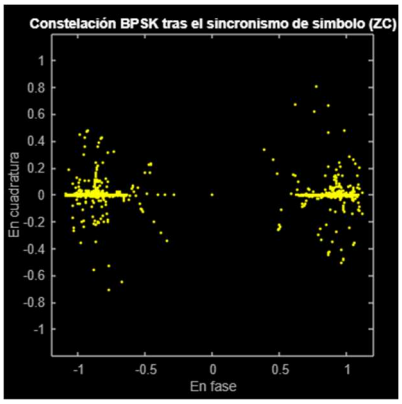

Figura 8.11. Sincronismo de símbolo con el algoritmo de cruce por cero en una transmisión con la ADALM-Pluto.

El MSE obtenido es 0,316. Este valor es alto ya que este algoritmo no funciona bien cuando hay problemas en presencia de desviación frecuencial, como ya se adelantó en el Capítulo 7. La evolución del error de temporización se muestra en la Figura 8.12.

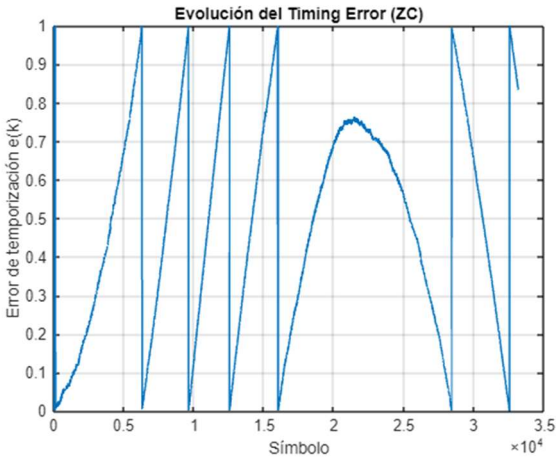

Figura 8.12. Evolución del error de temporización utilizando una transmisión con la ADALM-Pluto y el algoritmo de cruce por cero.

Finalmente, se utiliza el decisor y se obtiene una constelación BPSK sin los efectos del ruido, desviación frecuencial o retardo temporal.

8.2.2 SINCRONISMO DE TRAMA

Una vez aplicadas las correcciones propias del sincronismo de frecuencia y de símbolo, todavía es necesario introducir un nuevo concepto, el sincronismo de trama. Este se muestra en el receptor de la Figura 6.1 y se introdujo en la sección 8.1.1.

El sincronismo de trama [2] es un método que permite determinar el comienzo real de una trama recibida. Esto puede ser necesario ya que, cuando el receptor está escuchando, estará recibiendo ruido hasta que reciba la trama real, tal y como muestra la Figura 8.13. En estos casos, no se podría determinar en qué punto termina el ruido y en qué punto comienza la trama.

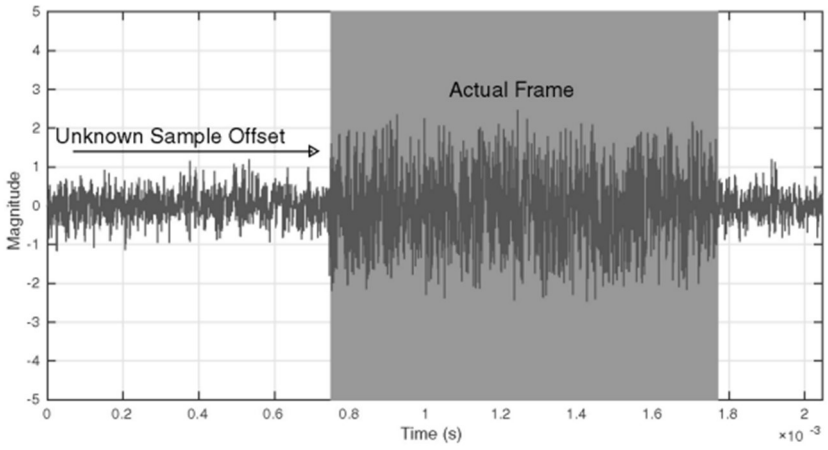

Figura 8.13. Ejemplo de una señal recibida con un comienzo de trama indeterminado [2].

Para ello se utilizan preámbulos que se añaden antes de realizar la modulación y que se conocen en el receptor. Estas secuencias tienen buenas propiedades de autocorrelación, por lo tanto, al calcular la correlación cruzada, se produce un pico que se utiliza para detectar la llegada de una nueva trama.

En concreto, en este proyecto se utilizan los códigos de Barker [2]. Este preámbulo se añade en la función mostrada en el Código 9 del ANEXO II. Estos códigos de Barker pueden tener

diferentes longitudes (Figura 8.14). En este caso, se utiliza el código de 11 números. Para evaluar que los picos de correlación se detectan adecuadamente, se ha concatenado tres veces el vector de muestras antes de la transmisión. Esto implica que se deben detectar tres picos de correlación (Figura 8.15).

N	*Code*
2	−1, +1
3	−1, −1, +1
4	−1, −1, +1, −1
5	−1, −1, −1, +1, −1
7	−1, −1, −1, +1, +1, −1, +1
11	−1, −1, −1, +1, +1, +1, −1, +1, +1, −1, +1
13	−1, −1, −1, −1, −1, +1, +1, −1, −1, +1, −1, +1, −1

Figura 8.14. Códigos de Barker según su longitud [2].

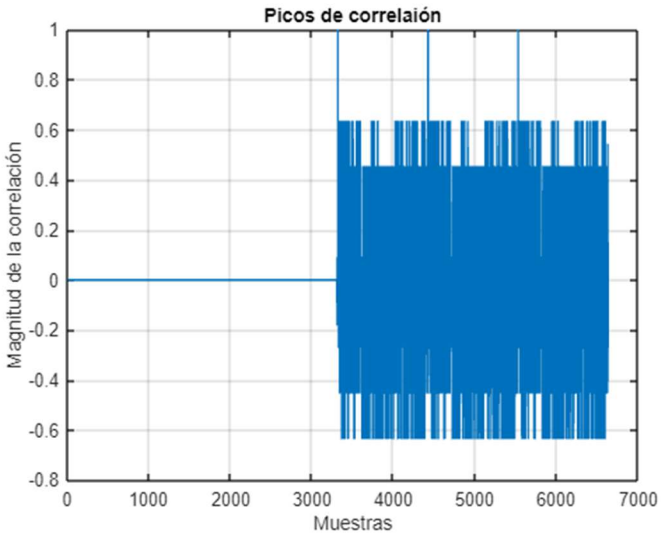

Figura 8.15. Picos de autocorrelación en la simulación BPSK tras el procesado de la señal.

El código que permite calcular estos picos de correlación para implementar este sincronismo de trama es el siguiente:

```
% Computing correlation peaks using barker codes
correlation = xcorr(bitsRx, barkerCode);
correlation = correlation/max(abs(correlation));
[vCorr,idxCorr] = max(abs(correlation));

beginFrame = idxCorr - length(bitsRx);
bitsRx = bitsRx(beginFrame + length(barkerCode) + 1:beginFrame +
length(barkerCode) + length(bitsTx));
```

8.2.3 AMBIGÜEDAD DE FASE

Tras realizar la transmisión BPSK con la ADALM-Pluto y realizar el tratamiento digital correspondiente, se obtiene que la correlación cruzada de la Figura 8.16. En ella, los picos de correlación son negativos en lugar de positivos. Esto se debe a la ambigüedad de fase.

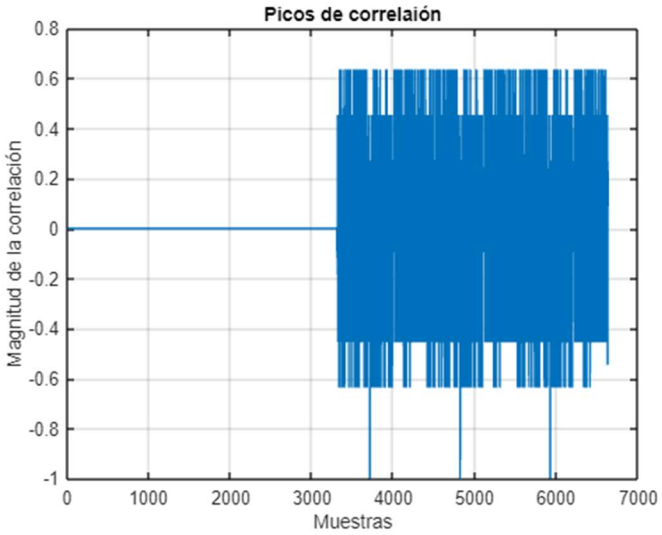

Figura 8.16. Señal BPSK recibida con la ADALM-Pluto afectada por la ambigüedad de fase.

La corrección fina no puede conocer la orientación real de la señal transmitida. De hecho, la documentación de MATLAB del *carrierSynchronizer* [15] advierte que el tratamiento digital de señales utilizando este objeto no resuelve problemas de ambigüedad de fase. La ambigüedad de fase [2] se da cuando en el sincronismo de portadora se corrige la fase, pero no hacia la orientación adecuada. En el caso MPSK, se tiene un total de M orientaciones posibles. Esta incertidumbre provoca que la señal sea demodulada con una rotación incorrecta.

Para resolver este problema, se pueden utilizar los códigos de Barker introducidos en la sección anterior. Si el preámbulo recibido es igual al esperado, entonces la orientación hacia la que se ha corregido la fase es correcta. De no ser así, es necesario aplicar una corrección de 180 grados a los símbolos recibidos. El Código 13 del ANEXO II permite incorporar esta funcionalidad.

Finalmente, se obtiene una BER de 0, lo cual indica que no se han producido errores en la transmisión y que los símbolos se han recibido correctamente.

8.3 SINCRONISMO DE SÍMBOLO CON LA ADALM-PLUTO

El último experimento del que se han obtenido resultados está relacionado con los algoritmos de sincronismo de símbolo estudiados en el Capítulo 7. El objetivo es evaluar el rendimiento de distintos TED en función de los principales parámetros que afectan a la sincronización de símbolo: el número de muestras por símbolo, el ancho de banda normalizado del lazo, el factor de amortiguamiento y el factor de *roll-off* del filtro de recepción.

Este experimento permite identificar configuraciones óptimas de estos parámetros para cada TED, de forma que se minimice el error de temporización y, por tanto, se mejore la calidad general de la recepción.

La metodología consiste en realizar 20 transmisiones por cada combinación de parámetros a evaluar. En cada una de ellas, se mide el MSE de la señal de error de temporización y se toma la media de los 20 MSE obtenidos, que se emplea como una medida del rendimiento.

Cuanto menor sea el MSE medio, mejor será el desempeño del PLL, ya que se traduce en una menor desviación respecto al instante óptimo de muestreo.

8.3.1 PARAMETRIZACIÓN DEL TIEMPO DE SÍMBOLO SEGÚN EL TED

En primer lugar, se debe decidir cuál es el número de muestras por símbolo que minimiza el error de temporización, considerando los tres algoritmos TED explicados en el Capítulo 7.

Para cada uno de estos TED se ha analizado el comportamiento del PLL en función del valor de $sps \in \{2, 4, 6, 8, 10\}$, manteniendo constante el resto de los parámetros (ancho de banda del lazo, amortiguamiento y roll-off).

Los resultados de este experimento se muestran en la Tabla 4. En los tres algoritmos, se observa una reducción progresiva del MSE medio al incrementar el número de muestras por

símbolo desde 2 hasta 10. Esto ilustra que un mayor número de muestras por símbolo mejora la estimación del instante óptimo de muestreo al proporcionar más información temporal para el TED. Aunque hay ligeras diferencias, estas se deben al número de iteraciones y los resultados son coherentes.

	Gardner	**Cruce por cero**	**Müller y Mueller**
$sps = 2$	0,715	0,684	0,736
$sps = 4$	0,603	0,502	0,527
$sps = 6$	0,399	0,442	0,415
$sps = 8$	0,358	0,375	0,371
$sps = 10$	0,392	0,354	0,326

Tabla 4. MSE medio según el TED y las muestras por símbolo tras 20 iteraciones.

También cabe mencionar que estos resultados también pueden verse afectados por un mejor rendimiento del PLL del corrector fino al aumentar el número de muestras por símbolo. Por ello, en adelante, se fija el valor del número de muestras por símbolo a 10.

8.3.2 PARAMETRIZACIÓN DEL ANCHO DE BANDA NORMALIZADO DEL LAZO Y DEL FACTOR DE AMORTIGUAMIENTO SEGÚN EL TED

El siguiente punto es optimizar los parámetros que controlan la dinámica del PLL y que ya se introdujeron en el Capítulo 7. Estos son el ancho de banda normalizado del lazo y el factor de amortiguamiento.

En el caso del ancho de banda normalizado de lazo, se han utilizado valores tales que $b_n \in \{0{,}001; 0{,}005; 0{,}01; 0{,}02; 0{,}05\}$ y manteniendo constante el factor de amortiguamiento a $\zeta = 1{,}2$. Los resultados obtenidos en este caso se muestran en la Tabla 5. De esta se puede concluir que en general, los mejores resultados se obtienen con anchos de banda pequeños entre 0,001 y 0,01, que dan una respuesta más lenta pero estable. Esto es coherente, ya que, en entornos ruidosos como este, estos valores mejoran el rendimiento del PLL. En la tabla se han resaltado los menores MSE medios. Ante estos resultados, el valor de b_n se fija a 0,001.

	Gardner	**Cruce por cero**	**Müller y Mueller**
$b_n = 0{,}001$	0,371	**0,339**	0,341
$b_n = 0{,}005$	0,420	0,355	**0,322**
$b_n = 0{,}01$	**0,337**	0,352	0,368
$b_n = 0{,}02$	0,361	0,438	0,414
$b_n = 0{,}05$	0,371	0,461	0,441

Tabla 5. MSE medio según el TED y el ancho de banda normalizado tras 20 iteraciones.

Posteriormente, se analiza el impacto del factor de amortiguamiento (ζ) en el rendimiento de los algoritmos. En este caso, se utilizan valores de $\zeta \in \{0{,}5;\ 0{,}707;\ 1;\ 1{,}5;\ 2\}$ Los resultados se muestran en la Tabla 6. En general, el MSE medio disminuye a medida que aumenta el factor de amortiguamiento[12]. Un factor de amortiguamiento mayor implica una respuesta más suave y estable, lo cual reduce las oscilaciones del error de temporización. Bajo este escenario experimental, se fija $\zeta = 2$. Los valores mínimos de MSE medio se resaltan.

	Gardner	**Cruce por cero**	**Müller y Mueller**
$\zeta = 0{,}5$	0,434	0,386	0,348
$\zeta = 0{,}707$	0,365	0,369	0,357
$\zeta = 1$	0,409	0,328	0,341
$\zeta = 1{,}5$	0,350	**0,311**	0,410
$\zeta = 2$	**0,292**	0,323	**0,320**

Tabla 6. MSE medio según el TED y el factor de amortiguamiento tras 20 iteraciones.

8.3.3 PARAMETRIZACIÓN DEL FACTOR DE ROLL-OFF DEL FILTRO SRRC SEGÚN EL TED

El último experimento analiza el impacto del factor de *roll-off* (β) del filtro SRRC en el rendimiento de los algoritmos de sincronismo de símbolo estudiados. Los diferentes valores que se utilizan son $\beta \in \{0{,}1;\ 0{,}25;\ 0{,}35;\ 0{,}5;\ 1\}$.

[12] El caso del algoritmo MM presenta una irregularidad para $\zeta = 1{,}5$, pero globalmente también se mejora el MSE al aumentar ζ.

La Tabla 7 muestra resultados que dependen del algoritmo de sincronismo utilizado. Para el caso de Gardner y de cruce por cero, el mejor rendimiento se obtiene para valores de $\beta \in (0{,}5;\ 1)$. Sin embargo, para el caso del algoritmo MM, se tiene un mejor rendimiento para valores de $\beta \in (0{,}1;\ 35)$. Es decir, para el caso de MM, es preferible utilizar valores de β más pequeños, mientras que para los casos de Gardner y cruce por cero, es conveniente utilizar valores mayores.

	Gardner	**Cruce por cero**	**Müller y Mueller**
$\beta = 0{,}1$	0,483	0,371	0,365
$\beta = 0{,}25$	0,453	0,345	**0,356**
$\beta = 0{,}35$	0,390	0,355	0,379
$\beta = 0{,}5$	0,400	**0,325**	0,387
$\beta = 1$	**0,379**	0,327	0,382

Tabla 7. MSE medio según el TED y el factor de roll-off tras 20 iteraciones.

Capítulo 9. BIBLIOGRAFÍA

[1] Mathworks, "Get Started with Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio," [Online]. Available: https://es.mathworks.com/help/comm/getting-started-with-communications-system-toolbox-support-package-for-pluto-radio.html?s_tid=CRUX_lftnav. [Accessed 14 5 2025].

[2] T. F. Collins, R. Getz, D. Pu and A. M. Wyglinski, Software-Defined Radio for Engineers, Artech House, 2018.

[3] "IEEE Approved Draft Standard for Definitions and Concepts for Dynamic Spectrum Access: Terminology Relating to Emerging Wireless Networks, System Functionality, and Spectrum Management Amendment: Addition of New Terms and Associated Definitions," *IEEE P1900.1a/D2.0,* pp. 1-18, 7 Dec 2012.

[4] MathWorks, "Communications Toolbox," [Online]. Available: https://es.mathworks.com/help/comm/index.html. [Accessed 14 5 2025].

[5] L. F. Novales Peleato, *Implementación de técnicas mejoradas de comunicaciones digitales aplicadas a docencia,* Trabajo de Fin de Grado, Universidad Pontificia Comillas, 2021.

[6] Teleco Renta, «Recursos para Escuelas de Teleco,» [En línea]. Available: https://github.com/Telecorenta?tab=repositories. [Último acceso: 26 05 2025].

[7] M. Lázaro, Teoría de la Comunicación, Universidad Carlos III de Madrid Open Course Ware, 2014.

[8] A. Artés Rodrígez, F. Pérez González, J. Cid Sueiro, R. López Valcarce, C. Mosquera Nartallo y F. Pérez Cruz, Comunicaciones Digitales, Prentice Hall, 2012.

[9] M. Lichtman, «PySDR: Guía de uso para SDR/DSP con Python,» [En línea]. Available: https://pysdr.org/es/content-es/digital_modulation.html. [Último acceso: 23 5 2025].

[10] A. V. Oppenheim y R. W. Schafer, Tratamiento de señales en tiempo discreto, Tercera edición, Pearson Educación, 2011.

[11] R. W. Stewart, K. W. Barlee, D. S. W. Atkinson y L. H. Crockett, Software Defined Radio using MATLAB & Simulink and the RTL-SDR, Strathclyde Academic Media, 2015.

[12] MathWorks, «Coarse Frequency Compensator,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/coarsefrequencycompensator.html. [Último acceso: 27 05 2025].

[13] Y. Wang, K. Shi y E. Serpedin, «Non-Data-Aided Feedforward Carrier Frequency Offset Estimators for QAM Constellations: A Nonlinear Least-Squares Approach,» *EURASIP Journal on Advances in Signal Processing,* nº 13, 2004.

[14] M. Luise y R. R., «Carrier Frequency Recovery in All-Digital Modems for Burst-Mode Transmission,» *IEEE Transactions on Communications 43,* nº 2/3/4, 1995.

[15] MathWorks, «Carrier Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/carriersynchronizer.html. [Último acceso: 27 05 2025].

[16] MathWorks, «Symbol Synchronizer,» [En línea]. Available: https://es.mathworks.com/help/comm/ref/symbolsynchronizer.html. [Último acceso: 27 05 2025].

[17] F. Gardner, «A BPSK/QPSK Timing-Error Detector for Sampled Receivers,» *IEEE Transactions on Communications,* vol. 34, nº 5, pp. 423-429, Mayo 1986.

[18] K. Mueller y M. Muller, «Timing Recovery in Digital Synchronous Data Receivers,» *IEEE Transactions on Communications,* vol. 24, nº 5, pp. 516-531, Mayo 1976.

ANEXO I: ALINEACIÓN DEL PROYECTO CON LOS ODS

El objetivo de desarrollo sostenible número 4 expone: "**Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos**".

Situaciones como la pandemia provocada por el Covid-19 ha provocado pérdidas de aprendizaje en 4 de cada 5 países de una muestra de 104. Esto es especialmente crítico, teniendo en cuenta que la educación es la base de la consecución de otros ODS.

En el contexto de la constante evolución tecnológica, la formación en cuestiones relacionadas con sistemas de telecomunicación es crítica. Por ello, este proyecto pretende enseñar el funcionamiento de sistemas de comunicaciones de una forma didáctica y económicamente asequible. De este modo, se garantiza una educación de calidad y disponible para todos.

ANEXO II. CÓDIGO DE FUNCIONES AUXILIARES

A lo largo de este anexo, se tiene el código empleado en MATLAB para el desarrollo del proyecto.

```
function samplesRx = plutoTransmitter(fc, fs, samples, replicationIdx)
    % Crear el objeto transmisor
    tx = sdrtx('Pluto', 'CenterFrequency', fc);
    tx.BasebandSampleRate = fs;

    % Transmitir las muestras
    samples = samples.';
    tx.transmitRepeat(samples)

    % Crear el objeto receptor
    rx = sdrrx('Pluto', 'CenterFrequency', fc);
    rx.BasebandSampleRate = fs;
    rx.SamplesPerFrame = length(samples) * replicationIdx;
    rx.OutputDataType = 'double';
    data = rx();

    % Para detener la transmisión:
    release(tx);

    % Para dejar de recibir:
    release(rx);

    % Señal recibida
    samplesRx = data.';
end
```

Código 1. Función "plutoTransmitter".

```
% Frecuencia de muestreo del transmisor (1 MHz)
% Frecuencia de la portadora (200 kHz)
% Índice de modulación (entre 0 y 1)
fs_tx = 1e6;
f_c = 200e3;
indice_modulacion = 1;
```

```
% Frecuencia del tono puro (10 kHz)
% Duración del tono en segundos
f_m = 10e3;
duracion = 0.001;

% Vector de tiempo
t_tx = (0 : 1/fs_tx : duracion - 1/fs_tx);

% Generar el tono puro (señal moduladora). Está normalizado, sino, sería
% necesario dividir entre su máximo.
moduladora = cos(2 * pi * f_m * t_tx);

% Señal portadora con exponencial compleja
portadora = exp(2 * pi * f_c * 1i * t_tx);

% Señal modulada en AM: (1 + m(t)) * portadora_exponencial_compleja
am_modulada = (1 + indice_modulacion * moduladora) .* portadora;
am_modulada_norm = am_modulada/max(am_modulada);
```

Código 2. Módem AM.

```
% Demodular la señal
portadora_recepcion =  exp(2 * pi * -f_c * i * t_tx);
signal_recibida = am_modulada_norm .* portadora_recepcion;

% Obtener la envolvente de la señal
envolvente = abs(signal_recibida);

% Se emplea un filtro diferenciador ya que no es necesario un filtrado paso bajo
% por la exponencial compleja.
% Eliminar la componente continua
signal_demodulada = envolvente - mean(envolvente);
```

Código 3. Receptor AM.

```
function [encodedText, encodingMap] = sourceEncoder(text)
    % ---------------------------------------------------------------
    % Handling input arguments
    if ~ischar(text) || isempty(text)
        error('Input must be a non-empty array of characters.');
    end
    % ---------------------------------------------------------------

    % Compute the frequency of each letter in the message
    letters         = unique(text);
    letterCounts    = zeros(1, length(letters));
    for i = 1:length(letters)
```

```
        letterCounts(i) = count(text, letters(i));
    end
    totalLetters    = sum(letterCounts);
    probabilities   = letterCounts / totalLetters;
    [~, sortIndex]  = sort(probabilities, 'descend');
    sortedLetters   = letters(sortIndex);

    % Assign to every letter a binary codification according to frequency.
    encoding        = containers.Map;
    for i = 1:length(sortedLetters)
        if i == 1
            encoding(sortedLetters(i)) = '0';
        elseif i == length(sortedLetters)
            encoding(sortedLetters(i)) = repmat('1', 1, i - 1);
        else
            encoding(sortedLetters(i)) = [repmat('1', 1, i -1), '0'];
        end
    end
    encodingMap = encoding;

    % Create intermidiate variable encodedStr
    encodedStr = '';
    for i = 1:length(text)
        encodedStr = strcat(encodedStr, encodingMap(text(i)));
    end
    % Convert encodedStr in the output variable with a numeric format
    % This code is like a comprhension python list to avoid loop
    encodedText = arrayfun(@(x) str2double(x), encodedStr);
end
```

Código 4. Función "sourceEncoder".

```
function decodedText = sourceDecoder(encodedText, encodingMap)

    % ---------------------------------------------------------------
    % Handling input arguments
    if ~isnumeric(encodedText) || ~all(ismember(encodedText, [0, 1]))
        error('Input encodedText must be a numeric array of 0s and 1s.');
    end
    if ~isa(encodingMap, 'containers.Map')
        error('Input encodingMap must be a containers.Map object.');
    end
    % ---------------------------------------------------------------
    % Create decoding map
    keys        = encodingMap.keys;
    values      = encodingMap.values;
    decodingMap = containers.Map(values, keys);
    encodedStr  = num2str(encodedText);

    % Decode the binary sequence
```

```
    decodedText = '';
    tempStr     = '';
    for i = 1:length(encodedStr)
        tempStr = strcat(tempStr, encodedStr(i));
        if isKey(decodingMap, tempStr)
            decodedText = strcat(decodedText, decodingMap(tempStr));
            tempStr = '';
        end
    end
end
```

Código 5. Función "sourceDecoder".

```
function encodedBits = channelEncoder(bitsSequence, N)

    % ------------------------------------------------------------------
    % Handling input arguments
    if any(bitsSequence ~= 0 & bitsSequence ~= 1)
        error('Bits sequence must contain only 0 and 1.')
    elseif ~isnumeric(N) || N <= 0
        error(['Number of desired redundant bits must be a positive ' ...
            'integer'])
    elseif size(bitsSequence, 1) ~= 1
        error('Bits sequence must be a one-row matrix')
    end
    % ------------------------------------------------------------------

    % Create a sequence of redundant bits
    encodedBits = repmat(bitsSequence, N, 1);
    encodedBits = reshape(encodedBits, 1, []);
end
```

Código 6. Función "channelEncoder".

```
function decodedBits = channelDecoder(bitsSequence, N)

    % ------------------------------------------------------------------
    % Handling input arguments
    if any(bitsSequence ~= 0 & bitsSequence ~= 1)
        error('Bits sequence must contain only 0 and 1.')
    elseif ~isnumeric(N) || N <= 0
        error(['Number of redundant bits must be a positive ' ...
            'integer'])
    elseif size(bitsSequence, 1) ~= 1
        error('Bits sequence must be a one-row matrix')
    elseif mod(length(bitsSequence), N) ~= 0
        error('Bits sequence is not valide (must be a multiple of N).')
    end
```

```
    % -------------------------------------------------------------------

    % Decode the bits sequence into the original version.
    % Criteria for decoding is majority voting.

    reshapedSequence = reshape(bitsSequence, N, []);
    decodedBits = zeros(1, size(reshapedSequence, 2));

    for i = 1:size(reshapedSequence, 2)
        bitsColumn = reshapedSequence(:, i);
        if sum(bitsColumn) >= ceil(N / 2)
            decodedBits(i) = 1;
        else
            decodedBits(i) = 0;
        end
    end

    decodedBits = decodedBits(:)';

end
```

Código 7. Función "channelDecoder".

```
function [h, t] = matchedFilter(beta, span, sps)

    % Initializing vectors
    t = (0:(sps * span + 1)) - (span * sps) / 2;
    h = 1:length(t);

    % Based on SDR4Engineers, page 192 and section 8.2.
    % 1st case: t = 0
    h(t == 0) = (1 / sqrt(sps)) * (1 - beta + 4 * beta / pi);

    % t == ±sps / (4 * beta)
    tsSps4beta = abs(t) == (sps / (4 * beta));
    h(tsSps4beta) = (beta / sqrt(2 * sps)) * ((1 + 2 / pi) * ...
        sin(pi / (4 * beta)) + (1 - 2 / pi) * cos(pi / (4 * beta)));

    % Otherwise
    tAux = t((t ~= 0) & (abs(t) ~= (sps / (4 * beta))));
    h((t ~= 0) & (abs(t) ~= (sps / (4 * beta)))) = (1 / sqrt(sps)) * ...
        (sin(pi * tAux * (1 - beta) / sps) + 4 * beta * tAux .* ...
        cos(pi * tAux * (1 + beta) / sps) / sps) ./ ...
        (pi * tAux .* (1 - (4 * beta * tAux / sps).^2) / sps);

end
```

Código 8. Función "matchedFilter".

```
function [samples, h] = BPSKmodulator(bits, sps, beta, span, preamble)

    % BPSK Mapping
    samples = complex((bits*2)-1);

    % Adds preamble
    samples = complex([preamble, samples]);

    % Interpolating
    samples = upsample(samples, sps);

    % Designining SRRC Filter
    [h, ~]    = matchedFilter(beta, span, sps);

    % Convolution
    samples = complex(conv(samples, h, 'same'));
end
```

Código 9. Función "BPSKmodulator".

```
% Time delay
if isTimeDelay
    hdelay = zeros(1, sampleDelay + 1);
    hdelay(end) = 1;
    samplesRx = complex(conv(samplesRx, hdelay, 'same'));
end

% Noise
if isNoiseEffect
    samplesRx = awgn(samplesRx, snr);
end

% Frequency Deviation
if isFreqDevEffect
    taux        = (0:length(samplesRx)-1) / rb;
    freqDev     = exp(1i * 2 * pi * freqOffset * taux);
    samplesRx   = samplesRx .* freqDev;
end
```

Código 10. Efectos del canal de comunicación.

```
function [correctedSamples, freqDevEstiamtion] = cfcCorrection(samples, fs)

    % Parameters
    M = 2;

    % Raise to signal power
    powerSamples = samples.^M;
```

```
    % FFT of the powered signal
    fftSignal = fftshift(abs(fft(powerSamples, length(powerSamples))) ...
        /length(powerSamples));
    fftAxis = linspace(-fs/2, fs/2, length(powerSamples));

    % Estimate Frecuency offset
    [~, maxIdx]         = max(fftSignal);
    maxFreq             = fftAxis(maxIdx);
    freqDevEstiamtion   = maxFreq/M;

    % NCO (Complex exponential)
    taux                = (0:length(samples)-1) / (fs);
    correctedSamples    = samples .* exp(1i * 2 * pi ...
        * -freqDevEstiamtion * taux);

end
```

Código 11. Función "cfcCorrection".

```
function [ber, numErrors] = computeBER(transmittedBits, receivedBits)

    % ---------------------------------------------------------------------
    % Handling input arguments
    if length(transmittedBits) ~= length(receivedBits)
        error('Transmitted and received bits must have the same length.')
    elseif any(transmittedBits ~= 0 & transmittedBits ~= 1)
        error('Transmitted bits must contain only 0 and 1.')
    elseif any(receivedBits ~= 0 & receivedBits ~= 1)
        error('Received bits must contain only 0 and 1.')
    end
    % ---------------------------------------------------------------------

    % Compute ber and number of errors
    bitErrors = xor(transmittedBits, receivedBits);
    numErrors = sum(bitErrors);
    ber = numErrors / length(transmittedBits);

end
```

Código 12. Función computeBER

```
% Phase Ambiguity Correction
receivedPreamble = bitsRx(1:length(barkerCode));
expectedPreamble = barkerCode;

expectedPreamble = expectedPreamble(:);
receivedPreamble = receivedPreamble(:);
```

```
corrNormal = sum(expectedPreamble == receivedPreamble);
corrInverted = sum(expectedPreamble == -receivedPreamble);

if corrInverted > corrNormal
    bitsRx = -bitsRx;
end
```

Código 13. Corrección de la ambigüedad de fase.

ANEXO III. DESARROLLO DE LA EXPRESIÓN ANALÍTICA DE UNA MODULACIÓN AM CONVENCIONAL

La expresión analítica de una señal moduladora es:

$$s(t) = A_c \cos(\omega_c t + \phi_c) + m(t) \cdot A_c \cos(\omega_c t + \phi_c)$$

$$s(t) = A_c \cdot [1 + m(t)] \cdot \cos(\omega_c t + \phi_c)$$

La señal $m(t)$ es la señal moduladora, A_c es la amplitud de la portadora, ω_c es la frecuencia angular de la portadora y ϕ_c su fase [7]. La amplitud de la señal a lo largo del tiempo viene definida por:

$$A_c \cdot [1 + m(t)]$$

Conviene definir la señal $m(t)$ de forma que $|m(t)| \leq 1$ para evitar que la señal esté *sobremodulada*. Esto dificultaría el proceso de demodulación. Por ello, se normaliza la señal $m(t)$ y se introduce el índice de modulación a [7]

:

$$m_n(t) = \frac{m(t)}{\text{máx}|m(t)|} = \frac{m(t)}{C_M}$$

$$m_a(t) = a \cdot m_n(t)$$

Donde, para prevenir el problema de la *sobremodulación*, se debe cumplir que:

$$0 < a \leq 1$$

De modo que la expresión general quedaría como:

$$s(t) = c(t) + m_a(t) \cdot c(t) = A_c \cdot [1 + a \cdot m_n(t)] \cdot \cos(2\pi f_c t + \phi_c)$$