UNIVERSIDAD ESTATAL

PENÍNSULA DE SANTA ELENA

FACSISTEL

INGENIERÍA EN TELECOMUNICACIONES

TRABAJO DE INTEGRACIÓN CURRICULAR

Implementación de técnicas de modulación digital en Python para determinar parámetros de transmisión en un canal inalámbrico simulado.

Reyes Ramírez Jorge Antonio

Dirigido por:

Ing. Daniel Jaramillo, Mgtr.

LA LIBERTAD – 2026

TRIBUNAL DE SUSTENTACIÓN

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

Ing. Ronald Rovira Jurado, PhD.
DIRECTOR DE LA CARRERA

Ing. Daniel Jaramillo, Mgtr.
TUTOR

Ing. Fernando Chamba, Mgtr.
DOCENTE ESPECIALISTA

Ing. Luis Amaya Fariño, Mgtr.
DOCENTE GUÍA UIC

Ing. Corina Gonzabay De La A, Mgtr.
SECRETARIA

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Reyes Ramírez Jorge Antonio**

DECLARO QUE:

El trabajo de Titulación, **Implementación de técnicas de modulación digital en Python para determinar parámetros de transmisión en un canal inalámbrico simulado** de un previo a la obtención del título en Ingeniero en Telecomunicaciones, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance del Trabajo de Titulación referido.

La Libertad, a los 04 días del mes de agosto del año 2026

EL AUTOR

Reyes Ramírez Jorge Antonio

CERTIFICACIÓN DEL TUTOR

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que luego de haber dirigido científica y técnicamente el desarrollo y estructura final del trabajo, este cumple y se ajusta a los estándares académicos, razón por el cual apruebo en todas sus partes el presente trabajo de titulación que fue realizado en su totalidad por **Reyes Ramírez Jorge Antonio**, como requerimiento para la obtención del título de Ingeniero en telecomunicaciones.

La Libertad, a los 04 días del mes de agosto del año 2026

TUTOR

Ing. Daniel Armando Jaramillo Chamba, Mgtr.

AUTORIZACIÓN

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Reyes Ramírez Jorge Antonio**

Autorizo a la Universidad Estatal Península de Santa Elena, para que haga de este trabajo de titulación o parte de él, un documento disponible para su lectura consulta y procesos de investigación, según las normas de la Institución.

Cedo los derechos en línea patrimoniales de artículo profesional de alto nivel con fines de difusión pública, además apruebo la reproducción de este artículo académico dentro de las regulaciones de la Universidad, siempre y cuando esta reproducción no suponga una ganancia económica y se realice respetando mis derechos de autor

Santa Elena, a los 04 días del mes de agosto del año 2026

EL AUTOR

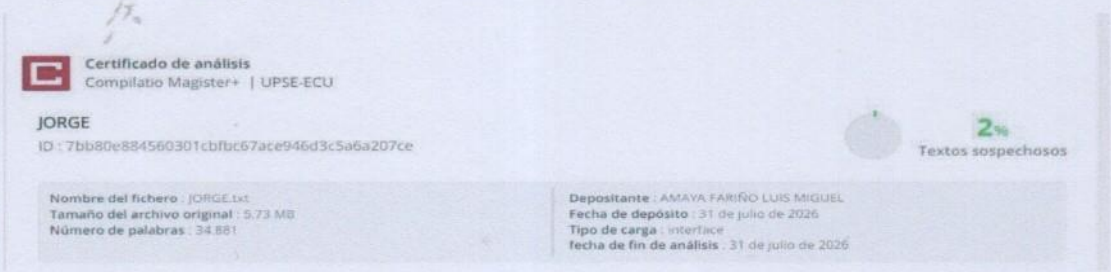
Reyes Ramírez Jorge Antonio

UPSE

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA
FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CERTIFICACIÓN DE ANTIPLAGIO

Certifico que después de revisar el documento final del trabajo de titulación denominado **Implementación de técnicas de modulación digital en Python para determinar parámetros de transmisión en un canal inalámbrico simulado,** presentado por el estudiante, **Reyes Ramírez Jorge Antonio** fue enviado al Sistema Antiplagio, presentando un porcentaje de similitud correspondiente al 02%, por lo que se aprueba el trabajo para que continúe con el proceso de titulación.



Certificado de análisis
Compilatio Magister+ | UPSE-ECU

JORGE
ID : 7bb80e884560301cbfbc67ace946d3c5a6a207ce

2%
Textos sospechosos

Nombre del fichero : JORGE.txt
Tamaño del archivo original : 5.73 MB
Número de palabras : 34.881

Depositante : AMAYA FARIÑO LUIS MIGUEL
Fecha de depósito : 31 de julio de 2026
Tipo de carga : interface
fecha de fin de análisis : 31 de julio de 2026

TUTOR

Ing. Daniel Armando Jaramillo Chamba, Mgtr.

DEDICATORIA

Dedico este trabajo con todo mi cariño a mi madre, quien con su esfuerzo, paciencia y amor incondicional ha sido el pilar fundamental en cada etapa de mi vida y de mi formación profesional. A mi hermano, por su apoyo constante y por creer en mí incluso en los momentos difíciles.

Asimismo, extiendo esta dedicatoria a mi familia, quienes de una u otra forma acompañaron este camino con su cariño y buenos deseos.

Con gratitud y cariño infinito,

Jorge Antonio Reyes Ramírez

AGRADECIMIENTO

Agradezco en primer lugar a mi familia, por su apoyo incondicional, su paciencia y su aliento constante durante todo el proceso de formación académica. Su respaldo ha sido fundamental para alcanzar esta meta.

Al Ing. Daniel Jaramillo, mi tutor, por su guía, disposición y valiosos conocimientos brindados durante el desarrollo de este proyecto, así como por su paciencia y compromiso en cada una de las etapas de este trabajo.

Al Ing. Fernando Chamba, mi especialista, por sus observaciones, recomendaciones y aportes que contribuyeron a mejorar y enriquecer el presente trabajo.

A mis compañeros de la carrera de Telecomunicaciones, con quienes compartí conocimientos, experiencias y momentos importantes a lo largo de este camino académico.

A todos mis más sinceros agradecimientos.

Jorge Antonio Reyes Ramírez

RESUMEN

La transmisión digital predomina hoy frente a la analógica por su mayor resistencia al ruido y eficiencia espectral, por lo que estudiar el comportamiento de las modulaciones digitales ante un canal inalámbrico simulado resulta relevante para la enseñanza de comunicaciones, frecuentemente limitada por la falta de laboratorios. El proyecto tuvo como objetivo desarrollar con fines educativos un sistema de comunicaciones digitales en Python que simule este comportamiento.

Para ello, se implementaron los esquemas ASK, FSK, BPSK, QPSK, 16-QAM y 64-QAM, codificación de errores Reed-Solomon y un canal con ruido AWGN junto a desvanecimiento Rayleigh, evaluando su desempeño mediante simulaciones Monte Carlo y la transmisión de imagen y audio bajo las métricas BER, MER, EVM y eficiencia espectral. Los resultados mostraron que BPSK y QPSK presentaron el menor BER en todo el rango de SNR evaluado, mientras que 16-QAM y 64-QAM alcanzaron mayor eficiencia espectral a costa de mayor sensibilidad al ruido. Se concluye que el sistema constituye una herramienta viable como laboratorio virtual para la enseñanza de comunicaciones digitales.

Palabras clave: Python, modulación digital, desvanecimiento Rayleigh.

ABSTRACT

Digital transmission predominates today over analog due to its greater noise immunity and spectral efficiency. Therefore, studying the behavior of modulations in a simulated wireless channel is relevant for teaching digital communications, which is often limited by a lack of laboratories. The project aimed to develop, for educational purposes, a digital communications system in Python that simulates this behavior.

To this end, the following signaling schemes were implemented: ASK, FSK, BPSK, QPSK, 16-QAM, and 64-QAM, along with Reed-Solomon error coding and an AWGN noise channel with Rayleigh fading. Performance was evaluated through Monte Carlo simulations and image and audio transmission using BER, MER, EVM, and spectral efficiency metrics. The results showed that BPSK and QPSK exhibited the lowest BER across the entire SNR range evaluated, while 16-QAM and 64-QAM achieved higher spectral efficiency at the cost of greater sensitivity to noise. It is concluded that the system constitutes a viable tool as a virtual laboratory for teaching digital communications.

Keywords: Python, digital modulation, Rayleigh fading.

GLOSARIO DE ABREVIATURAS

ASK: Amplitude Shift Keying o modulación por desplazamiento de amplitud.

AWGN: Additive White Gaussian Noise o ruido blanco gaussiano aditivo.

BER: Bit Error Rate o tasa de error de bit.

BPSK: Binary Phase Shift Keying o modulación por desplazamiento de fase binaria.

Eb/N0: Energy per-Bit to Noise Power Spectral Density Ratio o relación de energía por bit a densidad espectral de ruido.

EVM: Error Vector Magnitude o magnitud del vector de error.

FSK: Frequency Shift Keying o modulación por desplazamiento de frecuencia.

IQ: In-phase and Quadrature o componentes en fase y en cuadratura.

MER: Modulation Error Rate o tasa de error de modulación.

QAM: Quadrature Amplitude Modulation o modulación de amplitud en cuadratura.

QPSK: Quadrature Phase Shift Keying o modulación por desplazamiento de fase en cuadratura.

SNR: Signal-to-Noise Ratio o relación señal-ruido.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ECUACIONES

INTRODUCCIÓN

En este sentido, son de vital importancia las técnicas de modulación digital (ASK, FSK, BPSK, QPSK y QAM), ya que son las que soportan la eficiencia espectral y la robustez de los enlaces inalámbricos actuales. Sin embargo, la comprensión práctica de estos métodos se limita por la necesidad de costosos laboratorios y equipo especializado, infraestructura de la que muchas instituciones educativas, especialmente aquellas con recursos limitados, carecen.

Ante esta limitación, este trabajo plantea un entorno de simulación construido en Python que reproduce la cadena completa de la comunicación digital, desde la generación de bits hasta la detección en el receptor, pasando por canales con ruido blanco gaussiano, desvanecimiento Rayleigh y un esquema de corrección de errores. El sistema, basado en bibliotecas abiertas como NumPy, SciPy y Matplotlib, provee de forma ágil y reproducible métricas como el BER y la eficiencia espectral, lo que permite comparar cuantitativamente el desempeño de cada esquema bajo diferentes niveles de relación señal-ruido.

El aporte del proyecto no se limita al ámbito técnico, pues también responde a una necesidad formativa concreta: los estudiantes de Telecomunicaciones de la Universidad Estatal Península de Santa Elena podrán interactuar con un laboratorio virtual que traduce conceptos abstractos en resultados observables y medibles. De este modo, se afianza la comprensión de los fenómenos de propagación y se ofrece a los futuros ingenieros criterios concretos para seleccionar una modulación u otra según el escenario que enfrenten, lo cual aporta tanto a su formación académica como a su ejercicio profesional.

CAPÍTULO I: EL PROBLEMA DE LA INVESTIGACIÓN

1.1 Antecedentes

Basta revisar la producción académica reciente para notar una evolución marcada en el uso de herramientas de software para estudiar la modulación digital y los fenómenos que alteran la transmisión inalámbrica. [1] desarrolló en Python un sistema de radiocomunicaciones capaz de comparar QAM, PSK y sus variantes en OFDM, con la mira puesta en reducir el BER al mínimo. Se trataba de un enfoque marcadamente teórico-práctico, apoyado en librerías especializadas, que abrió paso a simulaciones de alto nivel sin necesidad de equipo de laboratorio costoso. Un año más tarde, [2] amplió el panorama combinando la transformada wavelet con redes neuronales para la clasificación automática de modulaciones analógicas y digitales. Sintéticamente, generó señales PAM, BPSK, ASK, FSK, PSK y QAM empleando algoritmos genéticos para extraer rasgos, probando así que el reconocimiento inteligente puede reforzar el diagnóstico del desempeño en escenarios reales.

En una línea de conexión entre teoría y aplicación didáctica, [3] diseñó en MATLAB un entorno interactivo para comparar modulaciones digitales bajo diferentes niveles de ruido, relacionando el orden de modulación con la eficiencia espectral y la velocidad de transmisión. Según sus resultados, las simulaciones detalladas ayudan a comprender mejor las tecnologías de banda ancha y, además, funcionan como banco de pruebas confiable para validar supuestos teóricos. En este mismo año, [4] llevó el debate al terreno práctico de la radio definida por software usando GNU Radio y el SDR HackRF One, y puso a prueba las modulaciones PSK y QAM bajo desvanecimiento y pérdida de información, mostrando cómo el hardware asequible en la enseñanza ayuda a entender mejor las limitaciones reales que afectan la calidad de servicio. En la misma línea, [5] desarrolló una interfaz gráfica en Python que actualiza en tiempo real constelaciones y curvas de BER. Sus pruebas demostraron que 16-QAM logra la misma calidad que 16-PSK con menor Eb/N0, lo que pone de relieve la ventaja energética de QAM y el valor didáctico de este tipo de simulaciones interactivas.

Juntos estos trabajos dejan ver que cada vez pesa más el uso de lenguajes de programación como Python, MATLAB o entornos SDR para modelar, simular y determinar los parámetros críticos de la comunicación inalámbrica. Juntar técnicas de modulación con modelos de canal como Rayleigh y esquemas de corrección de errores

tiende un puente real entre lo que se enseña en el aula y lo que se ejerce en la práctica profesional. Esta investigación se ubica justo en ese cruce: toma como base la simulación en Python que planteó Cañero, suma la mirada de clasificación inteligente de Alvarado, se apoya en el análisis de eficiencia espectral que exploró Maldonado y recoge la visión experimental de Saenz, todo ello con el objetivo de proporcionar a los estudiantes de telecomunicaciones una guía completa que abarque tanto la simulación como la verificación práctica de los sistemas digitales modernos.

1.2 Descripción del proyecto

La capacidad de red se demanda con ritmos exponenciales, mientras el espectro radioeléctrico disponible se mantiene prácticamente constante. Frente a esta limitación, la única alternativa factible consiste en optimizar el uso del espectro disponible, es decir, transmitir la mayor cantidad de información posible dentro del ancho de banda existente. En este sentido, la presente investigación resulta relevante al proponer un modelo integral que combina técnicas avanzadas de modulación digital, que determinan cuántos bits se transmiten por símbolo, con métodos de corrección de errores, que reducen la necesidad de retransmisiones y, con ello, el desperdicio adicional de espectro.

Este enfoque se caracteriza por implementar esas técnicas en un entorno simulado con Python que incorpora condiciones realistas de propagación, como ruido gaussiano y desvanecimiento Rayleigh, siendo los dos factores que más degradan la calidad de las señales. Se incluyen códigos de corrección de errores en el modelo para responder a una necesidad concreta de mantener tasas de error bajas en canales con interferencia, algo que se vuelve más urgente en zonas urbanas muy congestionadas o en zonas con cobertura marginal.

Por último, el sistema compara esquemas de modulación como QAM y PSK bajo distintas relaciones señal-ruido, lo que entrega datos cuantitativos útiles para seleccionar la configuración óptima, dependiendo de las condiciones del canal. De este modo, se ayuda a resolver el dilema entre eficiencia espectral y robustez frente al ruido, un reto habitual para cualquier diseñador de sistemas de comunicación [6].

1.3 Objetivos

1.3.1 General

Construir con herramientas de Python un sistema de comunicaciones digitales que aplique técnicas de modulación y corrección de errores, de modo que permita simular y analizar su comportamiento en condiciones adversas y cuantificar su rendimiento.

1.3.2 Específicos

- **Realizar** una investigación teórica sobre las técnicas de modulación digital ASK, FSK, BPSK, QPSK y QAM, y las técnicas de corrección de errores aplicables en sistemas de comunicaciones, así como el papel de Python como herramienta para simular y analizar estas modulaciones.
- **Implementar** un modelo de canal con desvanecimiento Rayleigh y ruido gaussiano en el sistema de comunicaciones para analizar su impacto en la calidad de la transmisión de datos y añadir un sistema de corrección de errores.
- **Evaluar** el rendimiento del sistema de comunicaciones implementado mediante el análisis de la tasa de error (BER), la tasa de error de modulación (MER), la magnitud del vector de error (EVM) y la eficiencia espectral, basándose en las técnicas de modulación digital estudiadas y en las herramientas desarrolladas en Python.
- **Desarrollar** una guía de prácticas que ayude a los estudiantes de sexto y séptimo semestre de la carrera de Telecomunicaciones a comprender mejor los contenidos vistos en clase, apoyándose en los escenarios planteados a lo largo de este proyecto de integración curricular.

1.4 Justificación

Esta investigación gana relevancia al considerar el panorama global de la conectividad digital que describe la [7] en su informe *Hechos y cifras 2023*: el acceso a internet sigue avanzando de manera desigual, y los países con bajos ingresos continúan enfrentando obstáculos en cuanto a infraestructura, asequibilidad y calidad de la conectividad. En estos escenarios, las redes suelen funcionar en condiciones desfavorables de ruido e interferencia, por lo que la selección del esquema de modulación

y el uso de técnicas de corrección de errores se convierten en factores críticos para establecer una comunicación fiable. Por ello, al combinar técnicas de modulación digital con métodos de corrección de errores, este trabajo aporta tanto en lo teórico como en soluciones potenciales para escenarios reales donde las redes móviles muestran deficiencias, como en regiones donde predomina el 3G o donde la penetración de banda ancha sigue siendo baja.

En la misma línea, se deben incorporar canales con desvanecimiento Rayleigh y ruido gaussiano en la simulación del modelo para considerar las condiciones a las que se enfrentan los usuarios en países con infraestructura precaria, donde las interferencias y las limitaciones geográficas impactan directamente en la calidad de la señal.

1.5 Alcance del proyecto

El presente trabajo se lleva a cabo en un entorno de simulación construido en lenguaje de programación Python, donde se modelan cadenas de comunicación digitales que utilizan los esquemas de modulación ASK, FSK, BPSK, QPSK y QAM, así como un mecanismo de detección y corrección de errores.

Las pruebas se realizan en canales sintéticos que combinan ruido blanco gaussiano aditivo con desvanecimiento Rayleigh, permitiendo el cálculo de cuatro métricas fundamentales: BER, MER, EVM y eficiencia espectral. El desarrollo se basa en librerías estándar como NumPy, SciPy y Matplotlib. Los beneficiarios son básicamente los estudiantes de la carrera de Telecomunicaciones de la UPSE y los profesores que enseñan asignaturas de comunicaciones digitales e inalámbricas.

Se les entrega un paquete de scripts en Python de libre uso y fácil ejecución, junto con una guía metodológica que detalla paso por paso las prácticas de laboratorio e incluye cuestionarios de autoevaluación.

La idea de esta propuesta es transformar conceptos teóricos en experiencias interactivas; por ejemplo, el estudiante podrá ver inmediatamente cómo cambia la calidad de la transmisión al pasar de BPSK a 16-QAM o al endurecer las condiciones del canal. Los resultados numéricos proporcionarán a los alumnos criterios objetivos para elegir la modulación más adecuada en función de las limitaciones de ancho de banda, potencia disponible o calidad de servicio, constituyendo así un recurso valioso tanto para el aula como para su futuro ejercicio profesional.

CAPÍTULO II: MARCO TEÓRICO

2.1 Introducción a los sistemas de comunicación digital

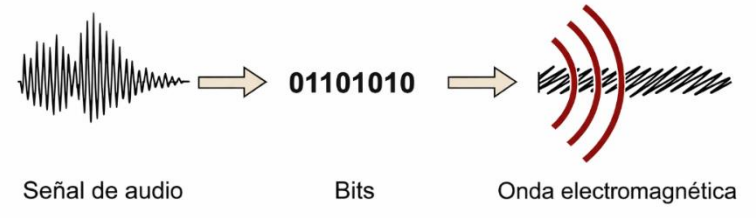
2.1.1 Definición de un sistema de comunicación digital

Un sistema de comunicación digital es aquel que transmite información representada en forma de señales discretas, es decir, secuencias de bits, en lugar de señales continuas como en los sistemas analógicos. Esta representación permite una mayor resistencia al ruido y facilita la detección junto a la corrección de errores, lo que posibilita el uso eficiente de técnicas de compresión y modulación. Actualmente, estos sistemas forman parte fundamental de la vida diaria: la música almacenada en formatos como el MP3 o la comunicación entre computadoras son ejemplos de que las tecnologías digitales están presentes en prácticamente cualquier ámbito.

Uno de los ejemplos más conocidos de digitalización es el llamado apagón analógico, aplicado en España en 2010. Este proceso dejó atrás las últimas emisiones de televisión analógica, reemplazándolas por un sistema digital conocido como TDT. En las secciones siguientes se repasan de forma resumida las ventajas de los sistemas digitales que motivaron este cambio, así como el diagrama de bloques característico de este tipo de comunicaciones.

La **Figura 1** representa justamente este proceso: cómo una señal de audio se convierte en onda electromagnética. La señal analógica se transforma primero en una secuencia de bits, que luego se transmite como onda electromagnética, ilustrando así el principio básico de la comunicación digital.

Figura 1. *Conversión de señal de audio a onda electromagnética.*



Nota: Investigado de [8].

2.1.2 Elementos que conforman el sistema

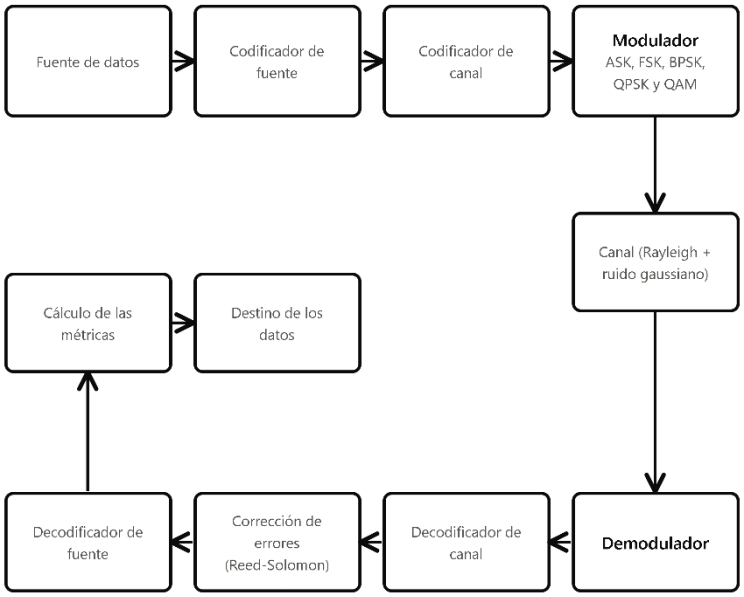
Los elementos de un sistema de comunicación digital conforman un proceso

secuencial que comienza con la generación del dato y termina con su recepción e interpretación en el destino. El proceso comienza con una fuente (imagen, audio o video) que se comprime con un codificador de fuente, con la finalidad de una mejor explotación del canal. A continuación, interviene el codificador del canal Reed-Solomon, que introduce redundancia con el fin de permitir la detección y corrección de errores más tarde. Una vez que la secuencia está lista, se modula usando esquemas como QPSK o QAM [9].

Posteriormente, la señal pasa por un canal expuesto a la atenuación (modelo de Rayleigh) y al ruido, los cuales son los factores que con mayor facilidad pueden alterar la información transmitida. Ya en el receptor se realiza la demodulación y se procede a la decodificación de canal —que corrige los errores presentes— y a la decodificación de fuente, recuperando así los datos originales. La BER se mide al mismo tiempo y es un indicador fundamental del desempeño del sistema y de la calidad de la transmisión. Finalmente, los datos ya reconstruidos llegan a su destino, ya sea un dispositivo receptor, un servidor o el usuario final.

Como se muestra en la **Figura 2**, el siguiente diagrama de bloques resume un sistema de comunicación digital completo: codificación de fuente y de canal, modulación, un canal con ruido de por medio, y finalmente corrección de errores.

Figura 2. *Diagrama de bloques del sistema de comunicación digital.*



Nota. Elaboración propia.

2.2 Procesamiento en el transmisor

2.2.1 Naturaleza de la fuente

En un sistema de comunicación digital, la fuente de información puede ser de naturaleza muy distinta —audio, imagen o video— y cada tipo presenta características particulares que determinan la forma más adecuada de procesarla para transmitirla con eficiencia. El video, por ejemplo, presenta redundancias tanto temporales como espaciales, las cuales pueden aprovecharse para reducir su tamaño sin que se perciba una pérdida notable de calidad. Conocer estas características es lo que permite seleccionar la técnica de procesamiento y compresión más adecuada para cada tipo de fuente [10].

2.2.2 Codificación de fuente y eliminación de redundancia

La compresión de datos consiste en eliminar la redundancia presente en la información para que esta ocupe menos espacio, lo que se traduce en una transmisión y un almacenamiento más eficientes. Existen dos enfoques principales para lograrlo. El primero es la compresión sin pérdida (lossless), que conserva toda la información original y resulta indispensable cuando la integridad del dato no admite concesiones, como en el caso de los archivos de texto. El segundo es la compresión con pérdida (lossy), que descarta la información menos perceptible para el usuario y se aplica comúnmente en audio, imágenes y video, donde sacrificar una parte mínima de calidad a cambio de un archivo considerablemente más liviano resulta una solución eficiente [11].

2.2.3 Importancia de la modulación y codificación

La modulación y la codificación son dos procesos fundamentales que sostienen todo sistema de comunicación digital, ya que sin ellos no sería posible garantizar una transmisión eficiente, segura y confiable frente a interferencias, ruido, atenuación y demás distorsiones que afectan la señal [12].

La modulación toma la secuencia de bits digitales y la convierte en una señal analógica capaz de propagarse por cables, fibra óptica o el espacio libre, adaptándola a las características del canal, reduciendo el ancho de banda requerido y aumentando su resistencia al ruido. Por su parte, la codificación de fuente elimina redundancias innecesarias con el objetivo de comprimir la información, mientras que la codificación de canal introduce una redundancia controlada que permite posteriormente la detección y

corrección de errores en la recepción. En conjunto, ambos procesos posibilitan una comunicación robusta incluso en entornos con altos niveles de ruido, mejorando la calidad del servicio y reduciendo la pérdida de información, aspecto esencial en aplicaciones críticas como las telecomunicaciones, las redes móviles y la transmisión satelital [13].

En estudios más recientes, se explora la integración de la modulación y la codificación dentro de la llamada comunicación digital semántica, en la cual el objetivo ya no es solo transmitir datos, sino transmitir el significado o la intención detrás de estos. Estos enfoques combinan el aprendizaje profundo con esquemas avanzados de modulación y codificación, con el fin de mejorar la eficiencia y la robustez del sistema [14].

En este proyecto, concretamente las técnicas de modulación digital —ASK, FSK, BPSK, QPSK y QAM— se implementan mediante herramientas de simulación en Python con el propósito de observar su comportamiento bajo condiciones de canal adversas como el desvanecimiento Rayleigh y el ruido gaussiano. Estas modulaciones son las que convierten la información digital en formas de onda aptas para propagarse por medios ruidosos, lo que permite determinar su eficiencia espectral y su capacidad de resistencia a los errores. A esto se suma Reed-Solomon como la técnica de corrección de errores orientada a reforzar la comunicación frente al ruido mediante la detección y corrección de fallos [15].

De este modo, implementar las modulaciones no solo cumple con el análisis técnico que se busca, sino que además sienta la base para construir una guía práctica que facilite el aprendizaje de estas técnicas, aplicado entre los estudiantes de telecomunicaciones.

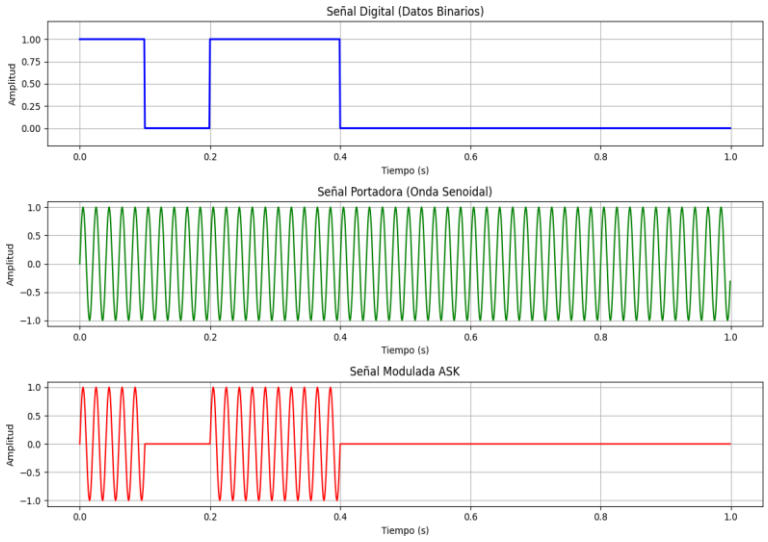
2.2.4 Técnicas de modulación digital

ASK (Amplitude Shift Keying)

ASK o modulación por desplazamiento de amplitud es una de las técnicas de modulación digital más básicas: consiste en variar la amplitud de la señal portadora para codificar y transmitir los datos, mientras que la frecuencia y la fase permanecen constantes. Debido a su simplicidad de implementación, esta técnica se emplea en sistemas de radio, televisión y transmisión de datos digitales, aunque su principal limitación es una mayor sensibilidad al ruido en comparación con otros esquemas de modulación [16].

Como se observa en la **Figura 3**, la señal modulada ASK cambia de amplitud según el bit que se transmite en cada momento.

Figura 3. *Ejemplo de una señal con modulación ASK.*



Nota. Elaboración propia.

Fórmulas de modulación ASK

De igual manera, para facilitar su interpretación desde el punto de vista matemático en la **ecuación 1**, se muestra la expresión general correspondiente a este tipo de modulaciones donde usualmente una de las amplitudes de las portadoras se fija en "0", mientras que la otra puede tomar distintos valores.

$$V_{AM} = [1 + V_m(t)] * \left[\frac{A}{2} * \cos(2\pi * f_c t)\right]$$

Ecuación 1. *ASK general [16].*

Donde:

- $\boldsymbol{V_{AM}(t)}$**:** Señal modulada ASK.
- $\boldsymbol{V_m(t)}$**:** Señal digital a modular que toma dos valores: +1 si el bit enviado es "1", y -1 si el bit enviado es "0".
- $\boldsymbol{A}$**:** Amplitud de la señal portadora.
- $\boldsymbol{f_c}$**:** Frecuencia de la señal portadora.

Se observa que cuando $V_m(t) = 1$, la ecuación adopta la siguiente forma:

$$V_{AM}(t) = [A * \cos(2\pi * f_c t)]$$

Ecuación 2. *ASK con Vm(t) =1 [16].*

Por otro lado, en el caso opuesto $V_m(t) = -1$, la ecuación toma la siguiente forma:

$$V_{AM}(t) = 0$$

Ecuación 3. *ASK con Vm(t) =-1 [16].*

Por otra parte, el ancho de banda se puede determinar mediante la siguiente ecuación, considerando que T_m representa la frecuencia máxima.

$$BW = 2 * F_m$$

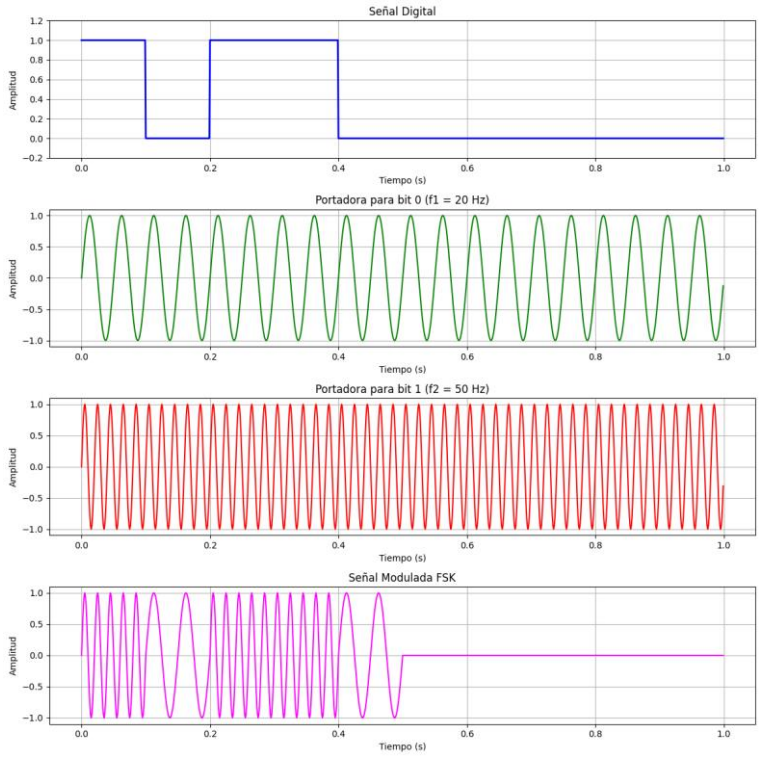
Ecuación 4. *Ancho de banda ASK [17].*

FSK (Frequency Shift Keying)

FSK o modulación por desplazamiento de frecuencia representa los bits 0 y 1 mediante frecuencias distintas de la señal portadora, mientras que la amplitud permanece constante. Al variar la frecuencia según los datos binarios, esta técnica logra una transmisión eficiente y presenta una mayor inmunidad al ruido en comparación con ASK, lo que la convierte en una opción adecuada para entornos con interferencias, como las redes de telecomunicaciones inalámbricas [18].

Como se observa en la **Figura 4**, aquí lo que cambia con el bit transmitido ya no es la amplitud, sino la frecuencia de la señal modulada.

Figura 4. *Ejemplo de una señal con modulación FSK.*



Nota. Elaboración propia.

Fórmula de modulación FSK

De manera similar, para facilitar su interpretación desde el punto de vista matemático en la **ecuación 5**, se presenta la expresión general correspondiente a la modulación FSK donde la información se codifica en la variación de la frecuencia de la señal portadora, la cual adopta distintos valores según el estado lógico de la señal transmitida.

$$V_{FSk}(t) = V_c * \cos\left[\left(w_c + \frac{V_m(t) * \Delta w}{2}\right) * t\right]$$

Ecuación 5. *FSK general [18].*

Donde:

- $\boldsymbol{V_{FSK}(t)}$**:** Señal modulada FSK.
- $\boldsymbol{V_c}$**:** Amplitud de portadora no modulada pico.
- $\boldsymbol{W_c}$**:** Frecuencia de la portadora (en radianes).
- $\boldsymbol{V_m(t)}$**:** Señal de entrada digital.
- $\boldsymbol{\Delta w}$**:** Cambio de frecuencia de la portadora.

Estas ecuaciones describen el comportamiento de la señal modulada en la que la

información digital se transmite variando la frecuencia de una señal portadora, como se observa en las **ecuaciones 6, 7** y **8**.

$$V_{FSK}(t) = V_c * \cos\left[2 * \pi\left(f_c + \frac{\Delta f}{2}\right) * t\right]$$

Ecuación 6. *FSK para "1" lógico [18].*

$$V_{FSK}(t) = V_c * \cos\left[2 * \pi\left(f_c - \frac{\Delta f}{2}\right) * t\right]$$

Ecuación 7. *FSK para "0" lógico [18].*

El ancho de banda de una señal FSK se calcula de la siguiente forma:

$$BW = 2 * (f_b + \Delta f)$$

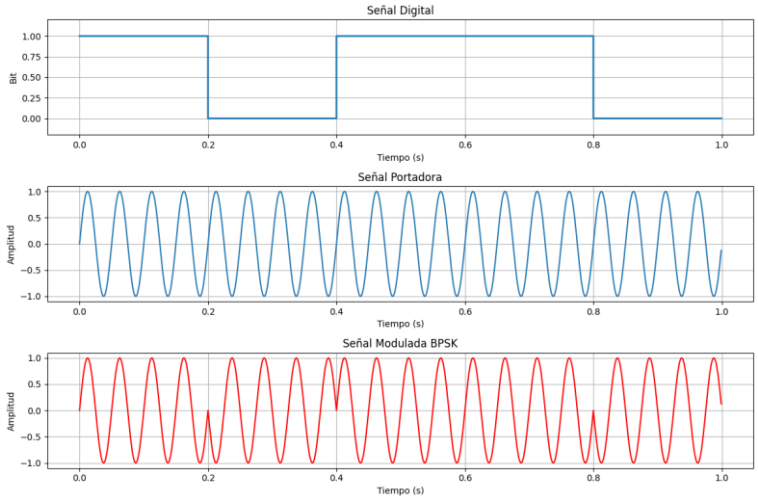
Ecuación 8. *Ancho de banda FSK [18].*

BPSK (Binary Phase Shift Keying)

BPSK transmite la información utilizando una sola frecuencia portadora y dos estados de fase, uno para el bit 1 y otro para el bit 0. Cuando la señal de entrada cambia de un bit a otro, la fase de la portadora se invierte, generando un desfase de 180° entre ambos estados. Este cambio se logra mediante el modulador balanceado, un dispositivo que funciona como conmutador para invertir la fase de la portadora. Gracias a esta característica, BPSK presenta una alta resistencia al ruido y a las interferencias, lo que la convierte en una de las modulaciones más robustas y ampliamente utilizadas en sistemas de comunicaciones digitales [19].

Como se observa en la **Figura 5**, en BPSK es la fase de la portadora la que cambia según el bit que se transmite.

Figura 5. *Ejemplo de una señal con modulación BPSK.*



Nota. Elaboración propia.

Fórmula de modulación BPSK

A partir de las formas de onda propias de este tipo de modulación BPSK, se presentan las ecuaciones matemáticas correspondientes, donde se define una expresión para el caso en que la entrada binaria es un "0" lógico y otra distinta para cuando se tiene un "1" lógico.

$$V_{BPSK}(t) = \begin{cases} A\cos(2\pi * f_t + \varphi_1) \rightarrow "0" \\ A\cos(2\pi * f_t + \varphi_2) \rightarrow "1" \end{cases}$$

Ecuación 9. *BPSK general [19].*

Donde:

$\boldsymbol{V_{BPSK}(t)}$: Es la señal modulada en función del tiempo.

$\boldsymbol{A}$: Es la amplitud de la señal portadora.

$\boldsymbol{2\pi * f_t}$: Representa la frecuencia angular.

$\boldsymbol{\varphi}$**:** Fases de la señal.

El ancho de banda de una señal BPSK será calculado como:

$$BW = 2 * f_b$$

Ecuación 10. *Ancho de banda BPSK [19].*

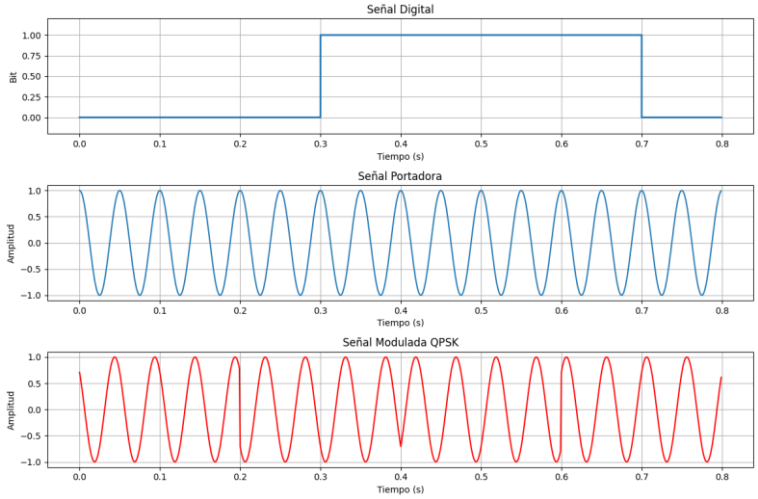
QPSK (Quadrature Phase Shift Keying)

QPSK es una modulación digital angular que, a diferencia de ASK, mantiene fija

la amplitud de la portadora. Al ser M-aria con M = 4, recurre a cuatro estados de fase para representar la información, lo que le permite meter dos bits en cada símbolo usando una sola frecuencia portadora. Esa característica la hace más eficiente espectralmente que BPSK, aunque sigue expuesta al ruido y a las interferencias del canal. Su buen desempeño en la transmisión de datos explica por qué se usa tanto en sistemas de comunicaciones digitales, sobre todo en enlaces de radiodifusión y comunicaciones satelitales [19].

Como se observa en la **Figura 6**, también en QPSK es la fase la que cambia según el bit transmitido.

Figura 6. *Ejemplo de una señal con modulación QPSK.*



Nota. Elaboración propia.

Fórmula de modulación QPSK

En términos generales, la modulación QPSK se deriva del esquema BPSK al incorporar desplazamientos de fase de 90°, lo que permite representar dos bits por cada símbolo transmitido.

$$V_{QPSK}(t) = A * b_p(t) * \cos(w_c t) + A * b_i(t) * \sin(w_c t)$$

Ecuación 11. *QPSK general [19].*

Donde:

A: Amplitud de la señal.

w_c: Frecuencia de la portadora en radianes

b_p: Bit par

b_i: Bit Impar

Por otra parte, para determinar el ancho de banda, este resulta ser aproximadamente la mitad de la velocidad de transmisión original, ya que, durante el proceso, la señal se divide en dos componentes con el fin de generar los dos bits necesarios para asignar los distintos desfases de la modulación.

$$BW = \frac{f_b}{N} = \frac{f_b}{2}$$

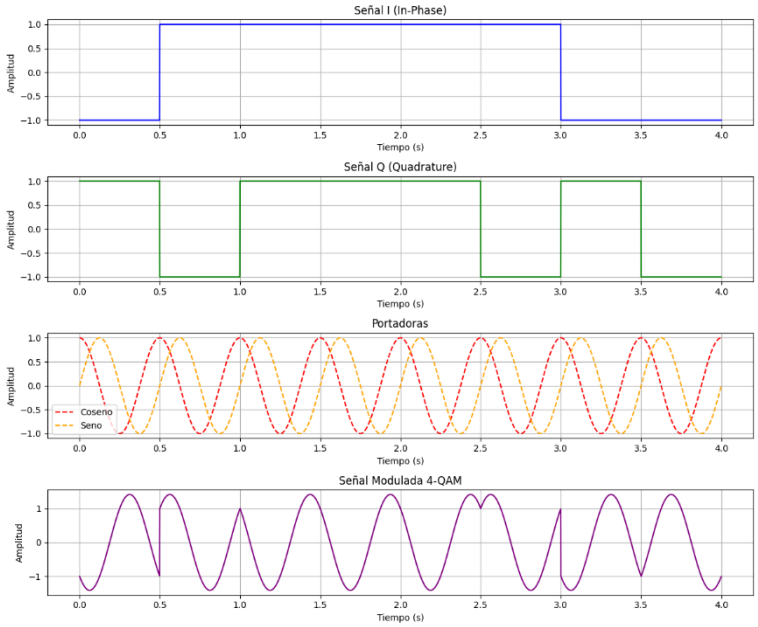
Ecuación 12. *Ancho de banda QPSK [19].*

QAM (Quadrature Amplitude Modulation)

QAM combina la modulación de amplitud (ASK) con la modulación de fase (PSK) para transmitir un mayor número de bits por símbolo sin necesidad de ampliar el ancho de banda del canal. Según el orden de la modulación (por ejemplo, 16-QAM o 64-QAM), cada símbolo puede representar cuatro, seis o más bits, lo que incrementa notablemente la eficiencia espectral respecto a modulaciones como BPSK y QPSK. Sin embargo, esta ganancia en eficiencia tiene un costo: al existir una menor separación entre los puntos del diagrama de constelación, QAM resulta más sensible al ruido y a las distorsiones del canal. Debido a este equilibrio entre eficiencia espectral y robustez, es considerada la modulación de referencia en las comunicaciones digitales de alta velocidad [8].

Como se observa en la **Figura 7**, aquí conviven los componentes I y Q, las portadoras, la señal ya modulada y el diagrama de constelación.

Figura 7. *Ejemplo de una señal con modulación QAM.*



Nota. Elaboración propia.

Fórmula de modulación QAM

La fórmula fundamental de la modulación QAM combina dos portadoras desfasadas (seno y coseno) moduladas en amplitud.

$$V_{QAM} = I(t)\cos(2\pi * f_c t) - Q(t)\sin(2\pi * f_c t)$$

Ecuación 13. *QAM general [20].*

Donde:

$\boldsymbol{V_{QAM}}$: Señal QAM.

$\boldsymbol{I(t)}$: Señal "en fase" (In-phase), portadora de los datos de amplitud.

$\boldsymbol{Q(t)}$: Señal "en cuadratura" (Quadrature), portadora de los datos de fase.

$\boldsymbol{f_c}$: Frecuencia de la portadora

El ancho de banda de una señal QAM será calculado como:

$$BW = R_s = \frac{R_b}{\log_2(M)}$$

Ecuación 14. *Ancho de banda QAM [21].*

2.3 Características del canal

El canal de transmisión es uno de los componentes más determinantes de cualquier sistema de comunicación digital, ya que puede degradar la señal enviada hasta comprometer seriamente la calidad y la confiabilidad de todo el sistema. Este trabajo se apoya en dos modelos para representar sus condiciones adversas: el ruido gaussiano aditivo (AWGN) y el desvanecimiento Rayleigh, que en conjunto permiten estudiar cómo responde el sistema ante perturbaciones típicas del mundo real, como el ruido térmico o la pérdida de señal por trayectos múltiples [12].

El ruido blanco gaussiano aditivo es ampliamente utilizado por ser un modelo simple y, al mismo tiempo, muy relevante en la práctica. Este modelo parte de suponer que la señal que llega al receptor presenta un ruido aleatorio con distribución gaussiana de media cero y varianza constante, lo cual constituye una manera razonable de representar fenómenos físicos como las fluctuaciones térmicas en los componentes electrónicos, además de resultar útil para establecer una línea base de rendimiento. En simulaciones digitales programadas en Python, resulta sencillo generarlo, ya que basta con funciones estadísticas, lo cual permite poner a prueba el sistema en distintas relaciones señal-ruido (SNR). Esta evaluación permite calcular la tasa de error de bits (BER), que en definitiva indica qué tan robusto es el sistema ante las perturbaciones.

El canal de Rayleigh, por otro lado, se utiliza para modelar entornos en los que no existe una línea de visión directa entre el transmisor y el receptor, como suele ocurrir en un escenario urbano repleto de obstáculos. En estos casos, la señal llega por múltiples caminos, producto de la reflexión, la difracción y la dispersión, lo que hace que su amplitud fluctúe rápidamente en lo que se conoce como desvanecimiento. El modelo supone que las partes real e imaginaria del canal son variables aleatorias gaussianas independientes, con lo que la amplitud de la señal recibida presenta una distribución Rayleigh. Para simular este tipo de canal en Python, se puede recurrir a la suma de sinusoides o a bibliotecas especializadas como CommPy, que incluyen modelos realistas de desvanecimiento Rayleigh [22].

En este trabajo se implementan ambos modelos de forma conjunta para simular escenarios más cercanos a la realidad: la señal ya modulada digitalmente se transmite a través de un canal sintético que combina el desvanecimiento Rayleigh con el ruido gaussiano. Esto permite evaluar el comportamiento del sistema ante distintas condiciones

de propagación mediante el seguimiento de la variación de la BER y la medición de la eficiencia espectral, ambas fundamentales para validar la robustez del diseño propuesto [23].

2.3.1 Canal AWGN

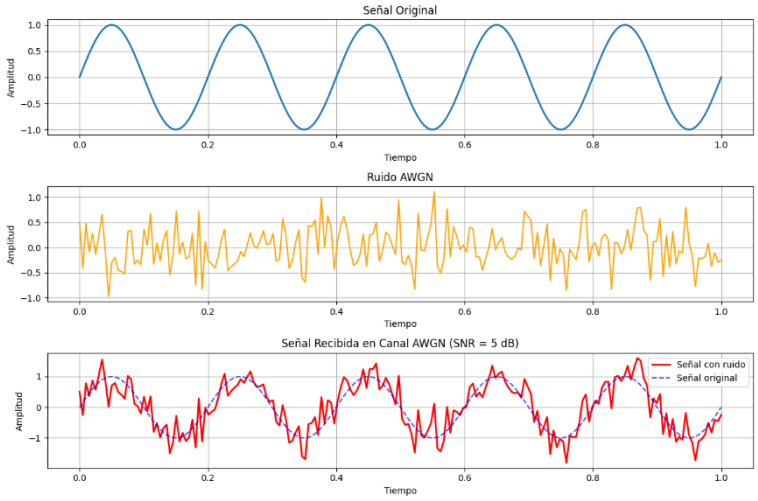
El ruido blanco gaussiano aditivo o AWGN es uno de los modelos de ruido más utilizados en los sistemas de comunicación y procesamiento de señales. Su característica principal es la distribución de probabilidad gaussiana, por la cual la mayoría de las muestras de ruido se sitúan cerca de la media, mientras que las desviaciones grandes resultan comparativamente improbables.

Este ruido afecta directamente la relación señal-ruido (SNR), indicador que determina la calidad de una transmisión. Si aparece ruido gaussiano, la SNR empeora y, con él, la fiabilidad del sistema. En las comunicaciones inalámbricas, donde las señales tienden a ser débiles y se ven afectadas por interferencias diversas, el ruido gaussiano puede generar varios efectos negativos sobre la transmisión.

El más frecuente es el incremento de los errores al decodificar los bits transmitidos, lo que compromete la integridad de la información. Otro efecto para considerar es la pérdida de sincronismo en los sistemas digitales, que puede desordenar la secuencia de datos y dar lugar a errores de comunicación. Desde el punto de vista teórico, el ruido gaussiano también limita la capacidad máxima del canal de transmisión, reduciendo la cantidad de información que puede transmitirse de forma eficiente [21].

Como se observa en la **Figura 8**, la señal original (sinusoidal) se ve afectada al pasar por un canal con ruido AWGN. El gráfico muestra cómo el ruido altera la forma de onda, evidenciando la diferencia entre la señal transmitida y la señal recibida con una relación señal-ruido (SNR) de 5 dB. Esto ilustra el impacto del ruido en la transmisión y la necesidad de técnicas de detección y corrección de errores en comunicaciones digitales.

Figura 8. *Efecto del ruido AWGN en una señal sinusoidal.*



Nota. Elaboración propia.

Fórmula del canal AWGN

De igual manera, para facilitar su interpretación desde el punto de vista matemático en la **ecuación 15**, se muestra la expresión general correspondiente al canal AWGN, el cual modela la incorporación de ruido aleatorio a la señal transmitida durante el proceso de comunicación.

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

Ecuación 15. *Representación matemática del canal AWGN [21].*

Donde:

- $r(t)$ representa la señal recibida.
- $s(t)$ corresponde a la señal original transmitida.
- $n(t)$ representa el ruido gaussiano aditivo incorporado al canal.

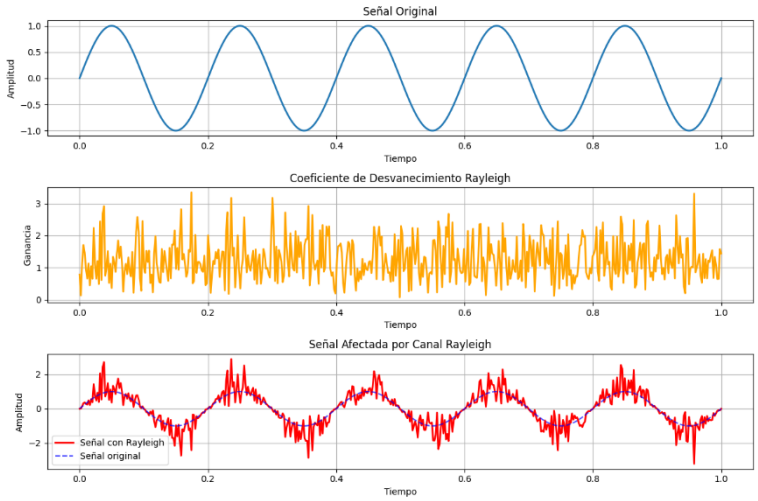
2.3.2 Canal con desvanecimiento Rayleigh

El desvanecimiento Rayleigh se representa estadísticamente asumiendo que la señal recibida es la suma de numerosos componentes aleatorios generados por la reflexión y la dispersión de la señal al chocar con obstáculos, como edificaciones, vegetación o terrenos irregulares. La intensidad de la señal resultante sigue una función de distribución de Rayleigh, lo que hace más probables los valores bajos, aunque los valores altos también son posibles. Este modelo resulta especialmente adecuado en entornos sin una trayectoria dominante entre emisor y receptor, como zonas urbanas densamente edificadas o espacios

cerrados [24].

Como se observa en la **Figura 9**, la señal original se ve afectada al atravesar un canal con desvanecimiento Rayleigh. El gráfico muestra cómo los coeficientes de ganancia del canal varían en el tiempo, provocando fluctuaciones en la amplitud de la señal recibida. Esta comparación evidencia la distorsión introducida por el desvanecimiento, un fenómeno característico en comunicaciones inalámbricas.

Figura 9. *Distorsión de señales en presencia de desvanecimiento Rayleigh.*



Nota. Elaboración propia.

Fórmula del canal de desvanecimiento Rayleigh

Para representar matemáticamente el efecto del desvanecimiento Rayleigh en un sistema de comunicación inalámbrica, se emplea la siguiente expresión, en la cual la señal transmitida experimenta variaciones aleatorias de amplitud debido a la propagación multicamino, antes de verse afectada por el ruido del canal. El coeficiente de desvanecimiento Rayleigh se obtiene mediante:

$$h_{mag} = \sqrt{h_I^2 + h_Q^2}$$

Ecuación 16. *Representación del desvanecimiento Rayleigh [21].*

donde h_I y h_Q son variables aleatorias gaussianas independientes asociadas a las componentes en fase y cuadratura del canal.

2.3.3 Factores que degradan la señal

En telecomunicaciones, los factores degradantes de la señal son aquellos elementos externos o inherentes al canal que alteran la calidad y estabilidad de las señales transmitidas. Entre los principales están el ruido, las interferencias electromagnéticas, los desvanecimientos de señal y otras anomalías que pueden reducir la eficiencia del sistema de comunicación.

Un ejemplo representativo de estos factores es la interferencia electromagnética, que puede causar daños graves a la integridad de las señales, aumentar la tasa de errores en la transmisión e, incluso, en casos extremos, interrumpir completamente el servicio. A fin de reducir este efecto, se utilizan dispositivos como los receptores de prueba EMI, que permiten detectar y reducir la presencia de interferencias, aportando estabilidad y rendimiento a las redes de telecomunicaciones.

2.4 Técnicas de detección y corrección de errores

2.4.1 Fundamentos de detección y corrección de errores

Con la comunicación digital moderna encaminándose hacia el 6G y con el internet de las cosas (IoT) en pleno auge, la detección y corrección de errores dejó de ser opcional: es lo que garantiza que las transmisiones sean confiables. Los métodos empleados en 3G, 4G y 5G —códigos turbo, LDPC y polares— contribuyeron a reducir la tasa de error y a mejorar la eficiencia, pero resultan insuficientes frente a las exigencias del 6G. Por tanto, la corrección de errores resulta fundamental para que el sistema gane en fiabilidad. Entre las técnicas actuales se incluyen el LDPC (código de baja densidad de comprobación de paridad), los códigos turbo y los códigos concatenados como Reed-Solomon, muy empleados en las comunicaciones inalámbricas y satelitales [25].

En este proyecto se emplea concretamente un código Reed-Solomon implementado mediante bibliotecas especializadas, con el fin de determinar cuánto mejora la tasa de error de bits (BER) frente a distintos niveles de SNR y condiciones de desvanecimiento Rayleigh. De este modo, los estudiantes pueden comprobar directamente cómo las decisiones de corrección de errores impactan en la robustez del sistema.

2.4.2 Código Reed-Solomon

El código Reed-Solomon (RS) es un método de corrección de errores por bloque muy

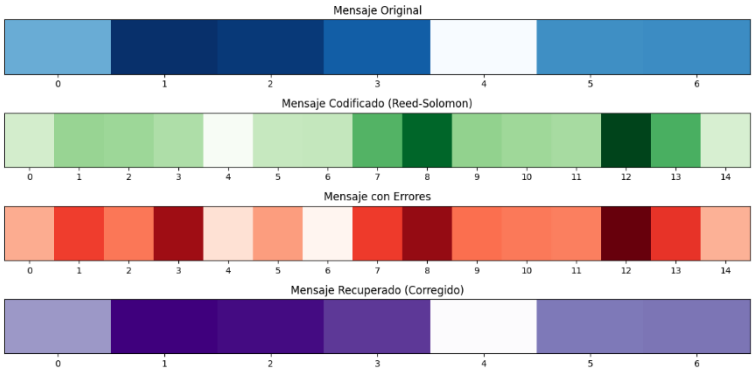
empleado en sistemas de comunicaciones digitales y en dispositivos de almacenamiento de datos, principalmente por su capacidad para corregir errores en ráfaga.

La codificación comienza identificando los datos como un polinomio sobre un campo finito, a partir del cual se generan símbolos redundantes adicionales que, en un paso posterior, permiten detectar y corregir errores en la transmisión. En el receptor, la decodificación consiste en calcular síndromes para identificar los errores y, posteriormente, localizarlos para corregirlos mediante algoritmos algebraicos.

Esta capacidad de corregir errores agrupados resulta particularmente valiosa en la transmisión de imágenes y audio, ya que ambos tipos de archivo se almacenan y procesan en bytes: cualquier alteración durante la transmisión suele afectar a varios bits dentro del mismo bloque de información, precisamente el tipo de error que Reed-Solomon corrige con mayor eficacia. Por esta misma razón, el código también es ampliamente utilizado en el almacenamiento óptico (CD y DVD) y en comunicaciones digitales en entornos ruidosos, ya que devuelve una recuperación exacta de los datos transmitidos, mejora la confiabilidad del sistema y evita tener que retransmitir constantemente la información perdida [26].

La **Figura 10** muestra el proceso de codificación y corrección de errores mediante el código Reed-Solomon. Se observa el mensaje original, seguido del mensaje codificado con redundancia. Posteriormente, se introducen errores que alteran la información transmitida, y finalmente, el decodificador Reed-Solomon logra recuperar completamente el mensaje original, evidenciando su alta capacidad de corrección de errores.

Figura 10. *Corrección de errores mediante código Reed-Solomon.*



Nota. Elaboración propia.

Fórmula del código de Reed-Solomon

Fórmula matemática para hallar la capacidad de corrección en Reed-Solomon:

$$t = \frac{n-k}{2}$$

Ecuación 17. *Código Reed-Solomon. [27]*

Donde:

n: Número total de símbolos en el bloque (datos + paridad).

k: Número de símbolos de datos.

n - k: Número de símbolos de paridad agregados.

t: Número máximo de símbolos erróneos que el código puede corregir.

2.5 Procesamiento en el receptor

2.5.1 Demodulación

Demodular consiste en recuperar la información original a partir de la señal modulada que llega al receptor, es decir, en detectar y extraer los parámetros que la modulación alteró: amplitud, frecuencia o fase, según el esquema empleado. Este proceso puede realizarse de forma coherente o no coherente. La demodulación coherente resulta más precisa, aunque también más compleja, ya que requiere sincronizarse exactamente con la portadora original. Investigaciones recientes exploran incluso el uso de inteligencia artificial con el fin de mejorar la precisión de la demodulación en entornos ruidosos, y han mostrado resultados eficaces al recuperar señales digitales bajo ruido blanco gaussiano aditivo [28].

2.5.2 Decodificación y reconstrucción de la señal

Decodificar no es solo traducir símbolos a bits, sino que también implica sincronizar la portadora, recuperar la fase y la frecuencia e igualar los efectos del canal, con el fin de reconstruir la señal de forma confiable. Las investigaciones más recientes en el campo de las comunicaciones ópticas señalan la importancia de emplear un procesamiento digital avanzado en el receptor, capaz de recuperar la sincronía mediante la compensación de frecuencia y portadora, así como la ecualización por polarización. Con ello se busca que la señal decodificada sea un reflejo fiel de lo transmitido, incluso en condiciones adversas [29].

2.6 Métricas de evaluación del desempeño

2.6.1 BER (Bit Error Rate)

La tasa de error de bit (BER) mide el número de errores de bits que se producen en comparación con el total de bits transmitidos, y sirve como un indicador esencial de la fiabilidad general del sistema entre la entrada y la salida. Esta métrica permite evaluar qué tan cerca se encuentra la transmisión real de una transmisión ideal, en la cual la señal recibida sería igual a la señal enviada. Durante la transmisión se aplican buenas prácticas de diseño para reducir los errores y mantener la integridad de los datos; sin embargo, todo sistema de transmisión presenta imperfecciones. El ruido es uno de los factores principales que afectan la calidad de la señal, ya que degrada la relación señal-ruido. Este ruido puede originarse en distintas etapas del diseño del circuito, por lo que su control resulta fundamental para asegurar una transmisión fiable [30].

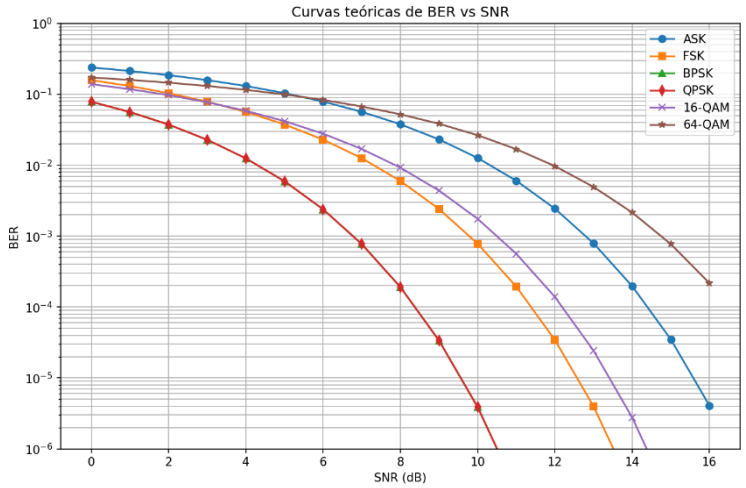
Curva teórica del BER

Como se observa en la **Figura 11**, se grafican las curvas teóricas de BER en función de SNR (0 a 16 dB) para las modulaciones investigadas. Todas las curvas muestran la forma de "cascada" típica de un canal AWGN: el BER cae de forma abrupta al aumentar Eb/N0.

BPSK y QPSK presentan el mejor desempeño y curvas idénticas, seguidas de FSK y 16-QAM en una posición intermedia, mientras que ASK y 64-QAM muestran el peor desempeño, requiriendo mayor SNR para alcanzar el mismo BER.

En conjunto, la figura evidencia el compromiso entre eficiencia espectral y eficiencia de potencia: las modulaciones de orden superior (16-QAM, 64-QAM) transmiten más bits por símbolo, pero necesitan mayor SNR para mantener una baja tasa de error frente a esquemas binarios como BPSK o QPSK.

Figura 11. *Curva teórica BER vs SNR de 16 dB.*



Nota. Elaboración Propia, investigado de [31]

Como se observa en la **Tabla 1**, el BER teórico decrece exponencialmente con el aumento de SNR, reflejando la mejora en la confiabilidad de la transmisión. Las modulaciones de menor orden, como BPSK y QPSK, alcanzan valores cercanos a cero con menor SNR, evidenciando su robustez frente al ruido. En contraste, modulaciones de orden superior, como 16-QAM y 64-QAM, requieren mayor SNR para lograr tasas de error similares, lo que refleja su mayor eficiencia espectral, pero también su sensibilidad a degradaciones del canal. Este análisis confirma la relación fundamental entre complejidad de modulación y tolerancia al ruido, siendo el BER teórico una métrica clave para el dimensionamiento de sistemas digitales de comunicación.

Tabla 1. *BER teórico vs SNR para diferentes modulaciones digitales.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.239750	0.158655	0.078650	0.078650	0.139160	0.172953
1	0.213776	0.130927	0.056282	0.056282	0.118346	0.160031
2	0.186681	0.104029	0.037506	0.037506	0.097559	0.146124
3	0.158942	0.078896	0.022878	0.022878	0.077415	0.131317
4	0.131210	0.056495	0.012501	0.012501	0.058618	0.115764
5	0.104298	0.037679	0.005954	0.005954	0.041892	0.099704
6	0.079142	0.023007	0.002388	0.002388	0.027871	0.083473
7	0.056709	0.012587	0.000773	0.000773	0.016967	0.067505
8	0.037852	0.006004	0.000191	0.000191	0.009247	0.052320

9	0.023136	0.002413	0.000034	0.000034	0.004390	0.038483
10	0.012674	0.000783	0.000004	0.000004	0.001754	0.026533
11	0.006055	0.000194	0.000000	0.000000	0.000565	0.016884
12	0.002439	0.000034	0.000000	0.000000	0.000139	0.009724
13	0.000793	0.000004	0.000000	0.000000	0.000024	0.004946
14	0.000197	0.000000	0.000000	0.000000	0.000003	0.002154
15	0.000035	0.000000	0.000000	0.000000	0.000000	0.000772
16	0.000004	0.000000	0.000000	0.000000	0.000000	0.000217

Nota. Elaboración Propia.

Fórmula matemática del BER

El BER se calcula de manera empírica, comparando directamente la secuencia transmitida con la recibida. Así, el cálculo se reduce a contar los errores y dividirlos entre el total de bits transmitidos, lo que ofrece una medida directa y sencilla del rendimiento observado.

La fórmula matemática del BER es:

$$BER = \frac{N_{errores}}{N_{total}}$$

Ecuación 18. *Tasa de error de bit [30].*

Donde:

- $\boldsymbol{N_{errores}}$: número de bits recibidos incorrectamente.
- $\boldsymbol{N_{total}}$: número total de bits transmitidos.

2.6.2 MER (Modulation Error Rate)

La tasa de error de modulación (MER), expresada en decibelios, mide la calidad de una señal digital comparando la potencia de la señal ideal recibida con la potencia del error introducido por el ruido y las distorsiones del canal. Cuanto mayor es el MER, menor es el error presente en la constelación recibida y, por lo tanto, mayor es la calidad de la transmisión.

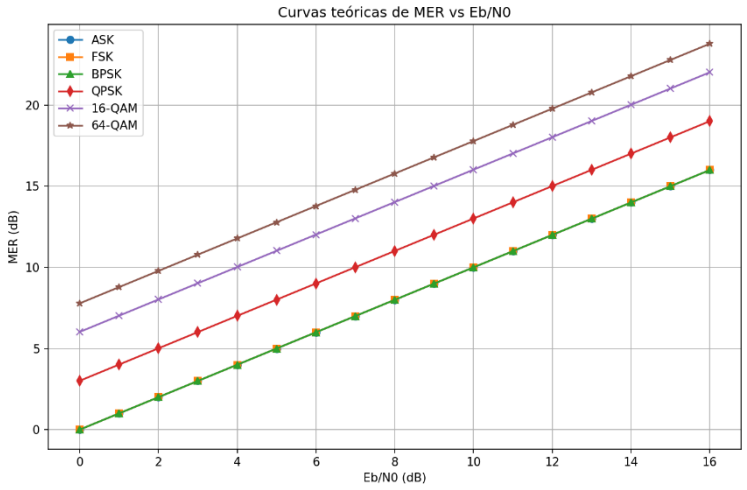
Curva teórica del MER

Como se observa en la **Figura 12**, se grafican las curvas teóricas de MER en función de Eb/N0. A diferencia de las curvas de BER, aquí todas las relaciones son crecientes y lineales: el MER aumenta de forma directamente proporcional a la SNR, sin el comportamiento de "cascada" observado antes.

ASK, FSK y BPSK muestran exactamente la misma curva con MER de 0 dB en SNR = 0 dB, ya que las tres transportan 1 bit por símbolo. QPSK aparece desplazada 3 dB por encima de ellas al transportar 2 bits por símbolo; 16-QAM se ubica aún más arriba (6 dB de desplazamiento), y 64-QAM presenta el MER más alto de todas (cerca de 7.8 dB de desplazamiento).

En conjunto, la figura confirma que el MER teórico depende directamente del número de bits por símbolo de cada modulación: para una misma SNR, las modulaciones de orden superior (16-QAM, 64-QAM) alcanzan un MER mayor, ya que concentran más energía por símbolo al transmitir más información en cada uno.

Figura 12. *Curva teórica MER vs SNR de 16 dB.*



Nota. Elaboración Propia, investigado de [32].

Como se observa en la **Tabla 2**, el MER teórico aumenta linealmente con la SNR, indicando una mejora continua en la relación señal-ruido y calidad de modulación. Las modulaciones de menor orden, como ASK, FSK y BPSK, presentan una pendiente de 1 dB por dB de SNR, reflejando una correspondencia directa entre energía y precisión. En cambio, modulaciones de orden superior, como QPSK, 16-

QAM y 64-QAM, muestran desplazamientos positivos en sus curvas, lo que evidencia su mayor eficiencia espectral y capacidad para mantener alta calidad de señal con el mismo nivel de ruido.

Este comportamiento confirma la relación directa entre MER, SNR y orden de modulación, siendo el MER una métrica clave para evaluar la linealidad, potencia efectiva y desempeño del transmisor en sistemas digitales de comunicación.

Tabla 2. *MER teórico vs SNR para diferentes modulaciones digitales.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.00	0.00	0.00	3.01	6.02	7.78
1	1.00	1.00	1.00	4.01	7.02	8.78
2	2.00	2.00	2.00	5.01	8.02	9.78
3	3.00	3.00	3.00	6.01	9.02	10.78
4	4.00	4.00	4.00	7.01	10.02	11.78
5	5.00	5.00	5.00	8.01	11.02	12.78
6	6.00	6.00	6.00	9.01	12.02	13.78
7	7.00	7.00	7.00	10.01	13.02	14.78
8	8.00	8.00	8.00	11.01	14.02	15.78
9	9.00	9.00	9.00	12.01	15.02	16.78
10	10.00	10.00	10.00	13.01	16.02	17.78
11	11.00	11.00	11.00	14.01	17.02	18.78
12	12.00	12.00	12.00	15.01	18.02	19.78
13	13.00	13.00	13.00	16.01	19.02	20.78
14	14.00	14.00	14.00	17.01	20.02	21.78
15	15.00	15.00	15.00	18.01	21.02	22.78
16	16.00	16.00	16.00	19.01	22.02	23.78

Nota. Elaboración Propia.

Fórmula del MER

En la ecuación de MER se establece una medida de calidad de la señal modulada digitalmente. El MER compara la potencia promedio de los símbolos transmitidos en sus posiciones ideales contra la potencia promedio del error, es decir, la desviación de los símbolos recibidos respecto a su ubicación teórica en el diagrama de constelación. Al

expresarse en decibelios, esta métrica permite evaluar de manera cuantitativa la fidelidad de la modulación y la robustez frente al ruido y las distorsiones del canal.

Fórmula teórica de MER:

$$MER(dB) = 10\log_{10}\left(\frac{P_{señal\_ideal}}{P_{error}}\right)$$

Ecuación 19. *Relación señal-error en modulación digital (MER) [33].*

Donde:

- $\boldsymbol{P_{señal\_ideal}}$: potencia promedio de los símbolos transmitidos en su posición teórica.
- $\boldsymbol{P_{error}}$: potencia promedio del vector de error (EVM) que mide la desviación entre símbolo recibido y símbolo ideal.

2.6.3 EVM (Error Vector Magnitude)

El EVM mide la distancia entre cada símbolo recibido y su símbolo ideal correspondiente en el plano IQ, promediada y expresada normalmente en porcentaje o en dB. Es un indicador clave de la fidelidad de la modulación y se especifica en los estándares de comunicaciones (Wi-Fi, LTE, 5G, satélite) [34].

Curva teórica del EVM

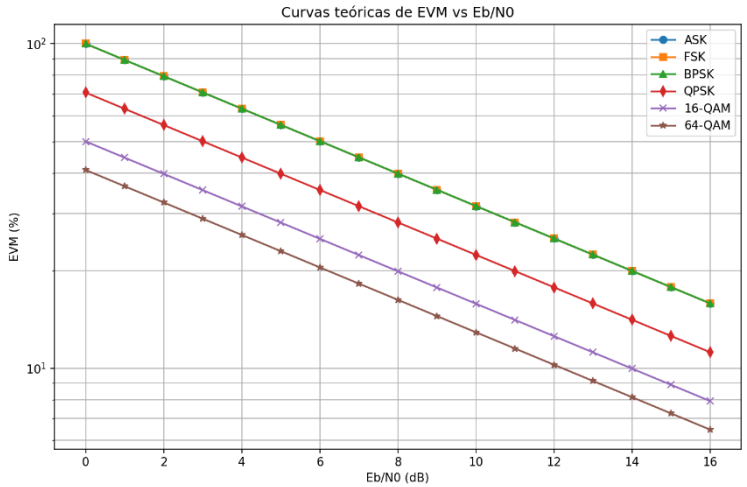
Como se observa en la **Figura 13**, se grafican las curvas teóricas de EVM en función de SNR con el eje vertical en escala logarítmica. A diferencia de las curvas de MER, aquí todas las relaciones son decrecientes. El EVM disminuye de manera monótona a medida que aumenta SNR, ya que un mayor nivel de energía por bit respecto al ruido reduce el error de vector de la señal recibida.

ASK, FSK y BPSK muestran exactamente la misma curva partiendo de un EVM del 100% en SNR= 0 dB al transportar 1 bit por símbolo. QPSK se ubica por debajo de ellas con un EVM menor para la misma SNR gracias a sus 2 bits por símbolo. 16-QAM presenta un EVM todavía más bajo, y 64-QAM muestra el EVM más bajo de todas las modulaciones, siendo la curva ubicada en la parte inferior de la gráfica en todo el rango.

En conjunto, la figura confirma que el EVM teórico es inversamente proporcional a la raíz cuadrada de la energía por símbolo: para una misma SNR, las modulaciones de orden superior (16-QAM, 64-QAM) presentan un EVM menor, ya que concentran más

energía por símbolo al transmitir más bits en cada uno, siendo así el comportamiento recíproco al observado en las curvas de MER.

Figura 13. *Curva teórica EVM vs SNR de 16 dB.*



Nota. Elaboración Propia, investigado de [32].

Como se observa en la **Tabla 3**, el EVM teórico disminuye de forma inversamente proporcional a la SNR, mostrando una mejora progresiva en la calidad de la señal conforme aumenta la relación señal-ruido. Las modulaciones de menor orden como BPSK y QPSK presentan valores de EVM más altos en bajas SNR, pero alcanzan rápidamente niveles inferiores al 15 % a partir de 10 dB, indicando alta robustez frente al ruido. En contraste, modulaciones de orden superior como 16-QAM y 64-QAM exhiben menor EVM teórico para la misma SNR, reflejando su mayor eficiencia espectral y mejor aprovechamiento de potencia, aunque requieren mayor linealidad y precisión en el sistema.

Este análisis confirma la relación directa entre EVM, SNR y orden de modulación, siendo el EVM una métrica esencial para evaluar la calidad de constelación y desempeño del transmisor en sistemas digitales de comunicación.

Tabla 3. *EVM teórico vs SNR para diferentes modulaciones digitales.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	100.00	100.00	100.00	70.71	50.00	40.82
1	89.13	89.13	89.13	63.02	44.56	36.39
2	79.43	79.43	79.43	56.17	39.72	32.43

3	70.79	70.79	70.79	50.06	35.40	28.90
4	63.10	63.10	63.10	44.62	31.55	25.76
5	56.23	56.23	56.23	39.76	28.12	22.96
6	50.12	50.12	50.12	35.44	25.06	20.46
7	44.67	44.67	44.67	31.59	22.33	18.24
8	39.81	39.81	39.81	28.15	19.91	16.25
9	35.48	35.48	35.48	25.09	17.74	14.49
10	31.62	31.62	31.62	22.36	15.81	12.91
11	28.18	28.18	28.18	19.93	14.09	11.51
12	25.12	25.12	25.12	17.76	12.56	10.25
13	22.39	22.39	22.39	15.83	11.19	9.14
14	19.95	19.95	19.95	14.11	9.98	8.15
15	17.78	17.78	17.78	12.57	8.89	7.26
16	15.85	15.85	15.85	11.21	7.92	6.47

Nota. Elaboración Propia.

Fórmula del EVM

En la **Ecuación 20** de EVM se define una métrica clave para evaluar la calidad de un transmisor o receptor digital:

$$EVM = \sqrt{\frac{\sum_{j=1}^{N} | S_{real,j} - S_{ideal,j} |^2}{\sum_{j=1}^{N} | S_{ideal,j} |^2}}$$

Ecuación 20. *Magnitud del vector de error. [34]*

Donde:

- $S_{real,j}$: símbolo recibido o medido.
- $S_{ideal,j}$: símbolo ideal de referencia.
- N: número total de símbolos analizados.

Esto corresponde exactamente a la definición estándar de EVM, utilizada en telecomunicaciones para determinar la fidelidad de la modulación digital y la precisión con la que los símbolos recibidos se aproximan a sus posiciones ideales en el diagrama de constelación.

2.6.4 Eficiencia espectral

La eficiencia espectral en redes móviles hace referencia a la capacidad de una red inalámbrica para aprovechar de la mejor manera posible el espectro de frecuencias disponible al transmitir datos de forma eficiente. Con el rápido crecimiento de los servicios de datos móviles y la cantidad cada vez mayor de dispositivos conectados, optimizar este recurso se ha vuelto indispensable para que los proveedores de servicios gestionen el aumento del tráfico sin sacrificar la experiencia del usuario.

En términos concretos, la eficiencia espectral mide qué tan bien una red utiliza el espectro disponible para enviar información: se calcula dividiendo la cantidad de datos transmitidos entre el ancho de banda empleado, de modo que una eficiencia mayor significa transmitir más información en el mismo rango de frecuencias, es decir, una red con mayor capacidad y mejor desempeño en general [35].

Fórmula de la eficiencia espectral

La fórmula matemática para la eficiencia espectral es:

$$\eta = \frac{R_b}{BW}$$

Ecuación 21. *Eficiencia Espectral [36].*

Donde:

- R_b: tasa de bits transmitidos en bits por segundo.
- BW: ancho de banda del canal en hertz.

Esta es, en esencia, la definición estándar de eficiencia espectral que se usa para determinar el rendimiento espectral de un sistema de comunicación digital y comparar distintos esquemas de modulación en función de cuánto aprovechan el espectro.

2.7 Herramientas para simulación y análisis

2.7.1 NumPy (Python numérico)

NumPy es la biblioteca de Python para cálculo numérico y análisis de datos, especialmente útil cuando el volumen de información es grande. Su elemento central es el array, una estructura de datos que almacena y gestiona colecciones de elementos del mismo tipo en múltiples dimensiones junto con funciones muy eficientes para

manipularlos.

Frente a las listas tradicionales de Python, su gran ventaja es la velocidad: los arrays de NumPy pueden llegar a ser hasta 50 veces más rápidos en operaciones vectorizadas, lo que convierte a esta biblioteca en la opción más adecuada para trabajar con vectores y matrices de gran tamaño [37].

2.7.2 SciPy (Python científico)

SciPy, por su parte, es una biblioteca de código abierto derivada de NumPy que extiende Python hacia la computación científica y técnica con módulos especializados en optimización, integración, álgebra lineal, ecuaciones diferenciales y otras áreas para resolver problemas complejos de ingeniería y ciencia. Sus funciones se basan en algoritmos optimizados que aseguran un buen desempeño incluso con grandes volúmenes de datos y se integran fácilmente con otras bibliotecas como NumPy, lo que facilita su aprendizaje y adopción por parte de nuevos usuarios.

Una comunidad global de científicos y desarrolladores la mantiene actualizada, razón por la cual se ha consolidado como una de las herramientas más completas y fiables para el análisis científico y numérico en Python [38].

2.7.3 Matplotlib (biblioteca para graficar al estilo MATLAB)

Matplotlib es una biblioteca versátil para generar visualizaciones estáticas, interactivas y animadas en Python, cuya funcionalidad se ha ampliado con paquetes externos como Seaborn, HoloViews y ggplot. Está inspirada en una lógica similar a la de MATLAB, lo que le permite integrarse por completo con Python. Al ser gratuita y de código abierto, permite representar datos mediante diagramas de dispersión, histogramas, gráficos de barras, entre otros, con una sintaxis que requiere apenas unas pocas líneas de código [39].

2.7.4 Scripts de análisis y visualización de resultados

De manera ilustrativa, se genera una señal que será contaminada y filtrada usando herramientas de Python para el análisis y procesamiento de señales, apoyándose en las tres bibliotecas antes mencionadas: NumPy, SciPy y Matplotlib.

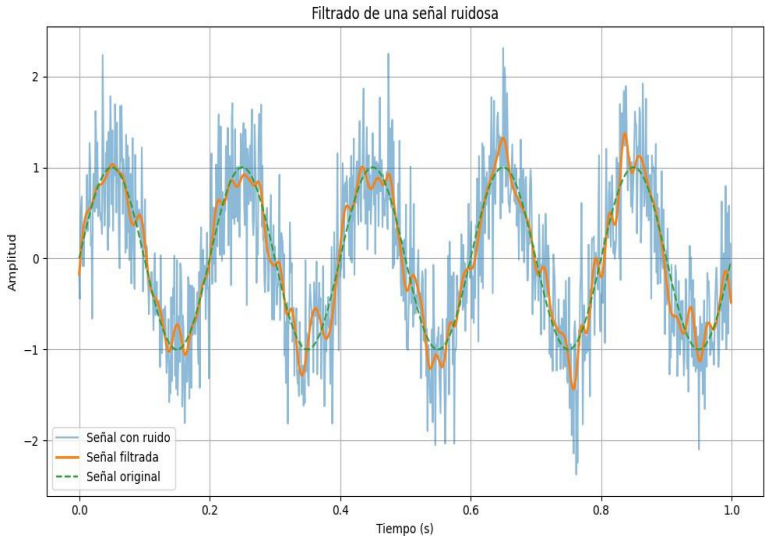
Primero, se genera con NumPy una señal senoidal a la que luego se le suma ruido blanco gaussiano para imitar una señal real afectada por interferencias.

Con SciPy se diseña y se aplica un filtro digital pasa-bajas, que atenúa el ruido de alta frecuencia.

Matplotlib permite visualizar el resultado: se grafican tres curvas —la señal original limpia, la contaminada con ruido y la ya filtrada— para que se aprecie a simple vista cómo el filtrado mejora la calidad de la señal.

Como se observa en la **Figura 14**, el filtro permite atenuar el ruido presente en la señal original, mejorando su forma de onda.

Figura 14. *Comparación entre una señal original, con ruido y filtrada.*



Nota. Elaboración propia.

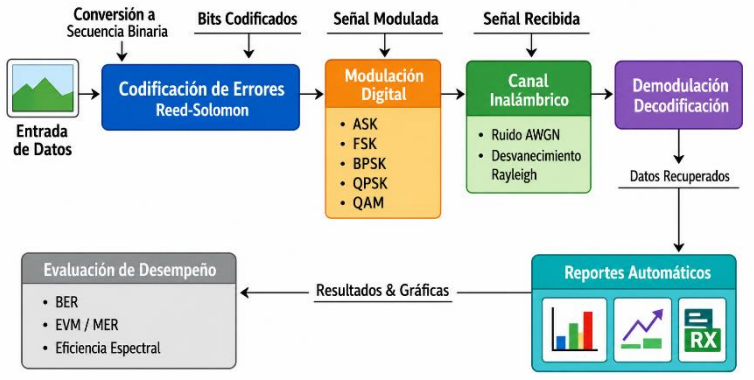
CAPÍTULO III: DISEÑO DE LA PROPUESTA

3.1 Arquitectura general del sistema implementado

El sistema digital desarrollado se organiza en una arquitectura modular. La señal de entrada (imagen o audio) se convierte en una secuencia binaria, la cual es sometida a codificación de errores Reed-Solomon. Luego, los datos se envían empleando distintos esquemas de modulación digital y se propagan por un canal inalámbrico simulado con ruido AWGN y desvanecimiento Rayleigh. En el receptor, la señal se demodula y se decodifica para recuperar la información original. Finalmente, se calculan las métricas de desempeño y se generan automáticamente reportes con gráficas e imágenes comparativas.

Como se observa en la **Figura 15**, el sistema se compone de módulos interconectados que abarcan desde la conversión de la fuente de entrada en bits hasta la generación de reportes automáticos. Cada bloque representa una etapa del proceso digital: codificación de errores, modulación, transmisión por el canal inalámbrico simulado, demodulación y evaluación del desempeño del enlace.

Figura 15. *Arquitectura general del sistema de comunicaciones implementado.*



Nota. Elaboración propia.

3.2 Preparación de la señal de entrada

3.2.1 Conversión del archivo cargado a secuencia binaria

Archivo en formato PNG para imágenes

La primera etapa del sistema consiste en transformar el archivo de entrada en una secuencia binaria, lo que permite representar la información en un formato adecuado para su transmisión digital. La imagen se carga como una matriz de píxeles y cada valor de intensidad de color se convierte en su representación binaria de ocho bits. De esta manera se obtiene un flujo continuo de datos binarios que constituye la señal de entrada del sistema, garantizando compatibilidad con las etapas posteriores de codificación, modulación y transmisión.

Como se observa en la **Figura 16**, la imagen cargada se convierte en una secuencia binaria mediante la función imagen_a_bits(). Este flujo de bits constituye la señal de entrada que será posteriormente codificada y modulada en el sistema.

Figura 16. *Función para convertir una imagen digital a secuencia binaria.*

```
# MÓDULOS GENÉRICOS
def imagen_a_bits(imagen):       # convierte una imagen en un array de bits
    return np.unpackbits(imagen.flatten()).astype(np.uint8)   # byte en 8 bits
```

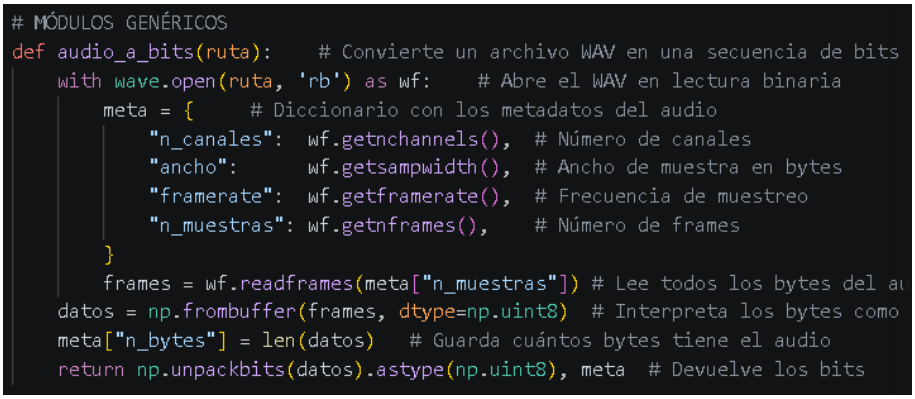
Nota. Elaboración propia.

Archivo en formato WAV para audios

En el caso de una señal de audio, la primera etapa consiste en convertir el archivo cargado en una secuencia binaria. El procedimiento inicia con la lectura del archivo WAV, obteniendo sus parámetros básicos (número de canales, frecuencia de muestreo, número de muestras). Luego, los datos de audio se extraen como bytes y cada uno se transforma en su representación binaria mediante operaciones de desempaquetado de bits. El resultado es un flujo continuo de bits que representa la señal sonora original y que puede ser procesado en las etapas posteriores de codificación, modulación y transmisión.

Como se observa en la **Figura 17**, con la función audio_a_bits() el archivo de audio cargado se convierte en una secuencia binaria mediante el desempaquetado de sus muestras, generando el flujo digital que constituye la señal de entrada del sistema.

Figura 17. *Función para convertir un archivo de audio a secuencia binaria.*



```
# MÓDULOS GENÉRICOS
def audio_a_bits(ruta):    # Convierte un archivo WAV en una secuencia de bits
    with wave.open(ruta, 'rb') as wf:    # Abre el WAV en lectura binaria
        meta = {     # Diccionario con los metadatos del audio
            "n_canales":  wf.getnchannels(),  # Número de canales
            "ancho":      wf.getsampwidth(),  # Ancho de muestra en bytes
            "framerate":  wf.getframerate(),  # Frecuencia de muestreo
            "n_muestras": wf.getnframes(),    # Número de frames
        }
        frames = wf.readframes(meta["n_muestras"]) # Lee todos los bytes del au
    datos = np.frombuffer(frames, dtype=np.uint8)  # Interpreta los bytes como
    meta["n_bytes"] = len(datos)   # Guarda cuántos bytes tiene el audio
    return np.unpackbits(datos).astype(np.uint8), meta  # Devuelve los bits
```

Nota. Elaboración propia.

3.3 Implementación del sistema de codificación de errores

3.3.1 Sistema de corrección de errores Reed-Solomon

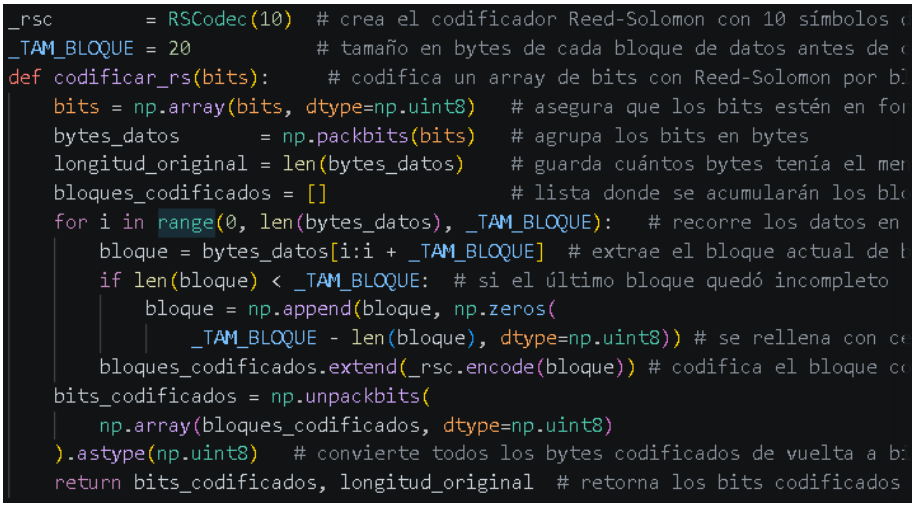
Codificación de Reed-Solomon

En el sistema desarrollado, se implementó el algoritmo Reed-Solomon (RS), ampliamente utilizado en aplicaciones como almacenamiento óptico, televisión digital y enlaces inalámbricos, debido a su capacidad para corregir múltiples errores dentro de un bloque de datos.

El procedimiento inicia con la segmentación de la secuencia binaria en bloques de tamaño fijo. Cada bloque es procesado por el codificador Reed-Solomon, el cual añade símbolos de redundancia que incrementan la longitud de la secuencia transmitida. Esta redundancia no altera la información original, pero proporciona al receptor la capacidad de reconstruir los datos incluso cuando se presentan errores durante la transmisión.

Como se observa en la **Figura 18**, la codificación Reed-Solomon añade símbolos de redundancia a los bloques de datos binarios, permitiendo la corrección de errores en la recepción y asegurando la integridad de la información transmitida.

Figura 18. *Función de codificación de Reed-Solomon.*



```
_rsc        = RSCodec(10)  # crea el codificador Reed-Solomon con 10 símbolos
_TAM_BLOQUE = 20           # tamaño en bytes de cada bloque de datos antes de
def codificar_rs(bits):     # codifica un array de bits con Reed-Solomon por b
    bits = np.array(bits, dtype=np.uint8)   # asegura que los bits estén en fo
    bytes_datos       = np.packbits(bits)   # agrupa los bits en bytes
    longitud_original = len(bytes_datos)    # guarda cuántos bytes tenía el me
    bloques_codificados = []                # lista donde se acumularán los bl
    for i in range(0, len(bytes_datos), _TAM_BLOQUE):   # recorre los datos en
        bloque = bytes_datos[i:i + _TAM_BLOQUE]  # extrae el bloque actual de
        if len(bloque) < _TAM_BLOQUE:  # si el último bloque quedó incompleto
            bloque = np.append(bloque, np.zeros(
                _TAM_BLOQUE - len(bloque), dtype=np.uint8)) # se rellena con c
        bloques_codificados.extend(_rsc.encode(bloque)) # codifica el bloque c
    bits_codificados = np.unpackbits(
        np.array(bloques_codificados, dtype=np.uint8)
    ).astype(np.uint8)   # convierte todos los bytes codificados de vuelta a b
    return bits_codificados, longitud_original  # retorna los bits codificados
```

Nota. Elaboración propia.

Decodificación de Reed-Solomon

El procedimiento para decodificar inicia con el ajuste de la secuencia de bits recibida, eliminando sobrantes para garantizar que la longitud sea múltiplo de ocho. Posteriormente, los bits se empaquetan en bytes, lo que permite reconstruir los bloques codificados. Cada bloque se procesa con el decodificador Reed-Solomon, que identifica y corrige los errores presentes en los símbolos. En caso de que un bloque no pueda ser decodificado correctamente, se aplica un manejo de excepciones para continuar con los demás bloques sin interrumpir el proceso.

Finalmente, los datos recuperados se convierten nuevamente en una secuencia binaria, obteniendo así los bits decodificados que representan la información original transmitida.

Como se observa en la **Figura 19**, la función de decodificación Reed-Solomon procesa los bloques recibidos, corrige los errores presentes y reconstruye la secuencia binaria original, asegurando la integridad de la información transmitida.

Figura 19. *Función de decodificación Reed-Solomon.*

```
def decodificar_rs(bits):   # decodifica un array de bits codificados con Reed-
    sobrantes = len(bits) % 8   # calcula si sobran bits que no completan un by
    if sobrantes:               # si hay bits sobrantes
        bits = bits[:-sobrantes]    # se descartan para trabajar solo con byte
    bytes_rx             = np.packbits(bits) # agrupa los bits recibidos en by
    tam_bloque_codificado = _TAM_BLOQUE + 10   # tamaño de cada bloque codifica
    datos_recuperados     = []    # lista donde se acumularán los datos decodi-
    for i in range(0, len(bytes_rx), tam_bloque_codificado):  # recorre los by
        bloque_rx = bytes_rx[i:i + tam_bloque_codificado]   # extrae el bloque
        if len(bloque_rx) < tam_bloque_codificado:   # si el bloque llegó incor
            datos_recuperados.extend(np.zeros(_TAM_BLOQUE, dtype=np.uint8))  #
            continue   # pasa al siguiente bloque sin intentar decodificar
        try:   # intenta decodificar el bloque con Reed-Solomon
            bloque_dec = _rsc.decode(bloque_rx)[0]  # decodifica y corrige err
            bloque_dec = np.asarray(bloque_dec, dtype=np.uint8) # convierte el
            if len(bloque_dec) < _TAM_BLOQUE:  # si quedó más corto de lo esper
                bloque_dec = np.append(
                    bloque_dec, np.zeros(_TAM_BLOQUE - len(bloque_dec), dtype=n
            datos_recuperados.extend(bloque_dec[:_TAM_BLOQUE])  # agrega el blo
        except Exception:    # si la decodificación falla
            datos_recuperados.extend(np.zeros(_TAM_BLOQUE, dtype=np.uint8))  #
    bits_dec = np.unpackbits(np.array(datos_recuperados, dtype=np.uint8))  # co
    return bits_dec.astype(np.uint8)  # retorna los bits decodificados finales
```

Nota. Elaboración propia.

3.4 Implementación computacional de los esquemas de modulación y demodulación

3.4.1 Implementación de ASK

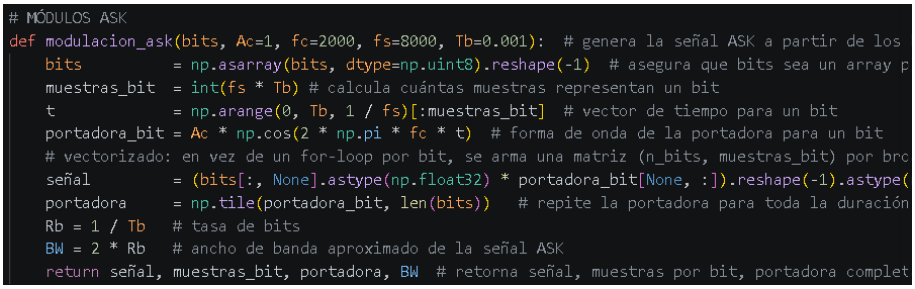
Modulación ASK

En el sistema desarrollado, la secuencia de bits codificados se multiplica por una portadora sinusoidal de frecuencia fija. La amplitud de dicha portadora se ajusta en función del valor del bit, generando una señal ASK que refleja directamente la secuencia binaria original. Este procedimiento permite observar cómo la variación de amplitud transporta la información y facilita la posterior demodulación en el receptor.

La modulación ASK es sencilla de implementar y constituye un punto de partida para comprender esquemas más complejos como PSK y QAM. Sin embargo, su desempeño frente al ruido es limitado, lo que la convierte en un buen caso de estudio para evaluar la influencia del canal AWGN y del desvanecimiento Rayleigh en la calidad de la transmisión.

Como se observa en la **Figura 20**, la modulación ASK representa los bits transmitidos mediante variaciones en la amplitud de la portadora, generando una señal modulada que constituye la entrada al canal de transmisión.

Figura 20. *Función de modulación digital ASK.*



```
# MÓDULOS ASK
def modulacion_ask(bits, Ac=1, fc=2000, fs=8000, Tb=0.001):  # genera la señal ASK a partir de los
    bits          = np.asarray(bits, dtype=np.uint8).reshape(-1)  # asegura que bits sea un array p
    muestras_bit  = int(fs * Tb) # calcula cuántas muestras representan un bit
    t             = np.arange(0, Tb, 1 / fs)[:muestras_bit]  # vector de tiempo para un bit
    portadora_bit = Ac * np.cos(2 * np.pi * fc * t)  # forma de onda de la portadora para un bit
    # vectorizado: en vez de un for-loop por bit, se arma una matriz (n_bits, muestras_bit) por bro
    señal         = (bits[:, None].astype(np.float32) * portadora_bit[None, :]).reshape(-1).astype(
    portadora     = np.tile(portadora_bit, len(bits))   # repite la portadora para toda la duración
    Rb = 1 / Tb   # tasa de bits
    BW = 2 * Rb   # ancho de banda aproximado de la señal ASK
    return señal, muestras_bit, portadora, BW  # retorna señal, muestras por bit, portadora complet
```

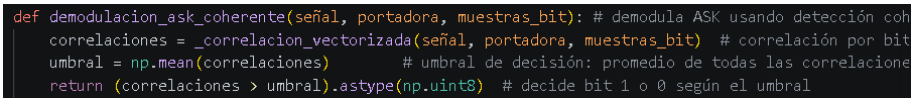
Nota. Elaboración propia.

Demodulación ASK

La función para demodular trabaja dividiendo la señal recibida en segmentos equivalentes a la duración de cada bit y calculando la correlación de cada segmento con la portadora de referencia; posteriormente, se establece un umbral de decisión como la media de las correlaciones, de modo que los valores superiores al umbral se interpretan como "1" y los inferiores como "0", reconstruyendo así los bits originales. Este procedimiento refleja el principio de la demodulación, que utiliza una portadora sincronizada para distinguir entre los niveles de amplitud y recuperar la información transmitida, incluso en presencia de ruido AWGN y desvanecimiento Rayleigh.

Como se observa en la **Figura 21**, la función de demodulación coherente ASK compara la señal recibida con la portadora de referencia y aplica un umbral de decisión para recuperar la secuencia binaria original.

Figura 21. *Función de demodulación digital ASK.*



```
def demodulacion_ask_coherente(señal, portadora, muestras_bit): # demodula ASK usando detección coh
    correlaciones = _correlacion_vectorizada(señal, portadora, muestras_bit)  # correlación por bit
    umbral = np.mean(correlaciones)        # umbral de decisión: promedio de todas las correlacione
    return (correlaciones > umbral).astype(np.uint8)  # decide bit 1 o 0 según el umbral
```

Nota. Elaboración propia.

3.4.2 Implementación de FSK

Modulación FSK

La función implementa la modulación digital FSK, donde los bits se representan mediante variaciones en la frecuencia de la portadora. El procedimiento inicia definiendo el número de muestras por bit y generando dos portadoras sinusoidales: una de frecuencia para el bit "1" y otra de frecuencia para el bit "0". La señal modulada se construye asignando la portadora correspondiente a cada bit de la secuencia, formando así una señal continua en el tiempo que alterna entre dos frecuencias según la información transmitida.

Además, la función devuelve parámetros útiles como las portadoras extendidas para toda la secuencia, la tasa de bits (R_b), la separación de frecuencias (Δf) y el ancho de banda estimado (BW). Esto permite analizar el desempeño del esquema FSK y evaluar su eficiencia espectral. La modulación FSK es más robusta frente al ruido que ASK, ya que la información se transmite en la frecuencia de la portadora, lo que facilita la detección en el receptor incluso en condiciones de canal adversas.

Como se observa en la **Figura 22**, la modulación FSK representa los bits transmitidos mediante variaciones en la frecuencia de la portadora, generando una señal modulada más robusta frente al ruido.

Figura 22. *Función de modulación digital FSK.*

```
# MÓDULOS FSK
def modulacion_fsk(bits, Ac=1, f1=3000, f0=1000, fs=8000, Tb=0.001):   # genera la señal FSK a part
    bits         = np.asarray(bits, dtype=np.uint8).reshape(-1)  # asegura formato plano de los bit
    muestras_bit = int(fs * Tb)   # muestras por bit
    t            = np.arange(0, Tb, 1 / fs)[:muestras_bit]  # vector de tiempo para un bit
    port_f1      = Ac * np.cos(2 * np.pi * f1 * t)  # portadora de frecuencia alta (bit = 1)
    port_f0      = Ac * np.cos(2 * np.pi * f0 * t)  # portadora de frecuencia baja (bit = 0)
    # vectorizado: selecciona por broadcasting la portadora f1 o f0 según el bit, sin for-loop en P
    señal        = np.where(bits[:, None] == 1, port_f1[None, :], port_f0[None, :]).reshape(-1).ast
    portadora_f1 = np.tile(port_f1, len(bits))  # portadora f1 repetida para toda la señal
    portadora_f0 = np.tile(port_f0, len(bits))  # portadora f0 repetida para toda la señal
    Rb      = 1 / Tb    # tasa de bits
    delta_f = (f1 - f0) / 2    # desviación de frecuencia
    BW      = 2 * (delta_f + Rb)    # ancho de banda aproximado (regla de Carson)
    return señal, muestras_bit, portadora_f1, portadora_f0, BW  # retorna señal, muestras/bit, amba
```

Nota. Elaboración propia.

Demodulación FSK

La función de demodulación FSK recupera los bits transmitidos a partir de la señal modulada en frecuencia. El procedimiento divide la señal recibida en segmentos

equivalentes a la duración de cada bit y calcula la correlación de cada segmento con las dos portadoras de referencia, una asociada al bit "1" y otra al bit "0".

A cada segmento de la señal se le calcula qué tanto se parece a cada una de las dos portadoras, ajustando ese valor según su energía para que las comparaciones sean justas. Cuando la correlación resulta más alta con la portadora de mayor frecuencia, el bit se interpreta como "1"; en caso contrario, se interpreta como "0".

Como se observa en la **Figura 23**, la demodulación coherente FSK compara la señal recibida con las portadoras de referencia y selecciona el bit correspondiente, según la mayor correlación obtenida.

Figura 23. *Función de demodulación digital FSK.*

```
def demodulacion_fsk_coherente(señal, portadora_f1, portadora_f0, muestras_bit):  # demodula FSK pc
    C1, C0 = _correlacion_vectorizada(señal, portadora_f1, muestras_bit), _correlacion_vectorizada(
    return (C1 > C0).astype(np.uint8)   # decide bit 1 si la correlación con f1 es mayor
```

Nota. Elaboración propia.

3.4.3 Implementación de BPSK

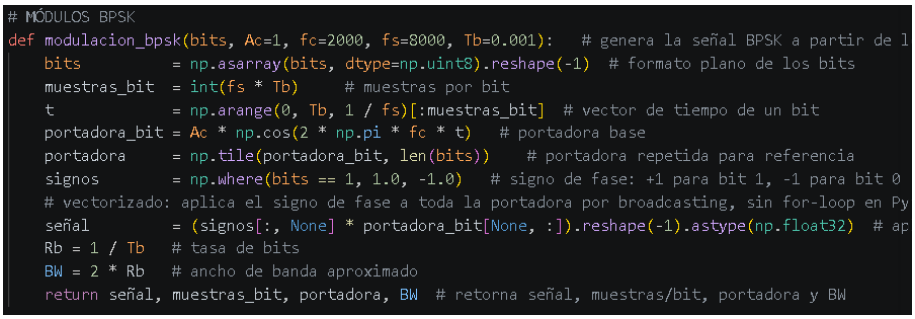
Modulación BPSK

La función implementa la modulación digital BPSK, donde los bits se representan mediante variaciones en la fase de la portadora. El procedimiento inicia calculando el número de muestras por bit y generando una portadora sinusoidal de frecuencia fija. Para cada bit de la secuencia se asigna un signo: el bit "1" mantiene la fase original de la portadora, mientras que el bit "0" invierte la fase (multiplicando por –1). De esta manera, la información binaria se transmite como cambios de fase en la señal modulada.

La función devuelve la señal modulada, el número de muestras por bit, la portadora extendida y el ancho de banda estimado. Este esquema es más robusto frente al ruido que ASK y FSK, ya que la información se codifica en la fase de la portadora, lo que facilita la detección en el receptor. La modulación BPSK constituye uno de los métodos más utilizados en sistemas de comunicación digital por su simplicidad y fiabilidad, siendo además la base para esquemas más avanzados como QPSK y QAM.

Como se observa en la **Figura 24**, la modulación BPSK transmite los bits mediante cambios de fase en la portadora, generando una señal más resistente al ruido.

Figura 24. *Función de modulación digital BPSK.*



```python
# MÓDULOS BPSK
def modulacion_bpsk(bits, Ac=1, fc=2000, fs=8000, Tb=0.001):   # genera la señal BPSK a partir de l
    bits          = np.asarray(bits, dtype=np.uint8).reshape(-1)  # formato plano de los bits
    muestras_bit  = int(fs * Tb)     # muestras por bit
    t             = np.arange(0, Tb, 1 / fs)[:muestras_bit]  # vector de tiempo de un bit
    portadora_bit = Ac * np.cos(2 * np.pi * fc * t)   # portadora base
    portadora     = np.tile(portadora_bit, len(bits))    # portadora repetida para referencia
    signos        = np.where(bits == 1, 1.0, -1.0)   # signo de fase: +1 para bit 1, -1 para bit 0
    # vectorizado: aplica el signo de fase a toda la portadora por broadcasting, sin for-loop en Py
    señal         = (signos[:, None] * portadora_bit[None, :]).reshape(-1).astype(np.float32)  # ap
    Rb = 1 / Tb   # tasa de bits
    BW = 2 * Rb   # ancho de banda aproximado
    return señal, muestras_bit, portadora, BW  # retorna señal, muestras/bit, portadora y BW
```

Nota. Elaboración propia.

Demodulación BPSK

La función de demodulación coherente BPSK recupera los bits transmitidos a partir de la señal modulada en fase. El procedimiento divide la señal recibida en segmentos equivalentes a la duración de cada bit y calcula la correlación de cada segmento con la portadora de referencia. Si el resultado de la correlación es positivo, se interpreta como un bit "1"; si es negativo, se asigna un bit "0". De esta manera, la función reconstruye la secuencia binaria original transmitida.

Este método refleja el principio de la demodulación coherente: utilizar una portadora sincronizada para distinguir entre los cambios de fase que representan los símbolos binarios. Gracias a esta técnica, el sistema puede recuperar la información incluso en presencia de ruido AWGN y desvanecimiento Rayleigh, evaluando su desempeño mediante métricas como la tasa de error de bits (BER). La demodulación BPSK es especialmente robusta, lo que la convierte en una opción ampliamente utilizada en sistemas de comunicación digital.

Como se observa en la **Figura 25**, la demodulación coherente BPSK compara la señal recibida con la portadora de referencia y aplica un criterio de correlación para recuperar la secuencia binaria original.

Figura 25. *Función de demodulación digital BPSK.*

```python
def demodulacion_bpsk_coherente(señal, portadora, muestras_bit):    # demodula BPSK por correlación
    correlaciones = _correlacion_vectorizada(señal, portadora, muestras_bit)  # correlación normali
    return (correlaciones > 0).astype(np.uint8)   # decide el bit según el signo de la correlación
```

Nota. Elaboración propia.

3.4.4 Implementación de QPSK

Modulación QPSK

La función implementa la modulación digital QPSK, donde cada símbolo transmite dos bits mediante variaciones en la fase de la portadora. El procedimiento inicia verificando que la secuencia de bits tenga longitud par; en caso contrario, se añade un bit de relleno. Luego, se calcula el número de muestras por símbolo y se generan dos portadoras ortogonales: una en fase (I) y otra en cuadratura (Q).

Cada par de bits se asigna a un símbolo QPSK mediante el mapeo definido en MAPA_QPSK, que determina los valores de las componentes I y Q. La señal modulada se construye combinando las portadoras con los coeficientes asignados, generando una secuencia continua que alterna entre cuatro posibles fases. Finalmente, la función devuelve la señal modulada, las portadoras completas, el ancho de banda estimado y la lista de símbolos IQ, lo que permite analizar tanto la eficiencia espectral como la constelación resultante. QPSK ofrece mayor eficiencia que BPSK al transmitir el doble de información usando el mismo ancho de banda y manteniendo buena robustez frente al ruido.

Como se observa en la **Figura 26**, la modulación QPSK transmite dos bits por símbolo mediante variaciones en la fase de la portadora, logrando mayor eficiencia espectral que BPSK.

Figura 26. *Función de modulación digital QPSK.*

```python
def modulacion_qpsk(bits, Ac=1, fc=2000, fs=8000, Tb=0.001): # genera la señal QPSK a partir de los
    bits = np.asarray(bits, dtype=np.uint8).reshape(-1)      # formato plano de los bits
    if len(bits) % 2 != 0:  # si el número de bits es impar
        bits = np.append(bits, 0)   # se rellena con un bit cero para formar pares
    muestras_sym = int(fs * Tb) * 2     # muestras por símbolo (2 bits por símbolo)
    t_sym        = np.arange(0, 2 * Tb, 1 / fs)[:muestras_sym]   # vector de tiempo de un símbolo
    port_I       = Ac * np.cos(2 * np.pi * fc * t_sym)       # portadora en fase (coseno)
    port_Q       = Ac * np.sin(2 * np.pi * fc * t_sym)       # portadora en cuadratura (seno)
    port_I_full  = np.tile(port_I, len(bits) // 2)   # portadora I repetida para toda la señal
    port_Q_full  = np.tile(port_Q, len(bits) // 2)   # portadora Q repetida para toda la señal
    # vectorizado: mapa de Gray sin diccionario ni for-loop (ver derivación: I~b0, Q~b1)
    b1 = bits[0::2]; b0 = bits[1::2]   # separa los bits pares (b1) e impares (b0) de cada símbolo
    I_s = np.where(b0 == 0, A_QPSK, -A_QPSK)   # bit b0=0 -> I=+A, b0=1 -> I=-A
    Q_s = np.where(b1 == 0, A_QPSK, -A_QPSK)   # bit b1=0 -> Q=+A, b1=1 -> Q=-A
    # construye la señal de todos los símbolos a la vez por broadcasting
    señal = (I_s[:, None] * port_I[None, :] - Q_s[:, None] * port_Q[None, :]).reshape(-1).astype(np
    Rb = 1 / Tb       # tasa de bits
    BW = 2 * (Rb / 2)    # ancho de banda aproximado (2 bits/símbolo)
    return señal, muestras_sym, port_I_full, port_Q_full, BW, (I_s, Q_s)   # retorna señal, muestra
```

Nota. Elaboración propia.

Demodulación QPSK

La función de demodulación coherente QPSK recupera los bits transmitidos a partir de la señal modulada en cuadratura. El procedimiento divide la señal recibida en segmentos equivalentes a la duración de cada símbolo y calcula la correlación de cada segmento con las portadoras ortogonales en fase (I) y en cuadratura (Q). Los valores de correlación se normalizan respecto a la energía de cada portadora y se interpretan según su signo, lo que permite determinar las componentes I y Q del símbolo recibido.

A partir de estas decisiones, se asigna el par de bits correspondiente según la constelación QPSK (00, 01, 11, 10). De esta manera, la función reconstruye la secuencia binaria original transmitida, reflejando el principio de la demodulación coherente: usar portadoras sincronizadas para distinguir entre las variaciones de fase que representan los símbolos.

Como se observa en la **Figura 27**, la demodulación coherente QPSK utiliza correlaciones con portadoras ortogonales para identificar los símbolos y recuperar la secuencia binaria original.

Figura 27. *Función de demodulación digital QPSK.*

```
def demodulacion_qpsk_coherente(señal, port_I, port_Q, muestras_sym):  # demodula QPSK por correlac
    CI =  _correlacion_vectorizada(señal, port_I, muestras_sym)   # correlación con la portadora I,
    CQ = -_correlacion_vectorizada(señal, port_Q, muestras_sym)   # correlación con la portadora Q
    n_simb = len(CI)   # número de símbolos demodulados
    bits = np.empty(2 * n_simb, dtype=np.uint8)   # reserva el array de bits de salida (2 bits por
    bits[0::2] = (CQ < 0).astype(np.uint8)   # b1: 0 si Q_dec>0 (CQ>=0), 1 si Q_dec<=0 (CQ<0)
    bits[1::2] = (CI < 0).astype(np.uint8)   # b0: 0 si I_dec>0 (CI>=0), 1 si I_dec<=0 (CI<0)
    return bits   # retorna los bits demodulados
```

Nota. Elaboración propia.

3.4.5 Implementación de 16-QAM

Modulación 16-QAM

La función implementa la modulación 16-QAM, donde cada símbolo transmite cuatro bits combinando variaciones de amplitud y fase en dos portadoras ortogonales (I y Q). El código ajusta la longitud de la secuencia de bits para que sea múltiplo de cuatro, calcula el número de muestras por símbolo y genera las portadoras en fase y cuadratura. Luego, cada grupo de cuatro bits se divide en dos pares: los dos primeros determinan el nivel de la componente I, y los dos siguientes, el nivel de la componente Q, utilizando un

mapeo Gray que reduce la probabilidad de error al asegurar que símbolos vecinos difieran en un solo bit.

La señal modulada se construye combinando las portadoras con los niveles asignados, generando una constelación de 16 puntos que representa todas las combinaciones posibles de cuatro bits. La función devuelve la señal modulada, las portadoras completas, el ancho de banda estimado y la lista de símbolos IQ, lo que permite analizar tanto la eficiencia espectral como la sensibilidad al ruido. Este esquema ofrece mayor capacidad de transmisión que QPSK, aunque con menor robustez frente a interferencias, siendo ampliamente utilizado en sistemas modernos de telecomunicaciones por su balance entre eficiencia y desempeño.

Como se observa en la **Figura 28**, la modulación 16-QAM transmite cuatro bits por símbolo mediante combinaciones de amplitud y fase, logrando alta eficiencia espectral a costa de mayor sensibilidad al ruido.

Figura 28. *Función de modulación digital 16-QAM.*

```
def modulacion_16qam(bits, Ac=1, fc=2000, fs=8000, Tb=0.001):  # genera la señal 16-QAM a partir de
    bits  = np.asarray(bits, dtype=np.uint8).reshape(-1)  # formato plano de bits
    resto = len(bits) % 4    # bits sobrantes que no completan un símbolo (4 bits)
    if resto != 0:           # si sobran bits
        bits = np.append(bits, np.zeros(4 - resto, dtype=np.uint8)) # se rellenan con ceros hasta c
    muestras_sym = int(fs * Tb) * 4  # muestras por símbolo (4 bits por símbolo)
    t_sym        = np.arange(0, 4 * Tb, 1 / fs)[:muestras_sym]  # vector de tiempo de un símbolo
    port_I       = Ac * np.cos(2 * np.pi * fc * t_sym) # portadora en fase
    port_Q       = Ac * np.sin(2 * np.pi * fc * t_sym) # portadora en cuadratura
    n_simbolos   = len(bits) // 4   # número total de símbolos
    port_I_full  = np.tile(port_I, n_simbolos) # portadora I repetida
    port_Q_full  = np.tile(port_Q, n_simbolos) # portadora Q repetida
    # vectorizado: separa los 4 bits de cada símbolo y usa la LUT en vez de un for-loop con diccion
    b3, b2 = bits[0::4], bits[1::4]   # primeros 2 bits -> componente I
    b1, b0 = bits[2::4], bits[3::4]   # últimos 2 bits -> componente Q
    I_s = LUT_ENCODE_16QAM[2 * b3 + b2]   # nivel I según la LUT de Gray
    Q_s = LUT_ENCODE_16QAM[2 * b1 + b0]   # nivel Q según la LUT de Gray
    señal = (I_s[:, None] * port_I[None, :] - Q_s[:, None] * port_Q[None, :]).reshape(-1).astype(np
    Rb = 1 / Tb     # tasa de bits
    BW = 2 * (Rb / 4)     # ancho de banda aproximado (4 bits/símbolo)
    return señal, muestras_sym, port_I_full, port_Q_full, BW, (I_s, Q_s)  # retorna señal, muestras
```

Nota. Elaboración propia.

Demodulación 16-QAM

La función de demodulación 16-QAM recupera los bits transmitidos a partir de la señal modulada en amplitud y fase. El procedimiento divide la señal recibida en segmentos equivalentes a la duración de cada símbolo y calcula la correlación de cada segmento con las portadoras ortogonales en fase (I) y cuadratura (Q). Los valores de

correlación se normalizan respecto a la energía de cada portadora y se comparan con los niveles definidos en la constelación 16-QAM.

El algoritmo busca para cada componente (I y Q) el nivel más cercano dentro de los posibles y lo convierte en su par de bits correspondiente usando el mapeo Gray, pensado precisamente para minimizar errores. Con esto, cada símbolo termina traduciéndose en cuatro bits hasta reconstruir la secuencia binaria original. En esencia, este proceso ilustra bien la lógica de la demodulación coherente: valerse de portadoras sincronizadas junto con niveles ya definidos de antemano para poder identificar las distintas combinaciones de amplitud y fase que representan cada símbolo.

Como se observa en la **Figura 29**, la demodulación coherente 16-QAM utiliza correlaciones con portadoras ortogonales y niveles predefinidos para identificar los símbolos y recuperar la secuencia binaria original.

Figura 29. *Función de demodulación digital 16-QAM.*

```
def demodulacion_16qam_coherente(señal, port_I, port_Q, muestras_sym):  # demodula 16-QAM por corre
    CI =  _correlacion_vectorizada(señal, port_I, muestras_sym)   # correlación con portadora I, ve
    CQ = -_correlacion_vectorizada(señal, port_Q, muestras_sym)   # correlación con portadora Q, ve
    idx_I = np.argmin(np.abs(CI[:, None] - NIVELES_16QAM_ARR[None, :]), axis=1)   # nivel I más cer
    idx_Q = np.argmin(np.abs(CQ[:, None] - NIVELES_16QAM_ARR[None, :]), axis=1)   # nivel Q más cer
    bI = LUT_DECODE_16QAM[idx_I]   # bits (b3,b2) recuperados del nivel I, forma (n_simb,2)
    bQ = LUT_DECODE_16QAM[idx_Q]   # bits (b1,b0) recuperados del nivel Q, forma (n_simb,2)
    n_simb = len(CI)   # número de símbolos
    bits = np.empty((n_simb, 4), dtype=np.uint8)   # arma los 4 bits por símbolo
    bits[:, 0:2] = bI; bits[:, 2:4] = bQ   # orden [b3,b2,b1,b0], igual que el original
    return bits.reshape(-1)   # retorna los bits demodulados, aplanados
```

Nota. Elaboración propia.

3.4.6 Implementación de 64-QAM

Modulación 64-QAM

La función implementa la modulación 64-QAM, donde cada símbolo transmite seis bits mediante combinaciones de amplitud y fase en dos portadoras ortogonales (I y Q). El procedimiento inicia ajustando la longitud de la secuencia de bits para que sea múltiplo de seis, calcula el número de muestras por símbolo y genera las portadoras en fase y cuadratura. Luego, cada grupo de seis bits se divide en dos ternas: los tres primeros determinan el nivel de la componente I, y los tres siguientes, el nivel de la componente Q, utilizando un mapeo Gray de tres bits que asegura que símbolos vecinos difieran en un solo bit, reduciendo la probabilidad de error.

La señal modulada se construye combinando las portadoras con los niveles asignados, generando una constelación de 64 puntos que representa todas las combinaciones posibles de seis bits. La función devuelve la señal modulada, las portadoras completas, el ancho de banda estimado y la lista de símbolos IQ, lo que permite analizar tanto la eficiencia espectral como la sensibilidad al ruido. Este esquema ofrece una capacidad de transmisión significativamente mayor que 16-QAM y QPSK, aunque con menor robustez frente a interferencias, siendo ampliamente utilizado en sistemas modernos como LTE y Wi-Fi por su balance entre velocidad y eficiencia espectral.

Como se observa en la **Figura 30**, la modulación 64-QAM transmite seis bits por símbolo mediante combinaciones de amplitud y fase, logrando alta eficiencia espectral pero con mayor sensibilidad al ruido.

Figura 30. *Función de modulación digital 64-QAM.*

```
def modulacion_64qam(bits, Ac=1, fc=2000, fs=8000, Tb=0.001):  # genera la señal 64-QAM a partir de
    bits  = np.asarray(bits, dtype=np.uint8).reshape(-1)       # formato plano de bits
    resto = len(bits) % 6   # bits sobrantes que no completan un símbolo (6 bits)
    if resto != 0:          # si sobran bits
        bits = np.append(bits, np.zeros(6 - resto, dtype=np.uint8)) # se rellenan con ceros hasta c
    muestras_sym = int(fs * Tb) * 6   # muestras por símbolo (6 bits por símbolo)
    t_sym        = np.arange(0, 6 * Tb, 1 / fs)[:muestras_sym] # vector de tiempo de un símbolo
    port_I       = Ac * np.cos(2 * np.pi * fc * t_sym)   # portadora en fase
    port_Q       = Ac * np.sin(2 * np.pi * fc * t_sym)   # portadora en cuadratura
    n_simbolos   = len(bits) // 6   # número total de símbolos
    port_I_full  = np.tile(port_I, n_simbolos) # portadora I repetida
    port_Q_full  = np.tile(port_Q, n_simbolos) # portadora Q repetida
    # vectorizado: separa los 6 bits de cada símbolo y usa la LUT en vez de un for-loop con diccion
    b5, b4, b3 = bits[0::6], bits[1::6], bits[2::6]   # primeros 3 bits -> componente I
    b2, b1, b0 = bits[3::6], bits[4::6], bits[5::6]   # últimos 3 bits -> componente Q
    I_s = LUT_ENCODE_64QAM[4 * b5 + 2 * b4 + b3]   # nivel I según la LUT de Gray
    Q_s = LUT_ENCODE_64QAM[4 * b2 + 2 * b1 + b0]   # nivel Q según la LUT de Gray
    señal = (I_s[:, None] * port_I[None, :] - Q_s[:, None] * port_Q[None, :]).reshape(-1).astype(np
    Rb = 1 / Tb   # tasa de bits
    BW = 2 * (Rb / 6)   # ancho de banda aproximado (6 bits/símbolo)
    return señal, muestras_sym, port_I_full, port_Q_full, BW, (I_s, Q_s)  # retorna señal, muestras
```

Nota. Elaboración propia.

Demodulación 64-QAM

La función de demodulación coherente 64-QAM recupera los seis bits transmitidos por cada símbolo. El procedimiento divide la señal recibida en segmentos equivalentes a la duración de cada símbolo y calcula la correlación de cada segmento con las portadoras ortogonales en fase (I) y cuadratura (Q). Los valores de correlación se normalizan respecto a la energía de cada portadora y se comparan con los niveles definidos en la constelación 64-QAM.

El algoritmo identifica para cada componente (I y Q) el nivel más cercano posible y lo traduce a sus tres bits correspondientes mediante un mapeo Gray de tres bits, diseñado justamente para reducir la probabilidad de error. De este modo, cada símbolo queda representado por seis bits, permitiendo reconstruir por completo la secuencia binaria original.

Como se observa en la **Figura 31**, la demodulación coherente 64-QAM utiliza correlaciones con portadoras ortogonales y niveles predefinidos para identificar los símbolos y recuperar la secuencia binaria original.

Figura 31. *Función de demodulación digital 64-QAM.*

```python
def demodulacion_64qam_coherente(señal, port_I, port_Q, muestras_sym): # demodula 64-QAM por correl
    CI =  _correlacion_vectorizada(señal, port_I, muestras_sym)   # correlación con portadora I, ve
    CQ = -_correlacion_vectorizada(señal, port_Q, muestras_sym)   # correlación con portadora Q, ve
    idx_I = np.argmin(np.abs(CI[:, None] - NIVELES_64QAM_ARR[None, :]), axis=1)   # nivel I más cer
    idx_Q = np.argmin(np.abs(CQ[:, None] - NIVELES_64QAM_ARR[None, :]), axis=1)   # nivel Q más cer
    bI = LUT_DECODE_64QAM[idx_I]   # bits (b5,b4,b3) recuperados del nivel I, forma (n_simb,3)
    bQ = LUT_DECODE_64QAM[idx_Q]   # bits (b2,b1,b0) recuperados del nivel Q, forma (n_simb,3)
    n_simb = len(CI)   # número de símbolos
    bits = np.empty((n_simb, 6), dtype=np.uint8)   # arma los 6 bits por símbolo
    bits[:, 0:3] = bI; bits[:, 3:6] = bQ   # orden [b5,b4,b3,b2,b1,b0], igual que el original
    return bits.reshape(-1)   # retorna los bits demodulados, aplanados
```

Nota. Elaboración propia.

3.5 Simulación del canal inalámbrico

3.5.1 Incorporación de ruido AWGN y el desvanecimiento Rayleigh

La función implementada simula un canal inalámbrico con desvanecimiento Rayleigh y ruido aditivo blanco gaussiano (AWGN), condiciones típicas en comunicaciones digitales. La señal de entrada se convierte en un arreglo numérico y se calcula su potencia promedio, lo que permite ajustar la potencia del ruido según la relación señal-ruido (SNR) definida.

El procesamiento se realiza en segmentos para optimizar el manejo de señales largas. En cada bloque se generan coeficientes aleatorios que modelan el desvanecimiento Rayleigh, simulando la variación de amplitud en un entorno multipropagado. Posteriormente, se añade un vector de ruido gaussiano con potencia ajustada, y la señal se normaliza, obteniendo una salida que representa de manera realista la señal recibida.

Este modelo combina los efectos del ruido AWGN y el desvanecimiento Rayleigh, permitiendo evaluar el desempeño del sistema bajo condiciones de propagación cercanas a la realidad.

Como se observa en la **Figura 32**, la función de canal incorpora ruido AWGN y desvanecimiento Rayleigh sobre la señal modulada, generando una salida que simula las condiciones reales de transmisión inalámbrica.

Figura 32. *Función del canal Rayleigh con ruido AWGN.*

```
def canal_rayleigh_awgn(señal, snr_db, muestras_por_simbolo=1, chunk_size=200_000): # simula un can
    señal   = np.asarray(señal, dtype=np.float32)  # convierte la señal de entrada a float32
    N       = len(señal) # obtiene la longitud total de la señal
    rx_ec   = np.empty(N, dtype=np.float32)  # reserva memoria para la señal ecualizada de salida
    potencia   = float(np.mean(señal ** 2))  # calcula la potencia promedio de la señal transmitida
    snr_lineal = 10 ** (snr_db / 10)  # convierte el SNR (Eb/N0) a escala lineal, por muestra
    pot_ruido  = potencia / snr_lineal if snr_lineal > 0 else 1e-10  # calcula la potencia de ruido
    muestras_por_simbolo = max(1, int(muestras_por_simbolo))  # asegura al menos 1 muestra por símb
    # el tamaño de bloque de procesamiento debe ser múltiplo del símbolo, para no partir un símbolo
    chunk_size = max(muestras_por_simbolo, (chunk_size // muestras_por_simbolo) * muestras_por_simb
    for inicio in range(0, N, chunk_size):   # procesa la señal en bloques para ahorrar memoria
        fin   = min(inicio + chunk_size, N)  # calcula el final del bloque actual
        seg   = señal[inicio:fin].astype(np.float64) # extrae el segmento de señal actual en doble
        L     = fin - inicio  # longitud del segmento actual
        n_simb = int(np.ceil(L / muestras_por_simbolo))  # número de símbolos completos (o parciale
        h_I_sym = np.random.randn(n_simb) / np.sqrt(2)  # componente en fase del desvanecimiento Ra
        h_Q_sym = np.random.randn(n_simb) / np.sqrt(2)  # componente en cuadratura del desvanecimie
        h_mag_sym = np.sqrt(h_I_sym ** 2 + h_Q_sym ** 2)  # magnitud del desvanecimiento por símbol
        h_mag = np.repeat(h_mag_sym, muestras_por_simbolo)[:L]  # mantiene el mismo desvanecimiento
        ruido = np.sqrt(pot_ruido / 2) * np.random.randn(L) # genera el ruido blanco gaussiano adit
        rx    = seg * h_mag + ruido  # aplica el desvanecimiento y suma el ruido a la señal
        h_mag_ec = np.maximum(h_mag, 0.15)  # limita el desvanecimiento mínimo para ecualizar (evit
        rx_ec[inicio:fin] = (rx / h_mag_ec).astype(np.float32)  # ecualiza dividiendo por el desvan
    return rx_ec # retorna la señal recibida y ecualizada
```

Nota. Elaboración propia.

3.6 Procedimiento para el cálculo de métricas

La función calcular_ber determina la tasa de error de bits (BER) comparando la secuencia original transmitida con la secuencia recibida. El resultado es la proporción de bits erróneos respecto al total, lo que refleja directamente la fiabilidad del sistema de comunicación frente al ruido, interferencias o distorsiones del canal.

La función calcular_evm se encarga de obtener la magnitud del vector de error (EVM), una medida que indica qué tan distintos son en promedio los símbolos recibidos respecto a los que fueron transmitidos, tomando como referencia la potencia de la señal transmitida.

La función calcular_mer calcula la tasa de error de modulación (MER), expresada en decibelios. Se obtiene como la razón entre la potencia promedio de la señal transmitida y la potencia del error.

Finalmente, la función eficiencia_espectral determina la eficiencia espectral definida como la razón entre la tasa de bits (R_b) y el ancho de banda ocupado (BW). Es una métrica clave para evaluar qué tan eficientemente se utiliza el espectro disponible en la transmisión de información.

Estas mismas funciones se aplican de manera análoga para cada esquema de modulación, simplemente añadiendo el sufijo correspondiente, lo que permite evaluar de forma consistente el desempeño de diferentes técnicas de modulación digital en términos de fiabilidad, calidad de señal y eficiencia espectral.

La validación de las métricas de desempeño se realizó mediante simulaciones Monte Carlo utilizando 30 iteraciones tanto para el cálculo del BER como para las métricas de EVM y MER. En cada nivel de SNR se calculó el promedio de las repeticiones junto con su desviación estándar poblacional, la cual fue calculada con NumPy mediante np.std() que es equivalente a ddof=0, es decir, con divisor n y no n-1, lo que permite estimar la dispersión de los resultados en cada punto de SNR. Esta información se incorporó al reporte generado por el sistema complementando el valor medio con una medida de variabilidad, especialmente relevante en condiciones de SNR muy bajas o altas, donde la dispersión entre iteraciones tiende a ser mayor.

Como se observa en la **Figura 33**, las métricas BER, EVM, MER y eficiencia espectral permiten evaluar la calidad y desempeño en condiciones de canal real.

Figura 33. *Funciones de métricas de desempeño generales en sistemas digitales.*

```python
def calcular_ber_ask(original, recibido):  # calcula la tasa de error de bit para ASK
    n = min(len(original), len(recibido))  # usa la longitud mínima entre ambos arrays
    return np.sum(original[:n] != recibido[:n]) / n if n else 0.0  # proporción de bits distintos e

def calcular_evm_ask(tx, rx):  # calcula el EVM (magnitud de error vectorial) para ASK
    n = min(len(tx), len(rx))  # usa la longitud mínima entre señales tx y rx
    tx, rx = tx[:n], rx[:n]    # recorta ambas señales a la misma longitud
    p_tx = np.mean(np.abs(tx)**2)   # potencia promedio de la señal transmitida
    return np.sqrt(np.mean(np.abs(tx - rx)**2) / p_tx) if p_tx else 0.0   # raíz del error cuadráti

def calcular_mer_ask(tx, rx):  # calcula el MER (relación de error de magnitud) para ASK
    n = min(len(tx), len(rx))  # usa la longitud mínima entre señales tx y rx
    tx, rx = tx[:n], rx[:n]    # recorta ambas señales a la misma longitud
    p_error = np.mean(np.abs(tx - rx)**2)  # potencia promedio del error
    if p_error == 0:  # si no hubo error alguno
        return np.inf  # el MER es infinito (señal perfecta)
    return 10 * np.log10(np.mean(np.abs(tx)**2) / p_error) # MER en dB: relación potencia señal / p

def eficiencia_espectral_ask(Rb, BW):  # calcula la eficiencia espectral de ASK
    return Rb / BW if BW else 0.0  # bits por segundo dividido entre el ancho de banda

Tb_ASK = 0.001    # duración de un bit para ASK (segundos)
```

Nota. Elaboración propia.

3.7 Validación de resultados de métricas de desempeño

Con el propósito de validar los resultados obtenidos en la simulación del sistema de comunicaciones digitales implementado en Python, se realizó el cálculo matemático de los principales parámetros de desempeño: la tasa de error de bit (BER) antes y después de la decodificación Reed-Solomon, la magnitud del vector de error (EVM) y la tasa de error de modulación (MER). Estos cálculos se desarrollan a partir de las ecuaciones teóricas de cada métrica y de los datos obtenidos durante la simulación, permitiendo comparar los valores teóricos con los resultados experimentales.

Ejemplo matemático de tasa de error de bit

Como ejemplo, a continuación se presenta el procedimiento detallado para la modulación QPSK, donde se muestra paso a paso el cálculo del BER pre-Reed-Solomon, verificando la correspondencia entre los resultados simulados y los obtenidos mediante el análisis matemático usando la **ecuación 18**:

Paso 1. Escribir la ecuación del BER

$$BER = \frac{N_e}{N_b}$$

Paso 2. Despejar el número de bits erróneos de la ecuación anterior

$$N_e = BER \times N_b$$

Paso 3. Sustituir los valores se reemplazan los datos del reporte:

$$N_e = 0.000856 \times 960000$$

Paso 4. Realizar la multiplicación

$$N_e = 821.76$$

Como el número de errores debe ser un entero, se aproxima a:

$$N_e \approx 822 \text{ bits}$$

Esto significa que, de los 960000 bits transmitidos, aproximadamente 822 llegaron incorrectamente antes de aplicar Reed-Solomon.

Paso 5. Verificar el BER

Se sustituye nuevamente en la ecuación del BER:

$$BER = \frac{822}{960000}$$

Se realiza la división y se redondea a decimales:

$$BER \approx 0.000856$$

Ejemplo matemático de error de magnitud del vector y tasa de error de modulación

Como ejemplo, a continuación se presenta el procedimiento detallado para la modulación QPSK, donde se muestra paso a paso el cálculo del MER y EVM Reed-Solomon, verificando la correspondencia entre los resultados simulados y los obtenidos mediante el análisis matemático usando la **ecuación 19** y **ecuación 20**:

Paso 1. Escribir la ecuación del MER

$$MER = 10\log_{10}\left(\frac{P_s}{P_e}\right)$$

Como en la simulación ya se conoce el EVM, también puede emplearse la relación:

$$MER = -20\log_{10}(EVM)$$

donde el EVM debe expresarse en forma decimal.

Paso 2. Convertir el EVM a valor decimal

El EVM obtenido fue:

$$EVM = 41.6425\%$$

Se convierte a decimal:

$$EVM = \frac{41.6425}{100} = 0.416425$$

Paso 3. Sustituir en la ecuación

$$MER = -20\log_{10}(0.416425)$$

Paso 4. Calcular el logaritmo

$$\log_{10}(0.416425) = -0.38046$$

Paso 5. Multiplicar por -20 y aproximarlo a cuatro decimales:

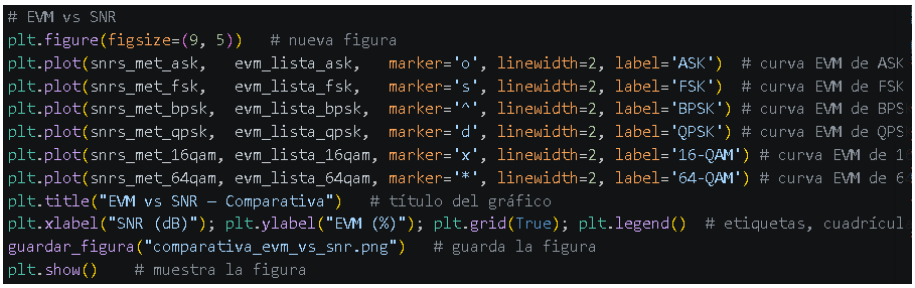
$$MER = -20(-0.38046)$$
$$MER = 7.6093 \text{ dB}$$

Aplicando error de magnitud del vector

El código implementa la comparación de la magnitud del vector de error (EVM) en función de la relación señal-ruido (SNR) para distintos esquemas de modulación digital. Este análisis gráfico es fundamental porque permite comparar la robustez de cada esquema de modulación frente al ruido. El resultado se guarda como comparativa_evm_vs_snr.png, integrando evidencia visual del desempeño de cada técnica.

Como se observa en la **Figura 34**, la comparación EVM vs SNR evidencia la diferencia en desempeño entre modulaciones simples y complejas, mostrando la sensibilidad de cada esquema frente al ruido.

Figura 34. *Comparativa de EVM vs SNR en modulaciones digitales.*



```
# EVM vs SNR
plt.figure(figsize=(9, 5))   # nueva figura
plt.plot(snrs_met_ask,   evm_lista_ask,   marker='o', linewidth=2, label='ASK')  # curva EVM de ASK
plt.plot(snrs_met_fsk,   evm_lista_fsk,   marker='s', linewidth=2, label='FSK')  # curva EVM de FSK
plt.plot(snrs_met_bpsk,  evm_lista_bpsk,  marker='^', linewidth=2, label='BPSK') # curva EVM de BPS
plt.plot(snrs_met_qpsk,  evm_lista_qpsk,  marker='d', linewidth=2, label='QPSK') # curva EVM de QPS
plt.plot(snrs_met_16qam, evm_lista_16qam, marker='x', linewidth=2, label='16-QAM') # curva EVM de 1
plt.plot(snrs_met_64qam, evm_lista_64qam, marker='*', linewidth=2, label='64-QAM') # curva EVM de 6
plt.title("EVM vs SNR – Comparativa")   # título del gráfico
plt.xlabel("SNR (dB)"); plt.ylabel("EVM (%)"); plt.grid(True); plt.legend()  # etiquetas, cuadrícul
guardar_figura("comparativa_evm_vs_snr.png")   # guarda la figura
plt.show()    # muestra la figura
```

Nota. Elaboración propia.

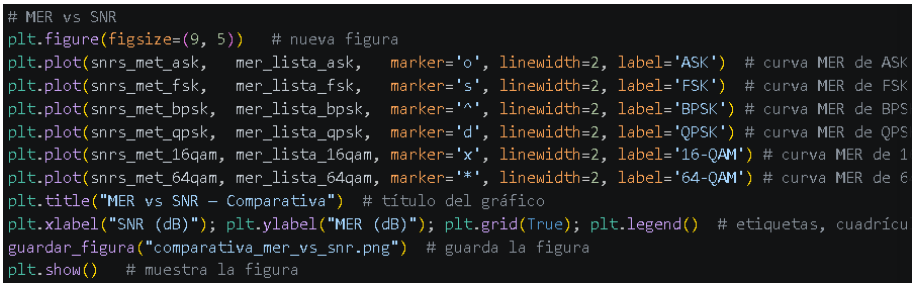
Aplicando tasa de error de modulación

El código mostrado implementa la comparación de la tasa de error de modulación (MER) en función de la relación señal-ruido (SNR) para distintos esquemas de modulación digital.

Este análisis es fundamental, porque permite comparar la robustez de cada esquema frente al ruido. El resultado se guarda como *comparativa_mer_vs_snr.png*, integrando evidencia visual del rendimiento de cada técnica.

Como se observa en la **Figura 35**, la comparación MER vs SNR muestra cómo cada esquema de modulación responde al ruido, destacando la diferencia entre modulaciones simples y de orden superior.

Figura 35. *Comparativa de MER vs SNR en modulaciones digitales.*



```
# MER vs SNR
plt.figure(figsize=(9, 5))   # nueva figura
plt.plot(snrs_met_ask,   mer_lista_ask,   marker='o', linewidth=2, label='ASK')  # curva MER de ASK
plt.plot(snrs_met_fsk,   mer_lista_fsk,   marker='s', linewidth=2, label='FSK')  # curva MER de FSK
plt.plot(snrs_met_bpsk,  mer_lista_bpsk,  marker='^', linewidth=2, label='BPSK') # curva MER de BPS
plt.plot(snrs_met_qpsk,  mer_lista_qpsk,  marker='d', linewidth=2, label='QPSK') # curva MER de QPS
plt.plot(snrs_met_16qam, mer_lista_16qam, marker='x', linewidth=2, label='16-QAM') # curva MER de 1
plt.plot(snrs_met_64qam, mer_lista_64qam, marker='*', linewidth=2, label='64-QAM') # curva MER de 6
plt.title("MER vs SNR – Comparativa")  # título del gráfico
plt.xlabel("SNR (dB)"); plt.ylabel("MER (dB)"); plt.grid(True); plt.legend()  # etiquetas, cuadrícu
guardar_figura("comparativa_mer_vs_snr.png")  # guarda la figura
plt.show()   # muestra la figura
```

Nota. Elaboración propia.

Aplicando tasa de error de bits después de la codificación de errores

El código mostrado implementa la comparación de la tasa de error de bits (BER) después de aplicar corrección Reed-Solomon (post-RS) en función de la relación señal-

ruido (SNR) para distintos esquemas de modulación digital: ASK, FSK, BPSK, QPSK, 16-QAM y 64-QAM.

Este análisis es fundamental, porque muestra la efectividad de los códigos Reed-Solomon en la reducción de errores, especialmente en modulaciones de orden superior que son más sensibles al ruido. El resultado se guarda como *comparativa_ber_post_rs.png*, integrando evidencia visual del rendimiento de cada técnica bajo condiciones de canal con corrección de errores aplicada.

Como se observa en la **Figura 36**, la comparación BER post-RS vs SNR evidencia la mejora en desempeño gracias a la corrección de errores, mostrando cómo cada esquema de modulación se beneficia de la codificación Reed-Solomon.

Figura 36. *Comparativa de BER post-RS vs SNR en modulaciones digitales.*

```
# BER simulado vs SNR
plt.figure(figsize=(9, 5))   # nueva figura
for snrs_m, lista, marca, nombre in [   # recorre cada modulación con su curva de BER simulado
    (snrs_ask,   ber_lista_ask,   'o', 'ASK'),     # datos de ASK
    (snrs_fsk,   ber_lista_fsk,   's', 'FSK'),     # datos de FSK
    (snrs_bpsk,  ber_lista_bpsk,  '^', 'BPSK'),    # datos de BPSK
    (snrs_qpsk,  ber_lista_qpsk,  'd', 'QPSK'),    # datos de QPSK
    (snrs_16qam, ber_lista_16qam, 'x', '16-QAM'),  # datos de 16-QAM
    (snrs_64qam, ber_lista_64qam, '*', '64-QAM'),  # datos de 64-QAM
]:
    x, y = preparar_ber_log(snrs_m, lista)  # prepara la curva
    if len(x):      # si quedaron puntos que graficar
        plt.semilogy(x, y, marker=marca, linewidth=2, label=nombre)  # grafica la curva en escala l
plt.title("BER (post Reed-Solomon) vs SNR — Comparativa")    # título del gráfico; BER simulado del
plt.xlabel("SNR (dB)"); plt.ylabel("BER"); plt.grid(True, which='both'); plt.legend()  # etiquetas,
guardar_figura("comparativa_ber_simulado.png")   # guarda la figura
plt.show()     # muestra la figura
```

Nota. Elaboración propia.

3.8 Generación de reportes automáticos del sistema

El fragmento de código mostrado implementa la generación automática de un reporte de modulación y el almacenamiento de la imagen recibida tras el proceso de transmisión. Primero, se crea un archivo de texto llamado reporte_modulaciones.txt en el que se registran parámetros clave del sistema de comunicación digital: el nombre de la imagen transmitida, el número total de bits y los resultados obtenidos bajo una relación señal-ruido (SNR) determinada. Entre las métricas incluidas se encuentran la tasa de error de bits (BER) antes y después de aplicar corrección de errores Reed-Solomon, la magnitud del vector de error (EVM), la tasa de error de modulación (MER) y la eficiencia espectral.

Además, el reporte documenta los resultados de la simulación, mostrando cómo

varían BER, EVM y MER en función de la SNR, lo que permite evaluar el desempeño del sistema bajo diferentes condiciones de canal. Finalmente, el código convierte la imagen recibida al formato BGR y la guarda como imagen_recibida.png, asegurando que tanto los datos numéricos como la evidencia visual del proceso queden registrados. Este procedimiento integra análisis cuantitativo y evidencia gráfica, facilitando la validación y comparación de distintos esquemas de modulación digital.

Este procedimiento se aplica de manera análoga para cada esquema de modulación, simplemente añadiendo el sufijo correspondiente, lo que permite evaluar de forma consistente el desempeño de diferentes técnicas de modulación digital en términos de fiabilidad, calidad de señal y eficiencia espectral.

Como se observa en la **Figura 37**, el sistema genera un reporte automático con métricas de desempeño (BER, EVM, MER, eficiencia) y guarda la imagen recibida para documentar el proceso de transmisión.

Figura 37. *Generación de reporte de modulación y almacenamiento de imagen recibida.*

```
# REPORTE UNIFICADO
ruta_reporte = os.path.join(carpeta_salida, "reporte_modulaciones.txt")   # ruta del archivo de rep
with open(ruta_reporte, "w", encoding="utf-8") as f:     # abre el archivo de reporte en modo escri
    def seccion(mod, snr_img, ber, evm_img, mer_img, eficiencia,
                snrs, ber_lista, ber_std, snrs_met, evm_lista, evm_std, mer_lista, mer_std, n_iter)
        sep = "="*52     # línea separadora de 52 signos igual
        f.write(f"\n{sep}\n")    # escribe la línea separadora superior
        f.write(f" REPORTE MODULACIÓN {mod}\n")  # escribe el encabezado con el nombre de la modula
        f.write(f"{sep}\n\n")   # escribe la línea separadora inferior
        f.write(f"Imagen        : {nombre_archivo}\n")      # escribe la ruta de la imagen usada
        f.write(f"Bits totales  : {len(bits_imagen)}\n\n")  # escribe el total de bits de la imagen
        f.write(f"RESULTADOS IMAGEN (SNR = {snr_img} dB)\n")  # encabezado de resultados puntuales
        f.write(f"  BER simulado (post-RS) : {ber:.6f}\n")     # escribe el BER simulado del sistem
        f.write(f"  EVM         : {evm_img*100:.4f} %\n") # escribe el EVM en porcentaje
        f.write(f"  MER         : {mer_img:.4f} dB\n")    # escribe el MER en dB
        f.write(f"  Eficiencia  : {eficiencia:.4f} bit/s/Hz\n\n")  # escribe la eficiencia espectra
        f.write(f"BER (post Reed-Solomon) vs SNR  ({n_iter} iter., se reporta media ± desviación es
        f.write(f"{'SNR':^10} {'BER post-RS (media)':^22} {'DE':^14}\n")   # encabezados de columna
        f.write("-"*50+"\n")    # línea separadora de la tabla
        for s, b, bs in zip(snrs, ber_lista, ber_std):  # recorre cada punto de SNR con su BER y su
            f.write(f"{s:^10.0f} {b:^22.6f} {bs:^14.6f}\n")   # escribe la fila de la tabla BER con
        f.write("\nEVM y MER vs SNR  (se reporta media ± desviación estándar)\n")     # encabezado
        f.write(f"{'SNR':^10} {'EVM (%)':^14} {'DE EVM':^12} {'MER (dB)':^14} {'DE MER':^12}\n")  #
        f.write("-"*50+"\n")   # línea separadora de la tabla
        for s, e, es, m, ms in zip(snrs_met, evm_lista, evm_std, mer_lista, mer_std):   # recorre c
            f.write(f"{s:^10.0f} {e:^14.4f} {es:^12.4f} {m:^14.4f} {ms:^12.4f}\n")    # escribe la
```

Nota. Elaboración propia.

CAPÍTULO IV: ANÁLISIS DE RESULTADOS

4.1 Escenarios de simulación evaluados

4.1.1 Fuentes de datos

Se utilizaron dos fuentes de datos para evaluar la robustez del sistema, cada una protegida con un esquema de codificación de canal distinto:

- Una imagen PNG convertida a bits mediante descomposición a nivel de bit con 960 000 bits en total.
- Un archivo de audio WAV convertido a bits de la misma manera, con 1 953 792 bits en total.

Estas cifras (960 000 y 1 953 792 bits) corresponden al tamaño total de cada archivo original y se emplean únicamente en las pruebas de transmisión completa, descritas en las secciones 4.2 y 4.3. Para el cálculo de las curvas de BER, EVM y MER en función de la SNR, se utilizó una cantidad de bits distinta e igual para ambas fuentes de datos, la cual se detalla en el apartado “Condiciones de evaluación”.

4.1.2 Modulaciones evaluadas

El sistema simulado evalúa seis esquemas de modulación digital que representan distintos niveles de complejidad y eficiencia espectral: ASK, FSK y BPSK, como esquemas binarios de menor orden, y QPSK, 16-QAM y 64-QAM, como esquemas de orden superior. A medida que aumenta el orden de la modulación, esta última alcanza una eficiencia espectral mucho mayor, aunque con una tolerancia al ruido progresivamente menor debido a la creciente densidad de puntos en su constelación.

Como se observa en la **Tabla 4**, a medida que aumenta el orden de la modulación, se incrementa el número de bits transmitidos por símbolo, manteniendo los mismos parámetros base de portadora en la mayoría de los esquemas.

Tabla 4. *Comparación de las técnicas de modulación digital evaluadas y sus parámetros de portadora.*

Modulación	Bits/símbolo	Frecuencia portadora	Frecuencia de muestreo	Duración de bit/símbolo
ASK	1	fc = 2000 Hz	8000 Hz	0.001 s

FSK	1	f1 = 3000 Hz (bit=1), f0 = 1000 Hz (bit=0)	8000 Hz	0.001 s
BPSK	1	fc = 2000 Hz	8000 Hz	0.001 s
QPSK	2	fc = 2000 Hz	8000 Hz	0.001 s
16-QAM	4	fc = 2000 Hz	8000 Hz	0.001 s
64-QAM	6	fc = 2000 Hz	8000 Hz	0.001 s

Nota. Elaboración propia.

4.1.3 Modelo de canal

El canal combina desvanecimiento Rayleigh con ruido AWGN ajustado a la SNR objetivo. En el receptor se aplicó ecualización utilizando la ganancia de canal conocida, con el fin de aislar el efecto del ruido sobre el desempeño de cada modulación. Este mismo modelo de canal se mantuvo constante para ambas fuentes de datos (imagen y audio), garantizando así que las diferencias observadas en los resultados respondan exclusivamente al esquema de modulación y codificación empleado y no a variaciones en las condiciones del canal.

4.1.4 Codificación de canal

- Para la imagen se empleó codificación Reed-Solomon (bloques de 20 bytes de datos + 10 de paridad), capaz de corregir hasta 5 bytes erróneos por bloque.
- Para el audio se empleó igualmente codificación Reed-Solomon RS(30,20) (bloques de 20 bytes de datos + 10 bytes de paridad), capaz de corregir hasta 5 bytes erróneos por bloque, aplicada sobre un canal Rayleigh + AWGN con SNR de referencia de 16 dB.

El empleo del mismo esquema de codificación Reed-Solomon tanto para la imagen como para el audio permitió mantener una base común de comparación entre ambos tipos de señal, evitando que las diferencias en el desempeño se debieran a variaciones en la capacidad de corrección de errores.

4.1.5 Condiciones de evaluación

Las simulaciones se realizaron bajo una configuración controlada para garantizar la consistencia de los resultados. Tanto en el script de imagen como en el de audio se fijó de forma independiente una semilla de aleatoriedad global (valor 42) al inicio de la ejecución. Esto garantizó que los bits de prueba generados aleatoriamente para el cálculo de las métricas fueran los mismos en cada corrida, permitiendo la reproducibilidad de los resultados tanto para la fuente de imagen como para la de audio.

En ambas fuentes de datos se distinguen dos tipos de experimentos con distinta cantidad de bits. Por un lado, la transmisión completa del archivo original: 960 000 bits para la imagen PNG y 1 953 792 bits para el audio WAV, utilizados para reconstruir la señal completa a una SNR fija de 16 dB.

Por otro lado, el cálculo de las curvas de BER, EVM y MER en función de la SNR (0 a 16 dB), para el cual se generaron tanto en imagen como en audio 25 000 bits aleatorios independientes para el cálculo de BER y otros 25 000 bits independientes para el cálculo de EVM y MER. Estas dos últimas cantidades son iguales para ambas fuentes de datos y no deben confundirse con los bits totales de cada archivo. En cuanto a la codificación de canal, la transmisión se implementó mediante Reed-Solomon en ambos casos.

4.2 Resultados de transmisión y reconstrucción de la imagen

Esta sección presenta los resultados obtenidos al transmitir la imagen (960 000 bits totales) a través del sistema simulado utilizando una relación señal-ruido (SNR) de 16 dB. Se analiza una transmisión representativa, mediante la cual se evalúa cuantitativamente la calidad de reconstrucción correspondiente a cada esquema de modulación.

4.2.1 Reconstrucción de las imágenes recibidas

A continuación, se presentan las imágenes recibidas tras atravesar el canal Rayleigh + AWGN y ser corregidas mediante codificación Reed-Solomon para cada uno de los seis esquemas de modulación evaluados a SNR fija. Las imágenes reconstruidas permiten observar de forma visual y cualitativa el efecto combinado del ruido del canal y la capacidad de corrección de cada modulación sobre la fidelidad de la información

recuperada, complementando así las métricas cuantitativas BER, EVM, MER y eficiencia espectral.

Imagen transmitida

Como se aprecia en la **Figura 38**, se presenta la imagen original utilizada como referencia para la comparación. Esta versión sin degradación permite evaluar de manera cualitativa el impacto del canal Rayleigh con ruido AWGN y la posterior corrección Reed-Solomon sobre la fidelidad de las imágenes reconstruidas en los diferentes esquemas de modulación digital.

Figura 38. *Imagen de referencia sin degradación.*

Nota. Elaboración propia.

Imagen receptada en ASK

Como se aprecia en la **Figura 39**, se muestra la reconstrucción de la imagen transmitida mediante modulación ASK. Se observa una degradación visual significativa con pérdida de nitidez y detalles, reflejando la limitada robustez de este esquema frente al ruido y complementando las métricas cuantitativas de desempeño.

Figura 39. Imagen reconstruida con modulación ASK.

Nota. Elaboración propia.

Imagen receptada en FSK

Como se aprecia en la **Figura 40**, se muestra la reconstrucción de la imagen transmitida mediante modulación FSK. La imagen presenta una degradación moderada con pérdida parcial de definición, reflejando la sensibilidad de este esquema frente al ruido y la capacidad limitada de corrección en coherencia.

Figura 40. Imagen reconstruida con modulación FSK.

Nota. Elaboración propia.

Imagen receptada en BPSK

Como se aprecia en la **Figura 41**, se muestra la reconstrucción de la imagen transmitida mediante modulación BPSK. La imagen evidencia una degradación notable con pérdida de nitidez y detalles, reflejando la alta sensibilidad de este esquema de modulación al ruido y su menor capacidad de corrección en comparación con modulaciones de mayor orden.

Figura 41. Imagen reconstruida con modulación BPSK.

Nota. Elaboración propia.

Imagen receptada en QPSK

Como se aprecia en la **Figura 42**, se muestra la reconstrucción de la imagen transmitida mediante modulación QPSK. La imagen presenta una degradación más evidente con pérdida de detalles y cierta pixelación, reflejando la mayor vulnerabilidad de este esquema frente al ruido en comparación con modulaciones de menor orden.

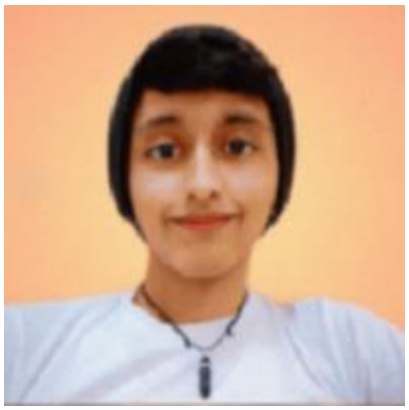
Figura 42. Imagen reconstruida con modulación QPSK.

Nota. Elaboración propia.

Imagen receptada en 16-QAM

Como se aprecia en la **Figura 43**, se muestra la reconstrucción de la imagen transmitida mediante modulación 16-QAM. La imagen conserva buena fidelidad visual con leves artefactos residuales, lo que refleja la mayor robustez de este esquema frente al ruido en comparación con las modulaciones de menor orden.

Figura 43. *Imagen reconstruida con modulación 16-QAM.*



Nota. Elaboración propia.

Imagen receptada en 64-QAM

Como se aprecia en la **Figura 44**, se muestra la reconstrucción de la imagen transmitida mediante modulación 64-QAM. La imagen conserva una alta fidelidad visual con mínima degradación.

Figura 44. *Imagen reconstruida con modulación 64-QAM.*

Nota. Elaboración propia.

La comparación de las imágenes reconstruidas bajo distintos esquemas de modulación digital evidencia cómo el orden de la modulación influye directamente en la fidelidad visual recuperada frente al canal Rayleigh con ruido AWGN y la corrección Reed-Solomon. Mientras las modulaciones de menor orden presentan una degradación visual más marcada en las condiciones evaluadas, las de orden superior conservan una mejor fidelidad de la imagen reconstruida. Este análisis cualitativo complementa las métricas cuantitativas y establece la base para el estudio de las constelaciones digitales, donde se observará de manera más precisa el efecto del ruido en la dispersión de los símbolos a

diferentes SNR.

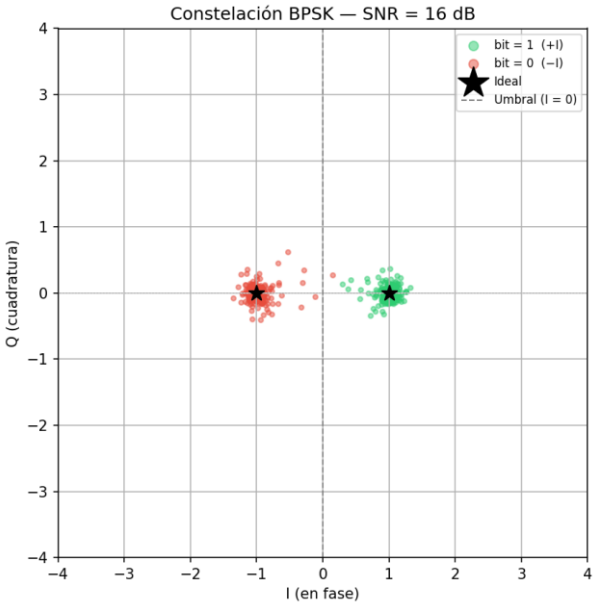
4.2.2 Comparación visual del efecto del ruido en las imágenes por medio de constelaciones

Para las modulaciones que admiten una representación en el plano de fase y cuadratura (BPSK, QPSK, 16-QAM y 64-QAM), se generaron diagramas de constelación a la SNR de referencia, permitiendo observar de manera directa cómo el desvanecimiento Rayleigh y el ruido AWGN dispersan los símbolos recibidos alrededor de sus puntos ideales. Las constelaciones de orden superior, al tener sus puntos más próximos entre sí, resultan visualmente más sensibles al mismo nivel de ruido que las de menor orden, lo cual explica su mayor BER pre-corrección observado en los resultados.

Constelación de modulación BPSK de imágenes

Como se observa en la **Figura 45**, se distinguen claramente los dos estados de símbolo alrededor de sus posiciones ideales, con dispersión reducida gracias al nivel de SNR, lo que refleja una alta fidelidad en la detección y baja probabilidad de error.

Figura 45. Constelación BPSK de imágenes SNR = 16 dB.

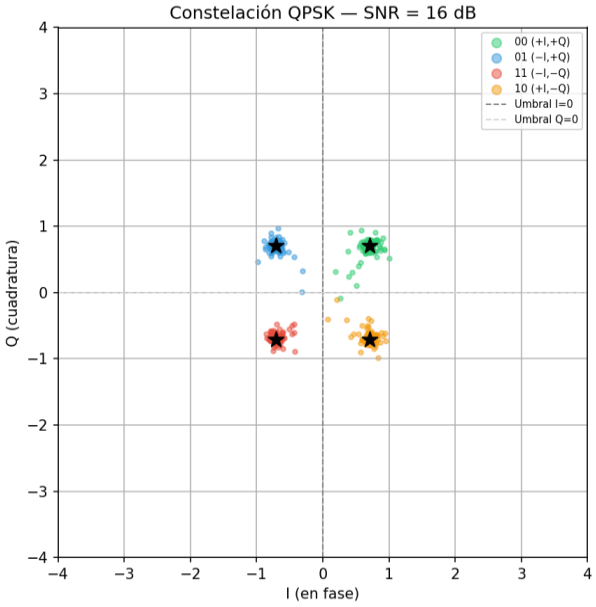


Nota. Elaboración propia.

Constelación de modulación QPSK de imágenes

Como se observa en la **Figura 46**, se presenta la constelación de la modulación QPSK con una relación señal-ruido de 16 dB. Los cuatro estados de símbolo se distinguen claramente alrededor de sus posiciones ideales, con dispersión reducida gracias al nivel de SNR, lo que refleja una adecuada separación de los puntos y una baja probabilidad de error en la detección.

Figura 46. *Constelación QPSK de imágenes SNR = 16 dB.*

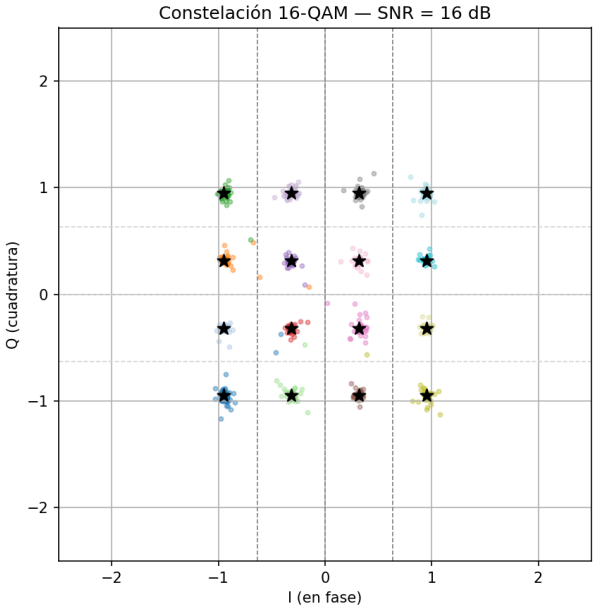


Nota. Elaboración propia.

Constelación de modulación 16-QAM de imágenes

Como se observa en la **Figura 47**, se presenta la constelación de la modulación 16-QAM con una relación señal-ruido de 16 dB. Los dieciséis estados de símbolo se distribuyen alrededor de sus posiciones ideales, mostrando una dispersión moderada producto del ruido. La separación entre los puntos aún permite una correcta detección, aunque se evidencia mayor complejidad y sensibilidad frente al canal en comparación con modulaciones de menor orden.

Figura 47. Constelación 16-QAM de imágenes SNR = 16 dB.

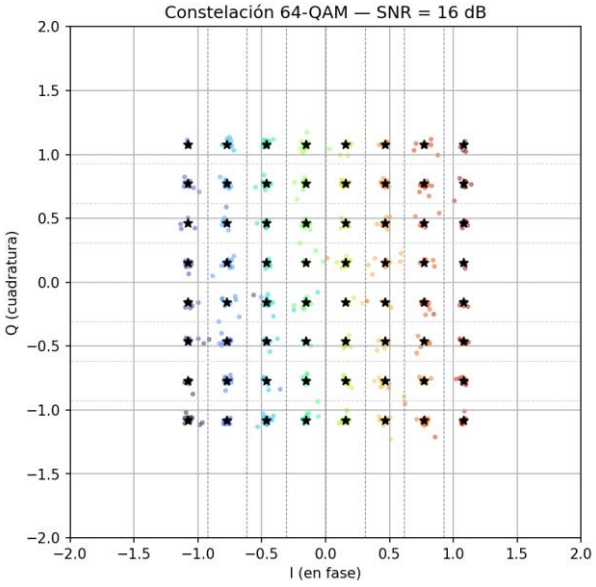


Nota. Elaboración propia.

Constelación de modulación 64-QAM de imágenes

Como se observa en la **Figura 48**, se presenta la constelación de la modulación 64-QAM con una relación señal-ruido de 16 dB. Los sesenta y cuatro estados de símbolo se distribuyen en una cuadrícula densa alrededor de sus posiciones ideales, mostrando una dispersión moderada producto del ruido. Aunque la separación entre puntos aún permite la detección, se evidencia la mayor complejidad y vulnerabilidad de este esquema frente al canal en comparación con modulaciones de menor orden.

Figura 48. Constelación 64-QAM de imágenes SNR = 16 dB.



Nota. Elaboración propia.

Como se observa en las constelaciones presentadas, el nivel de SNR determina directamente la dispersión de los símbolos alrededor de sus posiciones ideales. A medida que el ruido aumenta, los puntos se alejan de los centros de decisión y se incrementa la probabilidad de que el receptor clasifique erróneamente un símbolo. Este comportamiento visual se traduce en errores de bit durante la detección, lo que justifica la medición del BER como métrica fundamental para cuantificar el impacto del ruido y evaluar el desempeño de cada esquema de modulación.

4.2.3 Comparación de BER por técnica de modulación en imágenes

Se presenta la comparación del BER obtenido para cada esquema de modulación después de aplicar la corrección Reed-Solomon a la SNR de referencia de 16 dB.

El BER post-RS muestra el efecto de la codificación de canal sobre esos errores: para la mayoría de las modulaciones, el código Reed-Solomon logra corregir la totalidad de los errores residuales a esta SNR. Esta comparación evidencia que la efectividad de un mismo esquema de corrección de errores depende directamente de la tasa de error de entrada que debe corregir, la cual varía según la modulación empleada.

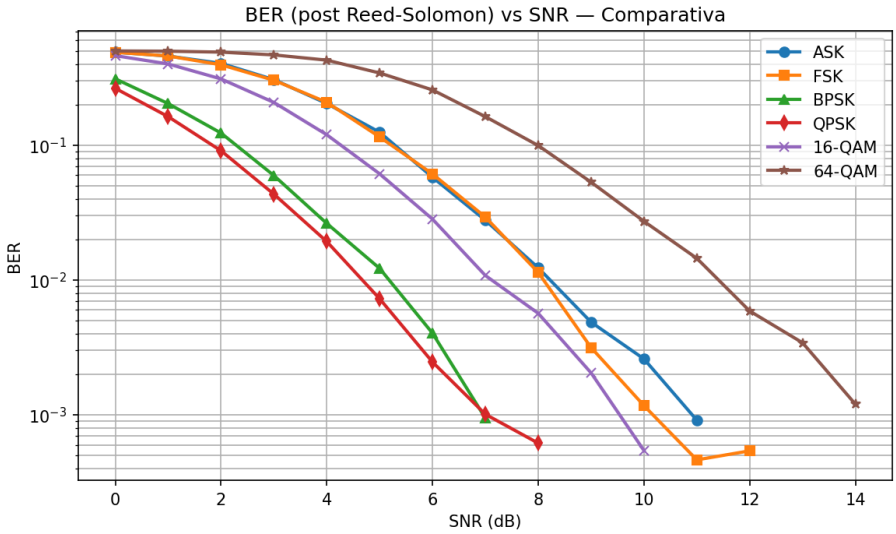
Tasa de error de bit en imágenes

Como se observa en la **Figura 49**, se presenta la comparación del BER posterior a la corrección Reed-Solomon en función de la SNR para los distintos esquemas de modulación digital. Se aprecia una reducción significativa en la tasa de error conforme aumenta la SNR, lo que evidencia la efectividad de la codificación Reed-Solomon en la mejora del desempeño del sistema.

La gráfica muestra que las modulaciones de menor orden, como BPSK y QPSK, alcanzan valores de BER cercanos a cero con menor SNR, confirmando su robustez frente al ruido. En contraste, las modulaciones de orden superior, como 16-QAM y 64-QAM, requieren niveles más altos de SNR para lograr tasas de error similares, reflejando su mayor eficiencia espectral, pero también su sensibilidad a degradaciones del canal.

Los picos o fluctuaciones en el BER se originan por la variabilidad estadística del canal y por fallos ocasionales del decodificador Reed-Solomon al corregir ráfagas de errores. Estas variaciones se acentúan en modulaciones donde la menor separación entre símbolos incrementa la probabilidad de confusión en SNR intermedias.

Figura 49. *Comparativa BER post-RS vs SNR.*



Nota. Elaboración propia.

Como se observa en la **Tabla 5**, el BER simulado decrece rápidamente con el aumento de la SNR, confirmando la mejora en la confiabilidad de la transmisión tras la corrección Reed-Solomon. Las modulaciones de menor orden, como BPSK y QPSK, alcanzan valores nulos de BER a partir de 9–10 dB, evidenciando su robustez frente al

ruido. En contraste, modulaciones de orden superior, como 16-QAM y especialmente 64-QAM, requieren mayor SNR para lograr tasas de error similares, reflejando su mayor eficiencia espectral, pero también su sensibilidad a degradaciones del canal. Este análisis confirma la relación entre complejidad de modulación, corrección de errores y tolerancia al ruido.

Tabla 5. *Resultados de BER vs SNR en imágenes.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.489704	0.488619	0.310735	0.264169	0.461308	0.499859
1	0.458321	0.458947	0.203565	0.162912	0.400449	0.497757
2	0.404607	0.394827	0.123112	0.090968	0.310129	0.489913
3	0.306219	0.303059	0.059612	0.043361	0.207601	0.467248
4	0.203783	0.208120	0.026316	0.019308	0.119573	0.427215
5	0.124792	0.115123	0.012204	0.007268	0.061348	0.343103
6	0.057851	0.061196	0.004035	0.002467	0.028305	0.256967
7	0.027747	0.029295	0.000947	0.001011	0.010807	0.162811
8	0.012316	0.011377	0.000312	0.000617	0.005664	0.099425
9	0.004872	0.003156	0.000113	0.000000	0.002048	0.053181
10	0.002600	0.001171	0.000000	0.000000	0.000544	0.027257
11	0.000913	0.000464	0.000000	0.000000	0.000000	0.014488
12	0.000285	0.000541	0.000000	0.000000	0.000097	0.005888
13	0.000213	0.000000	0.000000	0.000000	0.000000	0.003425
14	0.000000	0.000105	0.000000	0.000000	0.000000	0.001199
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000435
16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000312

Nota. Elaboración propia.

Resultados de BER vs SNR en modulaciones digitales para imágenes (Desviación estándar)

Como se observa en la **Tabla 6**, la desviación estándar del BER tiende a disminuir conforme aumenta la SNR, lo que indica una mayor estabilidad en los resultados de las simulaciones a medida que el canal se vuelve menos ruidoso. En modulaciones simples, como ASK y FSK, la variabilidad inicial es moderada y se reduce hasta valores

prácticamente nulos a partir de 13–14 dB. En modulaciones más robustas, como BPSK y QPSK, la dispersión es mayor en condiciones de baja SNR, pero desaparece rápidamente desde 9–10 dB, reflejando su consistencia frente al ruido.

Por otro lado, modulaciones de mayor orden, como 16-QAM y especialmente 64-QAM, presentan desviaciones estándar más altas en todo el rango de SNR, lo que evidencia una mayor sensibilidad y variabilidad en escenarios ruidosos.

Tabla 6. *Resultados de BER vs SNR en imágenes (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.004768	0.006070	0.016289	0.017760	0.010595	0.000714
1	0.008662	0.014403	0.015884	0.014897	0.012980	0.003013
2	0.018220	0.019093	0.013272	0.015714	0.022015	0.006607
3	0.020643	0.018208	0.016658	0.013297	0.017896	0.008219
4	0.018074	0.019834	0.008436	0.008303	0.016295	0.014790
5	0.018376	0.018153	0.006286	0.004325	0.013907	0.017349
6	0.012918	0.013545	0.003225	0.002521	0.008666	0.018306
7	0.008120	0.008043	0.001451	0.001738	0.004831	0.016582
8	0.006413	0.005291	0.000939	0.001237	0.003500	0.012011
9	0.004127	0.002552	0.000610	0.000000	0.002162	0.011649
10	0.002654	0.002218	0.000000	0.000000	0.001221	0.009019
11	0.001708	0.001187	0.000000	0.000000	0.000000	0.006813
12	0.000856	0.001211	0.000000	0.000000	0.000524	0.003179
13	0.000800	0.000000	0.000000	0.000000	0.000000	0.003809
14	0.000000	0.000567	0.000000	0.000000	0.000000	0.001763
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000936
16	0.000560	0.000000	0.000000	0.000000	0.000000	0.001112

Nota. Elaboración propia. Desviación estándar sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre BER teórico y BER simulado

Con el fin de complementar la validación visual de los algoritmos frente a las curvas teóricas conocidas, se realizó una comparación cuantitativa independiente entre el BER teórico (**Tabla 1**) y el BER simulado post Reed-Solomon (**Tabla 5**) para cada

esquema de modulación en el rango de 0 a 16 dB de SNR. Para ello, se calculó el error absoluto medio (MAE) y la raíz del error cuadrático medio (RMSE) tanto en escala lineal como en escala logarítmica ($\log_{10}$), además del coeficiente de determinación (R^2) sobre $\log_{10}$(BER). El uso de la escala logarítmica es necesario porque el BER varía en varios órdenes de magnitud a lo largo del rango de SNR, lo que haría que los valores más altos dominen el cálculo si se trabajara únicamente en escala lineal.

Como se observa en la **Tabla 7**, BPSK y QPSK presentan el mejor ajuste respecto al modelo teórico (R^2 = 0.968 y 0.942, respectivamente), confirmando que su comportamiento simulado sigue de cerca la curva teórica AWGN.

FSK, 16-QAM y 64-QAM muestran un ajuste moderado (R^2 entre 0.79 y 0.87), mientras que ASK presenta la mayor discrepancia relativa (R^2 = 0.648). En términos absolutos, el error medio (MAE lineal) se mantiene por debajo de 0.13 en todos los casos. Las mayores diferencias se concentran en SNR bajas e intermedias (0–6 dB), donde el BER simulado es consistentemente mayor al teórico, lo cual es esperable dado que la Tabla 5 no proviene de un canal AWGN puro, sino de la transmisión de imágenes con corrección Reed-Solomon, cuya efectividad depende de la longitud de las ráfagas de error y solo se vuelve comparable al modelo teórico a partir de SNR moderadas-altas, donde ambas curvas convergen hacia BER ≈ 0.

Tabla 7. *Comparación cuantitativa entre BER teórico y BER simulado por esquema de modulación.*

Modulación	**MAE (lineal)**	**RMSE (lineal)**	**MAE ($\log_{10}$)**	**RMSE ($\log_{10}$)**	**R^2 ($\log_{10}$)**
ASK	0.0627	0.1084	0.623	0.826	0.648
FSK	0.0861	0.1499	0.518	0.696	0.866
BPSK	0.0308	0.0705	0.257	0.347	0.968
QPSK	0.0221	0.0538	0.276	0.469	0.942
16-QAM	0.0613	0.1212	0.499	0.815	0.789
64-QAM	0.1318	0.1955	0.326	0.372	0.807

Nota. Elaboración propia. MAE: error absoluto medio; RMSE: raíz del error cuadrático medio; R^2: coeficiente de determinación calculado sobre $\log_{10}$(BER).

A modo de ejemplo, se detalla el cálculo del error para la modulación BPSK en

dos puntos representativos del rango de SNR, que ilustran la evolución del ajuste conforme aumenta la relación señal-ruido: SNR = 7 dB, donde ambos valores de BER son aún significativos y comparables en magnitud; SNR = 10 dB, donde el BER teórico se aproxima al piso de resolución numérica.

Ejemplo de cálculo — BPSK, SNR = 7 dB

- BER teórico (Tabla 1): 0.000773
- BER simulado (Tabla 5): 0.000947

Error absoluto:

$$\varepsilon_{abs} = | \, 0.000947 - 0.000773 \, | = 0.000174$$

Error relativo:

$$\varepsilon_{rel} = \frac{0.000174}{0.000773} \times 100\% = +22.5\%$$

En este punto ambos valores son de magnitud comparable, por lo que el error relativo es una métrica confiable. El BER simulado es un 22.5 % mayor que el teórico, diferencia atribuible a la variabilidad estadística del canal simulado y a la naturaleza probabilística de la corrección Reed-Solomon.

Ejemplo de cálculo — BPSK, SNR = 10 dB

- BER teórico (Tabla 1): 0.000004
- BER simulado (Tabla 5): 0.000000

Error absoluto:

$$\varepsilon_{abs} = | \, 0.000000 - 0.000004 \, | = 0.000004$$

Error relativo:

$$\varepsilon_{rel} = \frac{0.000004}{0.000004} \times 100\% = -100\%$$

Aunque el error relativo parece elevado, el error absoluto es despreciable, lo que ilustra por qué el error relativo pierde significancia cuando el BER teórico se aproxima a cero. En estos casos, el error absoluto o la comparación en escala logarítmica son las métricas más representativas.

4.2.4 Comparación de MER, EVM e histograma del error en imágenes

Se presenta la comparación de la magnitud del vector de error (EVM) y la tasa de error de modulación (MER) para las seis modulaciones evaluadas a la SNR de referencia, complementada con el histograma de la distribución del error de amplitud entre la señal transmitida y la recibida en cada caso.

El EVM cuantifica en términos porcentuales cuánto se aparta en promedio la señal recibida de su valor ideal, mientras que el MER expresa esa misma relación en escala logarítmica (dB), de forma análoga a una relación señal-ruido efectiva del enlace modulado. Ambas métricas son inversamente proporcionales entre sí y consistentes con el comportamiento esperado según el orden de cada modulación.

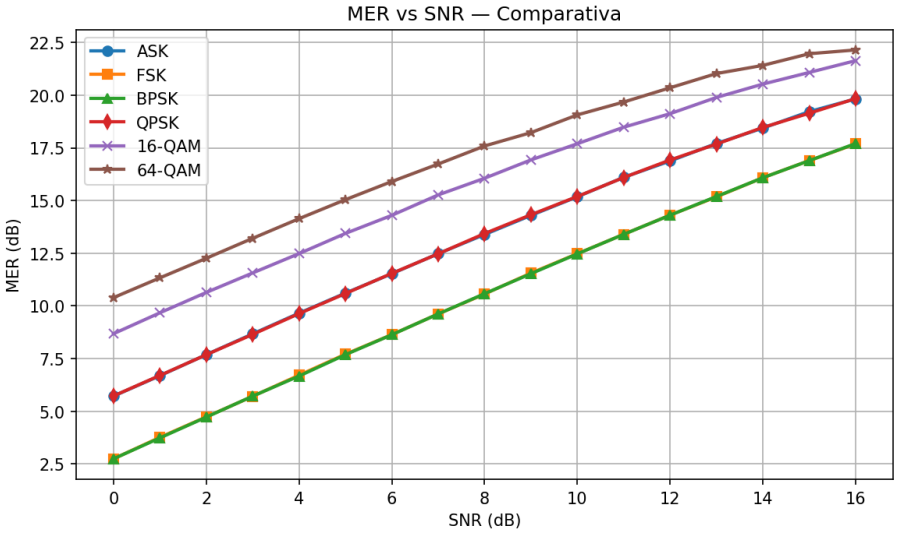
MER de las modulaciones evaluadas en imágenes

Como se observa en la **Figura 50**, se presenta la comparación del MER en función de la SNR para los distintos esquemas de modulación digital. Se aprecia una tendencia creciente del MER conforme aumenta la relación señal-ruido, lo que refleja una mejora en la calidad de la modulación y una menor dispersión de los símbolos respecto a sus posiciones ideales en la constelación.

La gráfica muestra que las modulaciones de orden superior, como 64-QAM y 16-QAM, alcanzan valores de MER más altos para la misma SNR, evidenciando su mayor eficiencia espectral y capacidad de transmitir más bits por símbolo. En contraste, modulaciones de menor orden, como ASK, FSK y BPSK, muestran valores más bajos de MER, confirmando su robustez frente al canal, aunque con menor eficiencia espectral.

Al graficar los valores de MER, algunos puntos pueden parecer solaparse entre distintas modulaciones. Esto ocurre porque varias curvas comparten pendientes similares y se representan en la misma escala. En modulaciones de orden bajo, los valores de MER crecen casi de forma paralela, lo que genera coincidencias visuales.

Figura 50. Comparativa MER vs SNR de imágenes.



Nota. Elaboración propia.

Como se observa en la **Tabla 8**, el MER simulado aumenta progresivamente con la SNR, reflejando la mejora en la calidad de la señal y la reducción de errores de modulación. Las modulaciones de menor orden, como BPSK y FSK, presentan valores de MER más bajos en todo el rango, lo que indica robustez, pero menor eficiencia espectral. En contraste, las modulaciones de orden superior, como 16-QAM y 64-QAM, alcanzan valores de MER más altos para la misma SNR, evidenciando su mayor eficiencia espectral y capacidad de transmitir más bits por símbolo, aunque con mayor sensibilidad a degradaciones del canal.

Este análisis confirma la relación entre MER, SNR y orden de modulación, siendo el MER una métrica clave para evaluar la linealidad, precisión de constelación y desempeño del transmisor en sistemas digitales de comunicación.

Tabla 8. Resultados de MER vs SNR en imágenes.

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	5.72	2.74	2.74	5.72	8.68	10.39
1	6.68	3.76	3.73	6.70	9.67	11.33
2	7.68	4.73	4.72	7.68	10.64	12.26
3	8.67	5.71	5.71	8.65	11.56	13.21
4	9.65	6.71	6.67	9.63	12.48	14.14
5	10.60	7.71	7.68	10.59	13.44	15.04
6	11.53	8.63	8.63	11.54	14.30	15.90

7	12.48	9.62	9.62	12.47	15.27	16.73
8	13.39	10.56	10.58	13.43	16.06	17.59
9	14.29	11.56	11.53	14.33	16.93	18.22
10	15.18	12.48	12.47	15.19	17.70	19.07
11	16.10	13.40	13.40	16.09	18.48	19.68
12	16.88	14.31	14.30	16.93	19.12	20.35
13	17.71	15.18	15.19	17.68	19.90	21.03
14	18.44	16.08	16.08	18.47	20.53	21.42
15	19.22	16.90	16.89	19.14	21.08	21.97
16	19.83	17.69	17.71	19.84	21.64	22.15

Nota. Elaboración propia.

Resultados de MER vs SNR en modulaciones digitales para imágenes (Desviación estándar)

La **Tabla 9** muestra la dispersión de los resultados en cada nivel de SNR. Se observa que la desviación estándar se mantiene baja en modulaciones robustas, como BPSK y QPSK, mientras que en modulaciones de mayor orden (16-QAM y 64-QAM) la variabilidad es más alta, reflejando su mayor sensibilidad al ruido.

Tabla 9. *Resultados de MER vs SNR en imágenes (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.08	0.07	0.07	0.07	0.11	0.17
1	0.09	0.08	0.05	0.10	0.12	0.14
2	0.08	0.08	0.08	0.11	0.12	0.16
3	0.06	0.07	0.07	0.09	0.18	0.21
4	0.09	0.10	0.07	0.10	0.11	0.16
5	0.09	0.09	0.07	0.12	0.17	0.19
6	0.09	0.09	0.08	0.12	0.15	0.19
7	0.10	0.08	0.06	0.11	0.15	0.18
8	0.09	0.08	0.07	0.10	0.17	0.23
9	0.07	0.10	0.07	0.09	0.15	0.27
10	0.08	0.10	0.07	0.11	0.22	0.24

11	0.12	0.10	0.06	0.15	0.24	0.37
12	0.11	0.08	0.09	0.12	0.22	0.35
13	0.11	0.09	0.08	0.14	0.25	0.34
14	0.14	0.10	0.08	0.13	0.24	0.36
15	0.13	0.10	0.08	0.16	0.28	0.45
16	0.18	0.11	0.09	0.17	0.29	0.39

Nota. Elaboración propia. Desviación estándar calculada sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre MER simulado y MER teórico

De forma análoga a la validación realizada para el BER, se efectuó una comparación cuantitativa entre el MER teórico (**Tabla 2**) y el MER simulado en imágenes (**Tabla 8**) para cada esquema de modulación en el rango de 0 a 16 dB de SNR. Dado que el MER ya se expresa en escala logarítmica (dB), las métricas de error se calcularon directamente en esta escala sin necesidad de una transformación adicional como en el caso del BER. Se calculó el error absoluto medio (MAE), la raíz del error cuadrático medio (RMSE) y el coeficiente de determinación (R^2) entre ambas series.

Como se observa en la **Tabla 10**, en el caso del MER las modulaciones de orden superior (16-QAM y 64-QAM) presentan el mejor ajuste relativo ($R^2 = 0.841$ y 0.863), mientras que ASK muestra un ajuste deficiente ($R^2 = -0.131$), lo que indica que el MER simulado de ASK no sigue adecuadamente la pendiente unitaria (1 dB/dB) esperada por el modelo teórico. En todas las modulaciones se observa un offset sistemático positivo entre el MER simulado y el teórico (entre 1.6 y 5.2 dB en promedio), es decir, el sistema simulado logra un MER ligeramente mayor al esperado por el modelo ideal para la misma SNR. Esta diferencia constante sugiere que el MER simulado no solo depende del ruido de canal modelado por la SNR, sino también de factores adicionales de la cadena de transmisión de imágenes que no están contemplados en el modelo teórico ideal.

Tabla 10. *Comparación cuantitativa entre MER teórico y MER simulado por esquema de modulación.*

Modulación	MAE (dB)	RMSE (dB)	R^2
ASK	5.179	5.209	−0.131

FSK	2.457	2.477	0.744
BPSK	2.450	2.469	0.746
QPSK	2.171	2.243	0.790
16-QAM	1.759	1.951	0.841
64-QAM	1.635	1.814	0.863

Nota. Elaboración propia. MAE: error absoluto medio en dB; RMSE: raíz del error cuadrático medio en dB; R²: coeficiente de determinación respecto al MER teórico.

A modo de ejemplo, se detalla el cálculo del error para la modulación QPSK en tres puntos representativos del rango de SNR.

SNR = 0 dB, donde la diferencia relativa es mayor; SNR = 8 dB, un punto intermedio; y SNR = 16 dB, donde el error relativo se reduce considerablemente, aunque sin llegar a converger por completo, a diferencia del comportamiento observado en el BER.

Ejemplo de cálculo — QPSK, SNR = 0 dB

- MER teórico (Tabla 2): 3.01 dB
- MER simulado (Tabla 8): 5.72 dB

Error absoluto:

$$\varepsilon_{abs} = | \, 5.72 - 3.01 \, | = 2.71 \; dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{2.71}{3.01} \times 100\% = +90.0\%$$

En SNR bajas, el error relativo es elevado porque el valor teórico de referencia es pequeño, por lo que cualquier offset absoluto se traduce en un porcentaje grande.

Ejemplo de cálculo — QPSK, SNR = 8 dB

- MER teórico: 11.01 dB
- MER simulado: 13.43 dB

Error absoluto:

$$\varepsilon_{abs} = | \, 13.43 - 11.01 \, | = 2.42 \; dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{2.42}{11.01} \times 100\% = +22.0\%$$

El error absoluto se mantiene prácticamente constante respecto al punto anterior, pero el error relativo disminuye considerablemente al crecer el valor de referencia, confirmando el comportamiento de offset sistemático mencionado en el análisis.

Ejemplo de cálculo — QPSK, SNR = 16 dB

- MER teórico: 19.01 dB
- MER simulado: 19.84 dB

Error absoluto:

$$\varepsilon_{abs} = |\ 19.84 - 19.01\ | = 0.83\ dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{0.83}{19.01} \times 100\% = +4.4\%$$

A diferencia del BER, donde el error tiende a desaparecer completamente en SNR altas, en el MER el offset se reduce, pero no se anula del todo. En conjunto, estos tres ejemplos muestran que, si bien el MER simulado sigue la tendencia creciente esperada por el modelo teórico, existe una brecha sistemática que no se cierra completamente incluso en SNR altas, comportamiento distinto al observado en el BER.

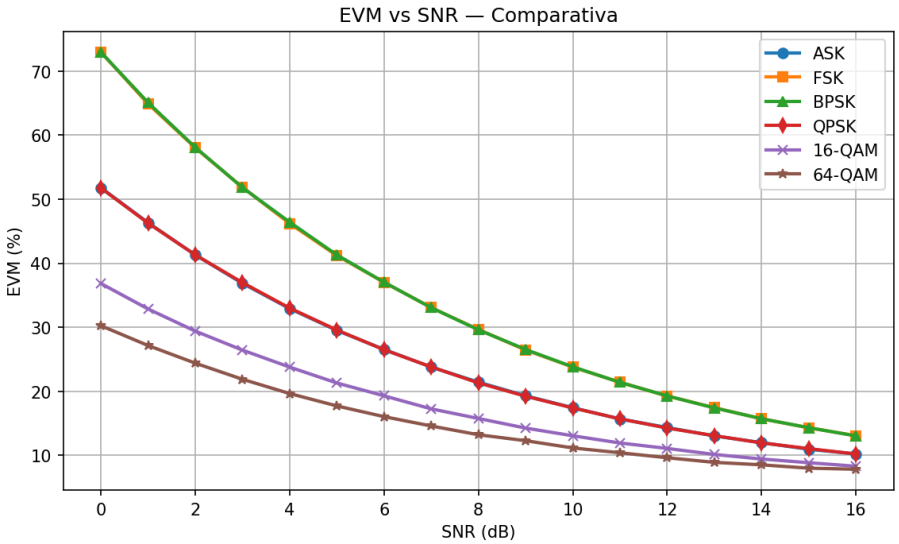
EVM de las modulaciones evaluadas en imágenes

Como se observa en la **Figura 51**, se presenta la comparación del EVM en función de la SNR para los distintos esquemas de modulación digital. Se aprecia una tendencia decreciente del EVM conforme aumenta la relación señal-ruido, lo que refleja una mejora en la calidad de la señal y menor dispersión de los símbolos.

Los picos en el EVM se originan por la variabilidad estadística del canal y el ruido en la estimación de los vectores de error, generando aumentos temporales en la dispersión de los símbolos. Son más notorios en modulaciones donde la cercanía entre puntos de la constelación incrementa la sensibilidad en SNR intermedias.

Al observar la nueva gráfica comparativa de EVM vs SNR, se aprecia que varias curvas de las modulaciones digitales tienden a solaparse visualmente en ciertos rangos de SNR. Esto no significa que los resultados sean idénticos, sino que las diferencias entre ellas son muy pequeñas y se encuentran dentro de la misma escala de representación.

Figura 51. Comparativa EVM vs SNR de imágenes.



Nota. Elaboración propia.

Como se observa en la **Tabla 11**, el EVM simulado decrece con el aumento de la SNR, reflejando la mejora en la calidad de la constelación y la reducción de errores de modulación. Las modulaciones de menor orden, como BPSK y FSK, presentan valores de EVM más altos en bajas SNR, lo que indica mayor dispersión de símbolos, pero también mayor tolerancia al ruido. En contraste, modulaciones de orden superior, como 16-QAM y 64-QAM, muestran valores de EVM más bajos para la misma SNR, reflejando su mayor eficiencia espectral y mejor aprovechamiento de potencia, aunque requieren mayor linealidad y precisión en el transmisor.

Este análisis confirma la relación inversa entre EVM y MER, siendo ambas métricas complementarias para evaluar la calidad de señal y desempeño de sistemas digitales de comunicación.

Tabla 11. Resultados de EVM vs SNR en imágenes.

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	51.79	72.97	72.96	51.74	36.83	30.23

1	46.33	64.89	65.08	46.26	32.85	27.14
2	41.29	58.02	58.10	41.31	29.39	24.39
3	36.86	51.85	51.81	36.96	26.42	21.87
4	32.91	46.20	46.41	32.99	23.77	19.64
5	29.50	41.19	41.29	29.56	21.29	17.70
6	26.52	37.01	37.03	26.48	19.28	16.03
7	23.78	33.06	33.05	23.80	17.25	14.57
8	21.41	29.64	29.59	21.30	15.75	13.20
9	19.29	26.44	26.52	19.22	14.24	12.27
10	17.42	23.78	23.80	17.39	13.04	11.14
11	15.67	21.39	21.39	15.68	11.91	10.39
12	14.33	19.26	19.27	14.25	11.06	9.61
13	13.02	17.43	17.39	13.06	10.13	8.89
14	11.97	15.70	15.70	11.93	9.41	8.50
15	10.94	14.30	14.30	11.04	8.83	7.98
16	10.20	13.05	13.02	10.19	8.28	7.82

Nota. Elaboración propia.

Resultados de EVM vs SNR en modulaciones digitales para imágenes (Desviación estándar)

La **Tabla 12** muestra que la desviación estándar se mantiene baja en modulaciones robustas, como BPSK y QPSK, mientras que en modulaciones de mayor orden (16-QAM y 64-QAM) la variabilidad es más alta, reflejando su mayor sensibilidad al ruido.

Tabla 12. *Resultados de EVM vs SNR en imágenes (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.46	0.63	0.57	0.42	0.49	0.59
1	0.48	0.56	0.40	0.54	0.45	0.44
2	0.37	0.54	0.57	0.50	0.42	0.43
3	0.27	0.40	0.40	0.38	0.54	0.52
4	0.33	0.51	0.36	0.39	0.30	0.37
5	0.29	0.44	0.32	0.41	0.42	0.38

6	0.27	0.38	0.33	0.36	0.34	0.35
7	0.27	0.30	0.22	0.29	0.30	0.31
8	0.21	0.28	0.25	0.25	0.31	0.36
9	0.16	0.30	0.20	0.20	0.25	0.38
10	0.16	0.28	0.19	0.22	0.33	0.31
11	0.22	0.25	0.15	0.27	0.33	0.44
12	0.18	0.18	0.19	0.20	0.28	0.38
13	0.17	0.17	0.16	0.20	0.29	0.35
14	0.19	0.19	0.15	0.18	0.26	0.35
15	0.16	0.16	0.13	0.21	0.29	0.41
16	0.22	0.16	0.14	0.19	0.28	0.36

Nota. Elaboración propia. Desviación estándar calculada sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre EVM simulado y EVM teórico

Se realizó una comparación cuantitativa entre el EVM teórico (**Tabla 3**) y el EVM simulado en imágenes (**Tabla 11**) para cada esquema de modulación en el rango de 0 a 16 dB de SNR. Al tratarse de una magnitud expresada en porcentaje con un rango dinámico moderado, las métricas de error se calcularon directamente en escala lineal: error absoluto medio (MAE), raíz del error cuadrático medio (RMSE) y coeficiente de determinación (R^2) entre ambas series.

Como se observa en la **Tabla 13**, el ajuste mejora de forma consistente al aumentar el orden de modulación: ASK presenta el peor ajuste ($R^2 = 0.026$, MAE = 21.55 %), mientras que 64-QAM muestra el mejor ($R^2 = 0.764$, MAE = 3.87 %). En todos los casos, el EVM simulado es sistemáticamente menor que el EVM teórico, con una razón promedio simulado/teórico de entre 0.55 (ASK) y 0.87 (64-QAM). Un EVM simulado menor implica necesariamente un MER simulado mayor, confirmando que ambos análisis describen el mismo fenómeno subyacente desde ángulos complementarios.

Tabla 13. *Comparación cuantitativa entre EVM teórico y EVM simulado por esquema de modulación.*

Modulación	**MAE (%)**	**RMSE (%)**	**R^2**	**Razón media (sim/teo)**

ASK	21.55	25.07	0.026	0.552
FSK	11.97	14.02	0.695	0.754
BPSK	11.94	13.98	0.697	0.755
QPSK	7.95	9.60	0.715	0.781
16-QAM	5.05	6.46	0.741	0.826
64-QAM	3.87	5.04	0.764	0.870

Nota. Elaboración propia. MAE: error absoluto medio en puntos porcentuales; RMSE: raíz del error cuadrático medio en puntos porcentuales; R²: coeficiente de determinación respecto al EVM teórico (Tabla 3); razón media: promedio de EVM simulado / EVM teórico en todo el rango de SNR.

A modo de ejemplo, se detalla el cálculo del error para la modulación ASK —la de mayor discrepancia relativa— en tres puntos representativos del rango de SNR: SNR = 0 dB, SNR = 8 dB y SNR = 16 dB.

Ejemplo de cálculo — ASK, SNR = 0 dB

- EVM teórico (Tabla 3): 100.00 %
- EVM simulado (Tabla 11): 51.79 %

Error absoluto:

$$\varepsilon_{abs} = |\ 51.79 - 100.00\ | = 48.21 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{-48.21}{100.00} \times 100\% = -48.2\%$$

Ejemplo de cálculo — ASK, SNR = 8 dB

- EVM teórico: 39.81 %
- EVM simulado: 21.41 %

Error absoluto:

$$\varepsilon_{abs} = |\ 21.41 - 39.81\ | = 18.40 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{-18.40}{39.81} \times 100\% = -46.2\%$$

Ejemplo de cálculo — ASK, SNR = 16 dB

- EVM teórico: 15.85 %
- EVM simulado: 10.20 %

Error absoluto:

$$\varepsilon_{abs} = |\ 10.20 - 15.85\ | = 5.65 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{-5.65}{15.85} \times 100\% = -35.6\%$$

A diferencia del BER y del MER, en el EVM de ASK el error relativo se mantiene notablemente estable a lo largo de todo el rango (entre −48 % y −36 %), lo que confirma que la discrepancia no es un efecto puntual de baja SNR, sino un sesgo proporcional constante presente en toda la curva.

En conjunto, los resultados de BER, MER y EVM cuentan una historia consistente: el sistema simulado se aproxima razonablemente al modelo teórico en tendencia general.

Histograma del error de las modulaciones evaluadas en imágenes

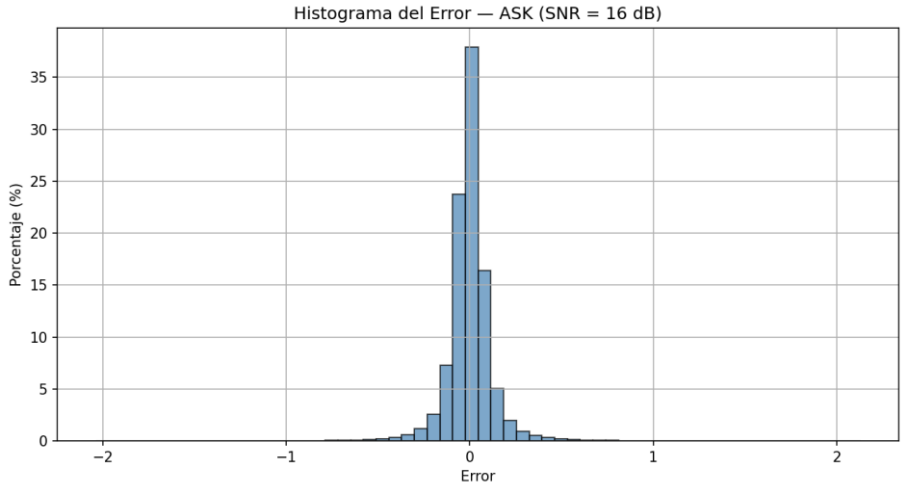
En el histograma del error se representa la diferencia de amplitud punto a punto entre la señal moduladora transmitida y la señal recibida tras atravesar el canal Rayleigh + AWGN. Para cada modulación, se calcula la resta directa entre ambas formas de onda (señal_tx − señal_rx), obteniendo así un vector de error continuo que refleja el efecto conjunto del desvanecimiento y el ruido sobre la señal antes de cualquier proceso de decisión o corrección de canal.

El histograma del error constituye así el complemento visual de las métricas EVM y MER: mientras estas últimas resumen la dispersión del error en un único valor numérico, el histograma muestra la distribución completa de los errores. Una distribución concentrada alrededor de cero es indicativa de un EVM bajo y un MER alto, evidenciando una señal recibida muy próxima a la transmitida; en cambio, una distribución más dispersa o con colas más anchas refleja una mayor degradación introducida por el canal, consistente con un EVM elevado y un MER reducido para esa modulación.

Histograma del error de ASK para imágenes

Como se observa en la **Figura 52**, la distribución se concentra alrededor de cero, indicando que la mayoría de los símbolos recibidos presentan un error mínimo. Este comportamiento refleja la estabilidad del esquema bajo condiciones de SNR relativamente altas.

Figura 52. *Histograma del error — ASK de imágenes.*

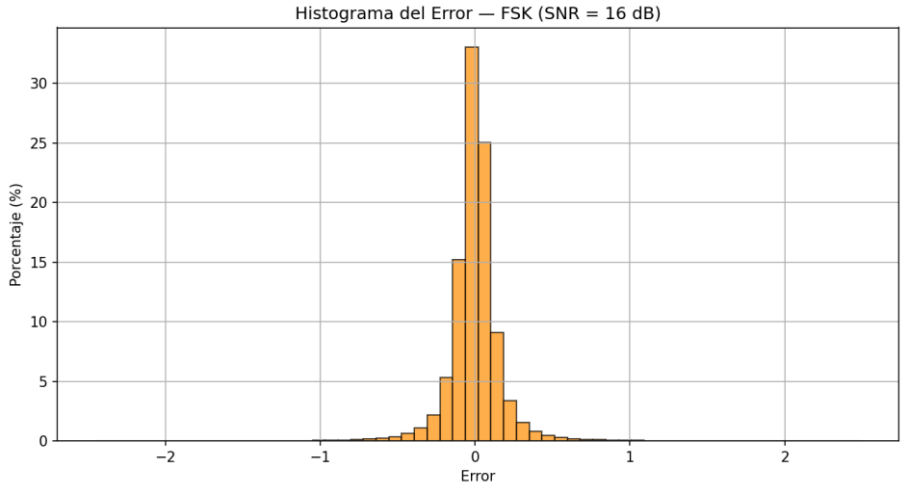


Nota. Elaboración propia.

Histograma del error de FSK para imágenes

Como se observa en la **Figura 53**, la distribución se concentra alrededor de cero, indicando que la mayoría de los símbolos recibidos presentan un error reducido. Este comportamiento refleja una adecuada estabilidad del esquema bajo condiciones de SNR relativamente altas y complementa las métricas de desempeño al mostrar la dispersión estadística de los errores.

Figura 53. *Histograma del error — FSK de imágenes.*

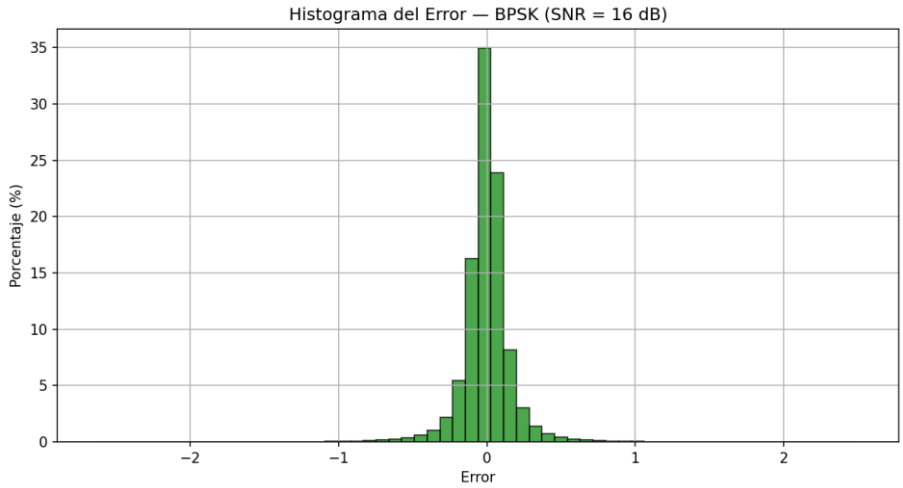


Nota. Elaboración propia.

Histograma del error de BPSK para imágenes

Como se observa en la **Figura 54**, la distribución se concentra alrededor de cero, mostrando que la mayoría de los símbolos recibidos presentan errores mínimos. Este comportamiento refleja la estabilidad del esquema.

Figura 54. *Histograma del error — BPSK de imágenes.*

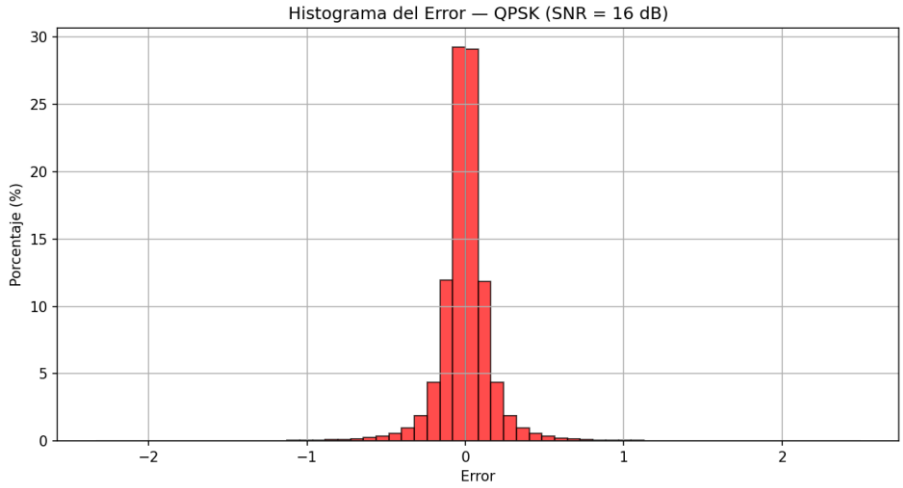


Nota. Elaboración propia.

Histograma del error de QPSK para imágenes

Como se observa en la **Figura 55**, la distribución se concentra alrededor de cero, mostrando que la mayoría de los símbolos recibidos presentan errores mínimos. Este comportamiento refleja la estabilidad del esquema bajo condiciones de SNR relativamente altas.

Figura 55. *Histograma del error — QPSK de imágenes.*

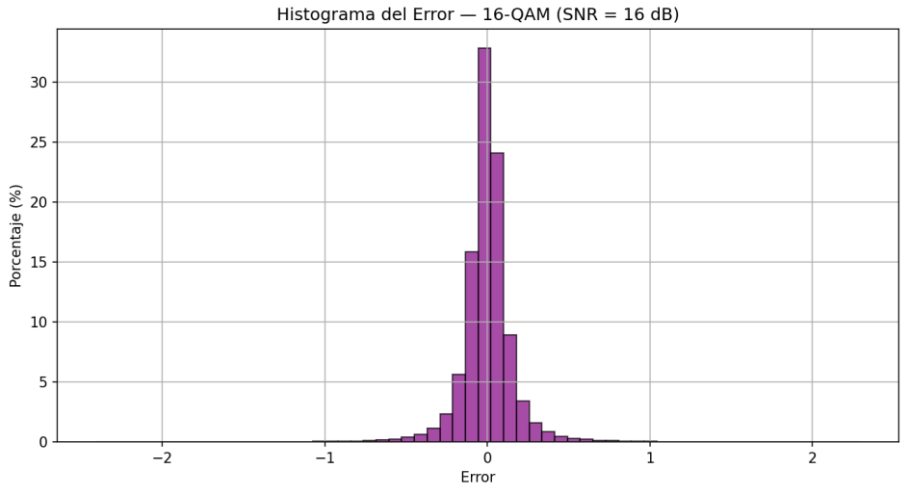


Nota. Elaboración propia.

Histograma del error de 16-QAM para imágenes

Como se observa en la **Figura 56**, la distribución se concentra alrededor de cero, aunque con una dispersión mayor que en modulaciones de menor orden, reflejando la mayor complejidad y sensibilidad de este esquema frente al ruido.

Figura 56. *Histograma del error — 16-QAM de imágenes.*

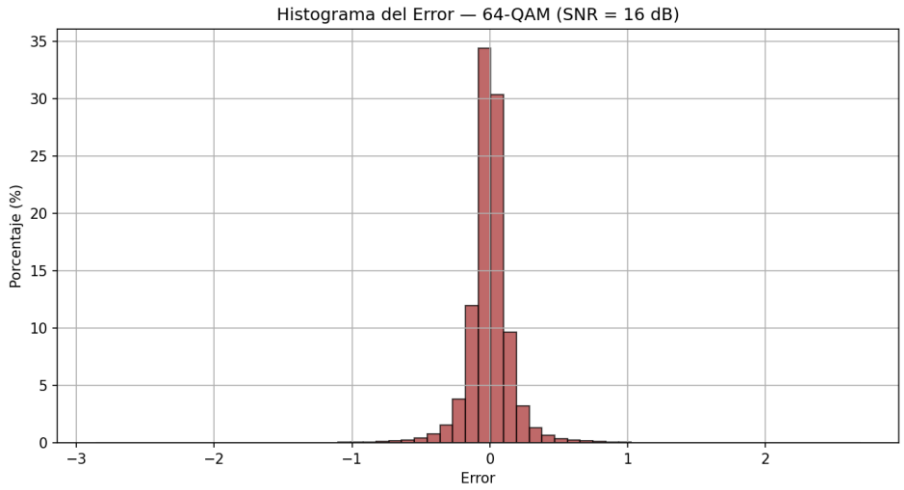


Nota. Elaboración propia.

Histograma del error de 64-QAM para imágenes

Como se observa en la **Figura 57**, la distribución se concentra alrededor de cero, aunque con una dispersión mayor que en modulaciones de menor orden, reflejando la complejidad y sensibilidad de este esquema frente al ruido.

Figura 57. *Histograma del error — 64-QAM de imágenes.*



Nota. Elaboración propia.

4.3 Resultados de transmisión y reconstrucción del audio

Esta sección presenta los resultados obtenidos al transmitir el audio completo (1 953 792 bits) a través del sistema simulado empleando un canal Rayleigh con ruido AWGN y fijando la SNR de referencia en 16 dB. Se presentan los resultados de una transmisión representativa, lo cual permite evaluar de forma cuantitativa, mediante métricas de BER, EVM y MER, la calidad de reconstrucción de la señal para cada uno de los seis esquemas de modulación evaluados.

4.3.1 Reconstrucción de las ondas sonoras recibidas

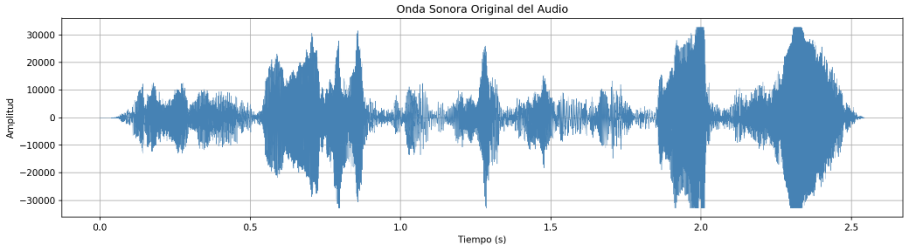
A continuación, se presentan los audios recibidos tras atravesar el canal Rayleigh + AWGN y ser corregidos mediante codificación Reed-Solomon para cada uno de los seis esquemas de modulación evaluados a SNR fija. Las ondas sonoras reconstruidas permiten apreciar de forma auditiva y cualitativa el efecto combinado del ruido del canal y la capacidad de corrección de cada modulación sobre la fidelidad de la información recuperada, complementando así las métricas cuantitativas (BER, EVM, MER).

Onda sonora del audio original enviado

Como se aprecia en la **Figura 58**, se presenta el audio original utilizado como referencia para la comparación. Esta versión sin degradación permite evaluar de manera cualitativa el impacto del canal Rayleigh con ruido AWGN y la posterior corrección

Reed-Solomon sobre la fidelidad de los audios reconstruidos en los diferentes esquemas de modulación digital.

Figura 58. Audio de referencia sin degradación.

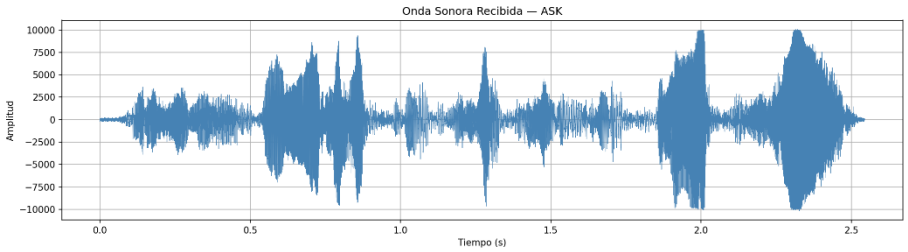


Nota. Elaboración propia.

Onda sonora del audio ASK receptado

En la **Figura 59**, se muestra la reconstrucción del audio transmitido mediante modulación ASK. Se observa una reducción notable en el rango dinámico de amplitud, acompañada de una pérdida de nitidez en los detalles sonoros. Esta degradación auditiva refleja la limitada robustez de este esquema frente al ruido, complementando las métricas cuantitativas de desempeño.

Figura 59. *Audio reconstruido con modulación ASK.*

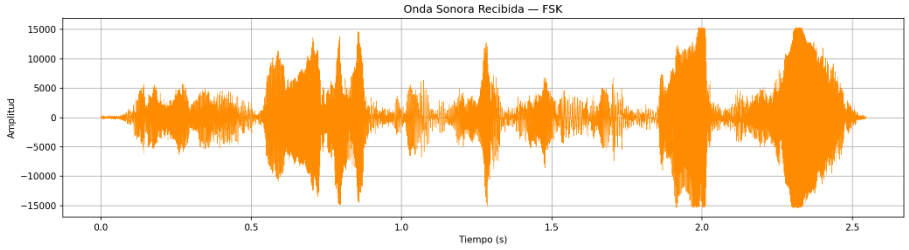


Nota. Elaboración propia.

Onda sonora del audio FSK receptado

En la **Figura 60**, se muestra la reconstrucción del audio transmitido mediante modulación FSK. Se identifica una reducción en la amplitud máxima alcanzada y una variación irregular en la envolvente temporal, lo que se traduce en una degradación moderada con pérdida parcial de definición.

Figura 60. *Audio reconstruido con modulación FSK.*

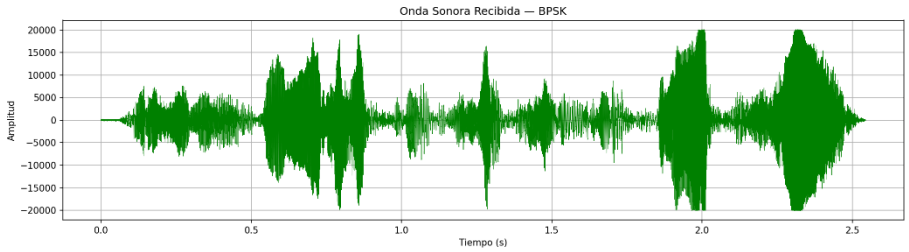


Nota. Elaboración propia.

Onda sonora del audio BPSK receptado

En la **Figura 61**, se muestra la reconstrucción del audio transmitido mediante modulación BPSK. Se evidencia una disminución en la amplitud promedio y una mayor dispersión en la forma de onda, lo que se traduce en una degradación notable con pérdida de nitidez y detalles.

Figura 61. *Audio reconstruido con modulación BPSK.*

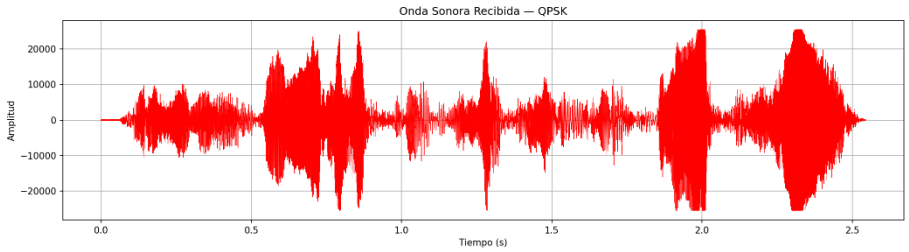


Nota. Elaboración propia.

Onda sonora del audio QPSK receptado

En la **Figura 62**, se muestra la reconstrucción del audio transmitido mediante modulación QPSK. Se detecta una mayor variabilidad en la amplitud de la señal y una distorsión en la envolvente temporal, lo que se traduce en una degradación más evidente con pérdida de detalles y cierta distorsión.

Figura 62. *Audio reconstruido con modulación QPSK.*

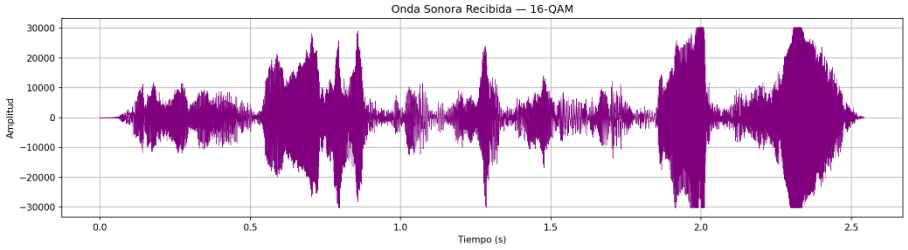


Nota. Elaboración propia.

Onda sonora del audio 16-QAM receptado

En la **Figura 63**, se muestra la reconstrucción del audio transmitido mediante modulación 16-QAM. Se mantiene una amplitud estable con baja dispersión en la forma de onda, lo que se traduce en buena fidelidad auditiva con solo leves artefactos residuales. Estos resultados reflejan la mayor robustez de este esquema frente al ruido en comparación con modulaciones de menor orden.

Figura 63. *Audio reconstruido con modulación 16-QAM.*

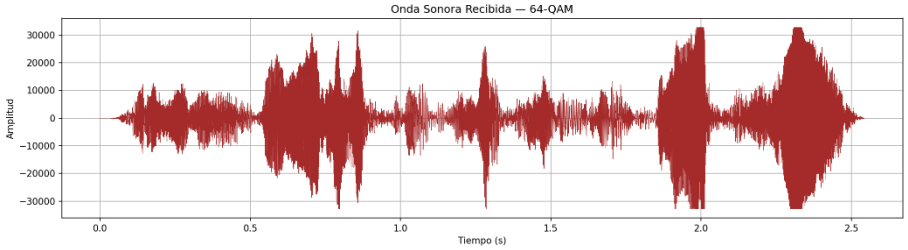


Nota. Elaboración propia.

Onda sonora del audio 64-QAM receptado

En la **Figura 64**, se muestra la reconstrucción del audio transmitido mediante modulación 64-QAM. Se mantiene una amplitud máxima elevada con baja variabilidad promedio, lo que se traduce en una alta fidelidad auditiva con mínima degradación. Estos resultados evidencian la capacidad de este esquema para preservar la calidad frente al ruido, aunque con mayor complejidad de procesamiento respecto a modulaciones de menor orden.

Figura 64. *Audio reconstruido con modulación 64-QAM.*



Nota. Elaboración propia.

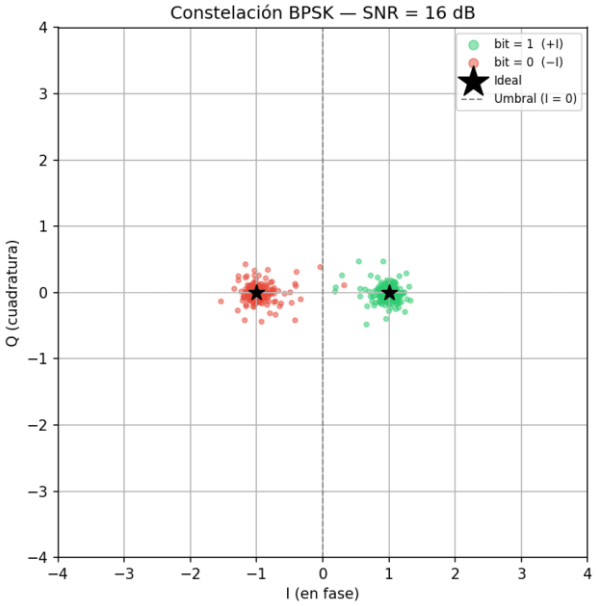
4.3.2 Comparación visual del efecto del ruido en los audios por medio de constelaciones

Para las modulaciones que pueden representarse en el plano de fase y cuadratura, se elaboraron diagramas de constelación utilizando una relación señal-ruido (SNR) de referencia.

Constelación de modulación BPSK de audios

Como se observa en la **Figura 65**, se presenta la constelación de la modulación BPSK con una relación señal-ruido de 16 dB. Los dos estados de símbolo se ubican en el eje "I" separados por el umbral de decisión en I = 0. La dispersión de los puntos alrededor de las posiciones ideales refleja el efecto del ruido, aunque la separación clara entre ambos estados asegura una detección confiable bajo estas condiciones.

Figura 65. *Constelación BPSK de audios SNR = 16 dB.*

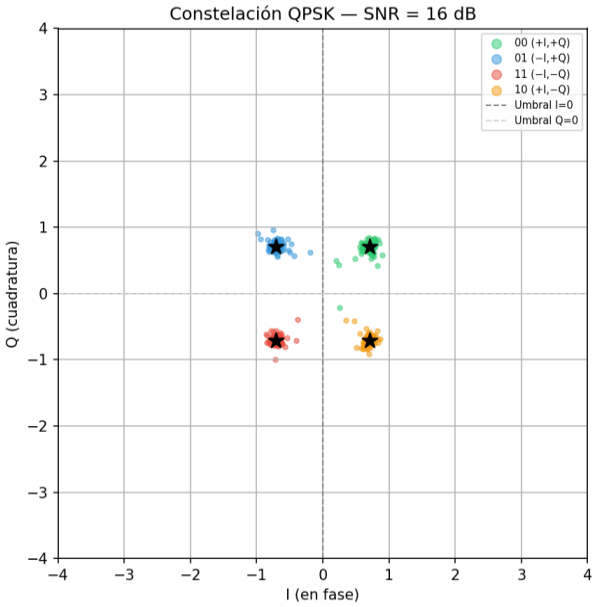


Nota. Elaboración propia.

Constelación de modulación QPSK de audios

Como se observa en la Figura 66, los cuatro estados de símbolo se ubican en los cuadrantes definidos por los umbrales en I = 0 y Q = 0, representando las combinaciones de bits. La dispersión de los puntos alrededor de las posiciones ideales refleja el efecto del ruido.

Figura 66. *Constelación QPSK de audios SNR = 16 dB.*

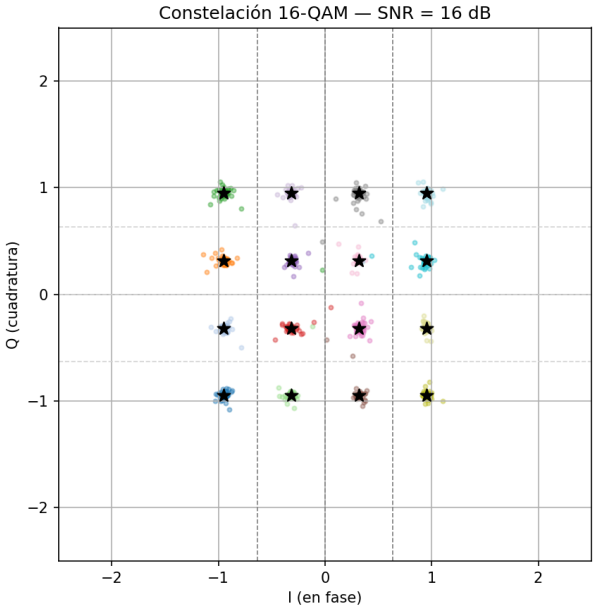


Nota. Elaboración propia.

Constelación de modulación 16-QAM de audios

Como se observa en la **Figura 67**, los dieciséis estados de símbolo se ubican en el plano I-Q alrededor de sus posiciones ideales representadas por los centroides. La dispersión de los puntos refleja el efecto del ruido, mostrando cómo el incremento en el orden de la modulación aumenta la densidad de símbolos y la sensibilidad frente a degradaciones del canal.

Figura 67. *Constelación 16-QAM de audios SNR = 16 dB.*

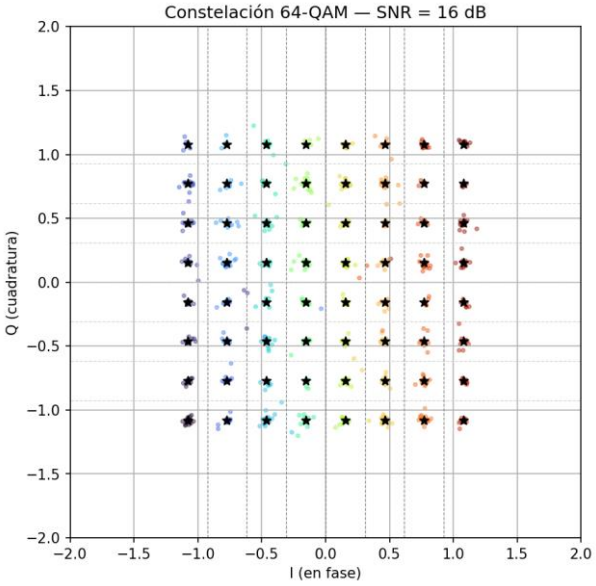


Nota. Elaboración propia.

Constelación de modulación 64-QAM de audios

Como se observa en la **Figura 68**, la dispersión de los puntos refleja el efecto del ruido, mostrando cómo el incremento en el orden de la modulación aumenta la densidad de símbolos y la sensibilidad frente a degradaciones del canal.

Figura 68. *Constelación 64-QAM de audios SNR = 16 dB.*



Nota. Elaboración propia.

Como se observa en las constelaciones, el nivel de SNR influye directamente en la dispersión de los símbolos alrededor de sus posiciones ideales. Cuando el ruido aumenta, los puntos se alejan de su posición correcta, incrementando la probabilidad de errores en la detección. Esto provoca un mayor número de errores de bit, por lo que la tasa de error de bits (BER) es una métrica fundamental para evaluar el desempeño de cada esquema de modulación.

4.3.3 Comparación de BER por técnica de modulación en audios

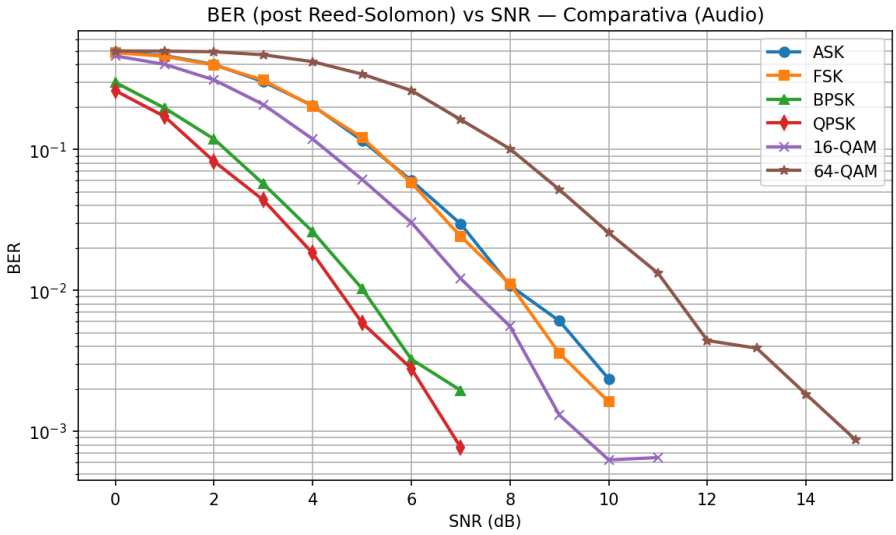
Se compara el BER de cada esquema de modulación, donde las modulaciones de menor orden presentan un BER más bajo, mientras que las de mayor orden muestran un BER más alto debido a la menor separación entre los símbolos de su constelación.

Tasa de error de bit en audios

Como se observa en la **Figura 69**, se presenta la comparación del BER posterior a la corrección Reed-Solomon en función de la SNR para distintos esquemas de modulación digital en transmisión de audio. Se aprecia que conforme aumenta la relación señal-ruido, la tasa de error disminuye de manera significativa, mostrando un desempeño mucho más robusto que en el caso sin codificación. Esto confirma la efectividad del código Reed-Solomon en la corrección de ráfagas de errores y en la mejora de la calidad de la señal reconstruida.

En este escenario, los picos o fluctuaciones en el BER se deben a la naturaleza aleatoria del canal y al comportamiento del decodificador Reed-Solomon, que, en ciertas condiciones de ruido y dispersión, puede fallar en la corrección de bloques completos de errores. Aunque la codificación reduce la probabilidad global de error, estas variaciones son más notorias en modulaciones de orden superior, donde la menor separación entre símbolos facilita la confusión en SNR intermedias y aumenta la sensibilidad frente al ruido.

Figura 69. *Resultado de BER post-RS vs SNR en audios.*



Nota. Elaboración propia.

Como se muestra en la **Tabla 14**, la codificación Reed-Solomon reduce drásticamente la tasa de error de bit en todas las modulaciones digitales. Se observa que en esquemas de orden bajo el BER se aproxima a cero a partir de SNR ≥ 8 dB, confirmando la robustez de estas modulaciones cuando se combinan con la corrección de errores.

En modulaciones de orden superior como 16-QAM y 64-QAM, la corrección sigue siendo efectiva, aunque se requieren niveles de SNR más altos para alcanzar valores de BER despreciables. Esto se debe a que la menor separación entre símbolos en la constelación incrementa la probabilidad de confusión en condiciones de ruido intermedio, lo que hace más exigente la tarea del decodificador Reed-Solomon.

En general, la tabla evidencia que Reed-Solomon logra compensar la degradación del canal, permitiendo que incluso modulaciones complejas alcancen un desempeño confiable en escenarios de audio digital.

Tabla 14. *Resultados de BER vs SNR en audios.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.490268	0.487141	0.299219	0.261343	0.461184	0.499779
1	0.463053	0.457196	0.196776	0.171247	0.401708	0.498181
2	0.400675	0.397097	0.118607	0.082341	0.311689	0.493236

3	0.302528	0.310481	0.057051	0.043795	0.208472	0.469661
4	0.204672	0.202429	0.026112	0.018348	0.118543	0.419473
5	0.115876	0.121484	0.010297	0.005893	0.061365	0.342332
6	0.060200	0.057999	0.003247	0.002788	0.030307	0.261912
7	0.029579	0.024188	0.001951	0.000768	0.012115	0.162756
8	0.010817	0.011073	0.000401	0.000225	0.005557	0.100913
9	0.006069	0.003576	0.000000	0.000327	0.001301	0.051857
10	0.002351	0.001620	0.000105	0.000000	0.000627	0.025592
11	0.000327	0.000337	0.000000	0.000000	0.000651	0.013231
12	0.000439	0.000000	0.000000	0.000000	0.000101	0.004405
13	0.000308	0.000000	0.000000	0.000000	0.000000	0.003885
14	0.000000	0.000000	0.000000	0.000000	0.000000	0.001831
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.000871
16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000313

Nota. Elaboración propia.

El análisis de las curvas BER muestra que la dispersión de los símbolos en las constelaciones genera errores durante la detección. A medida que aumenta la SNR, la tasa de error disminuye, evidenciando que las modulaciones de menor orden son más resistentes al ruido, mientras que las de mayor orden son más sensibles. La aplicación de la codificación Reed-Solomon reduce significativamente la tasa de error, lo que confirma la importancia del BER como una métrica para evaluar el desempeño de los sistemas de comunicación digital.

Resultados de BER vs SNR en modulaciones digitales para audios (Desviación estándar)

La **Tabla 15** muestra que la desviación estándar disminuye con el incremento de la SNR, indicando mayor estabilidad en los resultados. Las modulaciones de bajo orden presentan menor variabilidad, mientras que las de alto orden mantienen una dispersión más elevada en condiciones de ruido.

Tabla 15. *Resultados de BER vs SNR en audios (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.004235	0.006412	0.016113	0.019795	0.011387	0.000909
1	0.010209	0.012865	0.015309	0.018685	0.015428	0.002476
2	0.013174	0.016479	0.018690	0.012698	0.019358	0.004169
3	0.020701	0.019941	0.012202	0.009648	0.021783	0.006412
4	0.018967	0.017407	0.011037	0.007400	0.018930	0.011805
5	0.019872	0.014601	0.005384	0.004083	0.017403	0.018974
6	0.014031	0.012728	0.003939	0.003805	0.010217	0.021008
7	0.008836	0.007407	0.002292	0.002042	0.005914	0.017578
8	0.006061	0.004238	0.001024	0.000844	0.003351	0.013513
9	0.003136	0.004075	0.000000	0.000983	0.002280	0.011732
10	0.002513	0.002033	0.000567	0.000000	0.001476	0.010296
11	0.000982	0.001015	0.000000	0.000000	0.001305	0.005355
12	0.001124	0.000000	0.000000	0.000000	0.000546	0.004449
13	0.000933	0.000000	0.000000	0.000000	0.000000	0.003106
14	0.000000	0.000000	0.000000	0.000000	0.000000	0.002188
15	0.000000	0.000000	0.000000	0.000000	0.000000	0.001450
16	0.000000	0.000000	0.000000	0.000000	0.000000	0.000943

Nota. Elaboración propia. Desviación estándar calculada sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre BER teórico y BER simulado

Con el fin de complementar la validación visual de los algoritmos frente a las curvas teóricas conocidas, se realizó una comparación cuantitativa independiente entre el

BER teórico (**Tabla 1**) y el BER simulado post Reed-Solomon en transmisión de audio (**Tabla 14**) para cada esquema de modulación en el rango de 0 a 16 dB de SNR. Se calculó el error absoluto medio (MAE) y la raíz del error cuadrático medio (RMSE) tanto en escala lineal como en escala logarítmica ($\log_{10}$), además del coeficiente de determinación (R^2) sobre $\log_{10}$(BER).

Como se observa en la **Tabla 16**, QPSK y 16-QAM presentan el mejor ajuste relativo ($R^2 = 0.873$ y 0.849), mientras que 64-QAM muestra un ajuste inadecuado ($R^2 = -0.666$), lo que indica que el modelo teórico no describe apropiadamente su comportamiento simulado en audio. Esto se explica por una meseta de saturación: el BER simulado de 64-QAM permanece prácticamente constante (~0.50) entre 0 y 5 dB antes de comenzar a decrecer, mientras que el BER teórico decae de forma continua desde el primer punto, generando una discrepancia creciente en SNR bajas e intermedias (hasta +713 % de error relativo en SNR = 9 dB).

ASK también presenta un ajuste muy pobre ($R^2 = 0.071$). En contraste, BPSK y QPSK muestran errores absolutos (MAE lineal) considerablemente menores (0.056 y 0.068), aunque con picos puntuales de error relativo asociados a fallos ocasionales del decodificador Reed-Solomon.

Tabla 16. *Comparación cuantitativa entre BER teórico y BER simulado de audio por esquema de modulación.*

Modulación	MAE (lineal)	RMSE (lineal)	MAE ($\log_{10}$)	RMSE ($\log_{10}$)	R^2 ($\log_{10}$)
ASK	0.0618	0.1082	0.625	0.842	0.635
FSK	0.0861	0.1502	0.417	0.542	0.919
BPSK	0.0292	0.0670	0.376	0.586	0.909
QPSK	0.0218	0.0537	0.207	0.349	0.968
16-QAM	0.0615	0.1215	0.347	0.474	0.929
64-QAM	0.1320	0.1956	0.310	0.368	0.811

Nota. Elaboración propia. MAE: error absoluto medio; RMSE: raíz del error cuadrático medio; R^2: coeficiente de determinación calculado sobre $\log_{10}$(BER).

A modo de ejemplo, se detalla el cálculo del error para la modulación BPSK en tres puntos representativos del rango de SNR: SNR = 0 dB, donde la discrepancia inicial

es alta; SNR = 9 dB, un punto anómalo con error relativo extremo asociado a un fallo puntual del decodificador; y SNR = 16 dB, donde ambas curvas ya han convergido.

Ejemplo de cálculo — BPSK, SNR = 0 dB

- BER teórico (Tabla 1): 0.078650
- BER simulado (Tabla 14): 0.424270

Error absoluto:

$$\varepsilon_{abs} = | 0.424270 - 0.078650 | = 0.345620$$

Error relativo:

$$\varepsilon_{rel} = \frac{0.345620}{0.078650} \times 100\% = +439.4\%$$

Ejemplo de cálculo — BPSK, SNR = 9 dB

- BER teórico: 0.000034
- BER simulado: 0.000740

Error absoluto:

$$\varepsilon_{abs} = | 0.000740 - 0.000034 | = 0.000706$$

Error relativo:

$$\varepsilon_{rel} = \frac{0.000706}{0.000034} \times 100\% = +2076\%$$

Aunque el error relativo es extremo, el error absoluto sigue siendo muy pequeño. Este pico puntual visible en SNR = 9 dB responde a los fallos ocasionales del decodificador Reed-Solomon al corregir ráfagas de errores, más que a una falla sistemática del modelo.

Ejemplo de cálculo — BPSK, SNR = 16 dB

- BER teórico: 0.000000
- BER simulado: 0.000000

Error absoluto:

$$\varepsilon_{abs} = | 0.000000 - 0.000000 | = 0.000000$$

En este punto ambas curvas convergen completamente sin error relativo definido. En conjunto, el caso de audio confirma cuantitativamente lo que el texto ya describe cualitativamente: existe buena concordancia general con el modelo teórico en SNR altas, pero con mayor dispersión que en el escenario de imágenes, y picos puntuales atribuibles al decodificador Reed-Solomon.

4.3.4 Comparación de MER, EVM e histograma del error en audios

Se comparan las métricas EVM y MER para las seis modulaciones evaluadas, utilizando la SNR de referencia, junto con el histograma del error entre la señal transmitida y la recibida. El EVM mide el porcentaje de desviación de la señal recibida respecto a la ideal, mientras que el MER expresa esa relación en decibelios (dB). Ambas métricas permiten evaluar la calidad de la señal y muestran el comportamiento esperado para cada esquema de modulación.

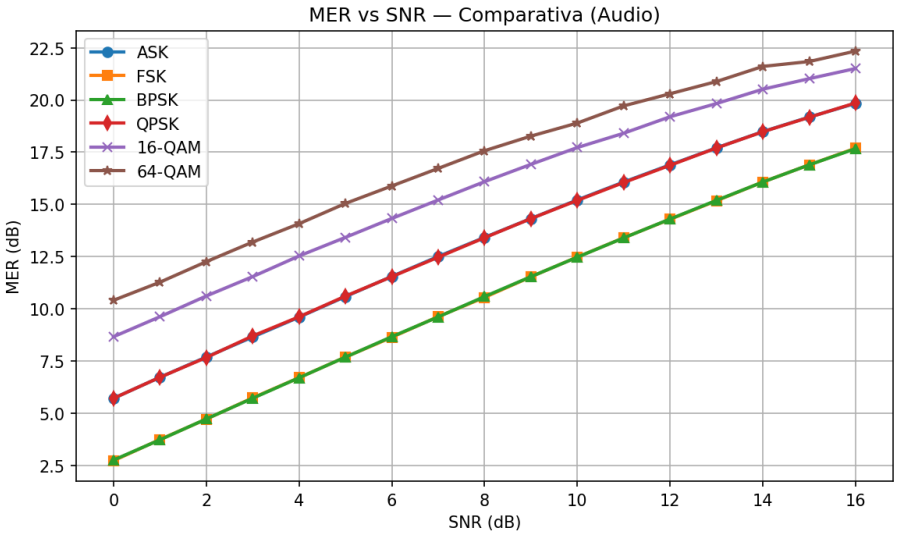
MER de las modulaciones evaluadas en audios

Como se observa en la **Figura 70**, se presenta la comparación del MER en función de la SNR para distintos esquemas de modulación digital en transmisión de audio. Se aprecia una tendencia creciente del MER conforme aumenta la relación señal-ruido, lo que refleja una mejora en la calidad de la modulación y una menor dispersión de los símbolos respecto a sus posiciones ideales en la constelación.

La gráfica muestra que las modulaciones de orden superior, como 16-QAM y 64-QAM, alcanzan valores de MER más altos para la misma SNR, evidenciando su mayor eficiencia espectral y capacidad de transmitir más bits por símbolo. En contraste, modulaciones de orden bajo, como ASK, FSK y BPSK, muestran valores más bajos de MER, confirmando su robustez frente al canal, pero con menor eficiencia espectral.

Al graficar los valores de MER, algunos puntos pueden parecer solaparse entre distintas modulaciones. Esto ocurre porque varias curvas comparten pendientes similares y se representan en la misma escala.

Figura 70. Comparativa MER vs SNR de audios.



Nota. Elaboración propia.

Como se observa en la **Tabla 17**, se presenta la evolución de los valores de desempeño (expresados en dB) para distintos esquemas de modulación digital bajo la aplicación de la codificación Reed-Solomon. Se aprecia que en bajas SNR (0–5 dB) los valores son negativos, lo que refleja una alta degradación del canal y la dificultad del decodificador para corregir ráfagas de errores en condiciones de ruido intenso.

A medida que la SNR aumenta, los valores se vuelven positivos y crecen de forma sostenida, indicando una mejora en la calidad de la señal reconstruida gracias a la acción de Reed-Solomon. En modulaciones de orden bajo, como BPSK y QPSK, la corrección es más efectiva y los valores positivos aparecen desde SNR intermedias (≈ 7–8 dB), confirmando su robustez frente al canal.

En el caso de modulaciones de orden superior, como 16-QAM y 64-QAM, la codificación Reed-Solomon también ayuda a estabilizar el desempeño, aunque aquí se necesitan niveles de SNR más altos para lograr valores positivos de manera consistente. En general, la tabla evidencia que Reed-Solomon actúa como un mecanismo de corrección eficaz, permitiendo que incluso modulaciones complejas alcancen un desempeño confiable cuando la SNR supera los 10–12 dB.

Tabla 17. *Resultados de MER vs SNR en audios.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	5.71	2.72	2.75	5.71	8.66	10.40

1	6.72	3.73	3.72	6.72	9.62	11.27
2	7.69	4.71	4.72	7.67	10.61	12.25
3	8.65	5.73	5.71	8.70	11.54	13.19
4	9.60	6.70	6.69	9.61	12.53	14.07
5	10.58	7.68	7.68	10.60	13.42	15.04
6	11.55	8.63	8.66	11.54	14.32	15.88
7	12.50	9.62	9.61	12.47	15.21	16.72
8	13.42	10.55	10.58	13.40	16.09	17.57
9	14.32	11.53	11.54	14.32	16.92	18.27
10	15.21	12.48	12.47	15.19	17.72	18.89
11	16.07	13.40	13.39	16.05	18.41	19.72
12	16.89	14.29	14.30	16.87	19.20	20.30
13	17.71	15.17	15.19	17.70	19.83	20.88
14	18.48	16.07	16.07	18.48	20.52	21.61
15	19.18	16.89	16.89	19.17	21.02	21.84
16	19.84	17.69	17.67	19.86	21.51	22.34

Nota. Elaboración propia.

Resultados de MER vs SNR en modulaciones digitales para audios (Desviación estándar)

La Tabla 18 muestra que la desviación estándar se mantiene baja en modulaciones robustas, como BPSK y QPSK, mientras que en modulaciones de mayor orden (16-QAM y 64-QAM) la variabilidad es más alta, reflejando su mayor sensibilidad al ruido.

Tabla 18. *Resultados de MER vs SNR en audios (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM

0	0.09	0.09	0.08	0.09	0.15	0.18
1	0.06	0.08	0.06	0.09	0.13	0.19
2	0.07	0.08	0.07	0.09	0.12	0.20
3	0.09	0.07	0.08	0.10	0.14	0.17
4	0.10	0.08	0.07	0.07	0.16	0.18
5	0.08	0.08	0.05	0.09	0.13	0.13
6	0.08	0.08	0.06	0.11	0.16	0.22
7	0.08	0.09	0.07	0.10	0.15	0.20
8	0.07	0.10	0.07	0.11	0.19	0.23
9	0.10	0.08	0.07	0.08	0.16	0.23
10	0.11	0.09	0.07	0.09	0.18	0.26
11	0.12	0.09	0.07	0.12	0.20	0.33
12	0.12	0.09	0.07	0.13	0.27	0.33
13	0.12	0.10	0.08	0.10	0.26	0.39
14	0.10	0.10	0.07	0.15	0.30	0.35
15	0.19	0.10	0.07	0.14	0.33	0.39
16	0.16	0.12	0.07	0.16	0.34	0.46

Nota. Elaboración propia. Desviación estándar calculada sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre MER simulado y MER teórico

Se realizó la comparación cuantitativa entre el MER teórico (**Tabla 2**) y el MER simulado en transmisión de audio (**Tabla 17**) para cada esquema de modulación en el rango de 0 a 16 dB de SNR, calculando el error absoluto medio (MAE), la raíz del error cuadrático medio (RMSE) y el coeficiente de determinación (R^2) directamente en escala de dB.

Como se observa en la **Tabla 19**, el ajuste entre el MER simulado en audio y el MER teórico es deficiente en todos los esquemas de modulación. Esto contrasta fuertemente con el comportamiento observado en el escenario de imágenes, donde el ajuste era razonable. La causa es visible directamente en la **Tabla 17:** en SNR bajas (0–6 dB) el MER simulado toma valores negativos, algo que el modelo teórico ideal no predice en absoluto, ya que un MER teórico siempre es positivo o cero por definición.

Tabla 19. *Comparación cuantitativa entre MER teórico y MER simulado por esquema de modulación.*

Modulación	MAE (dB)	RMSE (dB)	R^2
ASK	6.856	6.942	−1.008
FSK	7.347	7.500	−1.344
BPSK	7.238	7.312	−1.228
QPSK	9.876	9.904	−3.087
16-QAM	12.781	12.794	−5.820
64-QAM	14.620	14.643	−7.933

Nota. Elaboración propia. MAE: error absoluto medio en dB; RMSE: raíz del error cuadrático medio en dB; R^2: coeficiente de determinación respecto al MER teórico. Un R^2 negativo indica que el modelo teórico ajusta peor que simplemente usar el promedio de los datos simulados como predicción.

A modo de ejemplo, se detalla el cálculo del error para la modulación QPSK en tres puntos representativos del rango de SNR:

SNR = 0 dB, donde el MER simulado es negativo; SNR = 8 dB, donde el simulado ya es positivo, pero sigue muy por debajo del teórico; y SNR = 16 dB, el punto de mayor SNR evaluado, donde persiste una brecha considerable.

Ejemplo de cálculo — QPSK, SNR = 0 dB

- MER teórico (Tabla 2): 3.01 dB
- MER simulado (Tabla 17): −6.46 dB

Error absoluto:

$$\varepsilon_{abs} = | -6.46 - 3.01 | = 9.47\ dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{-9.47}{3.01} \times 100\% = -314.6\%$$

Ejemplo de cálculo — QPSK, SNR = 8 dB

- MER teórico: 11.01 dB
- MER simulado: 1.92 dB

Error absoluto:

$$\varepsilon_{abs} = | 1.92 - 11.01 | = 9.09\ dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{-9.09}{11.01} \times 100\% = -82.6\%$$

Ejemplo de cálculo — QPSK, SNR = 16 dB

- MER teórico: 19.01 dB
- MER simulado: 8.68 dB

Error absoluto:

$$\varepsilon_{abs} = | 8.68 - 19.01 | = 10.33\ dB$$

Error relativo:

$$\varepsilon_{rel} = \frac{-10.33}{19.01} \times 100\% = -54.3\%$$

A diferencia del escenario de imágenes, donde el offset se reducía progresivamente al aumentar la SNR, en audio el error absoluto se mantiene alrededor de 9–10 dB en todo el rango evaluado, sin mostrar una tendencia clara de convergencia hacia el modelo teórico. Esto indica que la brecha no es un simple offset constante corregible, sino que refleja una diferencia estructural en cómo se comporta el enlace de audio simulado frente al modelo ideal.

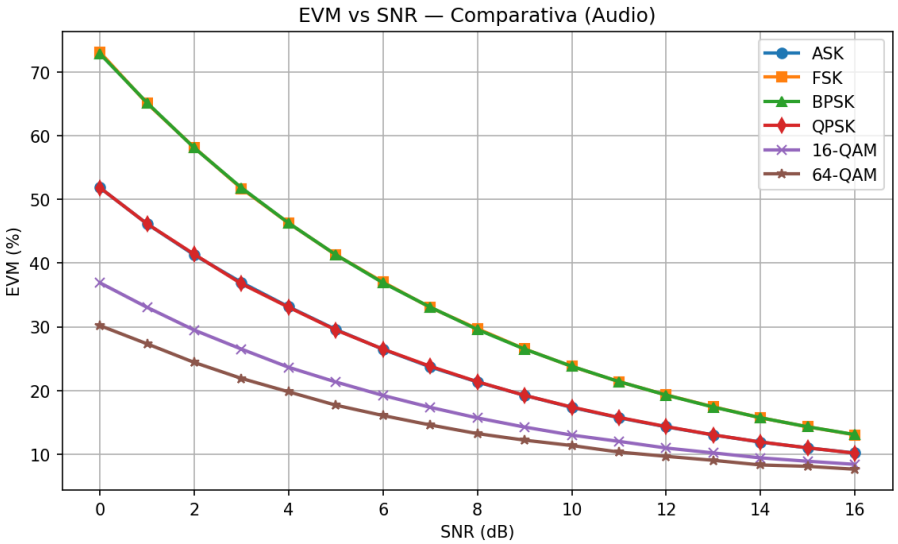
EVM de las modulaciones evaluadas en audios

Como se observa en la **Figura 71**, se presenta la comparación del EVM en función de la SNR para distintos esquemas de modulación digital en transmisión de audio. Se aprecia una tendencia decreciente del EVM conforme aumenta la relación señal-ruido, lo que refleja una mejora en la calidad de la señal y una menor dispersión de los símbolos respecto a sus posiciones ideales en la constelación.

La gráfica deja ver que las modulaciones de orden superior, como 16-QAM y 64-QAM, logran valores de EVM más bajos para un mismo nivel de SNR, lo que confirma su eficiencia espectral y su capacidad de transportar más bits por símbolo, aunque a cambio exigen mayor linealidad y precisión por parte del transmisor.

Al graficar los valores de EVM, algunos puntos pueden parecer solaparse entre distintas modulaciones. Esto ocurre porque varias curvas comparten pendientes similares y se representan en la misma escala.

Figura 71. Comparativa EVM vs SNR de audios.



Nota. Elaboración propia.

Como se visualiza en la **Tabla 20**, se presentan los valores de desempeño asociados al EVM en transmisión de audio para distintos esquemas de modulación digital en función de la SNR.

A medida que la SNR va subiendo, el EVM tiende a bajar de forma progresiva, lo que refleja una mejora en la linealidad y precisión de la modulación. En las modulaciones de orden bajo, esta reducción es más pronunciada. En cambio, en modulaciones de orden superior, el EVM también disminuye, pero hace falta una SNR más alta para que logre estabilizarse en rangos bajos.

Tabla 20. *Resultados de EVM vs SNR en audios.*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	51.83	73.10	72.85	51.82	36.90	30.19
1	46.15	65.05	65.15	46.16	33.05	27.32
2	41.27	58.13	58.07	41.38	29.48	24.40
3	36.96	51.70	51.81	36.75	26.50	21.90

4	33.13	46.26	46.27	33.06	23.64	19.79
5	29.57	41.29	41.30	29.50	21.34	17.70
6	26.46	37.02	36.89	26.50	19.22	16.07
7	23.70	33.03	33.08	23.80	17.36	14.59
8	21.34	29.70	29.57	21.38	15.69	13.23
9	19.23	26.52	26.49	19.24	14.26	12.21
10	17.35	23.78	23.80	17.40	13.00	11.36
11	15.72	21.39	21.40	15.77	12.02	10.34
12	14.30	19.30	19.28	14.34	10.97	9.67
13	13.02	17.43	17.40	13.03	10.20	9.04
14	11.91	15.72	15.72	11.92	9.43	8.32
15	10.99	14.30	14.30	11.00	8.90	8.10
16	10.19	13.05	13.08	10.17	8.41	7.65

Nota. Elaboración propia.

Resultados de EVM vs SNR en modulaciones digitales para audios (Desviación estándar)

La **Tabla 21** muestra la dispersión de los valores de EVM en cada nivel de SNR. Se observa que la desviación estándar se mantiene baja en modulaciones robustas como BPSK y QPSK mientras que en modulaciones de mayor orden (16-QAM y 64-QAM) la variabilidad es más alta, reflejando su mayor sensibilidad al ruido.

Tabla 21. *Resultados de EVM vs SNR en audios (Desviación estándar).*

SNR (dB)	ASK	FSK	BPSK	QPSK	16-QAM	64-QAM
0	0.57	0.79	0.65	0.53	0.65	0.62
1	0.33	0.62	0.44	0.50	0.50	0.59
2	0.36	0.56	0.45	0.44	0.39	0.56
3	0.37	0.42	0.46	0.42	0.44	0.44
4	0.38	0.40	0.38	0.28	0.43	0.40
5	0.26	0.39	0.25	0.29	0.31	0.26
6	0.24	0.34	0.26	0.32	0.35	0.41

7	0.22	0.33	0.26	0.26	0.31	0.33
8	0.17	0.33	0.23	0.28	0.34	0.35
9	0.23	0.24	0.20	0.18	0.27	0.32
10	0.23	0.23	0.19	0.18	0.27	0.34
11	0.21	0.21	0.16	0.22	0.27	0.39
12	0.21	0.20	0.16	0.22	0.34	0.36
13	0.18	0.20	0.15	0.14	0.30	0.40
14	0.13	0.18	0.13	0.20	0.32	0.33
15	0.24	0.17	0.12	0.18	0.33	0.37
16	0.18	0.17	0.11	0.19	0.33	0.40

Nota. Elaboración propia. Desviación estándar calculada sobre 30 iteraciones Monte Carlo por nivel de SNR.

Comparación cuantitativa entre EVM simulado y EVM teórico

Se realizó la comparación cuantitativa entre el EVM teórico (**Tabla 3**) y el EVM simulado en transmisión de audio (**Tabla 20**) para cada esquema de modulación en el rango de 0 a 16 dB de SNR, calculando el error absoluto medio (MAE), la raíz del error cuadrático medio (RMSE) y el coeficiente de determinación (R^2) en escala lineal (porcentaje).

Como se observa en la **Tabla 22**, el ajuste es muy deficiente en todos los esquemas, con R^2 fuertemente negativos que empeoran drásticamente con el orden de modulación; es decir, en audio el vector de error simulado llega a ser más de cinco veces mayor de lo que predice el modelo ideal para 64-QAM.

Tabla 22. *Comparación cuantitativa entre EVM teórico y EVM simulado por esquema de modulación.*

Modulación	MAE (%)	RMSE (%)	R^2	Razón media (sim/teo)
ASK	61.45	71.78	−6.984	2.303
FSK	69.33	86.62	−10.627	2.543
BPSK	65.91	78.08	−8.447	2.400
QPSK	72.29	82.08	−19.877	3.220
16-QAM	78.79	89.39	−48.521	4.447

64-QAM	84.53	97.00	−86.481	5.507

Nota. Elaboración propia. MAE: error absoluto medio en puntos porcentuales; RMSE: raíz del error cuadrático medio en puntos porcentuales; R²: coeficiente de determinación respecto al EVM teórico; razón media: promedio de EVM simulado / EVM teórico en todo el rango de SNR. Los valores de R² fuertemente negativos, que se agravan con el orden de modulación, indican ausencia de correspondencia funcional con el modelo teórico en este escenario.

A modo de ejemplo, se detalla el cálculo del error para la modulación QPSK en tres puntos representativos del rango de SNR: SNR = 0 dB, SNR = 9 dB y SNR = 16 dB, el extremo superior evaluado.

Ejemplo de cálculo — QPSK, SNR = 0 dB

- EVM teórico (Tabla 3): 70.71 %
- EVM simulado (Tabla 20): 213.23 %

Error absoluto:

$$\varepsilon_{abs} = |\ 213.23 - 70.71\ | = 142.52 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{142.52}{70.71} \times 100\% = +201.6\%$$

Ejemplo de cálculo — QPSK, SNR = 9 dB

- EVM teórico: 25.09 %
- EVM simulado: 104.97 %

Error absoluto:

$$\varepsilon_{abs} = |\ 104.97 - 25.09\ | = 79.88 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{79.88}{25.09} \times 100\% = +318.4\%$$

Ejemplo de cálculo — QPSK, SNR = 16 dB

- EVM teórico: 11.21 %

- EVM simulado: 37.28 %

Error absoluto:

$$\varepsilon_{abs} = |\ 37.28 - 11.21\ | = 26.07 \text{ puntos porcentuales}$$

Error relativo:

$$\varepsilon_{rel} = \frac{26.07}{11.21} \times 100\% = +232.6\%$$

En conjunto, el análisis de BER, MER y EVM en audio muestra un patrón claro y consistente entre sí: mientras que el BER logra converger razonablemente con la teoría en SNR altas, tanto el MER como el EVM presentan discrepancias sistemáticas y crecientes con el orden de modulación en todo el rango evaluado.

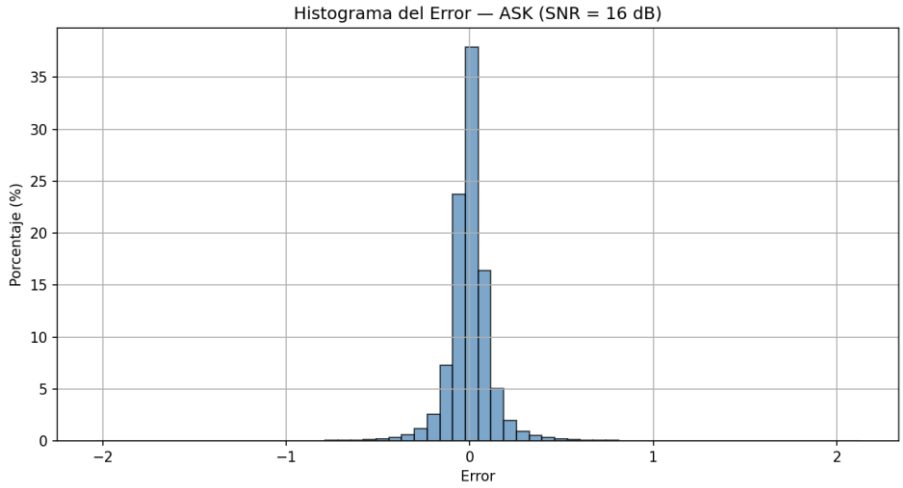
Histograma del error de las modulaciones evaluadas en audios

El histograma del error muestra la diferencia entre la señal transmitida y la señal recibida después de pasar por el canal Rayleigh + AWGN. Este permite visualizar el efecto del ruido y del desvanecimiento sobre la señal antes del proceso de detección.
Además, complementa las métricas EVM y MER. Cuando los errores se concentran alrededor de cero, la señal recibida es muy similar a la transmitida, lo que se refleja en un EVM bajo y un MER alto.

Histograma del error de ASK para audios

Como se observa en la **Figura 72**, la distribución es estrecha y simétrica alrededor de cero, lo que indica que la mayoría de los símbolos recibidos presentan errores muy pequeños. Este comportamiento refleja la buena calidad de la señal bajo condiciones de SNR relativamente altas.

Figura 72. *Histograma del error — ASK de audios.*

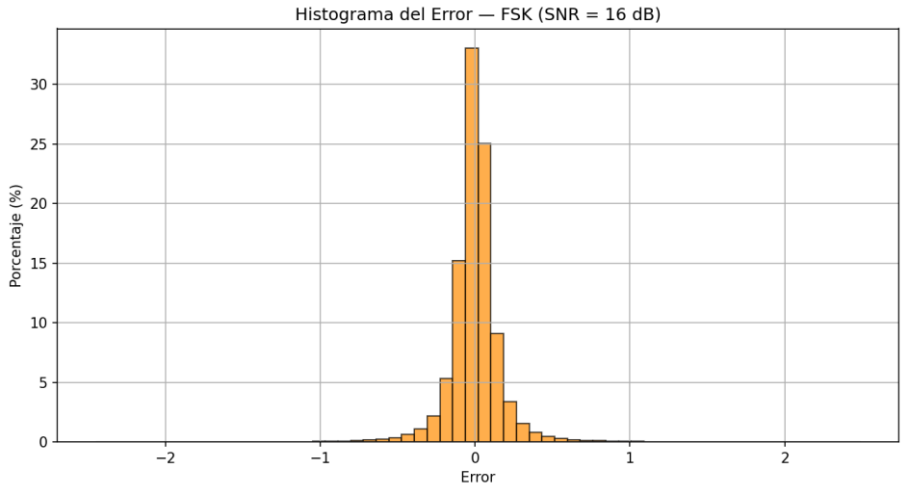


Nota. Elaboración propia.

Histograma del error de FSK para audios

Como se observa en **la Figura 73**, la distribución se concentra alrededor de cero, con una rápida disminución hacia ambos extremos, lo que indica baja varianza en los errores. Este comportamiento refleja la buena calidad de la señal.

Figura 73. *Histograma del error — FSK de audios.*

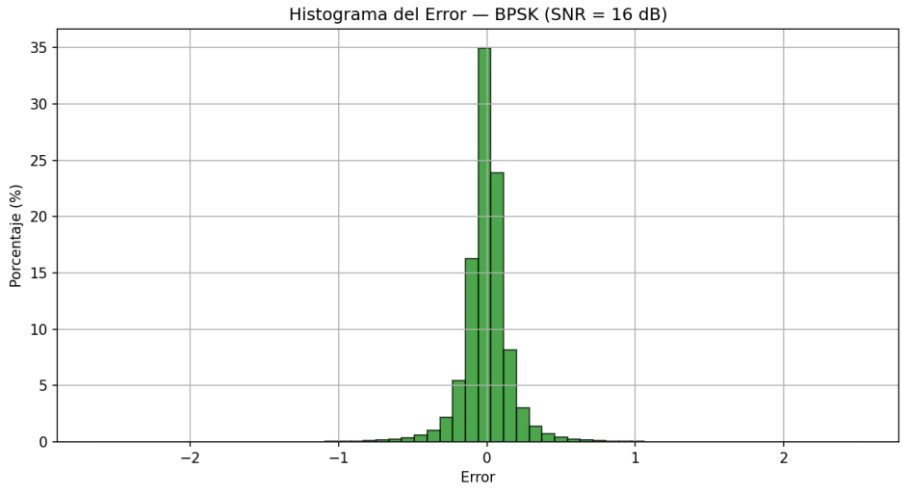


Nota. Elaboración propia.

Histograma del error de BPSK para audios

Como se observa en la **Figura 74**, la distribución se concentra alrededor de cero, mostrando que la mayoría de los símbolos recibidos presentan errores mínimos. Este comportamiento refleja la estabilidad del esquema bajo condiciones de SNR relativamente altas.

Figura 74. *Histograma del error — BPSK de audios.*

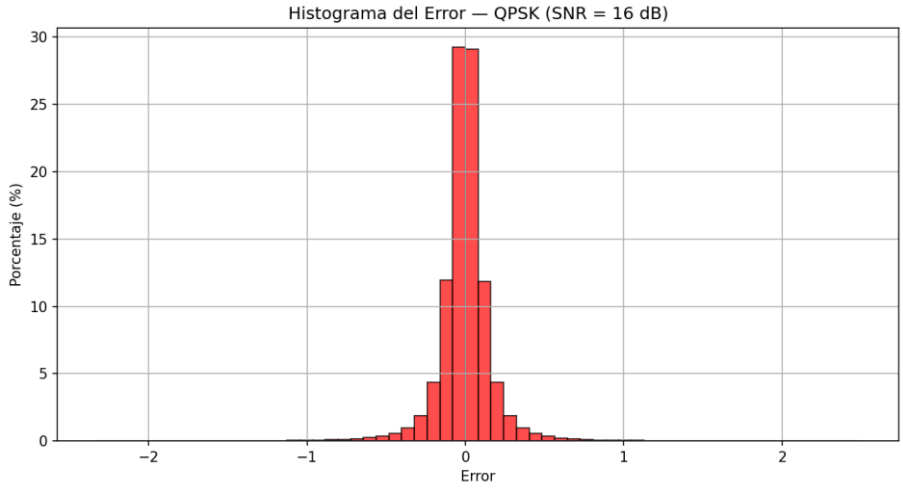


Nota. Elaboración propia.

Histograma del error de QPSK para audios

Como se observa en la **Figura 75**, la distribución es estrecha y centrada alrededor de cero, lo que indica que la mayoría de los símbolos recibidos presentan errores mínimos. Este comportamiento refleja la estabilidad del esquema.

Figura 75. *Histograma del error — QPSK de audios.*

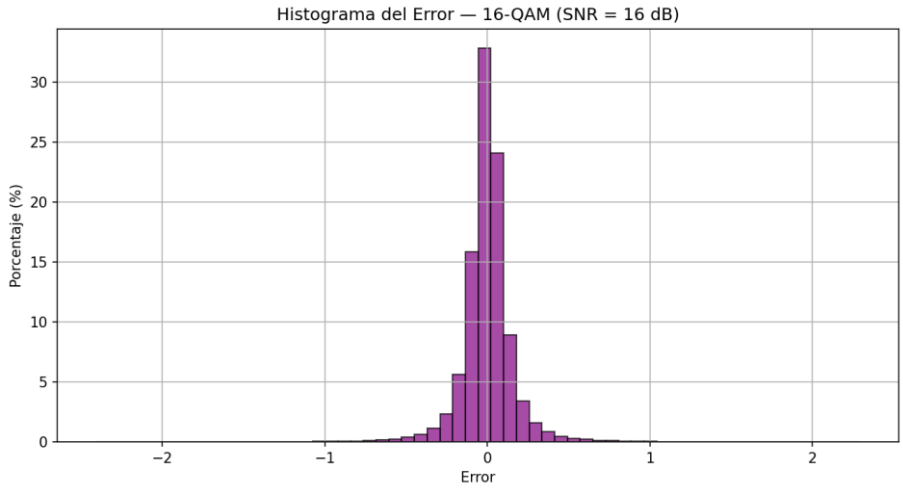


Nota. Elaboración propia.

Histograma del error de 16-QAM para audios

Como se observa en la **Figura 76**, la distribución se concentra alrededor de cero, aunque con una dispersión mayor que en modulaciones de menor orden, reflejando la mayor complejidad y sensibilidad de este esquema frente al ruido.

Figura 76. *Histograma del error — 16-QAM de audios.*

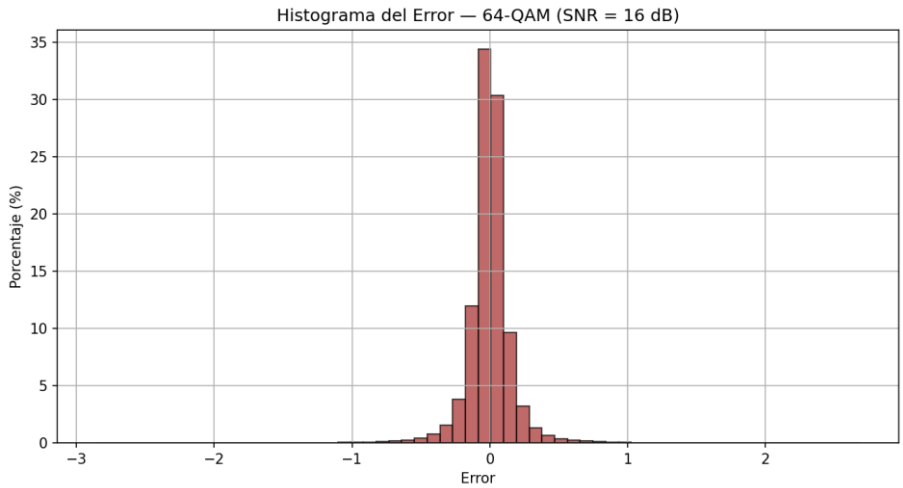


Nota. Elaboración propia.

Histograma del error de 64-QAM para audios

Como se observa en la **Figura 77**, la distribución se concentra alrededor de cero, aunque con una dispersión mayor que en modulaciones de menor orden, reflejando la complejidad y sensibilidad de este esquema frente al ruido.

Figura 77. *Histograma del error — 64-QAM de audios.*



Nota. Elaboración propia.

4.4 Comparación de eficiencia espectral

En la **Tabla 23**, se comparan distintas modulaciones digitales según sus bits por símbolo y su eficiencia espectral. Se observa que esquemas simples, como FSK, ASK y BPSK,

alcanzan hasta 0.5 bit/s/Hz, mientras que modulaciones de mayor orden, como QPSK, 16-QAM y 64-QAM, incrementan la eficiencia hasta 3 bit/s/Hz.

Tabla 23. *Eficiencia espectral de modulaciones digitales.*

Modulación	Bits/símbolo	Eficiencia espectral (bit/s/Hz)
FSK	1	0.25
ASK	1	0.50
BPSK	1	0.50
QPSK	2	1.00
16-QAM	4	2.00
64-QAM	6	3.00

Nota. Elaboración propia.

CONCLUSIONES

- Las métricas EVM y MER resultaron consistentes con el comportamiento del BER: a mayor SNR, menor EVM y mayor MER, lo que confirma la validez de estas métricas como indicadores complementarios de la calidad de la señal recibida en ambas fuentes de datos.
- El sistema de transmisión digital simulado, que combina un canal con desvanecimiento Rayleigh y ruido AWGN, permite recuperar el audio transmitido sin degradación perceptible en la mayoría de las modulaciones evaluadas cuando la SNR alcanza 16 dB. En el caso de las imágenes, en cambio, solo los esquemas de orden superior, 16-QAM y 64-QAM, logran una recuperación visualmente aceptable, mientras que en el resto de las modulaciones los errores residuales resultan claramente perceptibles, dado que una imagen evidencia visualmente alteraciones que en una señal de audio suelen pasar desapercibidas para el oído humano.
- BPSK y QPSK son consistentemente las modulaciones más robustas frente al ruido, al presentar el menor BER en todo el barrido de SNR tanto para la imagen como para el audio, aunque a costa de una menor eficiencia espectral. Sin embargo, como se señaló anteriormente, un BER bajo no se traduce necesariamente en una mejor calidad visual percibida en el caso de las imágenes, donde el número de bits transmitidos por símbolo resulta más determinante que la robustez frente al ruido.

RECOMENDACIONES

- Para la transmisión de audio se recomienda emplear BPSK o QPSK, dada su mayor resistencia al ruido frente a esquemas de orden superior; no obstante, dado que en este tipo de contenido los errores residuales no generan una degradación perceptible para el oído humano, esquemas como ASK, FSK, 16-QAM o 64-QAM también constituyen alternativas viables cuando se prioriza una mayor eficiencia espectral.
- Se recomienda evaluar el sistema con un rango de SNR más amplio (por ejemplo, hasta 20–25 dB) para 16-QAM y 64-QAM, con el fin de caracterizar con mayor precisión el punto de operación en el que estas modulaciones logran corrección completa bajo Reed-Solomon.
- Para la transmisión de contenido donde los errores sean fácilmente perceptibles por el usuario final, como imágenes o video, se recomienda emplear 16-QAM o 64-QAM en lugar de esquemas de orden inferior como ASK o FSK, ya que la degradación es más perceptible al ojo humano.

REFERENCIAS

[1] M. T. Jurado Cañero, «Simulación de sistemas de radiocomunicación mediante Python,» Sevilla, 2019.

[2] X. A. Alvarado Pihuave, «Implementación de un sistema de clasificación inteligente mediante la transformación Wavelets utilizados en modulaciones digitales,» Guayaquil, 2020.

[3] I. Maldonado, Diseño, pruebas y evaluación de simulaciones de modulaciones digitales como enfoque práctico para la materia de comunicaciones inalámbricas de banda ancha., Guayaquil, 2024.

[4] R. Saenz, "IMPLEMENTACIÓN DE SISTEMAS DEFINIDOS POR RADIO BASADO EN CÓDIGO ABIERTO PARA MODULACIONES PSK Y QAM EN FASE DE CUADRATURA, La Libertad, 2024.

[5] C. J. &. G. C. C. Gerena, «Desarrollo de una interfaz para simulación de modulaciones digitales utilizando lenguaje de programación python.,» 2021. [En línea]. Available: http://hdl.handle.net/11349/29127.

[6] R. N. Clarke, «Expansión de la capacidad inalámbrica móvil: Los desafíos que presentan la tecnología y la economía,» Septiembre 2014. [En línea]. Available: https://www.sciencedirect.com/science/article/pii/S0308596113001900.

[7] ITU, «Datos y cifras 2023,» 2023. [En línea]. Available: https://www.itu.int/itu-d/reports/statistics/facts-figures-2023/index/.

[8] M. &. T. R. F. Cabrera, Introducción a los sistemas de comunicaciones., Univerisad Oberta de Catalunya, 2013.

[9] M. S. M. A. R. M. A. Masud, «arxiv.org,» 29 marzo 2010. [En línea]. Available: https://arxiv.org/abs/1003.5629.

[10] J. Coalson, «Descripción general del formato,» Xiph.Org, 2022. [En línea]. Available: https://xiph.org/flac/documentation_format_overview.html.

[11] C. Vega, Codificación de Canal, Universidad de Cantabria, 2015.

[12] J. Mejuto Vázquez, Implementación software de un Simulador de un Sistema de Comunicaciones Inalámbricas, Universidad de Curuña, 2020.

[13] D. J. Huayas Flores, Las técnicas de modulación digital, Universidad Nacional de Educación Enrique Guzmán y Valle, 2021.

[14] K. Z. Y. C. Q. H. a. G. Y. Guangyi Zhang, «Towards Compatible Semantic Communication: A Perspective on Digital Coding and Modulation,» IEEE Communications Magazine, 2024.

[15] R. A. Escamilla, «Códigos para detección y corección de errores,» 2004.

[16] W. A. Ludeña Ignacio, La modulación y demodulación de ASK, Lima, Perú: Universidad Nacional de Educación Enrique Guzmán y Valle, 2021.

[17] W. Tomasi, Sistemas de Comunicaciones Electrónicas, Pearson Educación, 2003.

[18] J. J. &. A. P. F. Herrera Mejía, Monografía sobre modulación y demodulación FSK., Cartagena de Indias, 2009.

[19] M. Gallardo Tena, «La modulación y demodulación de la PSK,» 2021. [En línea]. Available: https://repositorio.une.edu.pe/handle/20.500.14039/6169.

[20] M. &. R. F. J. Cabrera-Bean, Transmisión digital a través de canales AWGN, Universitat Politècnica de Catalunya, 2009.

[21] A. Bensky, «Chapter 2 - Radio propagation,» 2019. [En línea]. Available: https://www.sciencedirect.com/science/chapter/monograph/abs/pii/B9780128154052000026.

[22] H. L. S. I. H. L. V. &. C. J. PARRA, Un simulador de canales inalámbricos, Valparaíso: Pontifica Universidad Católica de Valparaíso, 2006.

[23] M. M. Ali, «pmc.ncbi.nlm.nih.gov,» 12 junio 2024. [En línea]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11168661/.

[24] C. &. P. R. C. I. Torres Zambrano, «Análisis de un sistema de comunicaciones afectado por los desvanecimientos plano y lento tipo Rayleigh,» Ingeniería y Universidad, 2008.

[25] M. B. A. U. Jasone Astorga, Revisiting the Feasibility of Public Key Cryptography in Light of IIoT Communications, 2022.

[26] R. G. Acosta Arias, «Estudio teórico-práctico de los códigos no binarios de Reed-Salomon para la detección y corrección múltiple de errores utilizando el método matricial,» Escuela Politécnica Nacional, Quito, 1994.

[27] I. R. Martyn Riley, «Códigos Reed-Solomon,» 2022. [En línea]. Available: https://www.cs.cmu.edu/~guyb/realworld/reedsolomon/reed_solomon_codes.html.

[28] A. Hekal, Demodulation of Digital Signals Using Machine Learning Algorithms., Future Engineering Journal, 2025.

[29] C. K. S. N. P. P. J. L. S. C. &. R. K. M. Pothina, Diseño de bucle bloqueado de fase eficiente para aplicaciones de bajo consumo, Engineering Proceedings, 2023.

[30] F. M. D. J. C. V. O. J. C. &. V. A. F. Pertuz, Modelado y prueba de desempeño en tasa de errores de bits de un sistema de comunicaciones con modulación QAM y codificación Reed-Solomon utilizando LabVIEW para la enseñanza en pregrado, 2017.

[31] J. G. &. S. M. Proakis, Digital communications, New York: McGraw-hill, 2001.

[32] B. L. J.-H. L. Y.-C. K. Julius Ssimbwa, «Un estudio sobre los requisitos de modulación robusta para las comunicaciones satelitales personales de próxima generación,» 16 mayo 2022. [En línea]. Available: https://www.frontiersin.org/journals/communications-and-networks/articles/10.3389/frcmn.2022.850781/full.

[33] N. Rodríguez, Rodríguez, N. (2018). Monitoreo de la tasa de error de modulación en un transmisor digital de televisión, Quito, 2018.

[34] D. P.-L. K. &. G.-I. D. Ropero-Torres, «Evaluación de un sistema de comunicación óptico empleando modulación por desplazamiento de fase QPSK, 8PSK y 16PSK utilizando la técnica FDM,» Aibi revista de investigación, administración e ingeniería, 2020, pp. 76-83.

[35] S. Flores Asenjo, « Eficiencia espectral de una señal digital ASK multinivel,» 26 marzo 2018. [En línea]. Available: https://riunet.upv.es/entities/publication/61ec5ff3-e236-4a3b-98a7-717cbc44f4ae.

[36] J. M. T. &. P. H. P. Nova, Estudio y comparación en eficiencia espectral y probabilidad de error de los esquemas de modulación GMSK y DBPSK., Ingeniería e Investigación, 2008.

[37] O. Caja García, Librería Python para el aprendizaje y la implementación de redes neuronales, Valencia: Universidad Politécnica de Valencia, 2020.

[38] P. G. R. O. T. E. H. M. R. T. C. D. .. &. V. M. P. Virtanen, SciPy 1.0: fundamental algorithms for scientific computing in Python, Nature methods, 2020.

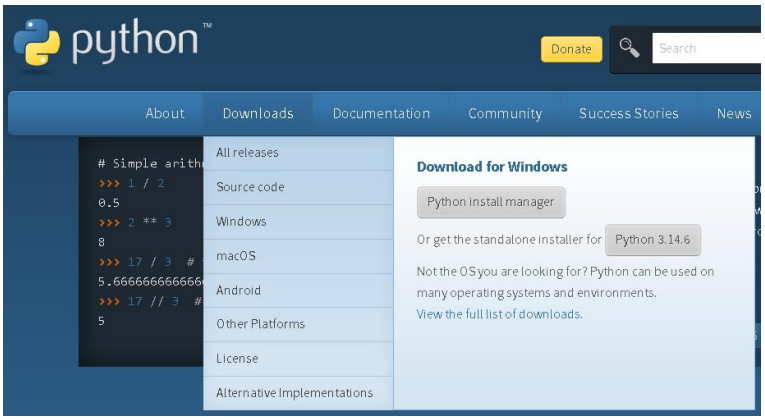
[39] F. O. R. D. L. M. J. O. &. M. J. S. Quintana, Matlab y Simulink con aplicaciones a la ingeniería-1ra edición, Ecoe Ediciones, 2024.

ANEXOS

Instalación de Python

Como se visualiza en la **Figura 78**, la página oficial de Python: https://www.python.org/downloads/ muestra la sección de Downloads, donde se puede seleccionar la versión adecuada para cada sistema operativo. En el panel derecho se selecciona la versión Python 3.14.6 y se descarga el instalador.

Figura 78. *Descarga oficial de Python desde la web.*



Nota. Elaboración propia.

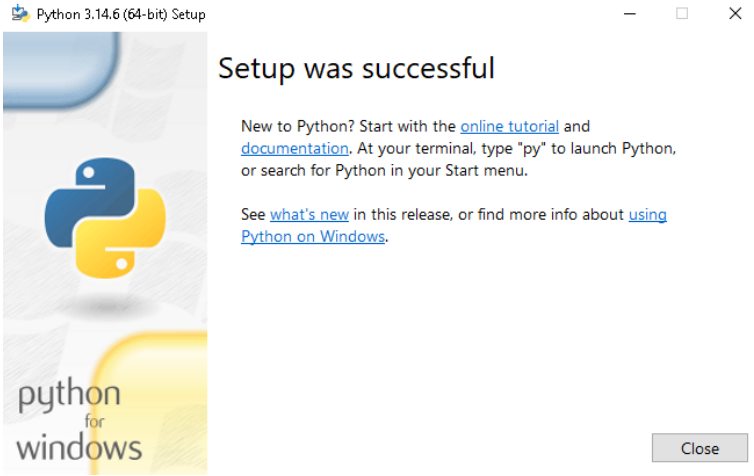
Como se observa en la **Figura 79**, al ejecutar como administrador, se abre la ventana de instalación de Python 3.14.6 en Windows. En este caso se ha seleccionado la opción rápida Install Now. Asimismo, se habilitaron las casillas “Use admin privileges when installing py.exe” y “Add python.exe to PATH”, lo que garantiza que Python se ejecute con privilegios administrativos y que el intérprete quede disponible desde la línea de comandos.

Figura 79. Instalador de Python en Windows.

Nota. Elaboración propia.

Como se aprecia en la **Figura 80**, la ventana del instalador confirma que la instalación de Python se ha realizado con éxito. Esta confirmación asegura que el intérprete y las herramientas asociadas ya están disponibles en el sistema.

Figura 80. *Instalación de Python completada en Windows.*

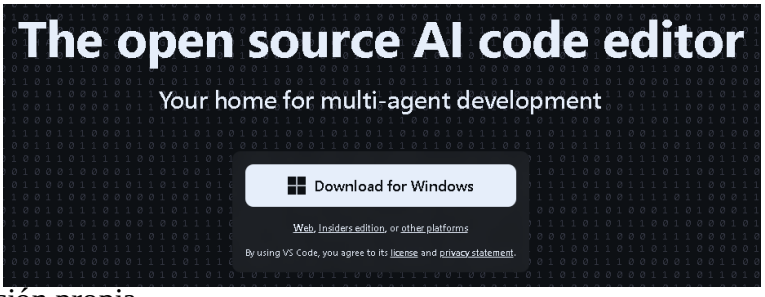


Nota. Elaboración propia.

Instalación de Visual Studio Code

Como se muestra en la **Figura 81**, la página oficial de Visual Studio Code presenta en el centro el botón “Download for Windows”, que permite obtener el instalador directamente para este sistema operativo.

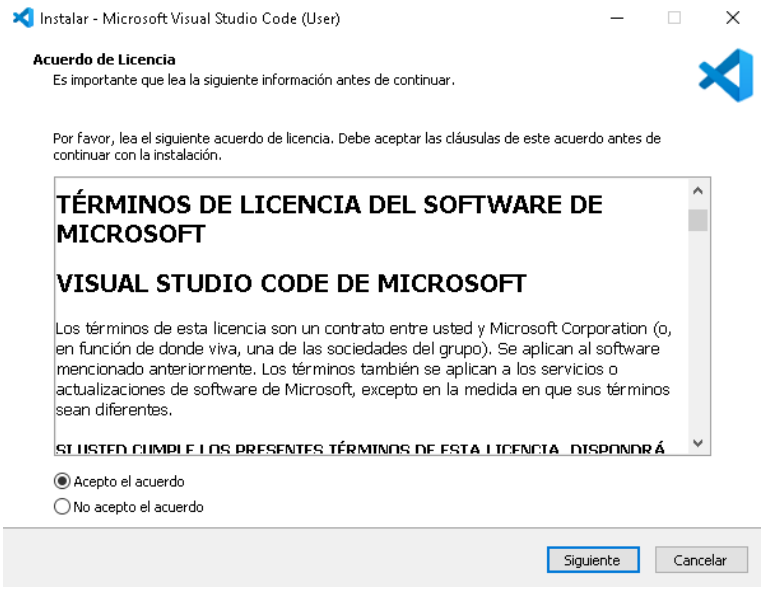
Figura 81. Descarga de Visual Studio Code.



Nota. Elaboración propia.

Como se observa en la **Figura 82**, durante la instalación de Visual Studio Code en Windows aparece la ventana del acuerdo de licencia. En este paso es necesario seleccionar la opción “Acepto el acuerdo” para poder continuar con la instalación.

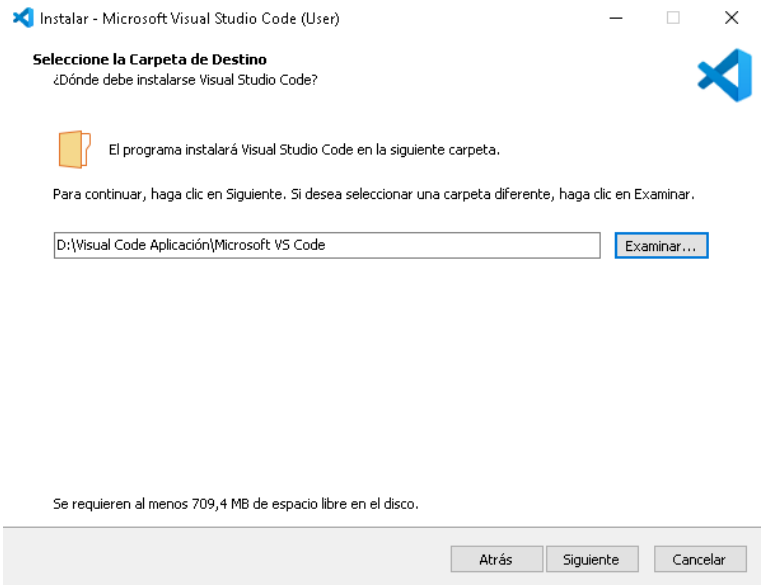
Figura 82. *Aceptación de términos de licencia en VS Code.*



Nota. Elaboración propia.

Como se muestra en la **Figura 83**, el instalador de Visual Studio Code solicita definir la carpeta de destino donde se instalará el programa. En este caso se indica la ruta y se continúa con la opción “Siguiente”.

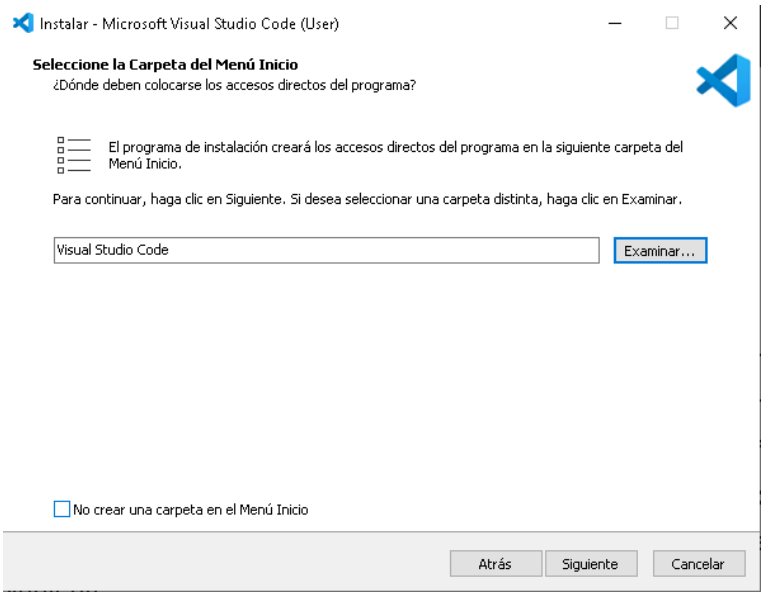
Figura 83. *Selección de carpeta de destino en VS Code.*



Nota. Elaboración propia.

Como se observa en la **Figura 84**, el instalador de Visual Studio Code solicita definir la carpeta del Menú Inicio donde se crearán los accesos directos del programa. Para continuar con la instalación, se debe hacer clic en “Siguiente”.

Figura 84. *Selección de carpeta en el Menú Inicio de VS Code.*

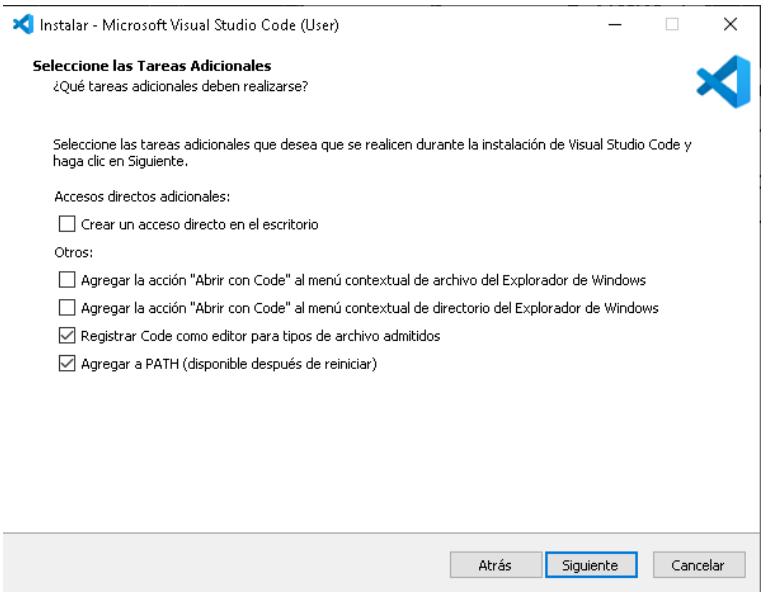


Nota. Elaboración propia.

Como se muestra en la **Figura 85**, el instalador de Visual Studio Code permite elegir tareas adicionales que integran el editor con el sistema operativo. En este caso se

mantienen las opciones predeterminadas y se hace clic en “Siguiente”.

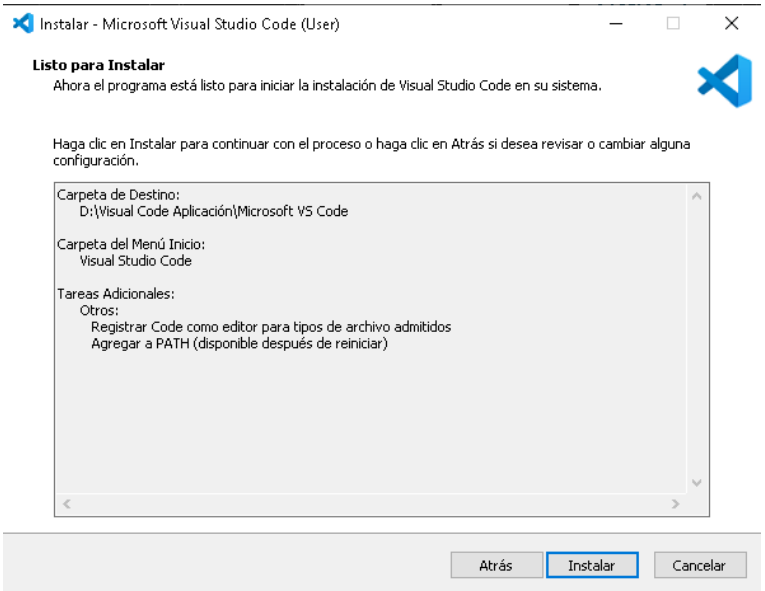
Figura 85. *Selección de tareas adicionales en VS Code.*



Nota. Elaboración propia.

Como se aprecia en la **Figura 86**, el asistente de instalación de Visual Studio Code muestra la ventana “Listo para Instalar”. Para continuar con la instalación se debe presionar el botón “Instalar”, mientras que las opciones “Atrás” y “Cancelar” permiten revisar o detener el proceso.

Figura 86. *Confirmación final antes de instalar VS Code.*



Nota. Elaboración propia.

Como se observa en la **Figura 87**, el asistente confirma que la instalación de Visual Studio Code ha finalizado correctamente. Para concluir el proceso basta con hacer clic en el botón “Finalizar”.

Figura 87. *Instalación de Visual Studio Code completada.*

Nota. Elaboración propia.

UNIVERSIDAD ESTATAL PENÍNSULA DE SANTA ELENA

FACULTAD DE SISTEMAS Y TELECOMUNICACIONES

CARRERA DE TELECOMUNICACIONES

GUÍA DE PRÁCTICAS

Implementación de técnicas de modulación digital en Python para determinar parámetros de transmisión en un canal inalámbrico simulado.

Autor:

Jorge Antonio Reyes Ramírez

Tutor:

Ing. Daniel Armando Jaramillo Chamba, Mgtr.

Dirigido a:

Estudiantes de sexto y séptimo semestre

de la carrera de Telecomunicaciones

La Libertad – Ecuador

2026

INTRODUCCIÓN

La presente guía de prácticas ha sido elaborada como material complementario para la asignatura de Comunicaciones Digitales. Su finalidad es fortalecer los conocimientos teóricos mediante actividades prácticas desarrolladas en Python, permitiendo al estudiante analizar el comportamiento de diferentes técnicas de modulación digital bajo condiciones reales de transmisión simuladas mediante ruido AWGN y desvanecimiento Rayleigh.

REQUISITOS DEL SISTEMA

REQUISITO	DESCRIPCIÓN
Sistema operativo	Windows 10 o superior
Lenguaje de programación	Python 3.x
Editor de código	Visual Studio Code
Bibliotecas	NumPy, SciPy, Matplotlib, OpenCV (cv2), Pillow, Reedsolo y CommPy
Espacio en disco	Al menos 2 GB disponibles
Memoria RAM	Mínimo 4 GB (8 GB recomendados)
Procesador	Intel Core i3 o equivalente (o superior recomendado)

La instalación y configuración de Python, Visual Studio Code y las bibliotecas necesarias se describen detalladamente en el apartado anterior del presente trabajo de integración curricular, específicamente en las secciones

- Instalación de Python
- Instalación de Visual Studio Code

donde se presenta el procedimiento paso a paso para la preparación del entorno de trabajo. Se recomienda al estudiante revisar previamente dichas secciones antes de iniciar las prácticas, con el propósito de garantizar que el entorno de desarrollo se encuentre

correctamente configurado y evitar inconvenientes durante la ejecución de los programas.

ORGANIZACIÓN DE LAS PRÁCTICAS PROPUESTAS EN LA GUÍA DE LABORATORIO

Cada práctica ha sido diseñada de manera secuencial para fortalecer el aprendizaje de los estudiantes, iniciando con la comprensión del funcionamiento general del sistema de comunicaciones digitales y avanzando hacia el estudio de las diferentes técnicas de modulación, la simulación del canal inalámbrico, los mecanismos de corrección de errores y el análisis de las principales métricas de desempeño. Esta estructura permite ir construyendo paso a paso los conocimientos teóricos y prácticos necesarios para entender cómo se comporta un sistema de comunicaciones digitales implementado en Python. Los códigos en Python pueden encontrarse en el siguiente repositorio: (M9vaCKmrrn7zGcGbyRk4-595tpQpGwYM8GBffEwUn4).

PRÁCTICA	TEMA
1	Modulación ASK
2	Modulación FSK
3	Modulación BPSK
4	Modulación QPSK
5	Modulación 16-QAM
6	Modulación 64-QAM
7	Comparación de métricas

PRÁCTICA 1

Objetivo

Implementar la modulación por desplazamiento de amplitud (ASK) en Python y analizar su comportamiento frente a un canal con ruido AWGN y desvanecimiento Rayleigh mediante las métricas de desempeño del sistema.

Fundamento teórico

La modulación ASK (Amplitude Shift Keying) es una técnica de comunicación

digital en la que la información binaria se transmite variando la amplitud de una señal portadora, mientras que su frecuencia y fase permanecen constantes. En esta práctica se evaluará el comportamiento de esta modulación mediante la observación de la señal transmitida, la señal recibida y el análisis de indicadores como la tasa de error de bits (BER), el Error Vector Magnitude (EVM) y el Modulation Error Rate (MER).

Procedimiento

- Correr el programa que simula la modulación ASK.
- Ajustar los parámetros de la simulación, como la SNR y el canal de transmisión.
- Realizar la transmisión de la información.
- Revisar la señal modulada, la señal recibida y el histograma del error.
- Anotar los valores obtenidos de BER, EVM, MER y eficiencia espectral.
- Analizar e interpretar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación ASK y podrá interpretar cómo el ruido y el desvanecimiento afectan la calidad de la transmisión a través de las métricas obtenidas durante la simulación.

Cuestionario

1. ¿De qué manera la modulación ASK representa los bits 0 y 1?
2. ¿Qué efecto tiene el ruido sobre la señal ya modulada?
3. ¿Qué nos dicen las métricas EVM y MER dentro de esta práctica?

PRÁCTICA 2

Objetivo

Implementar la modulación por desplazamiento de frecuencia (FSK) en Python y analizar su desempeño en un canal con ruido AWGN y desvanecimiento Rayleigh mediante el cálculo de las principales métricas de transmisión.

Fundamento teórico

La modulación FSK (Frequency Shift Keying) es una técnica de comunicación digital que transmite la información variando la frecuencia de la señal portadora para representar los estados binarios, mientras mantiene constante la amplitud. Debido a su mayor inmunidad al ruido en comparación con ASK, es ampliamente utilizada en

sistemas de comunicación digital donde se requiere una transmisión confiable. En esta práctica se evaluará su comportamiento mediante el análisis de la señal transmitida y recibida, así como de las métricas BER, EVM y MER.

Procedimiento

- Ejecutar el programa correspondiente a la modulación FSK.
- Configurar los parámetros de simulación (SNR y canal de transmisión).
- Realizar la transmisión de la información.
- Observar la señal modulada, la señal recibida y el histograma del error.
- Registrar los valores obtenidos de BER, EVM, MER y eficiencia espectral.
- Analizar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación FSK e identificará el efecto que producen el ruido y el desvanecimiento sobre la transmisión, interpretando los resultados obtenidos mediante las métricas de desempeño del sistema.

Cuestionario

1. ¿De qué manera representa la modulación FSK los bits 0 y 1?
2. ¿Cómo incide la relación señal-ruido (SNR) en el BER de esta modulación?
3. ¿Qué diferencias notas entre el comportamiento de ASK y FSK frente al ruido?

PRÁCTICA 3

Objetivo

Implementar la modulación por desplazamiento de fase binaria (BPSK) en Python y analizar su comportamiento frente a un canal con ruido AWGN y desvanecimiento Rayleigh mediante las métricas de desempeño del sistema.

Fundamento teórico

La modulación BPSK (Binary Phase Shift Keying) es una técnica de modulación

digital que representa la información binaria mediante dos estados de fase de la señal portadora separados por 180°. Debido a su alta resistencia al ruido y a las interferencias, es una de las modulaciones más robustas y ampliamente utilizadas en sistemas de comunicaciones digitales. En esta práctica se evaluará su desempeño mediante el análisis de la señal transmitida y recibida, así como de las métricas BER, EVM y MER.

Procedimiento

- ❖ Ejecutar el programa correspondiente a la modulación BPSK.
- ❖ Configurar los parámetros de simulación (SNR y canal de transmisión).
- ❖ Realizar la transmisión de la información.
- ❖ Observar la señal modulada, la señal recibida y el histograma del error.
- ❖ Registrar los valores obtenidos de BER, EVM, MER y eficiencia espectral.
- ❖ Analizar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación BPSK y analizará cómo las condiciones del canal afectan la calidad de la transmisión, interpretando los resultados obtenidos mediante las métricas de desempeño.

Cuestionario

1. ¿De qué manera la modulación BPSK representa los bits 0 y 1 a través de la fase de la portadora?
2. ¿Qué efecto produce la variación de la SNR sobre el BER en BPSK?
3. ¿Por qué se considera que BPSK es más robusta frente al ruido en comparación con ASK?

PRÁCTICA 4

Objetivo

Implementar la modulación por desplazamiento de fase en cuadratura (QPSK) en Python y analizar su comportamiento en un canal con ruido AWGN y desvanecimiento Rayleigh mediante las métricas de desempeño del sistema.

Fundamento teórico

La modulación QPSK (Quadrature Phase Shift Keying) es una técnica de modulación digital que transmite dos bits por cada símbolo mediante cuatro estados de fase de la señal portadora. Esta característica permite mejorar la eficiencia espectral en comparación con BPSK, manteniendo un buen desempeño frente al ruido. En esta práctica se evaluará el comportamiento de la modulación QPSK mediante el análisis de la señal transmitida y recibida, así como de las métricas BER, EVM y MER.

Procedimiento

- ❖ Ejecutar el programa correspondiente a la modulación QPSK.
- ❖ Configurar los parámetros de simulación (SNR y canal de transmisión).
- ❖ Realizar la transmisión de la información.
- ❖ Observar la señal modulada, la señal recibida y el diagrama de constelación.
- ❖ Registrar los valores obtenidos de BER, EVM, MER y eficiencia espectral.
- ❖ Analizar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación QPSK, interpretará su diagrama de constelación y analizará el efecto del ruido y el desvanecimiento sobre la calidad de la transmisión, utilizando las métricas de desempeño.

Cuestionario

1. ¿Cuántos bits logra transmitir cada símbolo en la modulación QPSK?
2. ¿Qué ventaja tiene QPSK frente a BPSK en cuanto a eficiencia espectral?
3. ¿Cómo se ven afectadas las métricas BER, EVM y MER en QPSK cuando disminuye la SNR?

PRÁCTICA 5

Objetivo

Implementar la modulación 16-QAM en Python y analizar su desempeño en un canal con ruido AWGN y desvanecimiento Rayleigh mediante el cálculo de las principales métricas de transmisión.

Fundamento teórico

La modulación 16-QAM (Quadrature Amplitude Modulation) combina variaciones de amplitud y fase para representar la información digital. Cada símbolo transmite cuatro bits, lo que incrementa la eficiencia espectral respecto a modulaciones como BPSK y QPSK. Sin embargo, al existir una menor separación entre los puntos de la constelación, esta modulación es más sensible al ruido y a las distorsiones del canal. En esta práctica se analizará su comportamiento mediante la observación del diagrama de constelación y las métricas BER, EVM y MER.

Procedimiento

- Correr el programa correspondiente a la modulación 16-QAM.
- Ajustar los parámetros de la simulación, como la SNR y el canal de transmisión.
- Llevar a cabo la transmisión de la información.
- Observar la señal modulada, la señal recibida y el diagrama de constelación.
- Anotar los valores obtenidos de BER, EVM, MER y eficiencia espectral.
- Analizar e interpretar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación 16-QAM, identificará la distribución de los símbolos en el diagrama de constelación y analizará cómo las condiciones del canal afectan el desempeño de la transmisión.

Cuestionario

- ¿Cuántos bits transmite cada símbolo dentro de la modulación 16-QAM?
- ¿Por qué 16-QAM logra una eficiencia espectral mayor que QPSK?
- ¿De qué manera influye el ruido del canal en el diagrama de constelación y en las métricas BER, EVM y MER?

PRÁCTICA 6

Objetivo

Implementar la modulación 64-QAM en Python y evaluar su desempeño en un canal con ruido AWGN y desvanecimiento Rayleigh mediante el análisis de las principales métricas de transmisión.

Fundamento teórico

La modulación 64-QAM (Quadrature Amplitude Modulation) es una técnica de alta eficiencia espectral que combina variaciones de amplitud y fase para representar la información digital. Cada símbolo transmite seis bits, permitiendo una mayor velocidad de transmisión sin incrementar el ancho de banda. Sin embargo, debido a la alta densidad de puntos en el diagrama de constelación, esta modulación es más susceptible al ruido y al desvanecimiento del canal. En esta práctica se analizará su comportamiento mediante el diagrama de constelación y las métricas BER, EVM y MER.

Procedimiento

- Ejecutar el programa correspondiente a la modulación 64-QAM.
- Definir los parámetros de la simulación, incluyendo la SNR y el canal de transmisión.
- Transmitir la información configurada.
- Revisar la señal modulada, la señal recibida y el diagrama de constelación resultante.
- Registrar los datos obtenidos de BER, EVM, MER y eficiencia espectral.
- Analizar los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante comprenderá el funcionamiento de la modulación 64-QAM, identificará la distribución de los símbolos en el diagrama de constelación y evaluará el compromiso existente entre la alta eficiencia espectral y la mayor sensibilidad al ruido y al desvanecimiento del canal.

Cuestionario

1. ¿Cuántos bits transmite cada símbolo en la modulación 64-QAM?
2. ¿Por qué la modulación 64-QAM es más sensible al ruido que 16-QAM?
3. ¿Qué relación existe entre la eficiencia espectral y el desempeño de la modulación 64-QAM en canales con ruido y desvanecimiento?

PRÁCTICA 7

Objetivo

Comparar el desempeño de las modulaciones digitales implementadas en Python mediante el análisis de las métricas BER, EVM, MER y eficiencia espectral con el fin de identificar sus ventajas y limitaciones bajo las mismas condiciones de transmisión.

Fundamento teórico

Las métricas de desempeño hacen posible evaluar de manera objetiva la calidad de un sistema de comunicaciones digitales. La tasa de error de bits (BER) indica cuántos errores se produjeron durante la transmisión, la magnitud del error del vector (EVM) mide qué tan alejados quedan los símbolos recibidos respecto a los transmitidos, la tasa de error de modulación (MER) expresa en decibelios qué tan buena es la calidad de la señal modulada y la eficiencia espectral señala cuánta información se logra transmitir por cada unidad de ancho de banda disponible. Al analizar estas métricas en conjunto, es posible comparar cómo se comportan las distintas técnicas de modulación trabajadas a lo largo del proyecto.

Procedimiento

- ❖ Ejecutar las simulaciones correspondientes a las modulaciones estudiadas.
- ❖ Anotar los valores de las métricas obtenidos en cada una.
- ❖ Comparar los resultados apoyándose en las tablas y gráficas que genera el programa.
- ❖ Analizar las diferencias de desempeño encontradas entre las distintas modulaciones.
- ❖ Elaborar conclusiones con base en los resultados obtenidos.

Resultados esperados

Al finalizar la práctica, el estudiante será capaz de interpretar las métricas de desempeño y comparar las diferentes técnicas de modulación, identificando cuál ofrece el mejor equilibrio entre robustez frente al ruido, calidad de la señal y eficiencia espectral según las condiciones de transmisión.

Cuestionario

- ¿Qué modulación presentó el menor BER y cuál obtuvo la mayor eficiencia espectral?
- ¿Qué relación existe entre las métricas BER, EVM y MER al evaluar la calidad de una transmisión?
- ¿Qué técnica de modulación recomendaría para un canal con alto nivel de ruido? Justifique su respuesta con base en los resultados obtenidos.