Visualización de Diagramas en el Tiempo para Sistemas de Comunicación Digitales Basados en Hardware Reconfigurable

Melissa Eugenia Diago Mosquera
William Augusto Gómez Paz

Director: M.Sc Víctor Fabián Miramá Pérez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL – Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Agosto del 2014

Visualización de Diagramas en el Tiempo para Sistemas de Comunicación Digitales Basados en Hardware Reconfigurable

Melissa Eugenia Diago Mosquera
William Augusto Gómez Paz

Trabajo de Grado presentado como requisito para obtener el título de Ingeniero en Electrónica y Telecomunicaciones

Director: M.Sc Víctor Fabián Miramá Pérez

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telecomunicaciones
GRIAL – Grupo de Radio e InALámbricas
GNTT – Grupo I+D Nuevas Tecnologías en Telecomunicaciones
Gestión Integrada de Redes, Servicios y Arquitecturas de Telecomunicaciones
Popayán, Agosto del 2014

AGRADECIMIENTOS

A Dios, la Virgen y el Espíritu Santo por guiar mi vida e iluminar mis decisiones.
A mis padres, Juan Carlos y Maria Fluvia por su incondicional apoyo, sus sonrisas e infinito amor que generan en mí la estabilidad que necesito para lograr las metas que trazo en mi vida, compartiendo conmigo este inmenso logro.
A mi hermanita menor, Sophia, que desde el momento que hizo parte de mi vida sueño con poder brindarle enorme apoyo.
Y a cada una de las personas que contribuyeron en mi formación y educación, aportando conocimiento y valiosas experiencias. Sin olvidar el gran afecto de mi tía Astrid que con su fiel amor, ha estado para mí en cada momento.

Melissa Eugenia Diago Mosquera

A Dios Todopoderoso que siempre me ha guiado por el camino correcto y me ha enseñado que de cada problema siempre hay un aprendizaje. Gracias por tu continua compañía y por permitirme realizar un sueño más.
A mi padre William por darme ejemplo de tenacidad y responsabilidad, gracias a su enorme sacrificio aprendí a valorar mucho más la vida y a ser una mejor persona cada día, a mi madre Alba Mery, ese ser especial que siempre está a mi lado dándome motivación y consejos, que siempre creyó en mis capacidades y con su inmenso amor me da fuerzas para seguir adelante. Me siento muy orgulloso de ser su hijo y le doy gracias a Dios por permitirme nacer en este hogar, de ahora en adelante espero contribuirles todo el apoyo que me brindaron.
A mis hermanos Eliana y Andrés que me impulsaron a mejorar todos los días para en un futuro convertirme en su guía y apoyo, en especial a mi hermano Joan que durante toda mi vida ha sido mi ejemplo a seguir debido a su constante dedicación y lucha, brindándome cariño y confianza en los momentos más difíciles.
A mis tíos que siempre estuvieron atentos para darme su ayuda, en especial mi tía socorro que siempre me ha dado su amor y ayuda en todo los momentos.

William Augusto Gómez Paz

TABLA DE CONTENIDO

LISTA DE FIGURAS

LISTA DE ACRONIMOS

AWGN	*Additive White Gaussian Noise*, Ruido Gaussiano Blanco Aditivo
BPSK	*Binary Phase Shift Keying*, Modulación por Desplazamiento de Fase Binario
DBPSK	*Differential Binary Phase Shift Keying*, Modulación por Desplazamiento de Fase Binario Diferencial
DQPSK	*Differential Quadrature Phase Shift Keying*, Modulación por Desplazamiento de Fase en Cuadratura Diferencial
FPGA	*Field Programmable Gate Array*, Campo de Matriz de Compuertas Programables
FSK	*Frecuency Shift Keying*, Modulación por Desplazamiento en Frecuencia
GUI	*Graphical User Interface*, Interfáz Gráfica de Usuario
GUIDE	*Graphical User Interface Development Environment*, Entorno de Desarrollo de Interfaz Gráfica de Usuario
MATLAB	*MATrix LABoratory*, Laboratorio de Matrices
M-PSK	*M-ary Phase Shift Keying*, Modulación por Desplazamiento de Fase *M-ario*
MSK	*Minimum Shift Keying,* Modulación por Desplazamiento Mínimo
OQPSK	*Offset Quadrature Phase Shift Keying*, Modulación por Desplazamiento de Fase en Cuadratura Compensada
PSK	*Phase Shift Keying*, Modulación por Desplazamiento de Fase
QAM	*Quadrature Amplitude Modulation*, Modulación de Amplitud en Cuadratura
QPSK	*Quadrature Phase Shift Keying*, Modulación por Desplazamiento de Fase en Cuadratura
RS-232	*Recommended Standard 232*, Estándar Recomendado No. 232

INTRODUCCIÓN

Con el pasar de los tiempos la era analógica se ha opacado por la continua evolución de los sistemas de comunicación digital, ya que estos brindan múltiples ventajas, entre ellas: el incremento de la velocidad en la trasmisión de datos, la inmunidad al ruido, la regeneración de la señal y un mayor procesamiento de información. Por todo lo anterior, se hace indispensable realizar un estudio detallado del comportamiento de este tipo de comunicaciones, enfocado en las diversas modulaciones digitales encargadas de optimizar la transmisión de datos.

Conforme la era digital ha trascendido a lo largo de los años, se ha generado y almacenado una enorme cantidad de información vital para los sistemas de comunicación digital, que al ser observada de forma lineal, consta de datos inconexos que fuera de contexto carecen de significado alguno. De forma que la visualización se presenta como una táctica hacia la comprensión e interpretación de la información. Sin embargo, obtener visualizaciones precisas implica ejecutar dos procesos: trabajar con datos de calidad, es decir, precisos, completos y bien tratados e identificar los datos adecuadamente, para realizar de forma estructurada una correcta extracción. En definitiva se deduce que la tarea de procesar adecuadamente la información debe ser efectuada antes de proceder al tratamiento gráfico.

Por todo lo planteado hasta ahora surge el presente trabajo de grado, el cual se basa en simplificar la tarea del análisis de modulaciones digitales mediante la elaboración de una herramienta software, capaz de visualizar los diagramas en el tiempo para posteriormente ser evaluados. Es relevante tener en cuenta que los trabajos de grado que hacen parte del proyecto macro “Diseño e Implementación de un Prototipo de Comunicación de Datos Basado en Hardware Reconfigurable Fase 1”, son los encargados de diseñar e implementar las modulaciones digitales.

Este documento presenta las generalidades de los sistemas de comunicación digital; describe el desarrollo e implementación de la herramienta y finalmente exhibe las pruebas y muestra el análisis de resultados.

CAPITULO 1

GENERALIDADES

La función esencial de un sistema de comunicaciones es transmitir la información a través de un canal hasta un destino o receptor, para esto se requiere un procesamiento de la información, en donde la modulación es importante, pues consiste en el traslado de frecuencia de la señal a transmitir. Una vez recibida la señal el receptor debe demodular la señal con el fin de obtener la información original.

En este capítulo se hace referencia a las componentes de un sistema de comunicación digital, algunas técnicas de modulación digital y los diagramas en el tiempo que se utilizan como herramientas de análisis para el trabajo de grado.

1.1. Componentes de un Sistema de Comunicación Digital

En la Figura 1.1 se observa el diagrama de bloques de los elementos que componen un sistema de comunicación digital.

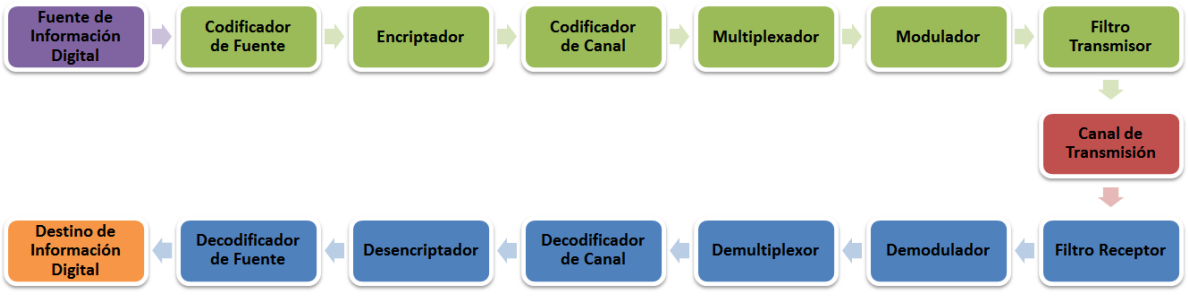


Figura 1.1. Diagrama de bloques de un sistema de comunicación digital.
Adaptado de [1]**.**

A continuación se realiza una breve descripción de cada uno de los bloques.

1.1.1. Fuente de información digital

Este bloque es el encargado de convertir cualquier mensaje de transmisión en una señal de tipo digital. Teniendo en cuenta que si es de tipo analógico se implementan las etapas de muestreo y cuantificación, con el fin de obtener la señal adecuada para el sistema.

1.1.2. Codificación de fuente

La labor de este elemento es eliminar la redundancia y la irrelevancia de la fuente [2], la primera implica la repetición o predicción de la información, lo que respecta a la segunda infiere la información que no es posible apreciar y cuya eliminación no afecta al contenido del mensaje, generando a su vez una compresión de datos. En recepción se cuenta con el decodificador de fuente, el cual se encarga de rescatar y restablecer la información original.

1.1.3. Encriptador

La confidencialidad, integridad y autenticidad en los sistemas de comunicación digital es brindada por este componente, en el que se realiza un proceso de cifrado [3], de tal manera que la información resultante es ilegible permitiendo la transmisión segura del mensaje original de tal forma que solo pueda ser recuperada por el bloque desencriptador que cuenta con la capacidad de obtener la señal inicial en función del mensaje encriptado.

1.1.4. Codificación de canal

Con el fin de prevenir la llegada de información errónea al destino debido al ruido que se añade en el canal, el codificador agrega redundancia a la señal [2], con lo cual es posible la detección y/o corrección de errores en transmisión.

1.1.5. Multiplexación

Este componente cuenta con la capacidad de seleccionar una de varias entradas de datos y llevar la señal seleccionada a una única salida [4], incrementando la cantidad de datos que pueden ser enviados sobre un canal dentro de cierta cantidad de tiempo y ancho de banda. Con lo que respecta a la demultiplexación, esta consiste en la extracción de una determinada señal de entre las múltiples que se pueden encontrar en el canal de comunicaciones.

1.1.6. Modulación

Consiste en modificar algunas de las características de una señal portadora de alta frecuencia, como su amplitud, frecuencia o fase, mediante la señal moduladora, lo cual permite alcanzar la mayor distancia posible y disminuir el impacto del ruido, además, al modular se pretende facilitar la radiación para la propagación de la señal, posibilitar la

multiplexación y superar las limitaciones de los equipos existentes [5]. La modulación es una de las funciones más importantes del sistema de comunicación digital, pues en su ausencia no sería posible el envío y la recepción de más de dos señales a través del mismo canal, por ello, gracias a este módulo se hace posible la identificación de las señales moduladas para realizar la demodulación en etapas posteriores, lo cual implica la utilización de técnicas que permiten recuperar la información original.

1.1.7. Filtro

En transmisión se ocupa de limitar y adecuar la potencia de transmisión de la señal modulada en función del ancho de banda establecido en el canal [6] y en cuanto a recepción, este bloque es el encargado de realizar el filtrado necesario para eliminar señales indeseables y minimizar el ruido de la señal.

1.2. Técnicas de Modulación

La primera fase del proyecto diseño e implementación de un prototipo de comunicación de datos basado en hardware reconfigurable [7] está fundamentada en la modulación por desplazamiento de fase (PSK, *Phase Shift Keying*), la modulación de amplitud en cuadratura (QAM, *Quadrature Amplitude Modulation*), la modulación por desplazamiento de frecuencia (FSK, *Frecuency Shift Keying*) y la modulación por mínimo desplazamiento (MSK, *Minimum Shift Keying*). Estas representan una señal digital mediante una señal analógica que varía sus parámetros conforme la señal de información digital tome diferentes valores o estados.

Es necesario resaltar que la transmisión de una señal digital se puede llevar a cabo de dos formas [8], la primera es transmitirla directamente sin emplear alguna técnica de modulación sobre una portadora, a este método se le conoce como transmisión en banda base. La segunda forma, consiste en efectuar algún tipo de modulación digital previa a la transmisión de la señal y es llamada transmisión en radiofrecuencia (RF, *Radio Frequency*).

Este apartado está dedicado a describir el comportamiento general de algunos tipos de modulación digital, como lo son M-PSK, DBPSK, DQPSK, QAM, FSK y MSK.

1.2.1. Modulación por desplazamiento de fase PSK

En esta técnica de modulación digital, se toma la fase de una portadora de amplitud constante como parámetro de variación para la transmisión de información y dependiendo de los valores de la entrada digital, la señal analógica modulada tendrá diversas fases de salida [9].

La cantidad de fases de salida depende del grado de modulación M-PSK que se realice, en donde M simboliza la cantidad de posibles estados que se obtendrán en la salida, a su vez, esto permite encontrar la cantidad de bits codificados N, a partir de la relación $N = \log_2 M$. Es por esto que PSK convencional presenta múltiples esquemas de modulación de acuerdo a los bits que se deseen analizar de la señal digital de entrada. Además de PSK convencional, existe otra alternativa denominada PSK diferencial en la cual la asignación de la fase a un símbolo no está determinada, lo que conlleva a generar cambios de fase en cada transición de bit. En esta técnica se observan los siguientes cambios de fase: si el siguiente símbolo es un uno, el cambio de fase es de 270°, pero si el símbolo a seguir es un cero, se dará un cambio de fase de 90°.

Este trabajo de grado considera las modulaciones convencionales BPSK, QPSK y 8-PSK, el caso particular OQPSK y las modulaciones diferenciales DBPSK y DQPSK. A continuación se describe cada una de ellas con las formas de onda de salida que le corresponde a cada tren de bits:

- **Modulación por desplazamiento de fase binaria BPSK**

Es la forma más simple de PSK ya que solo tiene dos fases de salida posible para una sola frecuencia portadora, la cual cambia su fase entre dos ángulos que están separados 180° a medida que la señal digital de entrada varía. En la Figura 1.2 se muestra la relación fase-tiempo de una forma de onda BPSK.

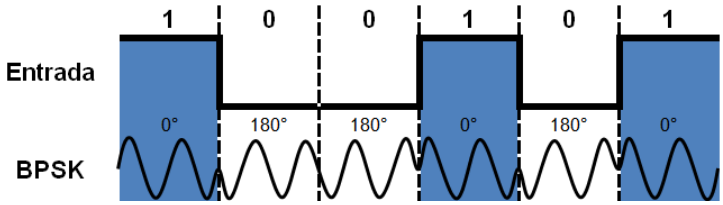


Figura 1.2. Modulación BPSK.

- **Modulación por desplazamiento de fase en cuadratura QPSK**

QPSK, es otra forma de modulación digital de tipo angular con amplitud constante, debido a que se tienen cuatro posibles fases de salida para una sola frecuencia de portadora, es necesario contar con cuatro condiciones de entrada diferentes, por lo que se requieren dibits[1], obteniendose: 00, 01, 10 y 11. En consecuencia, cada codigo dibit genera una de las cuatro fases (+135°, +45°, -45° y -135°). La Figura 1.3 muestra la relación de la fase de salida contra tiempo de un modulador QPSK.

[1] Grupo de 2 bits.

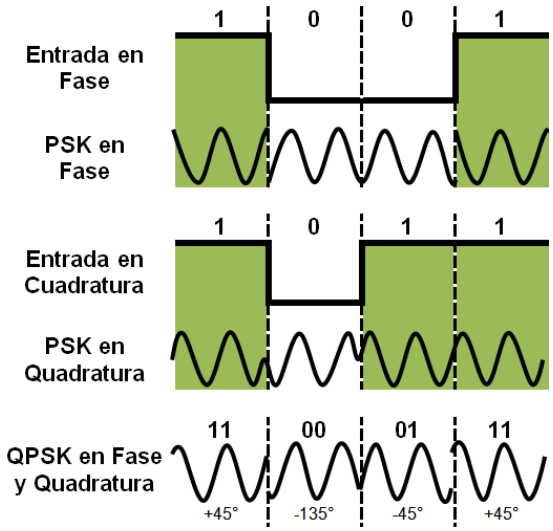


Figura 1.3. Modulación QPSK.

- **Modulación por desplazamiento de fase en cuadratura compensada OQPSK**

Es una forma modificada de QPSK en la que una de las formas de onda de bits en los canales I o Q es retrasada en la mitad de un tiempo de bit respecto al otro canal. En la Figura 1.4 se muestra como el bit de la entrada en cuadratura se desplaza medio periodo de bit con respecto al de la entrada en fase.

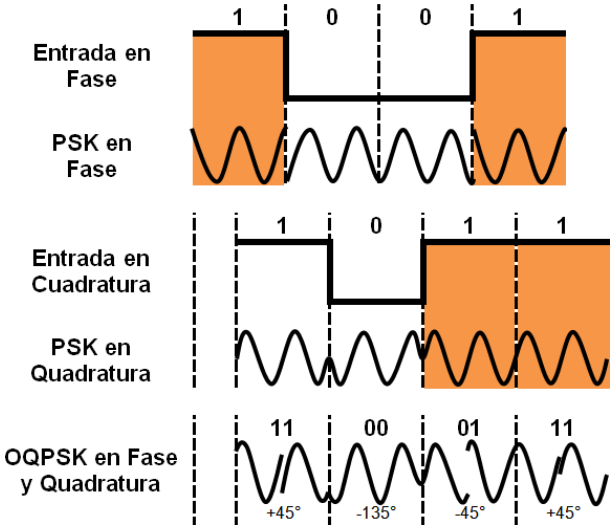


Figura 1.4. Modulación OQPSK.

- **Modulación por desplazamiento de fase binaria diferencial DBPSK**

A diferencia de PSK, en esta técnica de modulación diferencial la información de entrada binaria está contenida en la diferencia, entre dos elementos sucesivos de señalización y no en la fase absoluta. Es decir, cuando se transmite un uno, no se produce un cambio de

fase en la señal analógica. Por el contario, siempre que se transmita un cero, en la señal de salida se obtendrá un cambio de fase de +180°, tal y como se observa en la Figura 1.5.

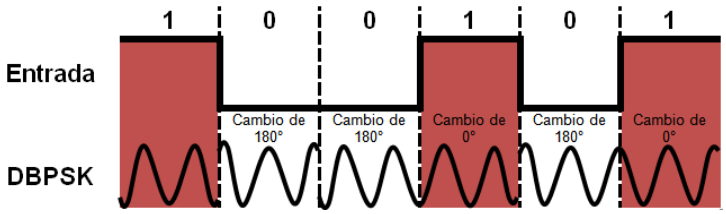


Figura 1.5. Modulación DBPSK.

- **Modulación por desplazamiento de fase en cuadratura diferencial DQPSK**

Al igual que DBPSK, este esquema de modulación trabaja con cambios de fase en la señal de salida en fase y cuadratura, obteniendo cambios cuando el bit transmitido ha sido un cero y manteniendo su forma de onda al trasmitir un uno, al sumar las señales en fase y cuadratura la señal DQPSK que se obtiene es la observada en la Figura 1.6.

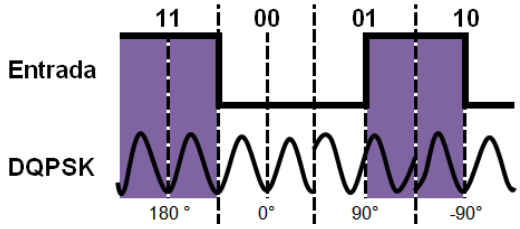


Figura 1.6. Modulación DQPSK.

1.2.2. Modulación de amplitud en cuadratura QAM

En la modulación de amplitud en cuadratura la información se modula tanto en amplitud como en fase [10], lo que conlleva a que la señal portadora sea modificada conjuntamente en estos dos parámetros, generando así una salida analógica QAM.

En este esquema de modulación se cuenta con sistemas M-arios, en los cuales, mediante grupos de k bits se obtienen M salidas diferentes relacionadas a través de la expresión $M = 2^k$, por ejemplo, en el caso específico de 16QAM existen 4 valores en cuadratura y 4 en fase, por lo que se tienen 16 posibles estados para la señal y debido a que $16 = 2^4$, se envían cuatro bits por símbolo, de los cuales dos bits se encuentran en fase y dos en cuadratura. Este trabajo de grado tiene en cuenta las técnicas de modulación 16QAM y 64QAM que contienen grupos de k bits de 4 y 6 respectivamente.

1.2.3. Modulación por desplazamiento de frecuencia FSK

En esta modulación a cada valor de la señal de información se le asignan valores de la frecuencia de la señal portadora con amplitud y fase constante, el caso más simple de

esta modulación es FSK binaria en el cual los dos estados lógicos son representados por ondas analógicas, donde cero se representa como una onda a una frecuencia específica y uno como una onda a otra frecuencia diferente como se muestra en la Figura 1.7, por lo que con cada cambio de la señal digital binaria se obtiene una variación en la frecuencia de la portadora.

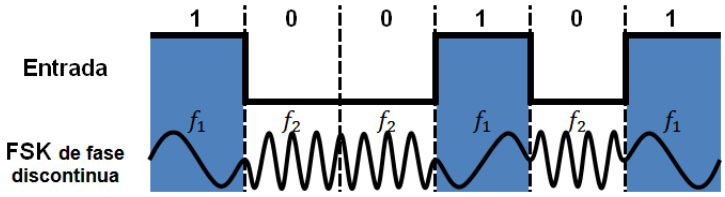


Figura 1.7. Modulación FSK.

1.2.4. Modulación por mínimo desplazamiento de frecuencia MSK

Se le conoce como modulación de mínimo corrimiento a la modulación FSK con un desplazamiento mínimo en la fase de más o menos 90° por símbolo, que produce una fase continua en la señal analógica de salida [11]. Representando el uno lógico como un salto en frecuencia de +90° y el cero lógico en un salto de −90°. En la Figura 1.8 se puede notar que cuando la frecuencia de salida cambia se presenta una transición continua suave, por ello no existen cambios abruptos de fase.

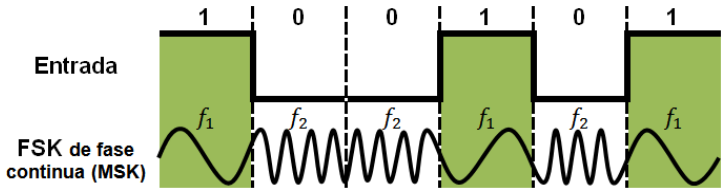


Figura 1.8. Modulación MSK.

En MSK las dos frecuencias están sincronizadas con la tasa de bits de la moduladora binaria, es decir que existe una relación temporal precisa entre ambas frecuencias. La selección de las frecuencias se realiza de manera que se encuentren separadas por un múltiplo impar exacto de la mitad de la tasa de bits de la frecuencia central, con lo que se asegura la existencia de una transición de fase continua en la señal analógica de salida cuando cambia de la frecuencia asignada a uno a la frecuencia de cero, o viceversa.

1.3. Diagramas en el Tiempo

Para el análisis del comportamiento de los sistemas de comunicaciones digitales se emplean diversas técnicas y mediciones de desempeño, a través de ellas se alcanza una apreciación de la calidad del enlace de transmisión, aunque en ciertas ocasiones muchas de ellas no logran establecer y visualizar la causa del problema. Es por esto que los diagramas en el tiempo son primordiales para el análisis de la señal, de sus formas de

onda y sus símbolos propagados en un enlace de comunicaciones, logrando así la visualización de los efectos de los diversos mecanismos de degradación de la señal como: el jitter, la interferencia intersimbólica (ISI, *IntersymbolInterference*), los niveles de ruido, la potencia de la señal, entre otros. Es así como el diagrama de constelación, la trayectoria de la señal y el diagrama del ojo permiten examinar la señal de transmisión a lo largo del enlace de comunicaciones.

1.3.1. Diagrama de constelación

Es un método de representación de los estados de los símbolos en el plano complejo en términos de amplitud y fase en los diversos[2] esquemas de modulación digital, las componentes de los símbolos que se encuentran en fase con la señal portadora hacen referencia al eje horizontal y las componentes en cuadratura (90°) al eje vertical [12]. La constelación ilustra el modulo del número complejo, el cual está dado por la distancia que separa el origen de las coordenadas y la ubicación del símbolo, y la fase que corresponde al ángulo que se forma con el eje horizontal de cada posible símbolo que conforma la modulación.

Los posibles datos ubicados en el diagrama son descritos como puntos fijos de la constelación que representan símbolos modulados a ser transmitidos. Además de esto el diagrama de constelación puede proporcionar información acerca del tipo de interferencia y distorsión de la señal en el medio de transmisión, como lo son el ruido gaussiano que distorsiona los puntos de la constelación, el ruido de fase que ocasiona la dispersión de la constelación en forma rotacional y la atenuación que genera movimiento de los puntos desde las esquinas hasta el centro del diagrama.

En la Figura 1.9 se muestra un ejemplo del diagrama de constelación de la modulación QPSK, donde el producto entre el bit I y la señal portadora ($\sin w_c t$) y el bit Q con una portadora desplazada en fase 90° ($\cos w_c t$) son sumados para obtener la señal QPSK.

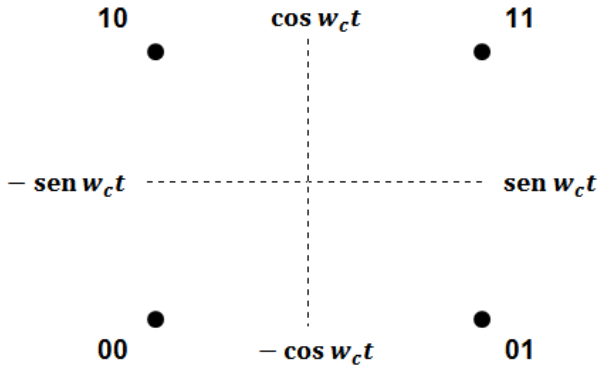


Figura 1.9. Diagrama de constelación QPSK.

[2] PSK, QAM, FSK y MSK: Modulaciones definidas en [7].

En la Tabla 1.1 se encuentran registrados los valores que puede tomar la señal de salida y su representación en fase.

Q	I	Señal de Salida	Fase de Salida
0	0	$-\cos w_c t - \sin w_c t$	-135°
0	1	$-\cos w_c t + \sin w_c t$	-45°
1	0	$+\cos w_c t - \sin w_c t$	$+135^{\circ}$
1	1	$+\cos w_c t + \sin w_c t$	$+45^{\circ}$

Tabla 1.1. Tabla de verdad de QPSK.

1.3.2. Trayectoria de la señal

Este tipo de diagrama en el tiempo permite visualizar los caminos de las transiciones entre todos los posibles estados de una señal modulada [13], debido a que grafica la componente de la señal en fase en función de la componente en cuadratura, como se observa en la Figura 1.10, en donde se muestra la trayectoria de la señal de la modulación QPSK.

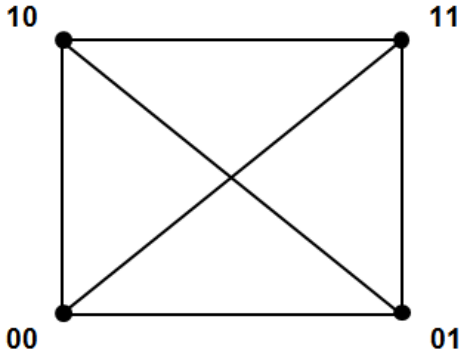


Figura 1.10. Trayectoria de la señal de QPSK.

En efecto, a través de este diagrama se puede observar como existen puntos de convergencia, que equivalen a los símbolos representados en el diagrama de constelación, con lo cual se puede hacer un análisis en presencia de ruido, efecto que conlleva la desviación del punto de convergencia.

1.3.3. Diagrama del ojo

La calidad de una señal digital y la integridad del sistema que lo transmite pueden ser medidas con satisfacción a través del diagrama del ojo. Este diagrama corresponde a la superposición de las combinaciones de los posibles unos y ceros en un rango de tiempo o cantidad de bits de la señal determinada. En gran parte esta técnica es cualitativa sin embargo se pueden obtener algunos datos cuantitativos útiles en términos de tendencia y fidelidad, es decir, se pueden detectar y apreciar las tendencias a las que conlleva el comportamiento de la señal, logrando establecer parámetros de diseño que mejoren el

sistema de comunicación y además verificar la calidad de la señal en tanto sea fiel a la original.

Cabe resaltar que es posible efectuar el análisis de este diagrama mediante dos formas, la primera examinando las características de la forma de onda del pulso referidas a cuatro propiedades fundamentales del diagrama del ojo, el nivel cero, nivel uno, cruce de amplitud y cruce en el tiempo que se pueden ver alteradas por diversos factores en el medio de transmisión y la segunda forma es posible comparando las máscaras obtenidas y las establecidas como referencia, las cuales son regiones específicas en el diagrama del ojo que representan zonas no permitidas para las señales ya que de ser así se presentan problemas y errores en la transmisión [14].

Para ejemplificar la formación del diagrama del ojo, se tiene en cuenta una secuencia de 3 bits, en la que la cantidad total de posibles combinaciones es de 8, tal y como se observa en la Figura 1.11. Al superponer las distintas combinaciones de unos y ceros generadas se obtiene el diagrama del ojo de esta secuencia de bits, en el cual no es necesario tener en cuenta las cadenas de 3 unos y 3 ceros consecutivos, puesto que debido a la superposición de las otras cadenas, estas quedan determinadas implícitamente.

En la Figura 1.11 se observan las propiedades fundamentales del ojo, en donde el nivel uno y cero corresponden a la medición del valor promedio del uno y cero lógico respectivamente; con respecto al cruce del ojo, se encuentra compuesto por dos partes, el cruce de tiempo y amplitud, donde se hace referencia al instante de tiempo y al nivel de voltaje en el que ocurre la apertura del ojo y su posterior cierre. Teniendo en cuenta estos dos parámetros se define el periodo de bit que establece el periodo de apertura y cierre del ojo.

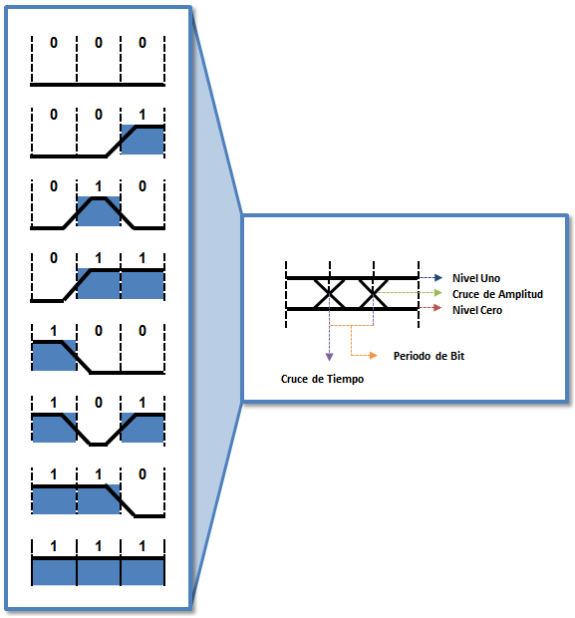


Figura 1.11. Formación del diagrama del ojo.

CAPITULO 2

DESARROLLO E IMPLEMENTACIÓN DE LA INTERFAZ DE VISUALIZACIÓN

Este capítulo describe el desarrollo de la herramienta de visualización, mediante la adaptación que se realiza a la secuencia establecida por el Modelo Lineal Secuencial y la metodología que sigue la programación extrema. El modelo general que se tiene en cuenta cumple con las secciones de planificación, diseño, implementación y pruebas. La etapa de diseño es adaptada a través de los fundamentos, principios y actividades de la programación extrema con el fin de lograr productividad y efectividad en el diseño de la herramienta.

Las secciones a continuación puntualizan el cumplimiento de las etapas de planificación, diseño y desarrollo ejecutadas a lo largo del trabajo de grado.

2.1. Planificación

Esencialmente la sección de planificación consiste en el análisis de requisitos, por tanto se hace indispensable la recopilación de información que conlleve a la correcta elección de las herramientas a utilizar. Para llevar a cabo la etapa de planificación es importante contar con las estrategias necesarias que permitan lograr los objetivos.

- Diseñar una herramienta software capaz de generar los diagramas del ojo, de constelación y trayectoria de la señal para diferentes[3] esquemas de modulación digital.

- Adaptar la herramienta a sistemas de comunicación digitales basados en hardware reconfigurable.

- Evaluar la interfaz software mediante el análisis de diagramas de tiempo para diferentes[4] esquemas de modulación digital.

Para el desarrollo e implementación de la herramienta de visualización se identifican los siguientes alcances:

- Generar y desplegar el diagrama de constelación, la trayectoria de la señal y el diagrama del ojo de los datos modulados extraídos antes y después del canal, lo cual permite observar los efectos que conlleva el procesamiento de la señal y el envío a través de un canal AWGN.

- Lograr la correcta formación de los estados según la modulación digital utilizada por parte del diagrama de constelación, los caminos que se trazan entre cambios de estados por parte de la trayectoria de la señal y la adecuada superposición de los cambios de la señal por parte del diagrama del ojo.

- La herramienta no presenta valores de mediciones concretas solo se enfoca en el despliegue de los diagramas en el tiempo con el fin de llevar a cabo un análisis de visualización.

Para concluir la etapa de planificación en la Tabla 2.1 se establecen las especificaciones que debe cumplir el sistema para lograr un óptimo desempeño.

Dispositivo	Requerimiento Hardware	Requerimiento Software
Hardware Reconfigurable	FPGA SPARTAN-3A de XILINX	ISE Design Suite 14.4 System Edition
Terminal de Usuario	Procesador: Pentium Dual-Core @ 1,87 GHz Velocidad de Recepción: 9600 Baudios	Windows XP Professional, Windows 7 (32 – 64 bits) MATLAB R2011a, R2011b, R2012a y R2012b

Tabla 2.1. Especiaciones hardware y software.

En los apartados a continuación se describen las características generales del hardware reconfigurable y la interfaz de usuario.

[3] PSK, QAM, FSK y MSK: Modulaciones definidas en [7].
[4] Los esquemas de modulación digital serán obtenidos a través de los trabajos de grado de [7].

2.1.1. Hardware reconfigurable

Un arreglo de compuertas lógicas programables es un dispositivo semiconductor que contiene bloques de lógica reprogramable dispuestos para la implementación de funcionalidades personalizadas en hardware, desarrolladas en tareas de cómputo digital en software y compiladas en un archivo de configuración (*bitstream*) que contiene información acerca de las conexiones que deben ser realizadas internamente [15]. Asimismo, estos dispositivos son reconfigurables por lo que toman una nueva lógica cuando se compilan con una configuración diferente, entre otras ventajas que presenta un FPGA se resalta su buen rendimiento que permite lograr más en cada ciclo de reloj, también sobresalen sus precios factibles que brindan soluciones en hardware basadas en FPGAs y fiabilidad muy importante al momento de poner en ejecución un proyecto, ya que mientras las herramientas software ofrecen un entorno de programación, los circuitos de un FPGA son una implementación segura de la puesta en marcha de un programa. Un software a nivel driver se encarga de administrar los recursos de hardware y el sistema operativo administra la memoria y el ancho de banda del procesador, además de esto, el núcleo de un procesador únicamente ejecuta una instrucción a la vez, lo que hace posible la obstrucción de las tareas entre sí. A diferencia de los FPGAs, que no necesitan sistemas operativos, minimizan los retos de fiabilidad y proveen hardware preciso que se encuentra dedicado a cada tarea.

FPGA SPARTAN-3A DE XILINX

La Figura 2.1 muestra la tarjeta de entrenamiento SPARTAN-3A *Starter Kid Board*, la cual es manufacturada y distribuida por Digilent Inc.

Figura 2.1. SPARTAN-3A *Starter Kid Board*.

La tarjeta cuenta con el FPGA XC3S700A-FG484 de XILINX, el cual contiene 700K compuertas que equivalen a 13248 celdas lógicas tal y como se observa en la Tabla 2.2[5].

Device	System Gates	Equivalent Logic Cells	CLB Array (One CLB = Four Slices)				Distributed RAM bits	Block RAM bits	Dedicated Multipliers	DCMs	Maximum User I/O	Maximum Differential I/O Pairs
			Rows	Columns	CLBs	Slices						
XC3S700A	700K	13248	48	32	1472	5888	92K	360K	20	8	372	165

Tabla 2.2. Resumen de los atributos del FPGA SPARTAN-3A.

La arquitectura de la SPARTAN-3A está compuesta por cinco elementos fundamentales [16], los primeros son los bloques lógicos configurables (CLBs, *Configurable Logic Blocks*) basados en tablas de búsqueda (LUTs, *Look Up Tables*) que implementan funciones lógicas (parte combinacional) y en Flip-Flops, usados como elementos de almacenamiento, los CLBs son la unidad básica de un FPGA. Los segundos corresponden a los bloques de entrada/salida (IOBs, *Input Output Blocks*) que controlan el flujo de datos y la lógica interna entre los puertos de entrada y salida. Los terceros son los bloques de memoria (RAM, *Random Access Memory*) que poveen almacenamiento de información. Los cuartos son los multiplicadores que aceptan palabras de 18 bits y entregan productos de 36 bits, logrando un manejo eficiente de los datos. Finalmente, los quintos elementos competen a la red de distribución de relojes (DCM, *Digital Clock Manager*) que proveen precisión en la señal de reloj, eliminando los cambios de fase así como desviaciones en la señal de reloj debido a efectos externos. Para mayor claridad la Figura 2.2 muestra los elementos mencionados, aclarando que el FPGA XC3S700A agrega dos DCMs en la mitad de las dos columnas de los bloques de memoria (RAM) y los multiplicadores.

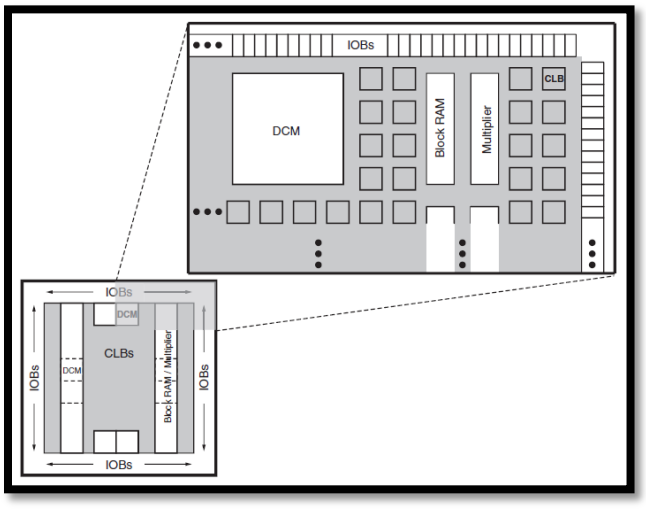


Figura 2.2. Arquitectura del FPGA SPARTAN-3A[6].

[5] Obtenida a partir del datasheet de la tarjeta SPARTAN-3A.
[6] Obtenida a partir del datasheet de la tarjeta SPARTAN-3A.

Entre los elementos que conforman la tarjeta SPARTAN-3A, se encuentran los dos puertos seriales RS-232 que juegan un papel muy importante en la implementación del presente trabajo de grado, razón por la cual se describe con mayor detalle. Los dos puertos hacen referencia al conector DB9 hembra (DCE, *Data Comumunications Equipment*) y el DB9 macho (DTE, *Data Terminal Equipment*) de los cuales solo se utiliza el DCE, pues este es el que va conectado directamente al puerto serial disponible en la mayoría de computadores a través de un cable serial, a diferencia del DTE que se encarga de controlar otros periféricos RS-232.

2.1.2. Interfaz de usuario

Una interfaz gráfica de usuario hace referencia a un software que funciona como puente de comunicación entre una computadora y su usuario, permitiéndole interactuar con esta a través del uso de símbolos, metáforas visuales y la opción de seleccionar y arrastrar elementos con el mouse en vez de introducir texto como comandos de línea. Los sistemas que funcionan junto a una GUI son más fáciles y simples de entender y manejar, esto debido a que el usuario no necesita conocer, memorizar o aprender comandos nuevos o adicionales, razones por las cuales hoy en día las interfaces graficas de usuario son empleadas por la mayoría de los sistemas operativos modernos y programas de aplicaciones.

- **Ventajas principales de una GUI**

La interacción con los sistemas se hace mediante múltiples ventanas permitiendo ir de una tarea a la otra sin perder de vista la información generada previamente.

Debido a su simplicidad a la hora de utilizar permite que usuarios que no tienen mucha experiencia en computadores no necesiten extensos periodos de tiempo para aprender a utilizar el entorno de una GUI.

Dado que la accesibilidad a los elementos de una GUI se hace mediante punteros y clics, hace más fácil para el usuario manejar las aplicaciones en vez de utilizar el tecleado para este mismo tipo de tarea.

Los errores pueden ser detectados y corregidos más fácilmente que con una interfaz de comandos de línea.

Mediante su uso, el usuario puede operar fácilmente el computador sintiendo que tiene más control sobre él.

- **Componentes básicos de una GUI**

Una interfaz gráfica cuenta con los siguientes conjuntos de componentes u objetos visuales:

→ Ventanas: Encargadas de desplegar la información requerida por el usuario.

→ Punteros: Símbolo que aparece en la pantalla y ayuda al usuario a seleccionar objetos y archivos.

→ Iconos: Pequeñas imágenes que representan comandos archivos o ventanas, los cuales son ejecutados dando le clic con el puntero, convirtiendo el icono en una ventana.

→ Menú: Agrupación de múltiples opciones que pueden ser escogidas por el usuario.

2.2. Diseño

En este apartado se definen la arquitectura y la funcionalidad de los pasos a seguir para llevar a cabo el desarrollo de la herramienta de visualización. Inicialmente se explica la metodología de la programación extrema aplicada a la sección de diseño; posteriormente se enfatiza en el diseño de las dos etapas indispensables para el desarrollo del objeto del presente trabajo de grado y finalmente se especifican las herramientas de diseño elegidas para cumplir con las etapas definidas.

Para el diseño de la herramienta se establecen dos etapas, la primera de adaptación y la segunda de visualización, cada una origina un diagrama de bloques general que permite representar el proceso que ocurre y la estructura con la que se cuenta.

2.2.1. Metodología de la programación extrema aplicada

En la búsqueda por obtener un software de calidad y alto rendimiento, se hace indispensable encontrar una metodología que permite mejorar la productividad y efectividad, razones por las cuales se adopta la programación extrema como metodología para el diseño del software, pues está orientada a la maximización de las cualidades de los programadores y de los sistemas, además está basada en cuatro valores principales: comunicación, necesaria para que los desarrolladores conozcan los requerimientos del sistema, simplicidad, requerida para ir siempre hacia la solución más simple, retroalimentación (*feedback*), muy relacionada con comunicación pues es importante para entender los requerimientos del usuario, y coraje, útil para enfrentar cada problema que pueda surgir, también cuenta con cuatro actividades que potencializan el desarrollo del software [17].

Teniendo en cuenta que el actual trabajo de grado cuenta con los requerimientos de esta metodología, la programación extrema proporciona grandes ventajas [18] para ser escogida como la metodología guía para el desarrollo de la interfaz gráfica que permita la visualización de diagramas en el tiempo para sistemas de comunicación digitales basados en hardware reconfigurable.

La Figura 2.3 muestra la adaptación del diagrama secuencial que se aplica para el diseño y desarrollo del actual sistema, aclarando que en el momento que la etapa de pruebas (*Testing*) satisfaga las necesidades y objetivos planteados se tendrá un diseño óptimo para el sistema.

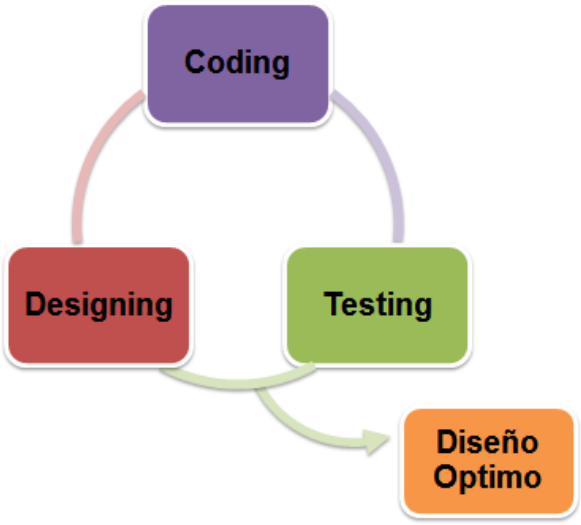


Figura 2.3. Adaptación del diagrama secuencial de la metodología XP aplicada.

En los apartados de esta sección se detalla la adaptación de cada actividad de la programación extrema al sistema actual.

- **Coding**

Para lograr la visualización de diagramas en el tiempo de los esquemas de modulación digital PSK, QAM, FSK y MSK basados en hardware reconfigurable, se hace indispensable la creación de instrucciones o códigos que logren generar los diagramas de constelación, la trayectoria de la señal y el diagrama del ojo en la interfaz gráfica.

Así mismo para evaluar la funcionalidad de la herramienta a través de pruebas es ineludible el diseño y desarrollo de un módulo hardware. Este módulo se implementó en el mismo FPGA en el que se encuentra el modulador y el canal AWGN desarrollado por los otros[7] proyectos; puesto que para el envío de las parejas de datos modulados y

[7] Trabajos de grado que conforman el proyecto macro de [7].

contaminados con ruido fue indispensable adaptarlos al medio, es decir, al formato en que van a ser transmitidos, lo cual exige la creación de una lógica necesaria para la adaptación.

- **Testing**

Una vez que se realice la primera actividad, es necesario comprobar que lo elaborado hasta el momento se está ejecutando correctamente, por lo que las pruebas son fundamentales para probar el funcionamiento de la interfaz.
Inicialmente las pruebas consisten en ingresar señales moduladas provisionales a la interfaz gráfica con el fin de observar los respectivos diagramas en el tiempo y verificar la respuesta de los códigos generados, con esto, se demuestra que la interfaz está cumpliendo con el objetivo de graficar. Para efectuar la adaptación se hace indispensable ajustar los datos provisionales obtenidos a partir del FPGA a la transmisión mediante el protocolo RS-232 [19], lo cual significa que las señales generadas en el FPGA deben ser enviadas hasta la interfaz gráfica a través del protocolo mencionado.

Las últimas pruebas se ejecutan de dos formas, la primera es observando las señales originadas por el FPGA en un osciloscopio para visualizar la existencia de alguna señal, y la segunda es conectando el FPGA al computador mediante un cable adaptador DB9 a USB para comprobar la recepción de datos. Una vez verificada la llegada de datos se puede probar la visualización gráfica de las señales, lo que demuestra que efectivamente los datos se han adaptado y se están visualizando.

- **Designing**

Con las etapas de adaptación y visualización se hace pertinente estructurar y organizar la lógica necesaria, de tal forma que el sistema trabaje en conjunto de la mejor manera. Ya que cada etapa se encuentra conformada por actividades de vital importancia, que se nombran a continuación:

Etapa de adaptación

→ Recepción de datos.
→ Protocolo de transmisión.
→ Serialización y envío de datos.

Etapa de visualización

→ Recepción de datos.
→ Visualización de diagramas en el tiempo.

En la Figura 2.4 se tiene el diagrama de bloques general del sistema, donde se detalla el orden de las etapas definidas en el diseño.

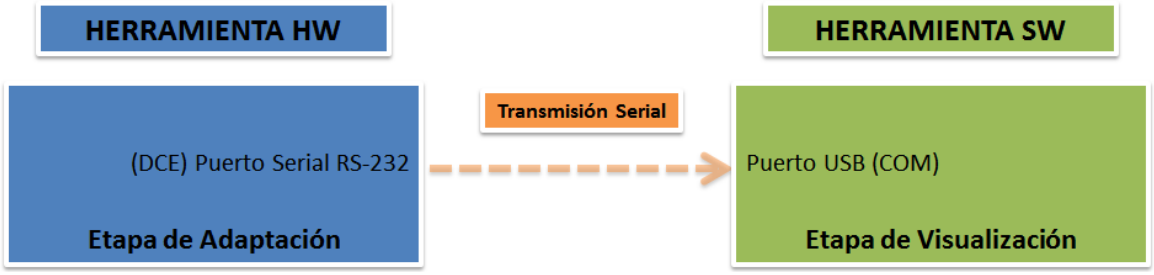


Figura 2.4. Diagrama de bloques general del sistema.

En los apartados 2.2.2 y 2.2.3 se describe con mayor detalle los módulos del diseño establecido, haciendo referencia a su funcionalidad en el sistema integrado.

2.2.2. Etapa de adaptación - Módulo hardware

La primera etapa por la que se interesa el presente trabajo de grado consiste en la adaptación de los datos obtenidos de las diferentes[8] modulaciones para la visualización de sus diagramas en el tiempo. Esta adaptación se logra a través del diagrama de bloques general de la Figura 2.5, en donde se resaltan tres actividades fundamentales que adecuan los datos para ser transmitidos al terminal de usuario.

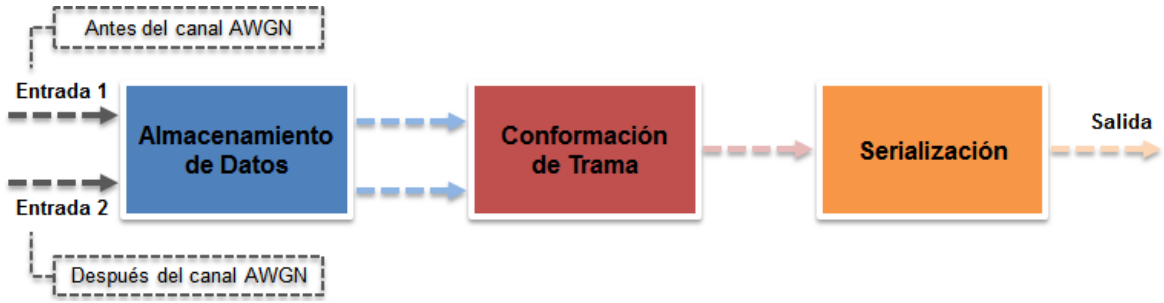


Figura 2.5. Diagrama de bloques de la etapa de adaptación.

La Figura 2.6 muestra el diagrama de flujo de la etapa de adaptación. Esta permite que se presente con mayor detalle las actividades de almacenamiento de datos, conformación de trama y serialización de la Figura 2.5.

[8] PSK, QAM, FSK y MSK: Modulaciones definidas en [7].

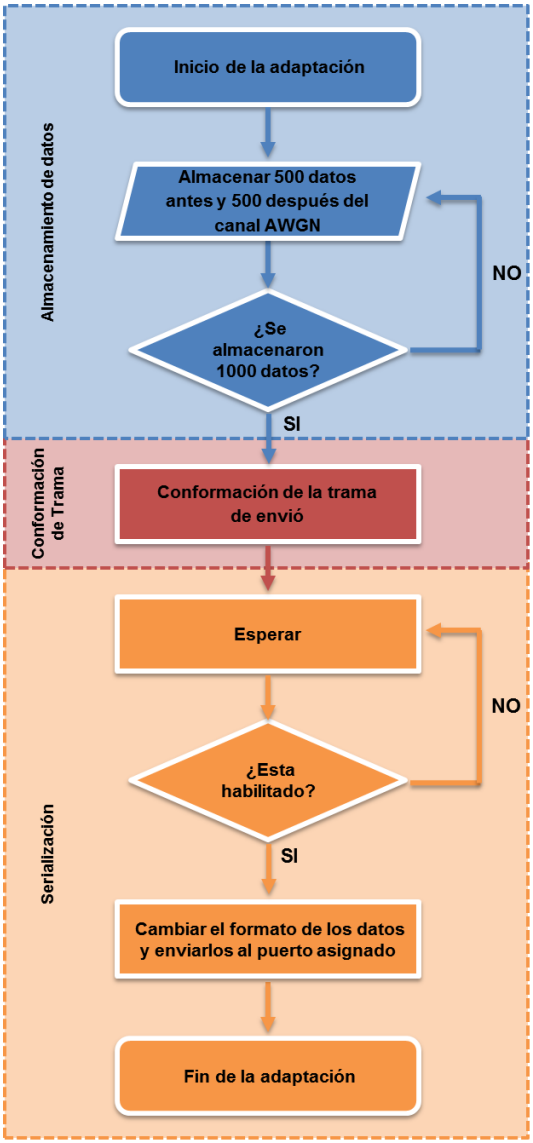


Figura 2.6. Diagrama de flujo de la etapa de adaptación.

A continuación se detalla el funcionamiento de cada actividad:

- **Almacenamiento de datos**

La entrada de esta etapa corresponde a las señales moduladas antes y después del canal AWGN, estas son representadas como números complejos de la siguiente forma:

Señal modulada antes del canal = a + ib
Señal modulada después del canal = c + id

En donde, a y c hacen referencia a las componentes reales de las señales y b y d a las componentes imaginarias de las señales.

Las señales se reciben en tiempo real, por lo que es necesario un bloque capaz de almacenar la entrada en vectores que puedan ser de fácil acceso. Para llevar a cabo esta función se crea un bloque de registro, encargado de generar un espacio de memoria temporal para almacenar 500 datos por cada entrada (4 entradas) durante el periodo de tiempo requerido para el procesamiento de la información. Una vez se cumpla el tiempo de procesamiento el bloque se reinicia, con lo que el vector vuelve a cargar 500 datos nuevos, listos para ser procesados nuevamente.

El bloque cuenta con 4 salidas, las cuales corresponden a las posiciones de los vectores en un tiempo determinado. Tomando como ejemplo la entrada a y su respectiva salida w, la lógica del bloque es la siguiente: por cada periodo de tiempo se tiene un dato en la entrada a, este es almacenado en la primera posición del vector; al entrar un segundo dato, este se almacena en la primera posición y el dato que ya se tenía pasa a la segunda posición. Por consiguiente, la primera posición se toma como referencia para establecer la salida del vector debido a que está se actualiza continuamente con el dato de entrada, tal como se muestra en la Figura 2.7. Transcurridos los 500 periodos de tiempo el vector se empieza a llenar con ceros, debido a que estos serán utilizados como bandera para el siguiente bloque.

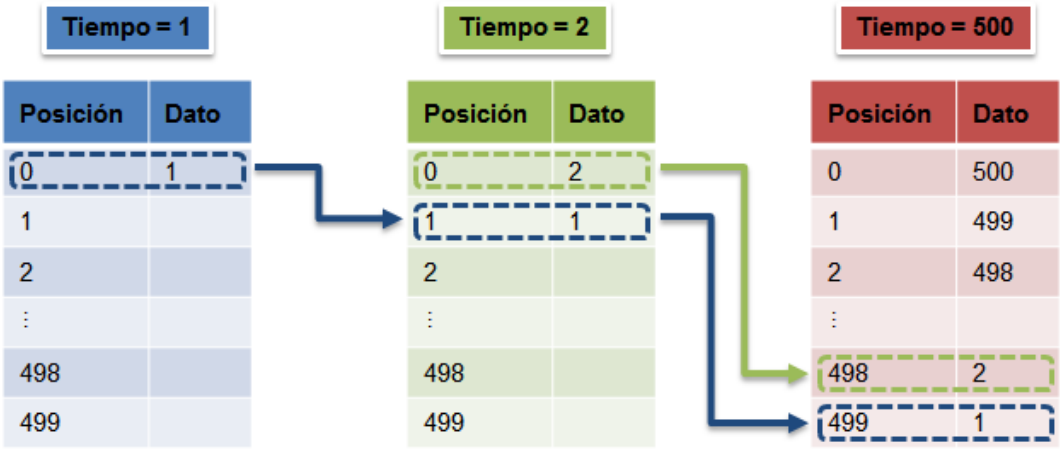


Figura 2.7. Inserción de datos al vector.

- **Conformación de trama**

Fue necesario diseñar un protocolo que permita organizar todos los datos de tal forma que se tenga acceso a ellos a través de un único vector. Lo anterior se estableció como funcionalidad principal de este bloque, quien está a cargo del almacenamiento de los números complejos y la estructuración de los mismos, es decir, este bloque guarda los 500 datos antes y después del canal junto con 3 banderas que informan partes especificas del vector, tal y como se muestra en la Figura 2.8.

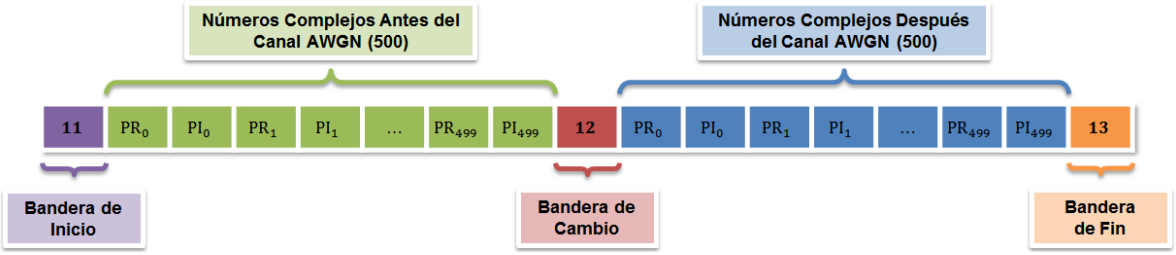


Figura 2.8. Conformación de la trama de transmisión.

Las partes que conforman el vector de 2002 posiciones de la Figura 2.8 son:

Bandera de inicio

→ Posición en el vector: 0
→ Definida en: 11
→ Descripción: esta bandera es la encargada de alertar el inicio del vector, por lo que se está a la espera del primer número complejo modulado antes del canal de transmisión.

Números complejos antes del canal AWGN

→ Posición en el vector: 1 - 1000
→ Descripción: esta parte del vector consta de 500 parejas de números complejos modulados, donde la parte real de cada pareja se encuentra en las posiciones impares del vector y la imaginaria en las posiciones pares, esto se realiza con el fin de recuperar inmediatamente el numero complejo completo en recepción.

Bandera de cambio

→ Posición en el vector: 1001
→ Definida en: 12
→ Descripción: esta bandera es la encargada de separar las señales moduladas sin ruido de las que se les ha adicionado ruido al salir del canal de transmisión AWGN, por lo que se espera recibir el primer número complejo con ruido.

Números complejos después del canal AWGN

→ Posición en el vector: 1002 - 2001
→ Descripción: esta parte del vector se conforma por 500 parejas de números complejos modulados a los cuales se les ha adicionado ruido, donde la parte real de cada pareja se encuentra en las posiciones pares del vector y la imaginaria en las posiciones impares.

Bandera de fin

→ Posición en el vector: 2002
→ Definida en: 13
→ Descripción: esta bandera es la encargada de informar que se han guardado satisfactoriamente 1000 datos modulados con y sin ruido.

Una vez conformado el vector con el protocolo de estructuración se procede a tener una única salida, en la cual se envía en cada periodo de tiempo todas las posiciones del vector en orden ascendente, lo cual significa que estas se recorren hasta liberar el vector completamente.

- **Serialización**

Debido a que los datos a transmitir tienen un formato de 16 bits y el envió de estos se realizó mediante el conector DB9, el cual cuenta con un solo pin para la transmisión y por ende solo es posible enviar un bit a la vez, se hace necesario serializar la salida de 16 bits; por lo tanto este bloque está compuesto por una máquina de 20 estados, 16 para la transmisión de cada bit y 4 más para la inserción de banderas, encargadas de la señalización en el protocolo de transmisión RS-232.

Al entrar el vector de datos de la Figura 2.8 a la máquina de estados se accede a su primera posición, es entonces cuando cada estado comienza a extraer desde el bit menos significativo (estado 2 ≡ Bit_0), hasta el más significativo (estado 19 ≡ Bit_15); el control que se realiza para pasar de un estado a otro se lleva a cabo cuando se recibe la señal equivalente a la tasa de muestreo proveniente del bloque habilitar. Una vez se han enviado todos los bits y se ha llegado al último estado, la lógica de la máquina se reinicia, pero esta vez con la siguiente posición del vector. Este proceso continua hasta llegar a la última posición y una vez enviado el bit más significativo, se termina el ciclo teniendo como resultado todos los bits en el formato requerido. En la Figura 2.9 se observa con mayor detalle la secuencia descrita.

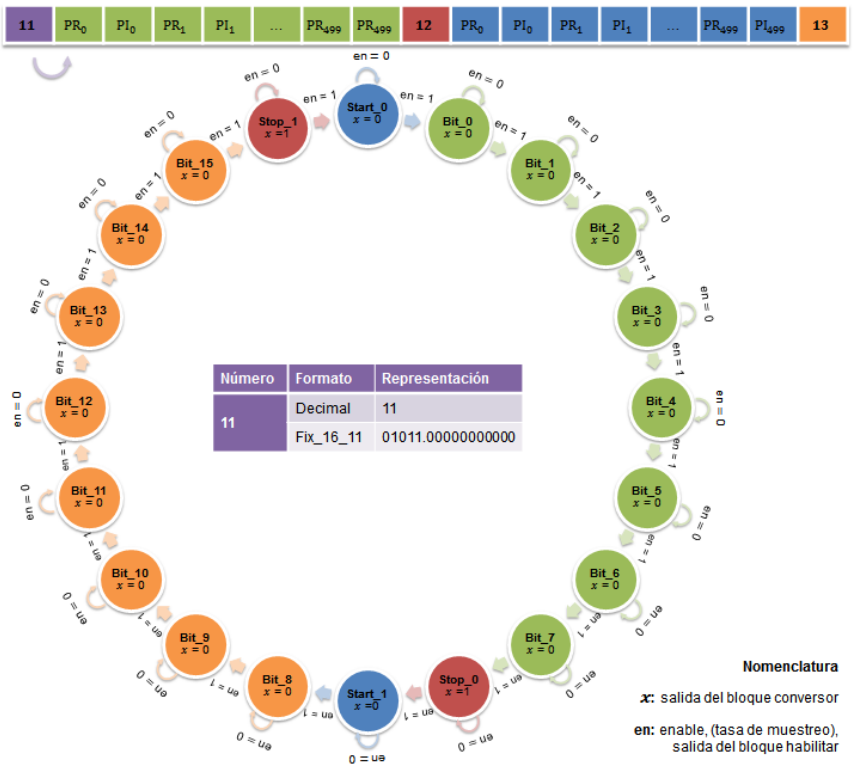


Figura 2.9. Máquina de estados de la serialización.

Es necesario enfatizar en la importancia de los 20 estados, ya que estos logran la transmisión satisfactoria de los datos desde la FPGA hasta el computador, para esto se debe seguir el cronograma de un envío RS-232 con 8 datos y sin bit de paridad. Para complementar la explicación de la Figura 2.9, en la Figura 2.10 se especifican los estados de la trama correspondiente al envió del número 11, conformado por 16 números binarios.

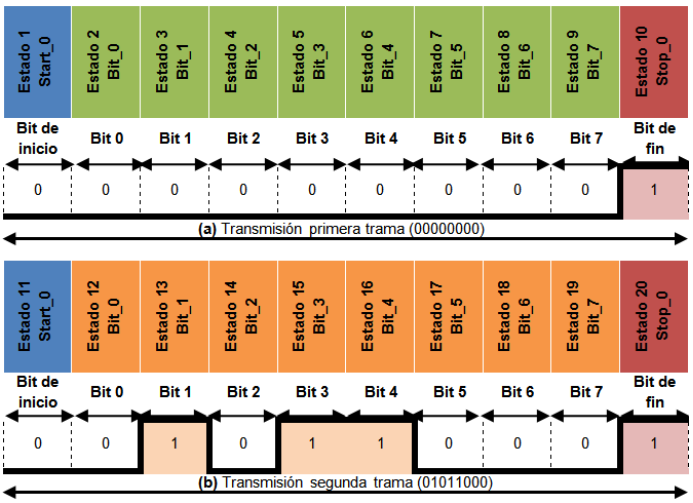


Figura 2.10. (a) Primera trama de envío. (b) Segunda trama de envió. Estándar RS-232, aplicado al número 11 en formato binario.

2.2.3. Etapa de Visualización - Herramienta software

En esta sección se detalla el diseño de la interfaz gráfica a través del diagrama de bloques de la Figura 2.11 y del diagrama de flujo que se muestra en la Figura 2.12, los cuales buscan detallar la estructura básica del diseño de la interfaz gráfica.

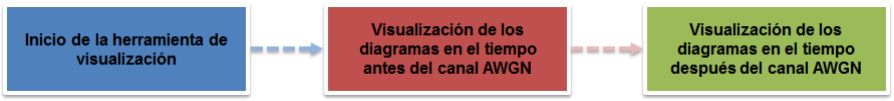


Figura 2.11. Diagrama de bloques de la etapa de visualización.

La etapa de visualización se desarrolla con el objetivo de entablar una comunicación entre el usuario y el equipo, de forma que simplifica su uso al momento de ser ejecutada gracias a la interfaz gráfica y realiza el proceso de recepción de datos provenientes del FPGA a través del puerto RS-232 en un segundo plano.

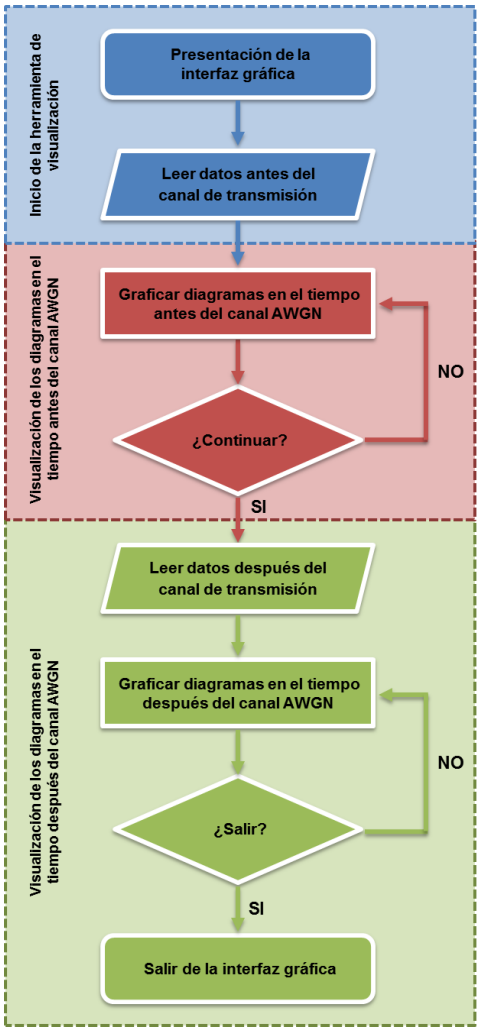


Figura 2.12. Diagrama de flujo de la etapa de visualización.

Mediante un diseño fácil de accionar, el usuario cuenta con botones y ventanas con las cuales puede visualizar los diagramas del tiempo antes y después del canal.

2.2.4. Herramientas de implementación elegidas

- **Herramienta software para el desarrollo de la etapa de adaptación**

La herramienta sobre la cual se desarrolla el diseño de la etapa de adaptación-transmisión es *System Generator*, plataforma software alojada en la herramienta Simulink de MATLAB. Esta herramienta permite crear modelos de algoritmos para su implementación en FPGAs de XILINX, mediante componentes de alto nivel de las librerías de bloques de Xilinx [20].

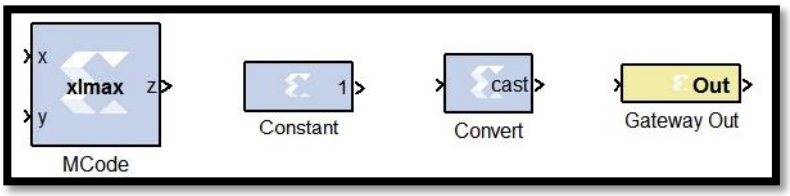


Figura 2.13. Bloques de *System Generator*.

En la Figura 2.13 se observan los bloques relevantes de la librería de XILINX (Xilinx Blockset) para el diseño en, los cuales corresponden a los bloques MCode, Constant, Convert y Gateway Out [21].

- **Herramienta software para el desarrollo de la etapa de visualización**

Se escogió la herramienta MATLAB R2012a y su entorno de desarrollo GUIDE para la creación de la interfaz gráfica, debido a que contiene todos los elementos necesarios tales como botones de opción y de selección, menús y barras de herramientas; además permite combinar programación en código que admite cualquier tipo de operación o acciones como: leer y modificar archivos, brindar comunicación con otras GUIs y desplegar datos como tablas o gráficos [22].

Cada componente de una GUI de MATLAB tiene la posibilidad de ser programada para ejecutar algún comando en particular en el momento que un usuario presione un botón o seleccione algún elemento con el puntero, lo que permite al creador de la interfaz desarrollar comandos que dan respuesta a cada función.

2.3. Implementación

Este apartado presenta la implementación de los módulos de adaptación y visualización sobre las herramientas seleccionadas.

2.3.1. Módulo de adaptación

La adaptación del hardware reconfigurable a la interfaz gráfica se logra mediante la herramienta de diseño *System Generator* de XILINX®, instalada en la plataforma de MATLAB. En la Figura 2.14 se observa el modelo diseñado para el desarrollo de la etapa de adaptación.

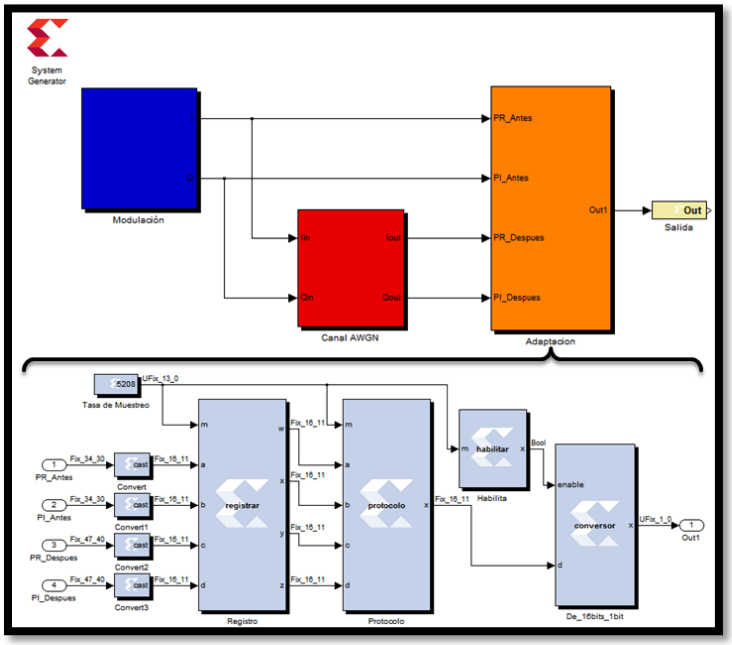


Figura 2.14. Modelo en Simulink con SysGen para la adaptación de los datos.

Al implementar la etapa de adaptación los datos fueron representados y almacenados como datos binarios con formato en punto fijo como se muestra en la Figura 2.15, este formato presenta dos caracteristicas: la longitud de la palabra que indica el numero de bits que se van a usar para almacenar el dato deseado y la longitud de la fraccion que corresponde a la cantidad de digitos destinados como parte fraccionaria del dato almacenado.

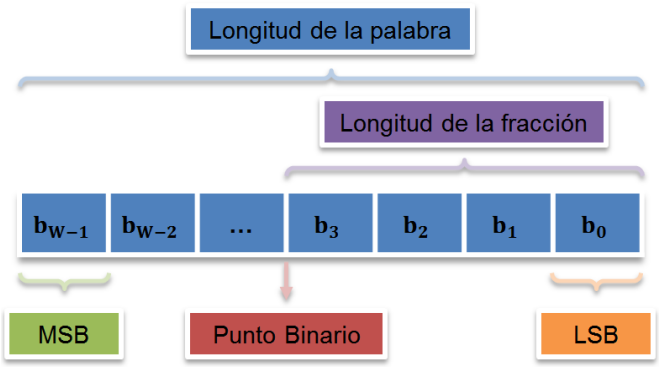


Figura 2.15. Formato de representación en punto fijo.

Es necesario tener en cuenta la notación que se observa en la Figura 2.14, ya que en esta se determina si el dato contiene o no signo (*signed* - *unsigned*), es decir, si existen números negativos. En la Tabla 2.3 se especifican los formatos con los que trabajan los bloques implementados en la etapa de adaptación.

Formato	Fix_16_11	UFix_1_0
Longitud de la palabra	16 bits	1 bit
Longitud de la fracción	11 bits	0 bits
Números negativos	Si	No

Tabla 2.3. Descripción de las notaciones utilizadas.

A continuación se describe la configuración de cada bloque del modelo de la Figura 2.14 implementado en *System Generator*.

- **Bloque "Constant"**

Utilizado para fijar el valor de la tasa de muestreo que determina el correcto funcionamiento del sistema. Para su configuración es necesario establecer el formato en punto fijo como el tipo de salida y asignar el valor de 13 números de bits sin punto binario, teniendo como resultado el valor requerido que corresponde a 5208; por último se fija el periodo de muestreo a 1 indicando que se desea trabajar a la frecuencia interna (50MHz) de la tarjeta SPARTAN-3A.

- **Bloque "Convert"**

Para garantizar la adaptabilidad a los diferentes sistemas encargados de modular las señales, se emplea el bloque convert que busca transformar los datos a un formato específico. En este caso se escogió el formato Fix_16_11 cuya notación es descrita en la Tabla 2.3.

- **Bloque "MCode"**

A cada bloque MCode le corresponde una de las funciones registrar, protocolo, habilitar y conversor detalladas en el Apéndice B, dado que posibilita la opción de cargar archivos .m con el lenguaje de programación de MATLAB previamente diseñados y guardados en el directorio del proyecto. De este modo el bloque ejecuta el archivo y genera los puertos de entrada y salida ya establecidos en la función diseñada.

- **Bloque "Gateway Out"**

Este bloque permite asignar el pin de transmisión del puerto serial RS-232 del FPGA a la salida del sistema. Para el caso de la tarjeta SPARTAN-3A se encuentra localizado en el pin F15.

2.3.2. Módulo de visualización

Para la implementación del módulo de visualización es pertinente la creación de funciones que ejecuten procesos de recepción y despliegue de diagramas, por lo que se procede a dar su explicación.

- **Recuperación de datos a su formato original**

Debido a que los datos transmitidos desde el FPGA se encuentran en formato binario, es necesario detallar la forma como se interpretan para su recepción y visualización; con lo que se hace indispensable la conversión al formato decimal para graficar correctamente los diagramas en el tiempo. Por lo anterior se desarrolló la función de recepción de datos (Apéndice B), en la cual se reciben los datos binarios reales e imaginarios antes y después del canal y posteriormente se modifica el formato mediante la 2.1.

$$\boldsymbol{valor_{dec}} = -b(1) \times 2^{B-L-1} + b(2) \times 2^{B-L-2} + \cdots + b(B) \times 2^{-L} \qquad \textbf{2.1}$$

Donde $b(1)$, $b(2)$, … , $b(B)$, son la secuencia de dígitos binarios, siendo $b(1)$ y $b(B)$ los bits menos y más significativos respectivamente, L la longitud de la fracción y B la longitud de la palabra; una vez terminado este proceso se obtiene el número complejo en el formato apropiado para visualizar los diagramas.

- **Funciones para el despliegue de diagramas**

→ Diagrama de constelación

El diagrama de constelación se obtiene al aplicar la función $scatter$ de MATLAB a la parte real e imaginaria de los datos en cuestión, esta se encarga del despliegue en el eje real e imaginario de los puntos localizados en las posiciones especificadas por el numero complejo, es decir, si se tiene $scatter(x, y)$ se va a graficar la ubicación en el plano del número complejo $x + iy$.

→ Trayectoria de la señal

Este diagrama se logra mediante la función $plot(x, y)$ de MATLAB, que crea un trazo correspondiente al dato y versus el valor de x, en este caso en particular el dato y representa la parte imaginaria y x la parte real de los datos a graficar.

→ Diagrama del ojo

La visualización de este diagrama se realiza a través de la función $myeyediagram$, la cual es una adaptación de la función $eyediagram$ de MATLAB para lograr visualizar los diagramas en los campos dispuestos en la interfaz gráfica. La función $myeyediagram(x, n)$, se encarga de desplegar el diagrama del ojo en fase y cuadratura del dato x, graficando n muestras en cada trazo.

CAPITULO 3

EXPERIMENTACIÓN Y ANALISIS DE RESULTADOS

Culminado el proceso de desarrollo e implementación de la interfaz gráfica, se procede a evaluar su funcionamiento mediante el análisis de los diagramas visualizados y su comportamiento ante los datos provenientes antes y después del canal. Para esto se cuenta con los resultados de simulación obtenidos con las herramientas *Discrete-Time Scatter Plot Scope*[9] [23], *Discrete-Time Signal trajectory Scope*[10] [13] y *Discrete-Time Eye Diagram Scope*[11] [24] de Simulink de MATLAB para compararlos con diagramas resultantes de los proyectos mencionados en [7].

En este capítulo se estudian los diagramas de tiempo de las modulaciones QPSK y FSK, los diagramas de las modulaciones restantes se encuentran en el Apéndice D. En la primera sección se muestran los resultados de los módulos de adaptación y visualización, seguidamente se detalla la forma en que se llevara a cabo la validación de resultados para aclarar la forma en que se hacen las simulaciones.

3.1. Resultados de la herramienta de visualización

3.1.1. Módulo de adaptación

Debido a que la serie 3A de las SPARTAN cuenta con un reloj interno de 50MHz y se ha predeterminado una velocidad de recepción de 9600Baudios, se requiere tener una razón entre estas dos frecuencias para lograr una transmisión acorde al protocolo RS-232. Por ende, resulta el bloque habilitar que se encarga del envió de datos cada 5208 periodos de

[9] Muestra el diagrama de constelación de una señal modulada.
[10] Muestra la trayectoria de la señal modulada.
[11] Muestra el diagrama del ojo en fase y cuadratura de la señal modulada.

tiempo, logrando la sincronización entre la tarjeta SPARTAN-3A y el computador utilizado como receptor.

El número obtenido a partir de las dos frecuencias identificadas se describe con mayor claridad a continuación:
Inicialmente se cuenta con una velocidad de recepción de 9600Baudios, la cual se debe convertir en un flujo de bps para lograr una relación con la frecuencia de trabajo del FPGA, así que se procede a aplicar la 3.1.

$$\boldsymbol{C = S \log_2 M} \quad \textbf{3.1}$$

Donde C equivale al número de bits transmitidos por segundo, S al número de símbolos transmitidos por segundo (9600Baudios) y M al número de estados por símbolo, en este caso se cuenta con dos estados ya que los símbolos solo pueden corresponder a un uno o un cero.

$$\boldsymbol{C} = 9600 \log_2 2$$

$$\boxed{\boldsymbol{C} = 9600\ bps}$$

Ahora, se sabe que la frecuencia de transmisión es de 50MHz, es decir que se estarían transmitiendo 50 millones de bits en un segundo, pero solo se requiere enviar 9600 bits en un segundo, por lo que se dividen estas frecuencias para lograr obtener el número deseado:

$$\boldsymbol{Tasa\ deseada} = \frac{50 \times 10^6}{9600} \quad \textbf{3.2}$$

$$\boldsymbol{Tasa\ deseada} = 5{,}208333 \times 10^3$$

$$\boxed{\boldsymbol{Tasa\ deseada} \approx 5208}$$

Al obtener el resultado se logra una muy buena aproximación a la frecuencia deseada, con esto se consigue el valor que habilita la transmisión.

3.1.2. Módulo de visualización

- **Inicio de la herramienta de visualización**

La ventana principal de la interfaz gráfica es la que se muestra en la Figura 3.1, en esta se encuentra la presentación del proyecto y el botón iniciar, que al ser presionado inicia la lectura de los datos y despliega la ventana de diagramas en el tiempo antes del canal al mismo tiempo que almacena los datos de la primera trama.

Figura 3.1. Presentación de la interfaz gráfica.

- **Implementación de diagramas en el tiempo**

Para la visualización de los diagramas en el tiempo antes y después del canal de transmisión se crearon dos interfaces de usuario, cada una cuenta con 4 espacios dispuestos para graficar el diagrama de constelación, la trayectoria de la señal y el diagrama del ojo en fase y cuadratura, además de esto contienen un botón que ejecuta un proceso.

En la Figura 3.2 se observa la interfaz de usuario que despliega los diagramas en el tiempo antes del canal; en el momento que ocurre la apertura inicial de esta ventana se observan todos los diagramas en el tiempo de la primera trama de datos recibidos antes del canal que han sido almacenados previamente en el despliegue de la ventana de presentación de la interfaz gráfica, al pasar 0,1 segundos la interfaz se refresca y grafica otra trama de datos con el fin de actualizar la información y lograr una aproximación a través de muestras del tiempo real.

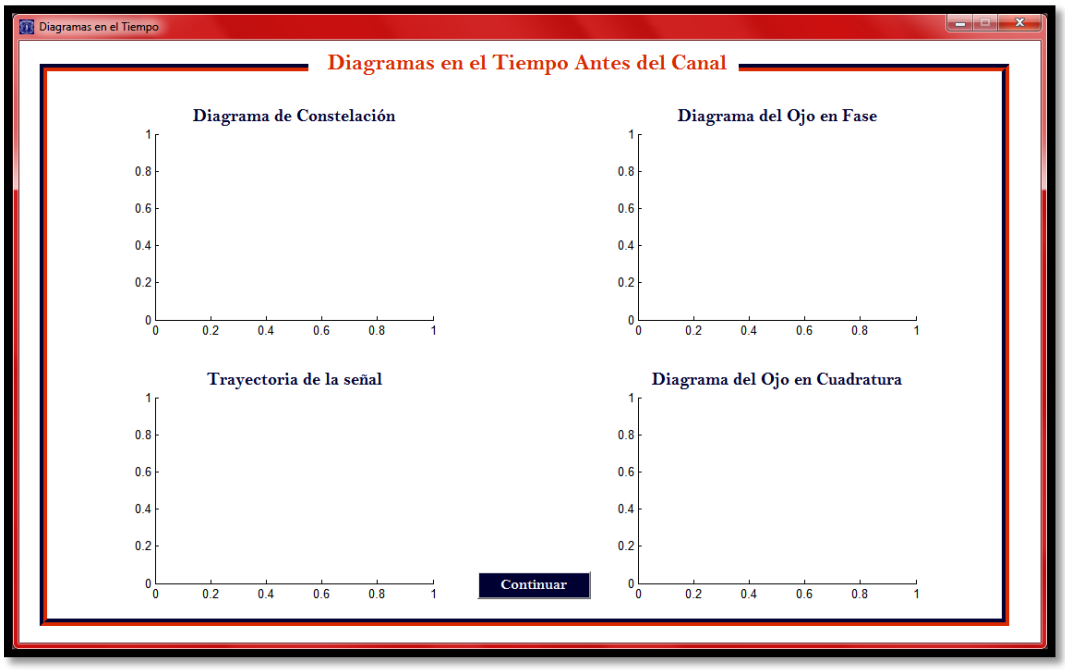


Figura 3.2. Visualización de diagramas en el tiempo antes del canal.

El proceso de la Figura 3.3 es similar al anterior, con la diferencia de que los datos a graficar corresponden a la señal después del canal de transmisión, visualizando diagramas con ruido, el cual es adicionado por el canal AWGN. El botón “Desconectar” finaliza el proceso y cierra la interfaz de usuario dando por concluida la visualización de los diagramas en el tiempo de la señal modulada.

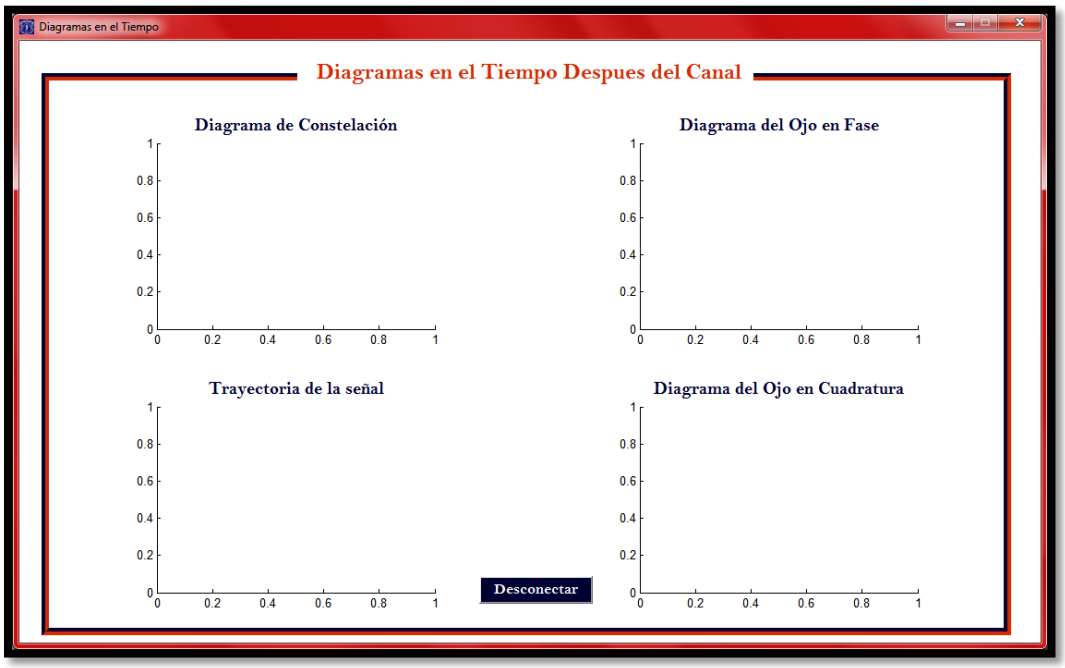


Figura 3.3. Visualización de diagramas en el tiempo después del canal.

3.2. Validación de Resultados

Se llevan a cabo tres simulaciones en Simulink por cada modulador para validar los diagramas en el tiempo que genera la herramienta de visualización: la primera simulación se utiliza como punto de comparación con la gráfica obtenida de los datos modulados (antes del canal), la segunda y tercera simulación ayudan a confrontar los diagramas graficados con los datos modulados y contaminados con ruido (después del canal). A continuación se especifica la cantidad de simulaciones que se llevan a cabo para la validación de resultados.

3.2.1. Antes del canal

Para este caso solo se hace necesaria una simulación ya que no existe alguna variable que afecte o varié los datos que generan el diagrama de constelación, la trayectoria de la señal o el diagrama del ojo antes del canal, esto debido al diseño implementado por los proyectos de [7].

3.2.2. Después del canal

Se efectúan dos simulaciones para el análisis de los datos generados en este tramo del modelo. Gracias a lo desarrollado e implementado por los proyectos de [7] cada canal AWGN se configura con un valor de desviación estándar (sigma, σ), cuyo valor está relacionado directamente con la razón entre la energía de bit y la densidad espectral de potencia de ruido (E_b/N_o), el cual permite adicionar un valor de ruido específico. Definiendo esta variable como parámetro de variación se pueden realizar diferentes simulaciones con distintos valores de sigma que generan diagramas diferentes, manteniendo una relación directamente proporcional entre el valor de E_b/N_o y la calidad de los diagramas, en otras palabras, se logra evidenciar que a mayor valor de E_b/N_o la definición de los diagramas en el tiempo mejora, debido a que la densidad espectral de potencia de ruido es menor.

A continuación se detalla el proceso que se realiza para variar el valor de sigma en cada simulación, teniendo en cuenta que se escogen únicamente dos valores de E_b/N_o para notar las diferencias y validar los resultados, pues se considera suficiente para verificar el funcionamiento de la herramienta de visualización. El valor de E_b/N_o implementado y simulado se encuentra en decibeles.

Inicialmente se cuenta con la 3.3 que indica la relación entre la energía de bit y la densidad espectral de potencia de ruido, donde M representa el número de estados de la modulación elegida.

$$\left(\frac{\boldsymbol{e_b}}{\boldsymbol{n_o}}\right) = \frac{p\ señal}{p\ ruido} \times \frac{1}{log_2(M)} \qquad \textbf{3.3}$$

A su vez la relación señal a ruido está dada por la 3.4.

$$\boldsymbol{snr} = \frac{p\ señal}{p\ ruido} \qquad \textbf{3.4}$$

Al reemplazar la 3.4 en la 3.3, surge la 3.5.

$$\left(\frac{\boldsymbol{e_b}}{\boldsymbol{n_o}}\right) = snr \times \frac{1}{log_2(M)} \qquad \textbf{3.5}$$

Al pasar la 3.5 a decibeles y despejar el valor de la relación señal a ruido resulta una nueva relación, dada por la 3.6.

$$\left(\frac{\boldsymbol{E_b}}{\boldsymbol{N_o}}\right)[\boldsymbol{dB}] = SNR + 10log\left(\frac{1}{log_2(M)}\right)[dB] \qquad \textbf{3.6}$$

$$\boxed{\boldsymbol{SNR}[\boldsymbol{dB}] = \left(\frac{E_b}{N_o}\right)[dB] - 10log\left(\frac{1}{log_2(M)}\right)[dB]} \qquad \textbf{3.7}$$

Debido a que la señal está normalizada en potencia y se utilizan dos fuentes de ruido AWGN según los proyectos de [7], la 3.4 toma la forma de la 3.8.

$$\boldsymbol{snr} = \frac{1}{{\sigma_R}^2 + {\sigma_I}^2} \quad , \quad \sigma_R = \sigma_I \qquad \textbf{3.8}$$

Al despejar, el valor de sigma asociado a un valor de relación señal a ruido encontrado corresponde a la 3.9.

$$\boxed{\boldsymbol{\sigma} = \frac{1}{\sqrt{2 \times snr}}} \qquad \textbf{3.9}$$

Aclarando el origen de las ecuaciones se puede obtener sigma de la siguiente manera: primero se reemplazan los valores de E_b/N_o y M en la 3.7 y luego se pasa el resultado de SNR a unidades lineales (snr) para sustituirlo en la 3.9, obteniendo un valor de sigma que fue utilizado en el modelo diseñado en *System Generator* de XILINX®.

La Tabla 3.1 contiene los valores de sigma que se implementan en el canal AWGN para los valores de E_b/N_o de 14dB y 21dB.

Modulación Digital		$\boldsymbol{M}$	$\boldsymbol{\sigma_1}$	$\boldsymbol{\sigma_2}$
PSK	**BPSK – DBPSK**	2	0,141086351	0,063020958
	QPSK – OQPSK – DQPSK	4	0,099763116	0,044562546
	8PSK	8	0,081456242	0,036385167

Tabla 3.1. Valores de sigma para $(E_b/N_o)_1 = 14dB$ y $(E_b/N_o)_2 = 21dB$.

Debido al número de estados (M) que presentan las modulaciones 16QAM y 64QAM la distancia entre los estados es pequeña lo que aumenta la probabilidad de error, por esto se hace necesario incrementar los valores de E_b/N_o para tener una mejor visualización de los diagramas en el tiempo.

Con respecto a la modulación FSK es claro que el número de estados no es grande según la Tabla 3.2, pero el proyecto encargado del diseño del modulador estableció un parámetro que configura la desviación de frecuencia para modificar el número de estados original. Para la simulación e implementación de este esquema de modulación la desviación de frecuencia se fija a un valor de 1/16, lo que genera 16 estados en el diagrama de constelación, de forma que se tiene un caso similar a la distancia entre los estados de 16QAM y 64QAM.

La Tabla 3.2 contiene los valores de sigma utilizados en el modelo de canal AWGN para los valores de E_b/N_o de 17dB y 24dB.

Modulación Digital		M	σ_1	σ_2
QAM	16QAM	16	0,049940743	0,02230771
	64QAM	64	0,040776446	0,018214169
FSK – MSK		1	0,099881487	0,044615421

Tabla 3.2. Valores de sigma para $(E_b/N_o)_1 = 17dB$ y $(E_b/N_o)_2 = 24dB$.

3.3. Visualizaciones a partir de las Herramientas de Simulink

El montaje general para desplegar los diagramas en el tiempo a través de los bloques suministrados por la biblioteca de Simulink se observa en la Figura 3.4, la cual cuenta con un bloque azul que se refiere a la modulación específica a visualizar, seguido de un bloque rojo que contiene el canal AWGN, finalmente se encuentran los bloques encargados de graficar los diagramas en el tiempo antes y después del canal. Teniendo en cuenta que estas pruebas se realizan con el fin de adquirir resultados de referencia.

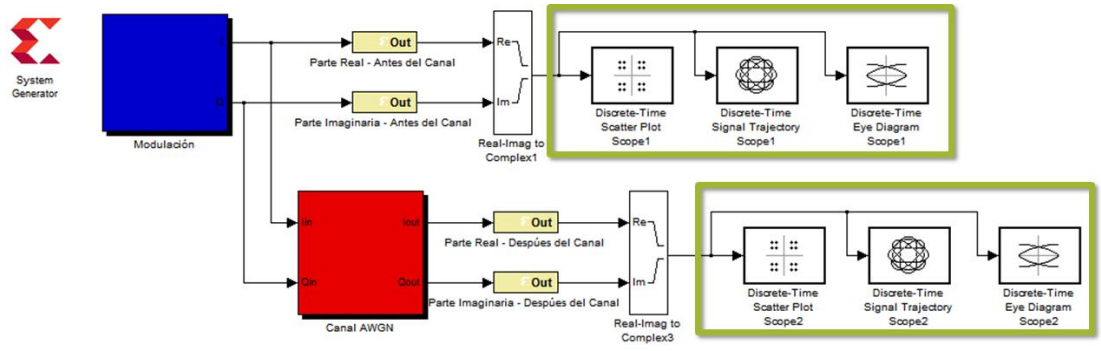


Figura 3.4. Bloques de Simulink para graficar diagramas en el tiempo.

Para cada simulación realizada el bloque “Modulación” y “Canal AWGN” cuenta con el modelo y diseño realizado por cada trabajo de grado desarrollado en el proyecto macro “Diseño e Implementación de un Prototipo de Comunicación de Datos Basado en Hardware Reconfigurable Fase 1”.

A continuación se visualizan los diagramas en el tiempo generados antes y después del canal AWGN de las dos modulaciones seleccionadas, realizando dos variaciones de E_b/N_o para observar los cambios que ocurren en los diagramas después del canal.

3.3.1. Modulación por desplazamiento de fase en cuadratura

- **Antes del canal**

La Figura 3.5 muestra los diagramas en el tiempo generados por los bloques de Simulink para el modulador QPSK diseñado por el proyecto “Análisis del desempeño de un sistema de comunicaciones con modulaciones BPSK, QPSK, 8PSK y OQPSK basado en hardware reconfigurable” del macro proyecto mencionado en [7].

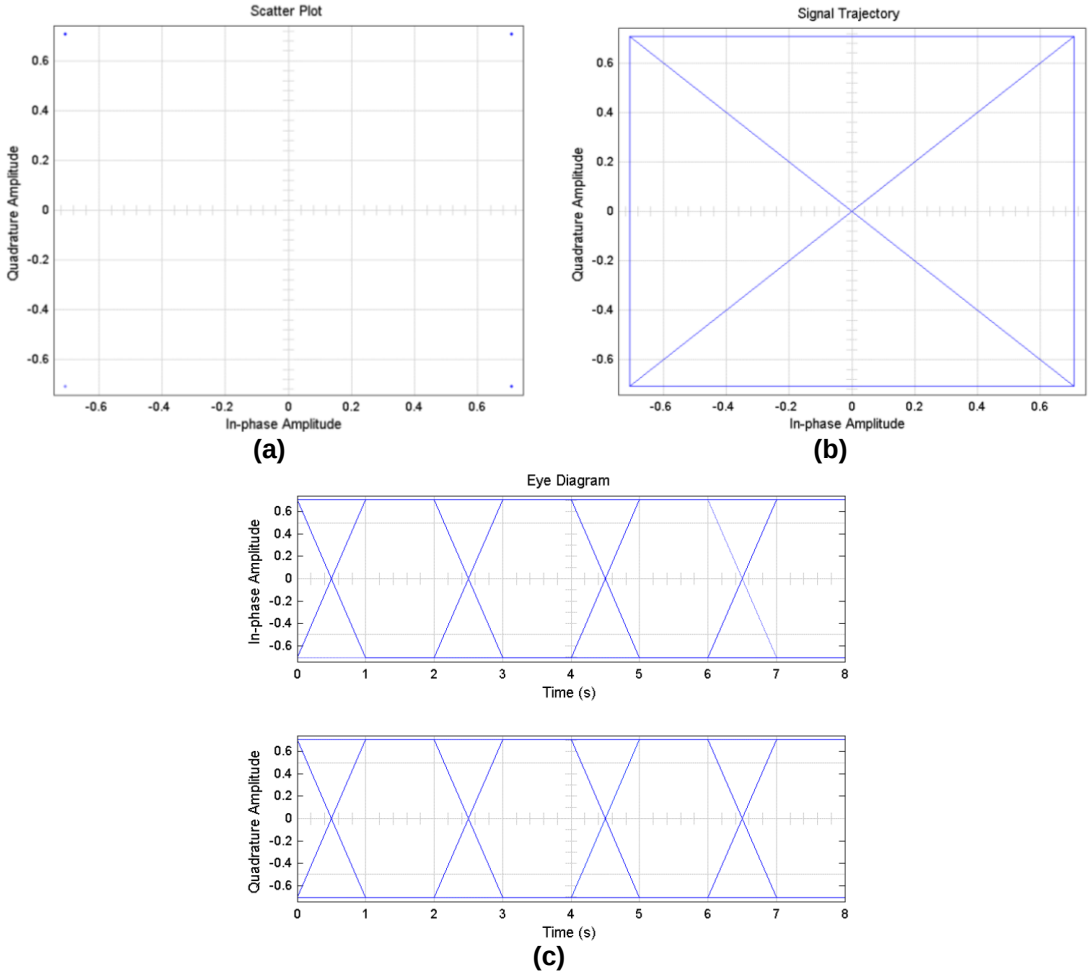


Figura 3.5. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación QPSK.

Validando el diagrama de constelación teórico del modulador QPSK de Simulink [25] con el de la Figura 3.5 (a), se observa la correcta formación del diagrama de constelación al igual que los caminos trazados entre los estados, como se aprecia en la Figura 3.5 (b). Con respecto al diagrama del ojo de la Figura 3.5 (c) se comprueba la teoría de la sección 1.3.3, en donde se superponen los cambios de la señal en un periodo de tiempo, observándose claramente el instante óptimo de muestreo en el eje x y el de decisión en el eje y.

- **Después del canal**

La Figura 3.6 muestra los diagramas en el tiempo generados por Simulink para el modulador QPSK con el parámetro E_b/N_o del canal AWGN configurado a 14dB.

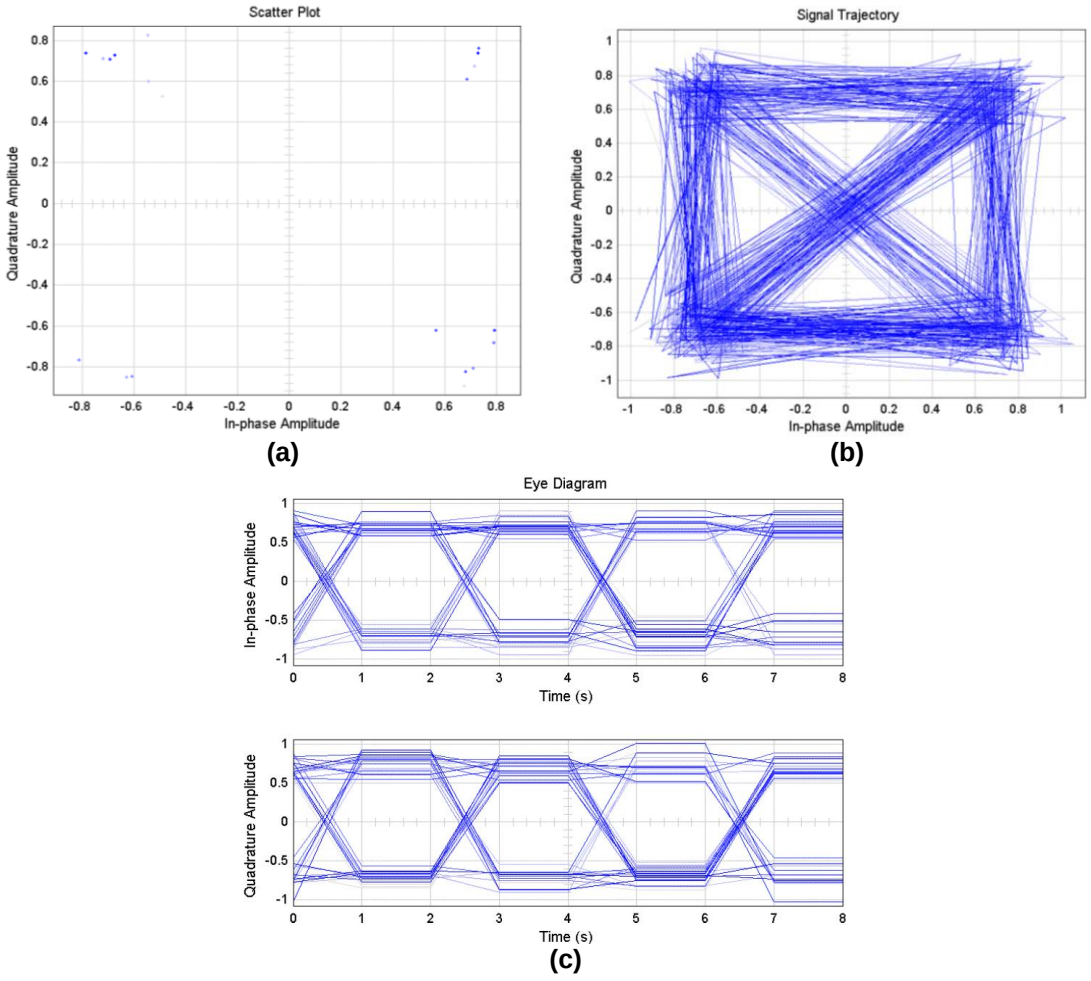


Figura 3.6. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación QPSK con $E_b/N_o = 14dB$.

La Figura 3.7 muestra los diagramas en el tiempo generados por Simulink para el modulador QPSK con el parámetro E_b/N_o del canal AWGN configurado en 21dB.

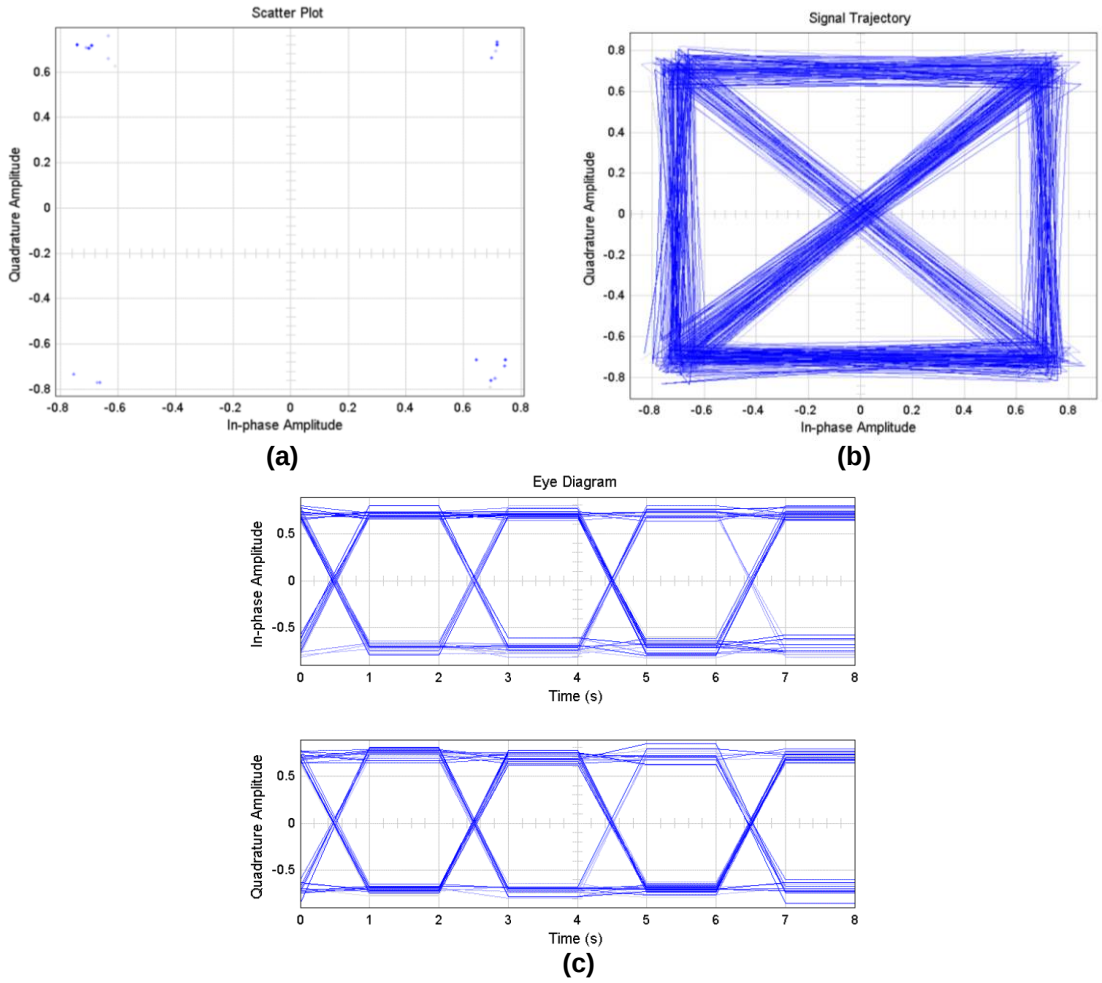


Figura 3.7. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación QPSK con $E_b/N_o = 21dB$.

Al observar los diagramas generados después del canal se evidencia la presencia del ruido que degrada claramente los diagramas. En el caso del diagrama de constelación los puntos se dispersan alrededor del estado inicial, el trazo de la trayectoria de la señal deja de ser una línea entre cambios de estado y el diagrama del ojo presenta continuas variaciones al ser analizado en un tiempo específico. Cada evento que ocurre en los diagramas del tiempo debe ser comprobado al momento de generar los diagramas a través de la interfaz gráfica, puesto que será esto lo que evidencia la validación del diseño de la interfaz gráfica.

3.3.2. Modulación por desplazamiento de fase

- **Antes del canal**

La Figura 3.8 muestra los diagramas en el tiempo generados por los bloques de Simulink para el modulador FSK diseñado por el proyecto “Análisis del desempeño de un sistema

de comunicaciones con modulación FSK/MSK basado en hardware reconfigurable" del macro proyecto mencionado en [7] con una desviación de frecuencia de 1/8.

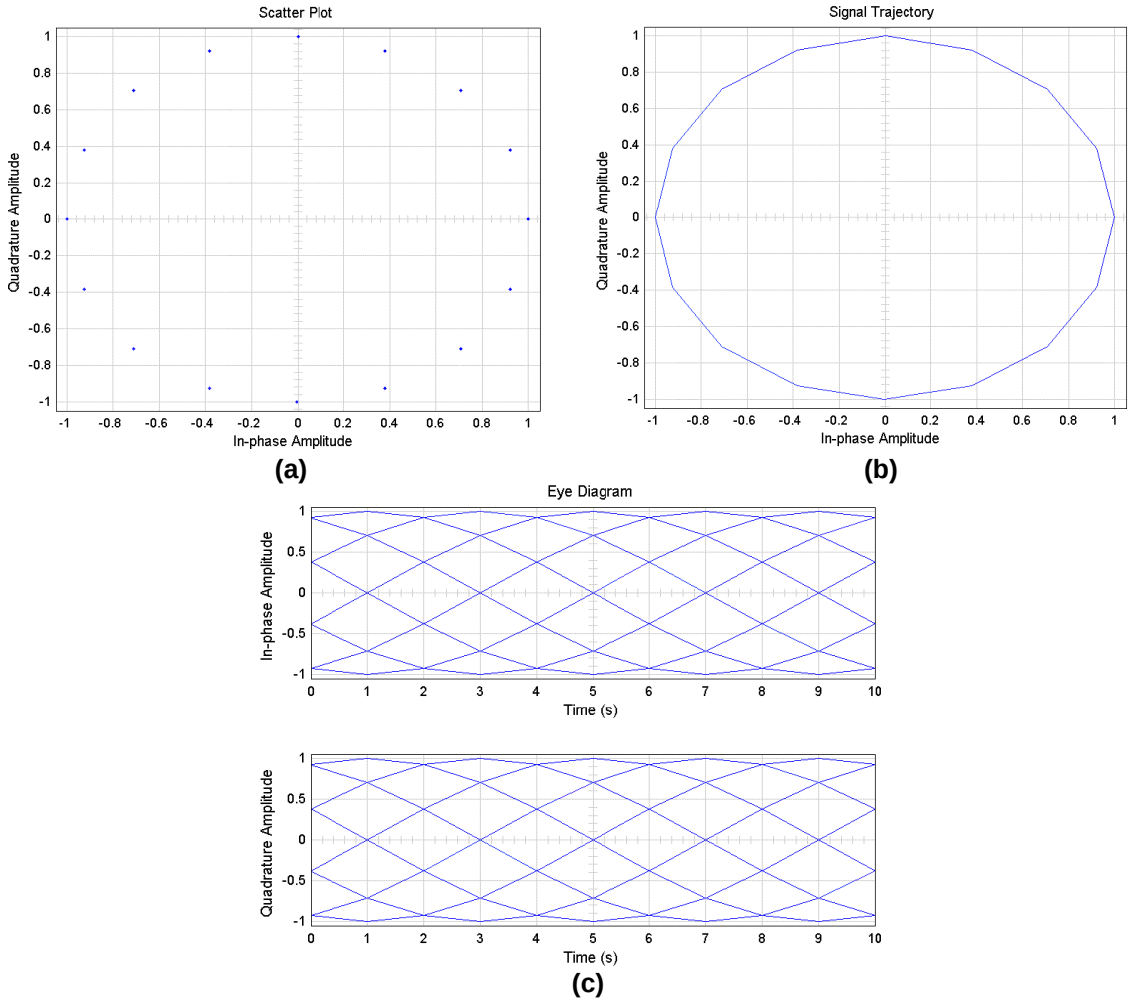


Figura 3.8. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación FSK.

Para comprobar la correcta formación del diagrama de constelación que se muestra en la Figura 3.8 (a) se compara con el teórico generado por el modulador CPFSK de Simulink [26] fijando el parámetro de configuración "desviación de frecuencia" en 1/8.

- **Después del canal**

La Figura 3.9 muestra los diagramas en el tiempo generados por Simulink para el modulador FSK con el parámetro E_b/N_o del canal AWGN configurado a 17dB.

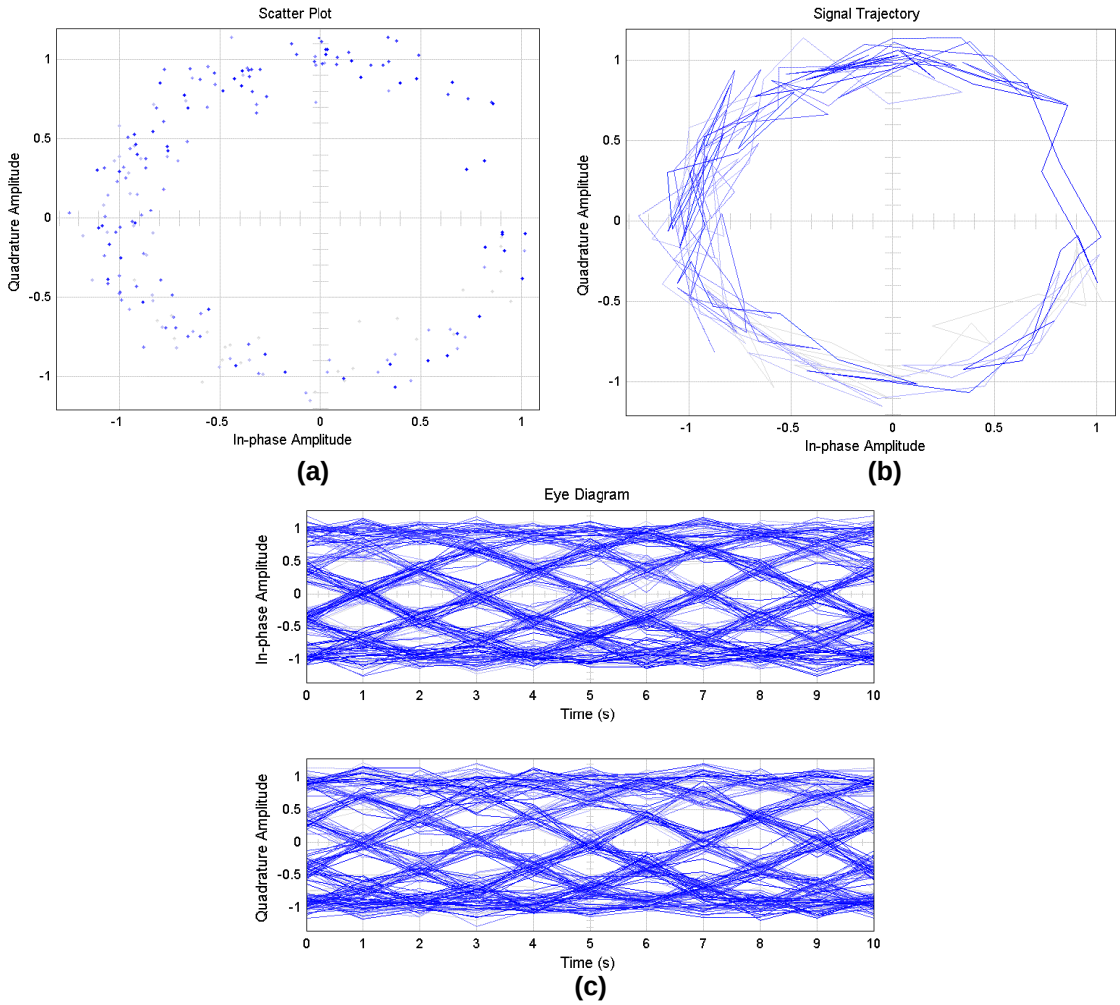


Figura 3.9. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación FSK con $E_b/N_o = 17dB$.

La Figura 3.10 muestra los diagramas en el tiempo generados por Simulink para el modulador FSK con el parámetro E_b/N_o del canal AWGN fijando en 24dB.

Una vez visualizados los diagramas en el tiempo después del canal para la modulación FSK, al igual que con la modulación QPSK se comprueba que los diagramas no mantienen su forma original debido a la presencia de ruido y se ven más afectados con el menor valor de E_b/N_o simulado.

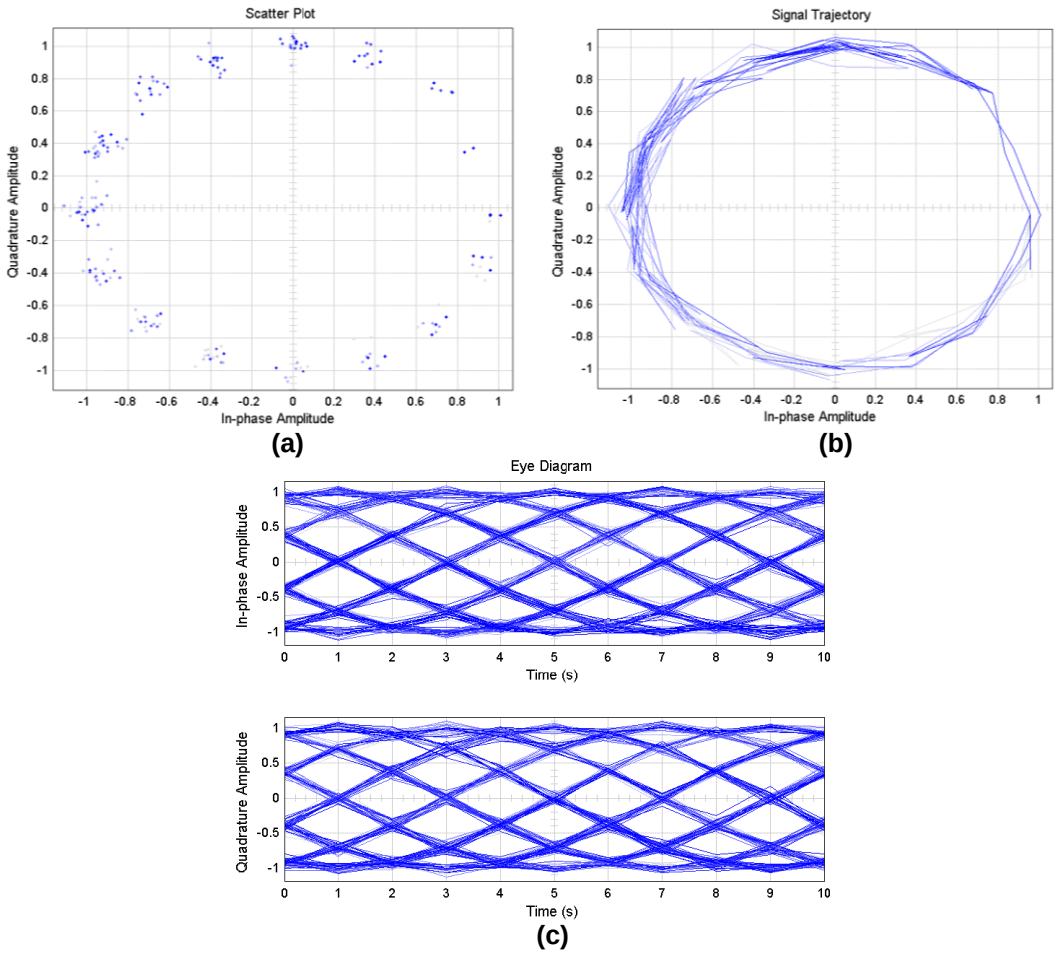


Figura 3.10. (a) Diagrama de constelación. **(b)** Trayectoria de la señal. **(c)** Diagrama del ojo. Modulación FSK con $E_b/N_o = 24dB$.

3.4. Visualizaciones a partir de la Interfaz Gráfica diseñada

A partir de la implementación realizada para visualizar los diagramas en el tiempo a través de la interfaz gráfica se puede resaltar en la Figura 3.11 el diagrama de bloques que sintetiza este proceso, donde las interfaces son las encargadas de desplegar los diagramas antes y después del canal mediante las funciones descritas en la sección 2.3.2.

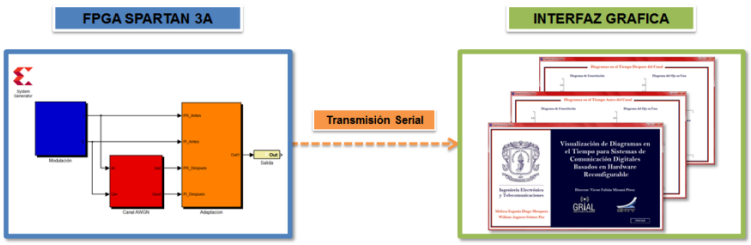


Figura 3.11. Diagrama de bloques para la visualización de diagramas en el tiempo a través de la interfaz gráfica.

Las figuras de los apartados 3.3.1 y 3.3.2 se toman como punto de comparación y verificación para el análisis de los diagramas en el tiempo, se observa que los diagramas generados después del canal se degradan por el efecto del ruido, pero entre los dos valores de E_b/N_o simulados la respuesta es mejor ante un valor mayor de este parámetro, efecto que debe ser visualizado en los diagramas generados por la interfaz.

3.4.1. Modulación por desplazamiento de fase en cuadratura

- **Antes del canal**

La Figura 3.12 muestra los diagramas en el tiempo generados por la interfaz gráfica para el modulador QPSK. Al comparar el diagrama de constelación y la trayectoria de la señal con los de la Figura 3.5 (a) y (b) se denota igualdad, observando los cuatro estados que representan a la modulación QPSK.

Con respecto al diagrama del ojo en fase y cuadratura, son semejantes a los de la Figura 3.5 (c) puesto que conservan la forma de la señal generada, a pesar de ser graficados en periodos de tiempos diferentes, se evidencian los dos posibles valores que puede tomar la señal modulada tanto en el eje real como en el eje imaginario.

Al comparar la Figura 3.12 con la Figura 3.5 se verifica la correcta generación de los diagramas, comprobando la funcionalidad de la herramienta implementada.

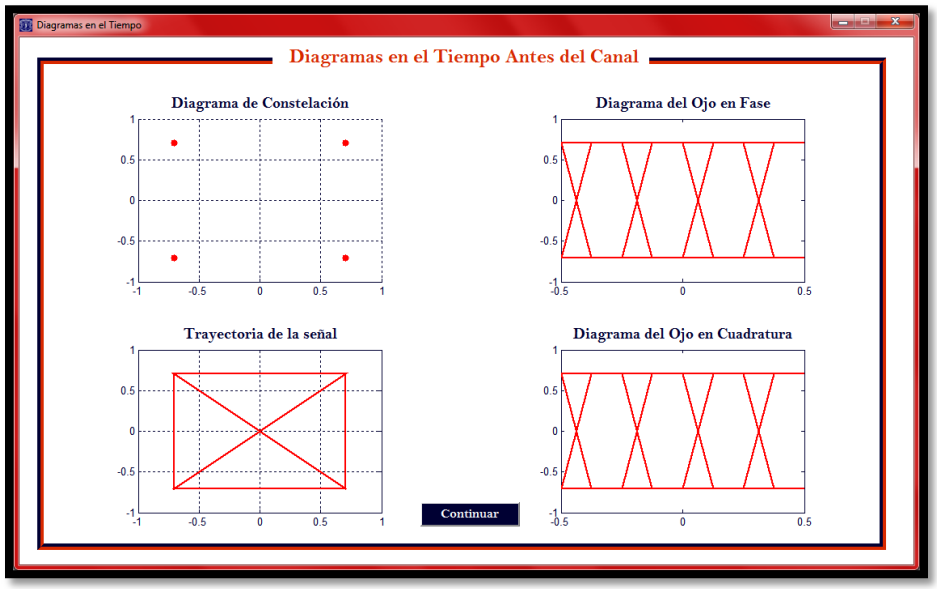


Figura 3.12. Diagramas en el tiempo generados por la interfaz gráfica para QPSK.

- **Después del canal**

Para visualizar los cambios que pueden ocurrir después del canal debido al ruido blanco gaussiano aditivo (AWGN) se realizan dos variaciones de E_b/N_o. La Figura 3.13 muestra

los diagramas en el tiempo generados por la interfaz para la modulación QPSK con un E_b/N_o de 14dB y la Figura 3.14 muestra los diagramas en el tiempo generados por la interfaz para la modulación QPSK con un E_b/N_o de 21dB.

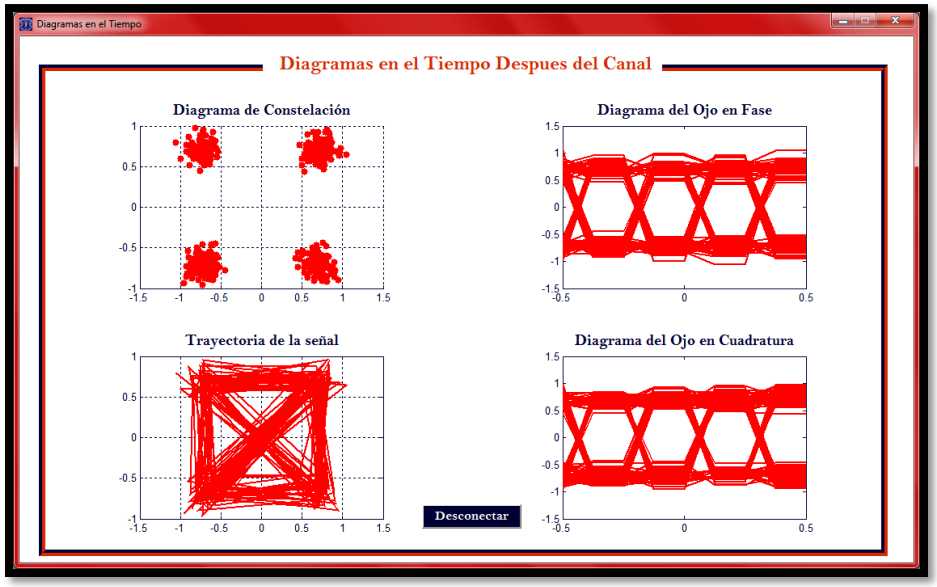


Figura 3.13. Diagramas en el tiempo generados por la interfaz gráfica para QPSK con $E_b/N_o = 14dB$.

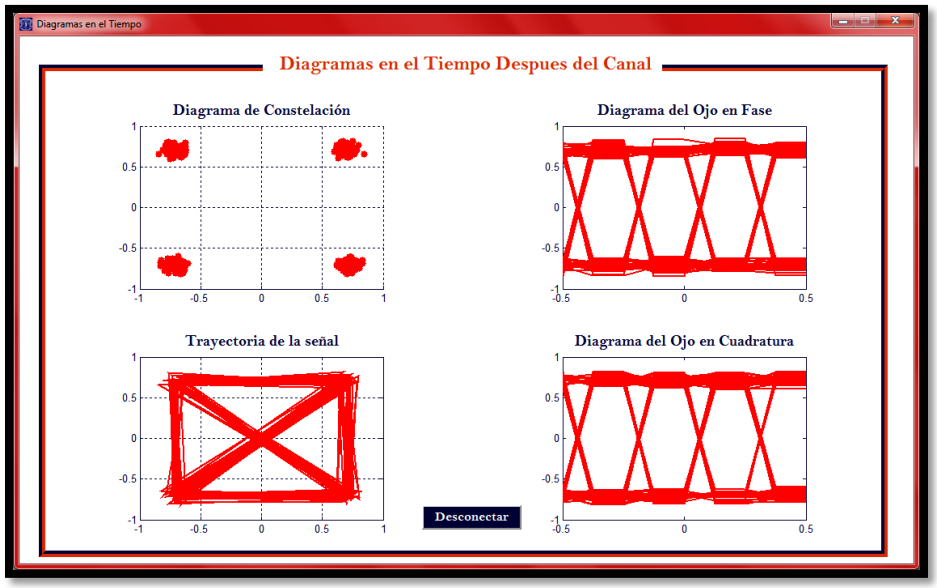


Figura 3.14. Diagramas en el tiempo generados por la interfaz gráfica para QPSK con $E_b/N_o = 21dB$.

Al obtener las gráficas después del canal con las variaciones en el valor de E_b/N_o , se demuestra que a través de la interfaz gráfica se puede ver como a un mayor valor de E_b/N_o las gráficas presentan mayor concentración sobre los puntos generados antes del

canal. Verificando el correcto funcionamiento de la herramienta de visualización diseñada con la modulación QPSK.

Comparando la Figura 3.6 con la Figura 3.13 y la Figura 3.7 con la Figura 3.14 se comprueba la similitud de los diagramas en presencia de ruido, con lo cual se deduce que el desempeño de la herramienta de visualización es óptimo.

3.4.2. Modulación por desplazamiento de fase

- **Antes del canal**

En la Figura 3.15 se cuenta con los diagramas en el tiempo generados para la modulación FSK. Tal y como se espera el diagrama de constelación de la Figura 3.15 equivale al de la Figura 3.8 (a), puesto que se observan todos los posibles símbolos que puede tener el diagrama con la configuración especifica del modulador diseñado. También se verifica la trayectoria que sigue la señal modulada, guardando total similitud con la Figura 3.8 (b). Finalmente, el diagrama del ojo en fase y cuadratura conserva los límites de amplitud y la forma de la señal que se aprecia en la Figura 3.8 (c).

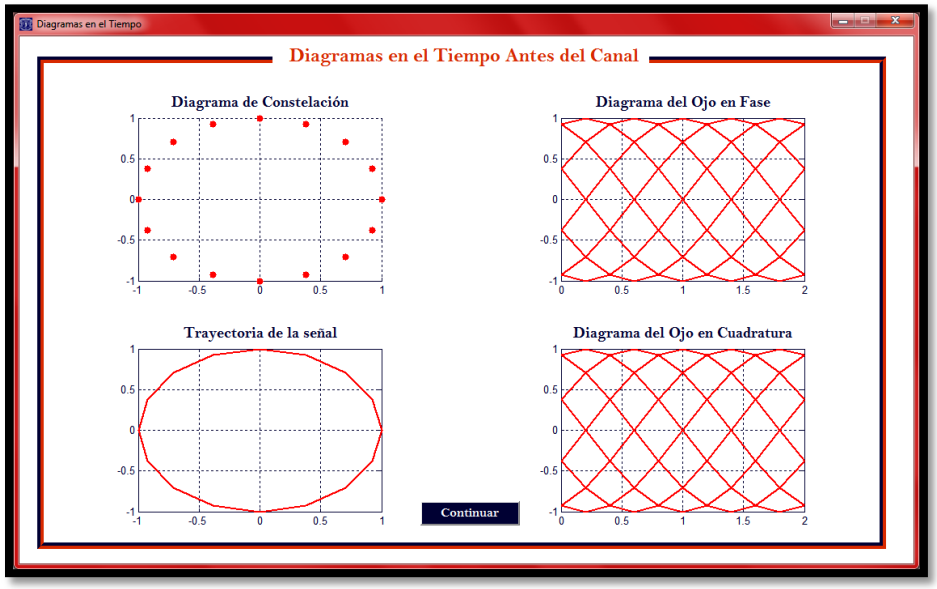


Figura 3.15. Diagramas en el tiempo generados por la interfaz gráfica para FSK.

- **Después del canal**

Se realiza una variación del valor de E_b/N_o, con el fin de evidenciar cambios en los diagramas, al ser analizados después del canal AWGN. La Figura 3.16 contiene los diagramas en el tiempo generados para el modulador FSK con el parámetro E_b/N_o fijado en 17dB.

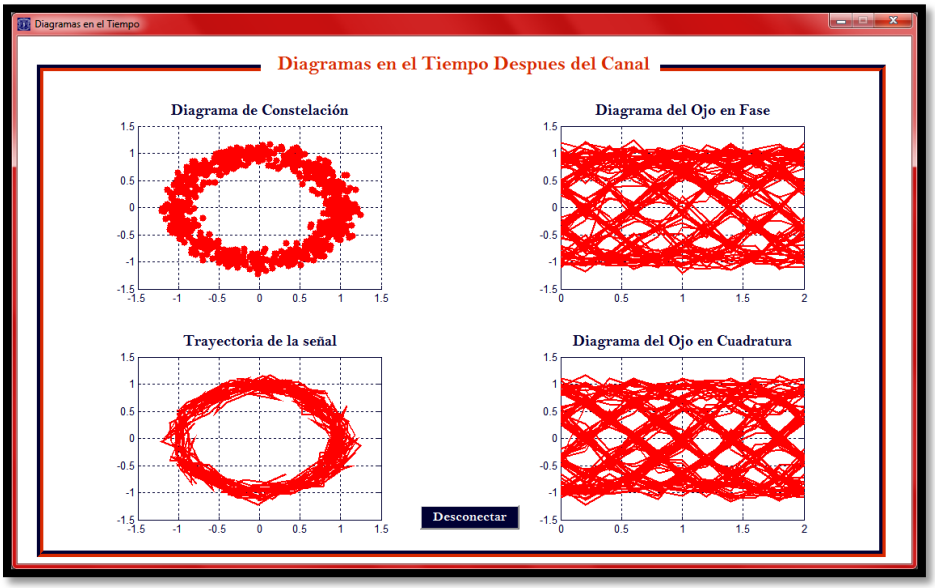


Figura 3.16. Diagramas en el tiempo generados por la interfaz gráfica para FSK con $E_b/N_o = 17dB$.

La Figura 3.17 muestra los diagramas en el tiempo generados para el modulador FSK para un valor de E_b/N_o fijado en 24dB.

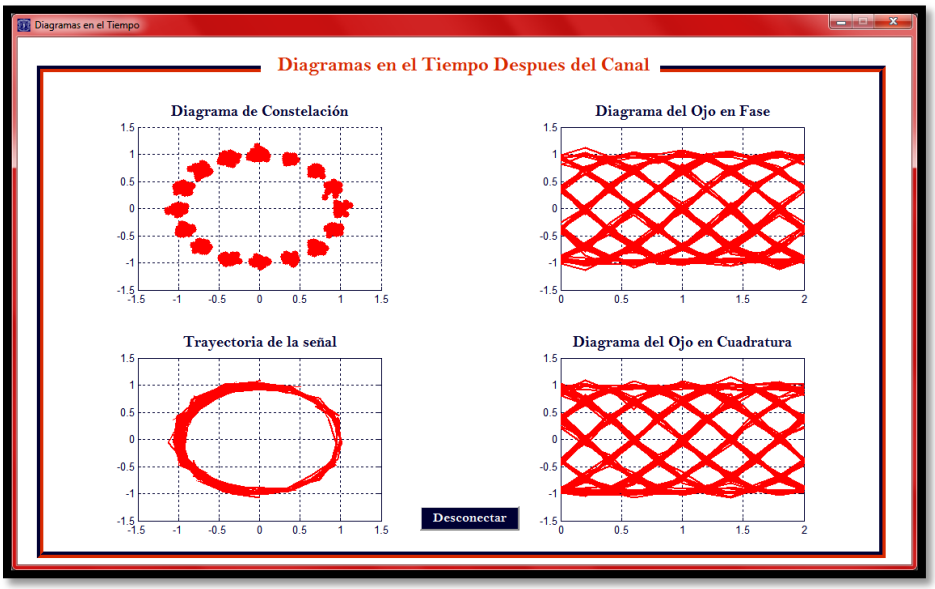


Figura 3.17. Diagramas en el tiempo generados por la interfaz gráfica para FSK con $E_b/N_o = 24dB$.

Al igual que en las simulaciones realizadas para el modulador QPSK se comprueba que con la modulación FSK también se observa una concentración alrededor de la gráfica original para un valor mayor de E_b/N_o. Evento que teóricamente se encuentra comprobado y verificado a través de las Figura 3.9 y Figura 3.10, debido a que la energía de la señal es mucho más grande respecto a la energía del ruido.

Mediante la visualización se puede construir un análisis y evaluar el comportamiento del sistema diseñado, por ejemplo, en el caso de los diagramas de ojo obtenidos es interesante observar que en ausencia de ruido o interferencia entre símbolos, todos los trazos pasan por los mismos puntos en los tiempos que corresponden a los niveles utilizados. En este caso para las dos simulaciones los valores de los niveles se encuentran entre ±1, de manera que al muestrear en estos instantes se obtendría el símbolo enviado en dicho momento.

La apertura vertical del ojo es una buena medida de la inmunidad al ruido, puesto que da una idea de qué amplitud de ruido sería necesaria para producir un error. En cuanto a la apertura horizontal, da una medida de la inmunidad a los errores en el instante elegido para muestrear la señal; en efecto, a medida que se desvía del instante óptimo de muestreo (en donde la apertura vertical es máxima) se reduce el margen necesario para que el ruido produzca errores, hasta el punto de que si se muestrea en instantes situados fuera del ojo, la probabilidad de error ya no será cero, incluso en ausencia de ruido [27].

Los diagramas en el tiempo de las señales generadas por los moduladores y canales AWGN restantes son validadas de igual forma, y se comprueba al visualizar las similitudes e igualdades entre los bloques de Simulink y la herramienta software encargada de generar el diagrama de constelación, la trayectoria de la señal y el diagrama del ojo para datos modulados y contaminados con ruido. En el Apéndice D se encuentra cada gráfica obtenida.

Debido a limitaciones en los recursos por parte de la tarjeta SPARTAN-3A, no es posible generar una interfaz gráfica que despliegue las múltiples modulaciones, por ende, se debe trabajar con cada una por separado, es decir, se requiere programar la tarjeta con la modulación a analizar y posteriormente programar otra si así se desea.

CAPITULO 4

CONCLUSIONES, RECOMENDACIONES Y TRABAJOS FUTUROS

4.1. Conclusiones

La etapa de adaptación diseñada e implementada en la tarjeta SPARTAN-3A permite la transmisión de los datos modulados antes y después del canal AWGN de una manera ordenada y secuencial siguiendo un protocolo de transmisión delimitado por banderas establecidas en el código incorporado. Haciendo del sistema un conjunto flexible, que permite la adaptación de cualquier formato de datos, presentando como único requerimiento la existencia de dos números complejos representados por su componente real e imaginaria.

Debido a las dificultades que se presentaron al momento de recibir datos en tiempo real a causa de las limitaciones en los recursos de la tarjeta, fue necesario realizar una aproximación a través de muestras del tiempo real, mediante actualizaciones de datos que generan una adecuada formación de cada diagrama permitiendo una correcta visualización de estos.

La interfaz gráfica presenta una fluida visualización de cada uno de los diagramas en el tiempo, lo cual permite al usuario realizar un análisis cualitativo acerca de la tendencia, fidelidad y calidad de la señal.

Al analizar y visualizar los resultados de las simulaciones realizadas con las modulaciones BPSK, 8PSK, QPSK, OQPSK, DBPSK, DQPSK, FSK, MSK, 16QAM y 64QAM se concluye que los diagramas en el tiempo son satisfactorios ya que los diagramas en el tiempo desplegados por los bloques de Simulink son similares a los obtenidos por la interfaz gráfica.

Se resalta la importancia que implica desarrollar una herramienta de visualización software capaz de adaptar un entorno hardware, pues a través de esta se proporciona un entorno visual sencillo que logra la interacción entre la tarjeta SPARTAN-3A y el computador mediante el cable convertidor de serial a USB, posibilitando la visualización del diagrama de constelación, la trayectoria de la señal y el diagrama del ojo en fase y cuadratura de los datos modulados antes y después del canal AWGN.

4.2. Recomendaciones

Identificar el tipo de formato de datos que se requiere para desarrollar el modelo en *System* Generator de XILINX® y comprobar que se mantenga a medida que se incrementa la lógica del modelo, de lo contrario el sistema puede presentar inconsistencias en el alcance de los valores requeridos, dejando inconcluso el diseño establecido.

Para llevar a cabo una transmisión mediante el puerto serial RS-232, verificar previamente el formato de los datos de salida antes de asignar puertos en la tarjeta que se va a implementar el diseño, para determinar si se encuentran serializados ya que este estándar únicamente asigna un puerto para transmitir un bit a la vez.

Identificar la frecuencia de trabajo de la tarjeta que se esté utilizando y la frecuencia a la que se desea recibir para lograr la sincronización entre la transmisión y recepción de datos, puesto que a partir de la frecuencia del FPGA se va a obtener la frecuencia de recepción.

Durante el diseño de la lógica a implementar, identificar todas las variables dependientes e independientes para definir el código en función de variables globales, lo cual permite alcanzar la flexibilidad y adaptabilidad que el sistema requiere.

Utilizar el bloque “Resource Estimator” de la librería de Xilinx de Simulink que permite advertir al diseñador y dar una aproximación del área ocupada en el FPGA, consiguiendo visualizar los límites de los recursos y determinar si el sistema presenta excesos de estos.

Para las pruebas previas a la recepción final se recomienda hacer uso de programas externos a la interfaz gráfica, ya que mediante la visualización de los datos se puede comprobar si estos corresponden a los transmitidos. Programas como Hyperterminal, Advanced Serial Port Monitor, HyperACCESS, entre otros brindan la posibilidad de visualizar los datos en diferentes formatos.

4.3. Trabajos Futuros

Se plantea como mejora y trabajo futuro lograr la visualización de la curva de BER Vs E_b/N_o comparando con una teórica que este por defecto en la interfaz gráfica.

Almacenar los datos de todas las modulaciones para lograr la visualización conjunta.

Lograr la transmisión de datos en paralelo a través de microcontroladores permitiendo la recepción y visualización de datos en tiempo real.

REFERENCIAS

[1] J. Rondon y C. Sandoval, «Diseño de un co-laboratorio remoto basado en programación modular de dispositivos VHDL aplicado a telecomunicaciones,» *Revista de la Facultad de Ingeniería Universidad Central de Venezuela,* vol. 25, nº 2, pp. 7-12, 2010.

[2] J. B. Anderson y S. Mohan, «Coding,» de *Source and Channel Coding: An Algorithmic Approach*, Springer, 1991, pp. 8 - 10.

[3] G. Maral y M. Bousquet, Satellite Communications Systems: Systems, Techniques and Technology, Wiley, 2010.

[4] A. R. Castro Lechtaler y R. J. Fusario, «Definición de Multiplexación,» de *Teleinformatica Para Ingenieros en Sistemas de Informacion*, Barcelona, Reverté, S.A, 1999, pp. 226 - 228.

[5] M. Faúndez Zanuy, «La Modulación,» de *Sistemas de Comunicaciones*, Barcelona, Marcombo, 2001, pp. 20 - 23.

[6] M. D. Lutovac, D. V. Tošić y B. Lawrence Evans, «Introduction to Analog Filters,» de *Filter Design for Signal Processing Using MATLAB and Mathematica*, Miroslav Lutovac, 2001, pp. 139 - 142.

[7] V. Quintero y P. Jojoa, «Diseño e implementación de un prototipo de comunicación de datos basado en hardware reconfigurable Fase 1,» *Universidad del Cauca, Popayán,* 2013.

[8] J. Luque Rodríguez y S. Clavijo Suero, «Modulación de Señales Digitales,» Universidad de Sevilla, Departamento de Tecnología Electrónica, Sevilla, 1995.

[9] K. V. Prasad, Principles of Digital Communication Systems and Computer Networks, Charles River Media, 2003.

[10] A. M. Wyglinski, Digital Communication Systems Engineering with Software-Defined Radio, Artech House, 2013.

[11] J. Chitode, «Minimum Shift Keying (MSK),» de *Digital Communications*, Technical Publications Pune, 2002, pp. 66 - 67.

[12] H. Packard, «Digital Modulation in Communications Systems - An Introduction,» Hewlett-Packard Company, U.S.A, 1997.

[13] I. The MathWorks, «Discrete-Time Signal Trajectory Scope,» 1994. [En línea]. Available: http://www.mathworks.com/help/comm/ref/discretetimesignaltrajectoryscope.html. [Último acceso: 04 Febrero 2014].

[14] A. Bateman, Comunicaciones Digitales: diseño para el mundo real, Marcombo, S.A., 2003.

[15] XILINX, «XILINX, All Porgrammable,» [En línea]. Available: http://www.xilinx.com/training/fpga/fpga-field-programmable-gate-array.htm. [Último acceso: 10 Febrero 2014].

[16] XILINX, «Spartan-3A FPGA Family,» 19 Agosto, 2010.

[17] K. Beck, Extreme Programming Explained: Embrace Change, Addison - Wesley, 1999.

[18] J. C. Cortizo Pérez, D. Expósito Gil y M. Ruiz Leyva, «¿Cuándo y dónde usar la XP?,» de *eXtreme Programming*, 2002, pp. 12 - 14.

[19] J. Balcells, J. Autonell, V. Barra, B. Joan, F. Fornieles, B. García y J. Ros, «Enlaces físicos,» de *Eficiencia en el Uso de la Enregía Eléctrica*, Marcombo, 2011, pp. 222 - 225.

[20] XILINX, «XILINX, All Programmable,» [En línea]. Available: http://www.xilinx.com/tools/sysgen.htm. [Último acceso: 10 Febrero 2014].

[21] J. A. Muñoz Hidalgo y J. C. Zemanate Zuñiga, «Análisis del Desempeño de un Sistema de Comunicaciones con Modulación 16/64 QAM Basado en Hardware Reconfigurable,» *Universidad del Cauca, Popayán,* p. Anexos, 2014.

[22] I. The MathWorks, «GUI de MATLAB,» 1994, 1994. [En línea]. Available: http://www.mathworks.com/discovery/matlab-gui.html. [Último acceso: 10 Febrero 2014].

[23] I. The MathWorks, «Discrete-Time Scatter Plot Scope,» 1994. [En línea]. Available: http://www.mathworks.es/es/help/comm/ref/discretetimescatterplotscope.html. [Último acceso: 19 Marzo 2013].

[24] I. The MathWorks, «Discrete-Time Eye Diagram Scope,» 1994. [En línea]. Available: http://www.mathworks.es/es/help/comm/ref/discretetimeeyediagramscope.html. [Último acceso: 19 Marzo 2013].

[25] I. The MathWorks, «QPSK Modulator Baseband,» 1994. [En línea]. Available: http://www.mathworks.es/es/help/comm/ref/qpskmodulatorbaseband.html. [Último acceso: 21 Marzo 2014].

[26] I. The MathWorks, «CPFSK Modulator Baseband,» 1994. [En línea]. Available: http://www.mathworks.es/es/help/comm/ref/cpfskmodulatorbaseband.html. [Último acceso: 21 Marzo 2014].

[27] A. Artés Rodríguez, F. Pérez González, J. Cid Sueiro, R. López Valcarce, C. Mosquera Nartallo y F. Pérez Cruz, «Comunicaciones Digitales,» DERECHOS RESERVADOS Los autores, 2012. [En línea]. Available: http://www.tsc.uc3m.es/~antonio/libro_comunicaciones/El_libro_files/comdig_artes_perez.pdf. [Último acceso: 26 Marzo 2014].

APENDICE A

FUNCIONES PARA LA ADAPTACION DE LOS DATOS EN EL FPGA

La lógica necesaria para la adaptación de los datos en el FPGA provenientes de cada modulación realizada por los proyectos de [7] se detalla a continuación, resaltando que los bloques de color amarillo, violeta, azul y café de la Figura A.1 se implementan a través de bloques “mcode” de la librería de Xilinx en Simulink.

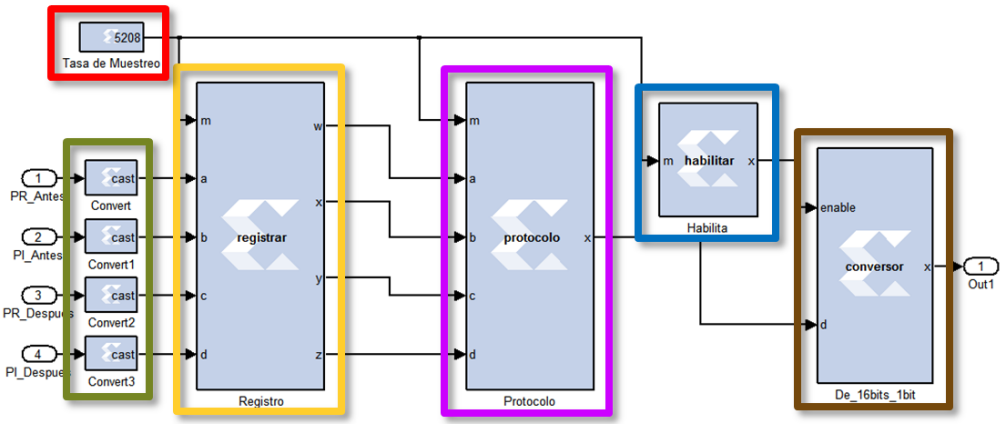


Figura A.1. Diagrama de bloques para la adaptación de los datos.

A.1. Tasa de Muestreo

En la Figura A.1 se observa esta etapa en el bloque enmarcado de color rojo, este se encarga de definir la tasa de muestreo y la frecuencia de salida que tendrá toda la adaptación y serialización de los datos, que para este caso será de 5208 como se explicó en el apartado 3.1.1 (bloque “Habilitar”). Es por esto que para cumplir con esta función el bloque corresponde a un bloque “constant” de la librería de Xilinx en Simulink.

A.2. Conversión a Fix_16_11

Esta etapa se resalta en verde en la Figura A.1, se implementa con la finalidad de dar un único formato a todos los datos provenientes de etapas anteriores a través del bloque "convert" de la librería de Xilinx en Simulink, especificando que los datos después de este punto se encuentran en formato Fix_16_11, es decir, datos con signo, con una longitud de palabra de 16 bits con punto fijo en el bit 11.

A.3. Registro

```
%----------------------------------------
%Universidad Del Cauca
%Nombre:
%         registrar
%Archivo:
%         registrar.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%----------------------------------------

function [w,x,y,z] = registrar(m,a,b,c,d)

persistent v, v = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v1, v1 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v2, v2 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v3, v3 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent cont, cont = xl_state(0, {xlUnsigned,30,0});

cont = cont+1;
if cont <= 501
    w = v.front;
    v.push_front_pop_back(a);
    x = v1.front;
    v1.push_front_pop_back(b);
    y = v2.front;
    v2.push_front_pop_back(c);
    z = v3.front;
    v3.push_front_pop_back(d);
else
    w = 30;
    x = 30;
    y = 30;
    z = 30;
    if cont == ((40060*m)+2504)
        cont = 0;
    end
end
%------------------------- Descripción -----------------------------
%Esta función encarga del almacenamiento de los primeros 500 datos de
%los números complejos antes y después del canal, por lo que se requieren
%4 vectores, 2 para el primer número complejo (parte real e imaginaria
%antes del canal) y dos para el siguiente (parte real e imaginaria
%después del canal). Función que se reinicia con un periodo de
```

```
%(40060*m)+2504 necesario para el reenvió de datos. Cuando el vector ya
%ha almacenado los 500 datos necesarios se inicia a llenarlo con datos
%que indican el fin del almacenamiento de datos (30).
%-------------------------------------------------------------------------
```

A.4. Protocolo

```
%---------------------------------------
%Universidad Del Cauca
%Nombre:
%         protocolo
%Archivo:
%         protocolo.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%---------------------------------------

function x = protocolo(m,a,b,c,d)

inicio = 0;
bandera_1 = 1;
p_real_a = 2;
p_imag_a = 3;
bandera_2 = 4;
p_real_d = 5;
p_imag_d = 6;
bandera_3 = 7;
fin = 8;

persistent state, state = xl_state(inicio, {xlUnsigned, 4, 0});
persistent v, v = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v1, v1 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v2, v2 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent v3, v3 = xl_state(zeros(1,501), {xlSigned,16,11});
persistent n, n = xl_state(1, {xlUnsigned,9,0});
persistent n1, n1 = xl_state(1, {xlUnsigned,9,0});
persistent n2, n2 = xl_state(1, {xlUnsigned,9,0});
persistent n3, n3 = xl_state(1, {xlUnsigned,9,0});
persistent cont1, cont1 = xl_state(0, {xlUnsigned,30,0});
persistent t1, t1 = xl_state(0, {xlUnsigned,9,0});

x = 0;
    switch state
        case inicio
            x = 30;
            if t1 < 500
                state = inicio;
                t1 = t1+1;
            else
                state = bandera_1;
            end
        case bandera_1
            state = p_real_a;
            x = 11;
        case p_real_a
```

```
        state = p_imag_a;
        x = v(n);
        n = n+1;
    case p_imag_a
        if n1 == 500
            state = bandera_2;
            x = v1(n1);
        else
           state = p_real_a;
            x = v1(n1);
            n1 = n1+1;
        end
    case bandera_2
        state = p_real_d;
        x = 12;
    case p_real_d
        state = p_imag_d;
        x = v2(n2);
        n2 = n2+1;
    case p_imag_d
        if n3 == 500
            state = bandera_3;
            x = v3(n3);
        else
            state = p_real_d;
            x = v3(n3);
            n3 = n3+1;
        end
    case bandera_3
        state = fin;
        x = 13;
    case fin
        if cont1 < ((40060*m)-1)%Tiempo que debe pasar para que el
        %dato 2003 se envíe, no se suman 2503 porque cuando se llega
        %a este punto ya han pasado los 2503 tiempos.
            x = 30;
            cont1 = cont1+1;
        else
            state = inicio;
            cont1 = 0;
            t1 = 0;
            n = 1;
            n1 = 1;
            n2 = 1;
            n3 = 1;
            x = 30;
        end
end

if a ~= 30
    v.push_front_pop_back(a);
end
if b ~= 30
    v1.push_front_pop_back(b);
end
if c ~= 30
```

```
        v2.push_front_pop_back(c);
    end
    if d ~= 30
        v3.push_front_pop_back(d);
    end

%---------------------------- Descripción ----------------------------
%Esta función se encarga de establecer el protocolo que crea las %reglas
y normas que permiten que la tx y rx del sistema se comuniquen %sin
importar el tipo de variación que exista, por lo que se encarga %de
formar la trama de tx de la siguiente manera:
%            11-Datos_Antes_Canal-12-Datos_Despues_Canal-13
%Donde: 11, 12 y 13 representan las banderas, indicando el inicio o %fin
de los datos. La función se basa en una máquina de estados que %pasa a un
siguiente nivel solo si se satisface el llenado de las %partes
específicas de la trama.
%--------------------------------------------------------------------
```

A.5. Habilitar

```
%----------------------------------------
%Universidad Del Cauca
%Nombre:
%         habilitar
%Archivo:
%         habilitar.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%----------------------------------------

function x = habilitar(m)

a = xfix({xlBoolean},0);
b = xfix({xlBoolean},1);

persistent cont, cont = xl_state(0, {xlUnsigned,13,0});
persistent j, j = xl_state(1, {xlUnsigned,1,0});

if cont == m*j %*Tasa de Muestreo
    x = b;
    cont = 0;
else
    x = a;
end

cont = cont+1;

%---------------------------- Descripción ----------------------------
%Esta función se encarga de obtener una frecuencia de transmisión de
%9600Hz a partir de los 50MHz de la FPGA, por lo que la salida de esta
%función se pone a uno cada 5208 tiempos, puesto que:
%                           50MHz/9600Hz = 5208
%--------------------------------------------------------------------
```

A.6. Conversor

```
%----------------------------------------
%Universidad Del Cauca
%Nombre:
%         conversor
%Archivo:
%         conversor.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%----------------------------------------

function x = conversor(enable,d)

start = 0;
bit_0 = 1;
bit_1 = 2;
bit_2 = 3;
bit_3 = 4;
bit_4 = 5;
bit_5 = 6;
bit_6 = 7;
bit_7 = 8;
stop = 9;
start_1 = 10;
bit_8 = 11;
bit_9 = 12;
bit_10 = 13;
bit_11 = 14;
bit_12 = 15;
bit_13 = 16;
bit_14 = 17;
bit_15 = 18;
stop_1 = 19;

persistent v, v = xl_state(zeros(1,2004), {xlSigned,16,11});
persistent n, n = xl_state(0, {xlUnsigned,11,0});
persistent state, state = xl_state(start, {xlUnsigned, 5, 0});
x = 0;

    switch state
        case start
            if enable == true
                state = bit_0;
                x = xl_slice(v(n),0,0);
            else
                state = start;
                x = 0;
            end
        case bit_0
            if enable == true
                state = bit_1;
                x = xl_slice(v(n),1,1);
            else
                state = bit_0;
```

```
            x = xl_slice(v(n),0,0);
        end
    case bit_1
        if enable == true
            state = bit_2;
            x = xl_slice(v(n),2,2);
        else
            state = bit_1;
            x = xl_slice(v(n),1,1);
        end
    case bit_2
        if enable == true
            state = bit_3;
            x = xl_slice(v(n),3,3);
        else
            state = bit_2;
            x = xl_slice(v(n),2,2);
        end
    case bit_3
        if enable == true
            state = bit_4;
            x = xl_slice(v(n),4,4);
        else
            state = bit_3;
            x = xl_slice(v(n),3,3);
        end
    case bit_4
        if enable == true
            state = bit_5;
            x = xl_slice(v(n),5,5);
        else
            state = bit_4;
            x = xl_slice(v(n),4,4);
        end
    case bit_5
        if enable == true
            state = bit_6;
            x = xl_slice(v(n),6,6);
        else
            state = bit_5;
            x = xl_slice(v(n),5,5);
        end
    case bit_6
        if enable == true
            state = bit_7;
            x = xl_slice(v(n),7,7);
        else
            state = bit_6;
            x = xl_slice(v(n),6,6);
        end
    case bit_7
        if enable == true
            state = stop;
            x = 1;
        else
            state = bit_7;
```

```
            x = xl_slice(v(n),7,7);
        end
    case stop
        if enable == true
            state = start_1;
            x = 0;
        else
            state = stop;
            x = 1;
        end
    case start_1
        if enable == true
            state = bit_8;
            x = xl_slice(v(n),8,8);
        else
            state = start_1;
            x = 0;
        end
    case bit_8
        if enable == true
            state = bit_9;
            x = xl_slice(v(n),9,9);
        else
            state = bit_8;
            x = xl_slice(v(n),8,8);
        end
    case bit_9
        if enable == true
            state = bit_10;
            x = xl_slice(v(n),10,10);
        else
            state = bit_9;
            x = xl_slice(v(n),9,9);
        end
    case bit_10
        if enable == true
            state = bit_11;
            x = xl_slice(v(n),11,11);
        else
            state = bit_10;
            x = xl_slice(v(n),10,10);
        end
    case bit_11
        if enable == true
            state = bit_12;
            x = xl_slice(v(n),12,12);
        else
            state = bit_11;
            x = xl_slice(v(n),11,11);
        end
    case bit_12
        if enable == true
            state = bit_13;
            x = xl_slice(v(n),13,13);
        else
            state = bit_12;
```

```
            x = xl_slice(v(n),12,12);
        end
    case bit_13
        if enable == true
            state = bit_14;
            x = xl_slice(v(n),14,14);
        else
            state = bit_13;
            x = xl_slice(v(n),13,13);
        end
    case bit_14
        if enable == true
            state = bit_15;
            x = xl_slice(v(n),15,15);
        else
            state = bit_14;
            x = xl_slice(v(n),14,14);
        end
    case bit_15
        if enable == true
            x = 1;
            if n < 2002
                state = stop_1;
                n = n+1;
            else
                state = stop_1;
                n = 0;
            end
        else
            state = bit_15;
             x = xl_slice(v(n),15,15);
        end
    case stop_1
        if enable == true
            state = start;
            x = 0;
        else
            state = stop_1;
            x = 1;
        end
end
if d ~= 30
    v.push_front_pop_back(d);
end
```

```
%---------------------------- Descripción ----------------------------
%Esta función se encarga de la transmisión serial de los datos, debido a
%que cada número contiene el formato Fix_16_11 lo que indica que existen
%16 bits en paralelo por número y se requiere un bit por el protocolo de
%tx que se utiliza(RS-232), por lo que se serializa la salida a través de
%una máquina de estados que además de serializar los 16 bits pone un bit
%de start al inicio y un bit de stop al pasar 8 bits, es así como la
%máquina contiene 20 estados, distribuidos de la siguiente manera:
%           start-priemros_8bits-stop-start-utimos_8bits-stop
%El start y stop son necesarios y fundamentales puesto que cumplen con
%las reglas y normas del protocolo de tx y rx RS-232.
%--------------------------------------------------------------------
```

APENDICE B

FUNCIONES PARA LA RECEPCIÓN DE DATOS

Las funciones necesarias para la recepción de los datos en la interfaz gráfica se realizan en tres partes, un inicio (Figura 3.1) y dos visualizaciones (Figura 3.2 y Figura 3.3), las cuales se detallan en los siguientes apartados.

B.1. Inicio

```
%---------------------------------------
%Universidad Del Cauca
%Nombre:
%         inicio
%Archivo:
%         inicio.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%---------------------------------------

function varargout = inicio(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @inicio_OpeningFcn, ...
                   'gui_OutputFcn',  @inicio_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
```

```
end

function inicio_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;

guidata(hObject, handles);
javaFrame = get(hObject,'JavaFrame');
javaFrame.setFigureIcon(javax.swing.ImageIcon('n.jpg'));

e=imread('e.jpg');
set(handles.axes1,'Visible','on');
image(e,'Parent',handles.axes1)
axis off

function varargout = inicio_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)

%------------------ Recepción y conversión --------------------%
m = 1;
w = 1;
ab_16 = zeros(998,16);
db_16 = zeros(998,16);
g=1;

B = 16;
L = 11;
z = zeros(1,16);
z1 = zeros(1,16);
datos_antes_ch = zeros(499,1);
z2 = zeros(1,16);
z3 = zeros(1,16);
datos_despues_ch = zeros(499,1);

j=1;
u=2;

%************************ Recepción ************************%
delete(instrfind);
S = serial('COM4','BaudRate',9600);
set(S,'StopBits',1);
set(S,'DataBits',8);
set(S,'Parity','none');
set(S,'FlowControl','hardware');
set(S,'InputBufferSize' ,10000);
set(S,'OutputBufferSize' ,10000);
set(S,'Timeout',10);
fopen(S);

b = fread(S,[10000,1],'char');
fclose(S);
delete(S);


[b_13 b_12 b_11] = banderas(b);
```

```
despuesch = zeros(b_12-b_13,1);

for i = b_13:b_12,
        despuesch(m) = b(i);
        m = m+1;
end

antesch = zeros(b_11-b_12,1);

for i = b_12:b_11,
        antesch(w) = b(i);
        w = w+1;
end

achbin=dec2bin(antesch);
dchbin=dec2bin(despuesch);

for i = 2:2:length(achbin)-4,
    ab_16(g,1:16)=[achbin(i+1,1:8) achbin(i,1:8)];
    db_16(g,1:16)=[dchbin(i+1,1:8) dchbin(i,1:8)];
    g=g+1;
end

   ab_16(ab_16==48)=0;
   db_16(db_16==48)=0;
   ab_16(ab_16==49)=1;
   db_16(db_16==49)=1;

%************************ Fin Recepción ************************%

%************************ Función convertir *********************%

for l = 1:499,
    for k = 1:16,
         if k == 1
             z(k) = -ab_16(j,k)*2^(B-L-k);
             z1(k) = -ab_16(u,k)*2^(B-L-k);
             z2(k) = -db_16(j,k)*2^(B-L-k);
             z3(k) = -db_16(u,k)*2^(B-L-k);
         else
             z(k) = ab_16(j,k)*2^(B-L-k);
             z1(k) = ab_16(u,k)*2^(B-L-k);
             z2(k) = db_16(j,k)*2^(B-L-k);
             z3(k) = db_16(u,k)*2^(B-L-k);
         end
         if k==16
             j=j+2;
             k=1;
             u=u+2;
         end
    end

    valor_dec_r = sum(z);
    valor_dec_i = sum(z1);
    valor_dec_r1 = sum(z2);
    valor_dec_i1 = sum(z3);
```

```
    valor_dec_complex = complex(valor_dec_r,valor_dec_i);
    datos_antes_ch(l) = valor_dec_complex;
    valor_dec_complex1 = complex(valor_dec_r1,valor_dec_i1);
    datos_despues_ch(l) = valor_dec_complex1;
end

%********************** Fin Función convertir ******************%
 save resultados datos_antes_ch;
 save resultados1 datos_despues_ch;

%--------------------- Fin Recepción y conversión ----------------%

diagramas;
close(gcbf);

%-------------------------- Descripción -----------------------------
%Al presionar el botón inicio de la ventana principal de la interfaz se
%activa esta función, la cual inicia estableciendo los parámetros de
%recepción para posteriormente comenzar la detección de las banderas,
%fijadas previamente en la función protocolo, necesarias para seleccionar
%que datos son antes y que otros son después del canal, una vez terminado
%este proceso de recepción los datos que se encuentran en formato binario
%son convertidos por medio de la ecuación a decimal con punto fijo y
%separados en números reales e imaginarios, al obtener los datos en el
%formato requerido se forman los números complejos necesarios para
%graficar los respectivos diagramas en el tiempo
%--------------------------------------------------------------------
```

B.2. Diagramas

Las ventanas responsables de la visualización de los diagramas en el tiempo, cargan los datos recibidos en la ventana principal de la interfaz y aplican las funciones correspondientes para generar cada diagrama (sección 2.3.2), para obtener una aproximación a una representación en tiempo real, los datos son renovados cada 0,1 segundos, lo que permite una actualización constante de las gráficas; este proceso se realiza exactamente igual en los archivos diagramas.m y diagramas1.m con excepción que en el primero los datos recibidos y graficados hacen alusión a los generados antes del canal, mientras que el segundo a los de después del canal.

```
%---------------------------------------
%Universidad Del Cauca
%Nombre:
%         diagramas
%Archivo:
%         diagramas.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gómez Paz
%---------------------------------------

function varargout = diagramas(varargin)
gui_Singleton = 1;
```

```
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @diagramas_OpeningFcn, ...
                   'gui_OutputFcn',  @diagramas_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function diagramas_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);
javaFrame = get(hObject,'JavaFrame');
javaFrame.setFigureIcon(javax.swing.ImageIcon('n.jpg'));

load resultados;
x = real(datos_antes_ch);
y = imag(datos_antes_ch);

axes(handles.axes1);
scatter(x,y,'r','fill');
set(gca,'Color','w');
set(gca,'ycolor',[0 0 0.2]);
set(gca,'xcolor',[0 0 0.2]);
hold on
axes(handles.axes3);
plot(x,y,'-r','LineWidth',2);
set(gca,'Color','w');
set(gca,'ycolor',[0 0 0.2]);
set(gca,'xcolor',[0 0 0.2]);
hold on
myeyediagram(datos_antes_ch,5,handles);
hold on
pause(0.1)
%------------------ Recepción y conversión --------------------%
w = 1;
ab_16 = zeros(998,16);
g=1;

B = 16;
L = 11;
z = zeros(1,16);
z1 = zeros(1,16);
datos_antes = zeros(499,1);

j=1;
u=2;
```

```
%****************** Recepción ******************%

delete(instrfind);
S = serial('COM4','BaudRate',9600);
set(S,'StopBits',1);
set(S,'DataBits',8);
set(S,'Parity','none');
set(S,'FlowControl','hardware');
set(S,'InputBufferSize' ,10000);
set(S,'OutputBufferSize' ,10000);
set(S,'Timeout',10);
fopen(S);

b = fread(S,[10000,1],'char');
fclose(S);
delete(S);

[b_13 b_12 b_11] = banderas(b);

antesch = zeros(b_11-b_12,1);

for i = b_12:b_11,
        antesch(w) = b(i);
        w = w+1;
end

achbin=dec2bin(antesch);

for i = 2:2:length(achbin)-4,
    ab_16(g,1:16)=[achbin(i+1,1:8) achbin(i,1:8)];
    g=g+1;
end

   ab_16(ab_16==48)=0;
   ab_16(ab_16==49)=1;

%****************** Fin Recepción ***************%

%****************** Función convertir ************%

for l = 1:499,
    for k = 1:16,
         if k == 1
             z(k) = -ab_16(j,k)*2^(B-L-k);
             z1(k) = -ab_16(u,k)*2^(B-L-k);
         else
             z(k) = ab_16(j,k)*2^(B-L-k);
             z1(k) = ab_16(u,k)*2^(B-L-k);
         end
         if k==16
             j=j+2;
             k=1;
             u=u+2;
         end
    end
```

```
    valor_dec_r = sum(z);
    valor_dec_i = sum(z1);

    valor_dec_complex = complex(valor_dec_r,valor_dec_i);
    datos_antes(1) = valor_dec_complex;
end

  save resultados2 datos_antes;

%****************** Fin Función convertir ***************%

%------------------ Fin Recepción y conversión --------------------%
load resultados2;
t= real(datos_antes);
o = imag(datos_antes);

axes(handles.axes1);
scatter(t,o,'r','fill');

axes(handles.axes3);
plot(t,o,'-r','LineWidth',2);

myeyediagram(datos_antes,5,handles);

function varargout = diagramas_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton1_Callback(hObject, eventdata, handles)

diagramas1;
close(gcbf);

%----------------------------------------
%Universidad Del Cauca
%Nombre:
%         diagramas1
%Archivo:
%         diagramas1.m
%Autores:
%         Melissa Eugenia Diago Mosquera
%         William Augusto Gomez Paz
%----------------------------------------

function varargout = diagramas1(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @diagramas1_OpeningFcn, ...
                   'gui_OutputFcn',  @diagramas1_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
```

```
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end

function diagramas1_OpeningFcn(hObject, eventdata, handles, varargin)

handles.output = hObject;

guidata(hObject, handles);
javaFrame = get(hObject,'JavaFrame');
javaFrame.setFigureIcon(javax.swing.ImageIcon('n.jpg'));

load resultados1;
x = real(datos_despues_ch);
y = imag(datos_despues_ch);

axes(handles.axes1);
scatter(x,y,'r','fill');
set(gca,'Color','w');
set(gca,'ycolor',[0 0 0.2]);
set(gca,'xcolor',[0 0 0.2]);
hold on
axes(handles.axes3);
plot(x,y,'-r','LineWidth',2);
set(gca,'Color','w');
set(gca,'ycolor',[0 0 0.2]);
set(gca,'xcolor',[0 0 0.2]);
hold on
myeyediagram1(datos_despues_ch,5,handles);

hold on
pause(0.1)
%------------------ Recepción y conversión --------------------%

m = 1;
db_16 = zeros(998,16);
g=1;

B = 16;
L = 11;
z2 = zeros(1,16);
z3 = zeros(1,16);
datos_despues = zeros(499,1);

j=1;
u=2;

%**************** Recepción ****************%

delete(instrfind);
S = serial('COM4','BaudRate',9600);
set(S,'StopBits',1);
set(S,'DataBits',8);
set(S,'Parity','none');
```

```
set(S,'FlowControl','hardware');
set(S,'InputBufferSize' ,10000);
set(S,'OutputBufferSize' ,10000);
set(S,'Timeout',10);
fopen(S);

b = fread(S,[10000,1],'char');
fclose(S);
delete(S);


[b_13 b_12 b_11] = banderas(b);
despuesch = zeros(b_12-b_13,1);

for i = b_13:b_12,
        despuesch(m) = b(i);
        m = m+1;
end



dchbin=dec2bin(despuesch);

for i = 2:2:length(dchbin)-4,
    db_16(g,1:16)=[dchbin(i+1,1:8) dchbin(i,1:8)];
    g=g+1;
end

   db_16(db_16==48)=0;
   db_16(db_16==49)=1;

%**************** Fin Recepción *************%

%**************** Función convertir ************%

for l = 1:499,
    for k = 1:16,
         if k == 1
             z2(k) = -db_16(j,k)*2^(B-L-k);
             z3(k) = -db_16(u,k)*2^(B-L-k);
         else
             z2(k) = db_16(j,k)*2^(B-L-k);
             z3(k) = db_16(u,k)*2^(B-L-k);
         end
         if k==16
             j=j+2;
             k=1;
             u=u+2;
         end
    end

    valor_dec_r1 = sum(z2);
    valor_dec_i1 = sum(z3);

    valor_dec_complex1 = complex(valor_dec_r1,valor_dec_i1);
    datos_despues(l) = valor_dec_complex1;
```

```
end

 save resultados3 datos_despues;

%**************** Fin Función convertir ***********%


%------------------ Recepción y conversión --------------------%
load resultados3;
t= real(datos_despues);
o = imag(datos_despues);

axes(handles.axes1);
scatter(t,o,'r','fill');

axes(handles.axes3);
plot(t,o,'-r','LineWidth',2);

myeyediagram1(datos_despues,5,handles);

function varargout = diagramas1_OutputFcn(hObject, eventdata, handles)

varargout{1} = handles.output;

function pushbutton2_Callback(hObject, eventdata, handles)
```

APENDICE C

DIAGRAMAS EN EL TIEMPO

Esta sección del trabajo de grado presenta las gráficas obtenidas de dos formas, la primera hace referencia a las generadas por la herramienta Simulink a través de los bloques *Discrete-Time Scatter Plot Scope*, *Discrete-Time Signal trajectory Scope* y *Discrete-Time Eye Diagram Scope* y la segunda por medio de las visualizaciones de la Interfaz Gráfica diseñada en el actual trabajo de grado.

C.1. Modulación por Desplazamiento de Fase

C.1.1 Modulación por desplazamiento de fase con M = 8

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

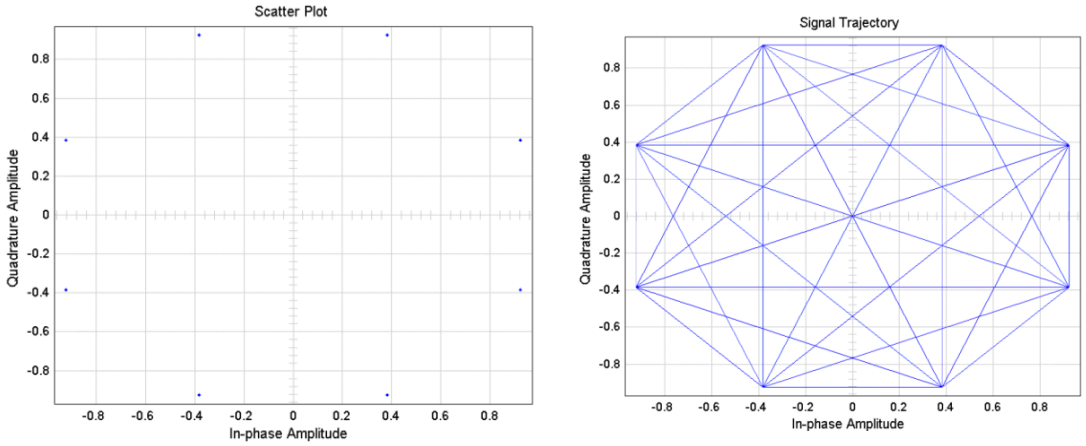


Figura C.1. Diagramas en el tiempo generados por Simulink para 8PSK.

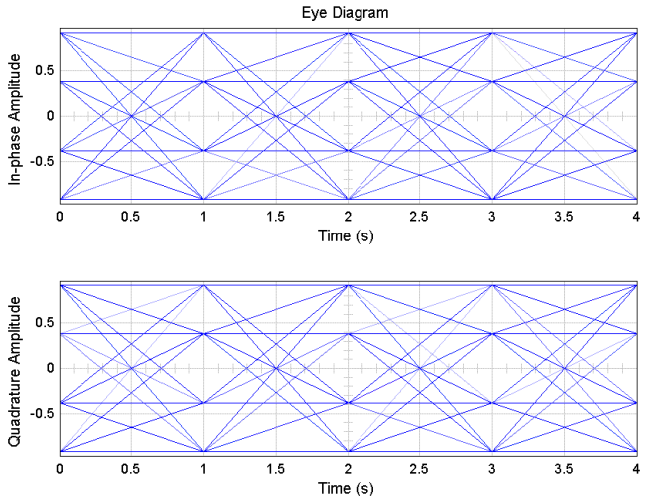


Figura C.2. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

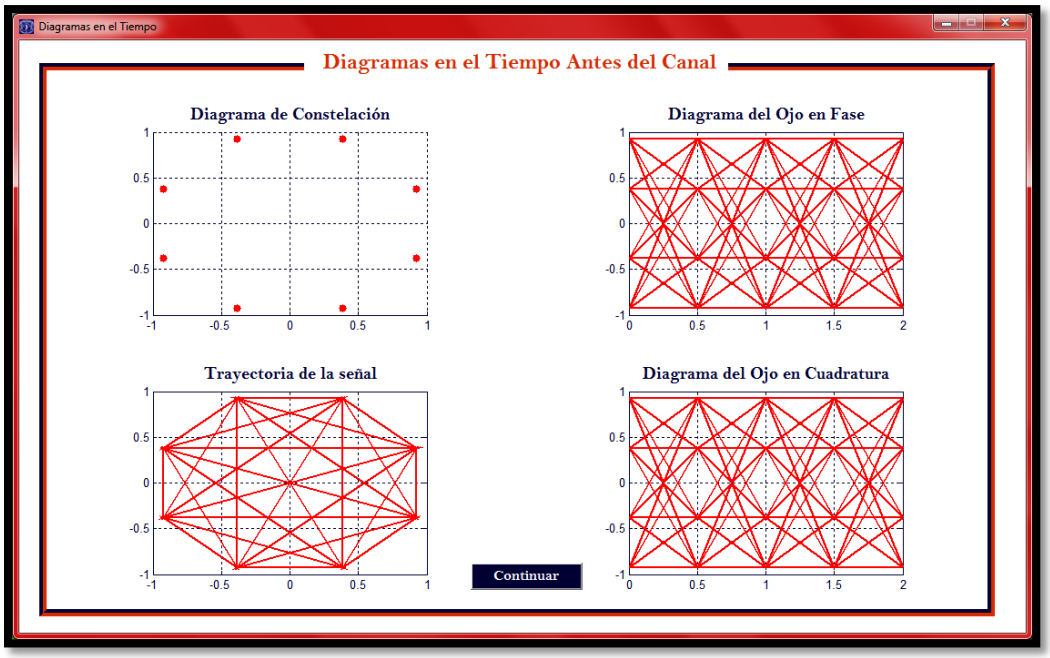


Figura C.3. Diagramas en el tiempo generados por la interfaz gráfica para 8PSK.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

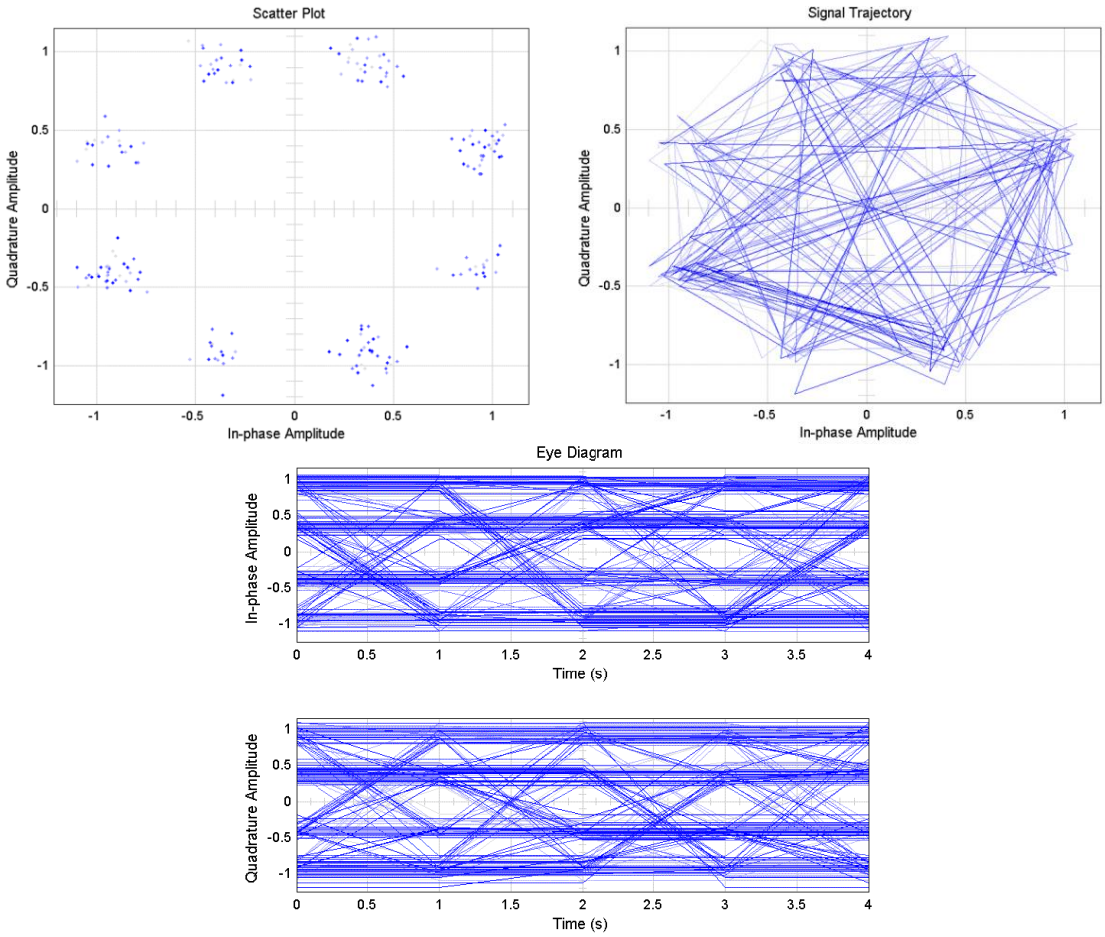


Figura C.4. Diagramas en el tiempo generados por Simulink para 8PSK con $E_b/N_o = 14dB$.

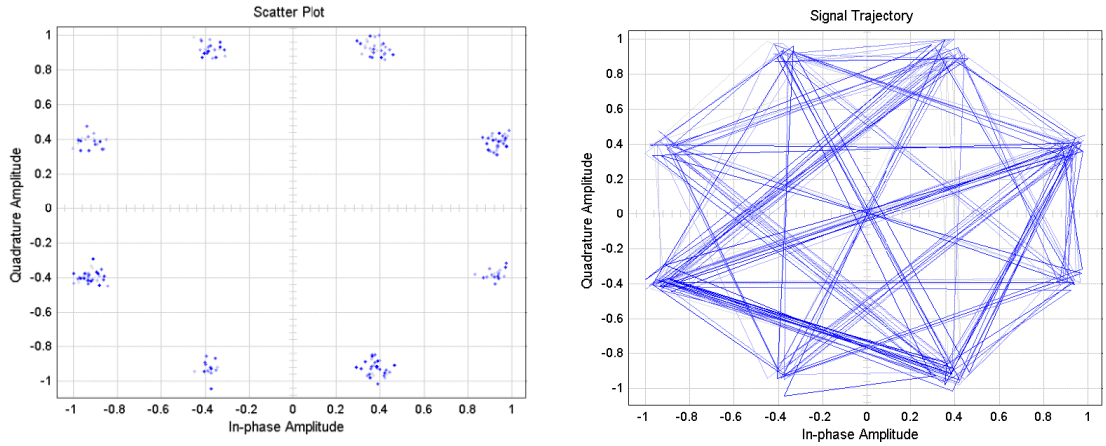


Figura C.5. Diagramas en el tiempo generados por Simulink para 8PSK con $E_b/N_o = 21dB$.

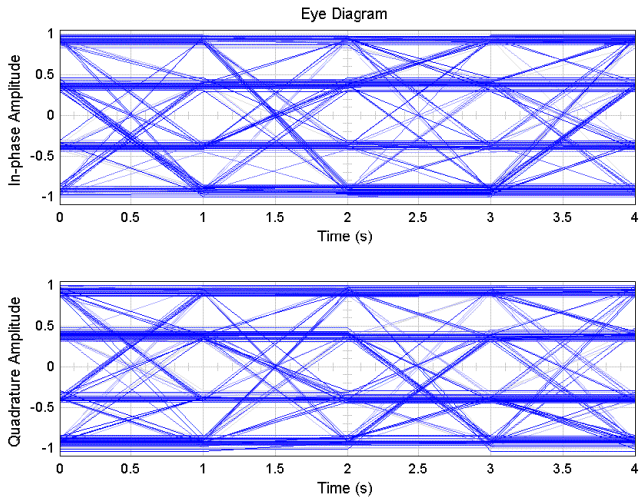


Figura C.6. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

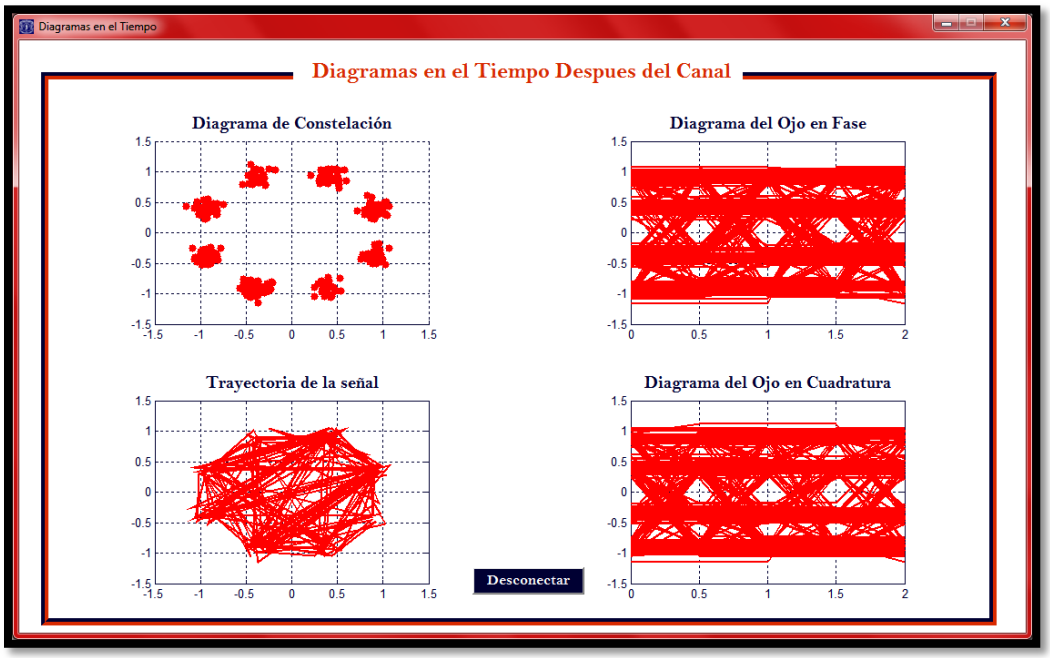


Figura C.7. Diagramas en el tiempo generados por la interfaz gráfica para 8PSK con $E_b/N_o = 14dB$.

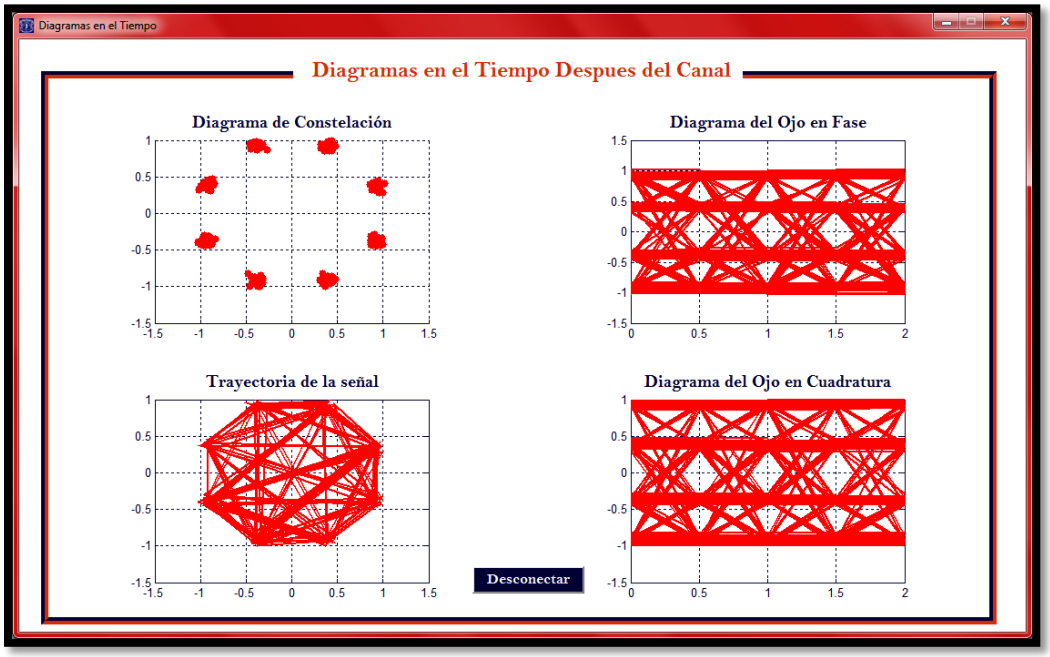


Figura C.8. Diagramas en el tiempo generados por la interfaz gráfica para 8PSK con $E_b/N_o = 21dB$.

C.1.2 Modulación por desplazamiento de fase binaria

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

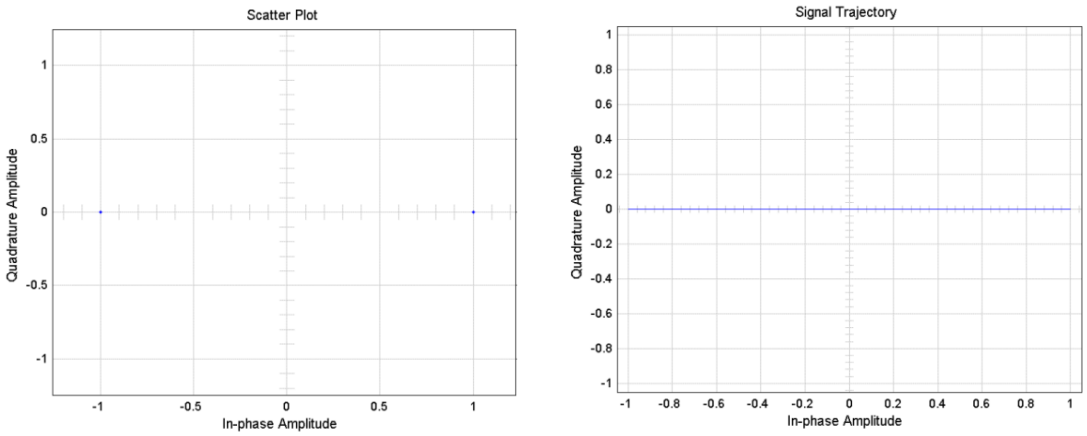


Figura C.9. Diagramas en el tiempo generados por Simulink para BPSK.

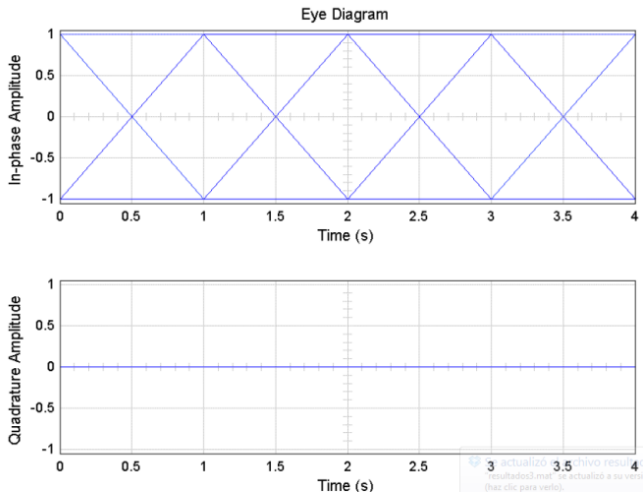


Figura C.10. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

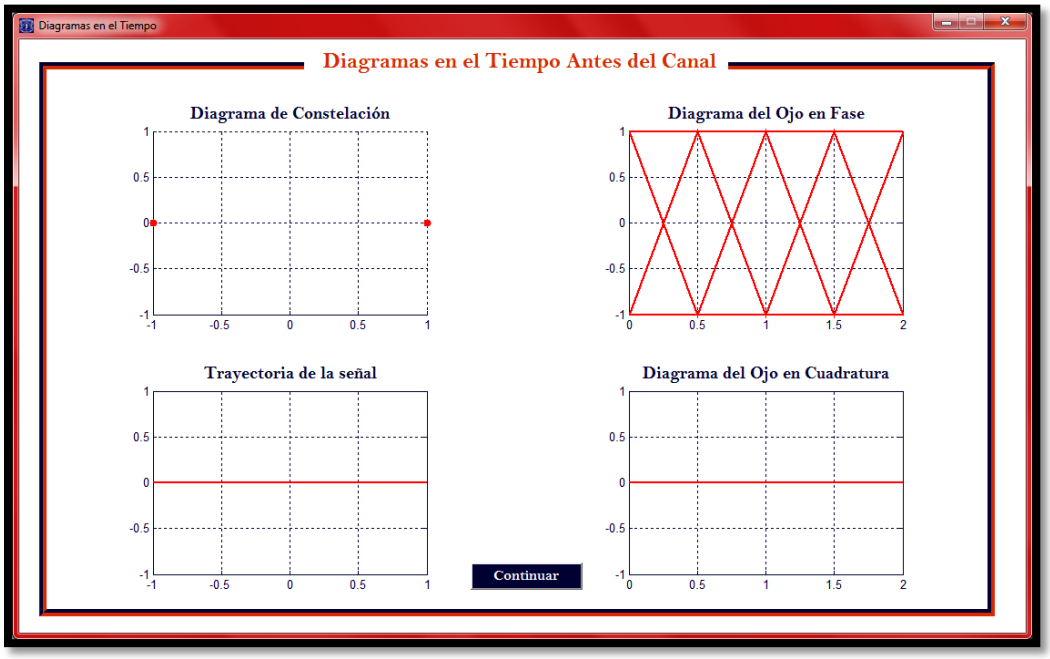


Figura C.11. Diagramas en el tiempo generados por la interfaz gráfica para BPSK.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

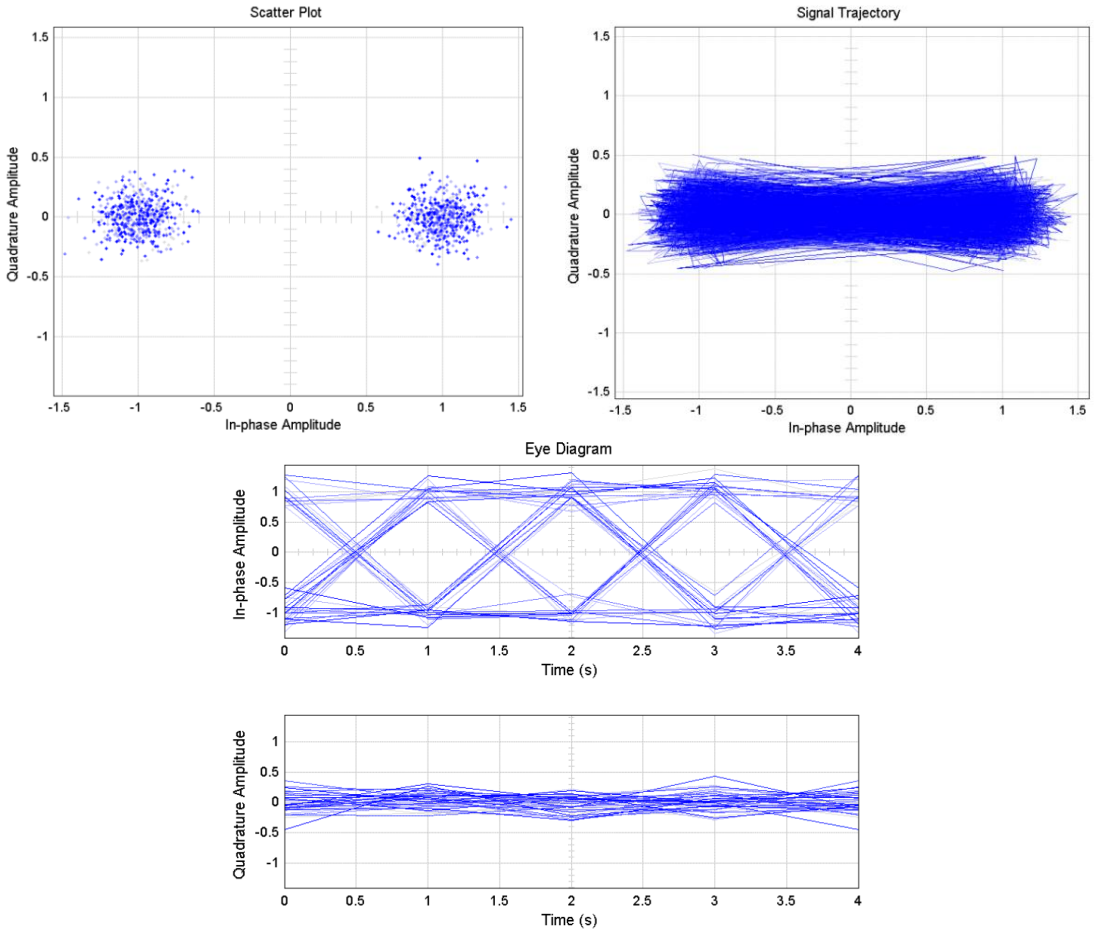


Figura C.12. Diagramas en el tiempo generados por Simulink para BPSK con $E_b/N_o = 14dB$.

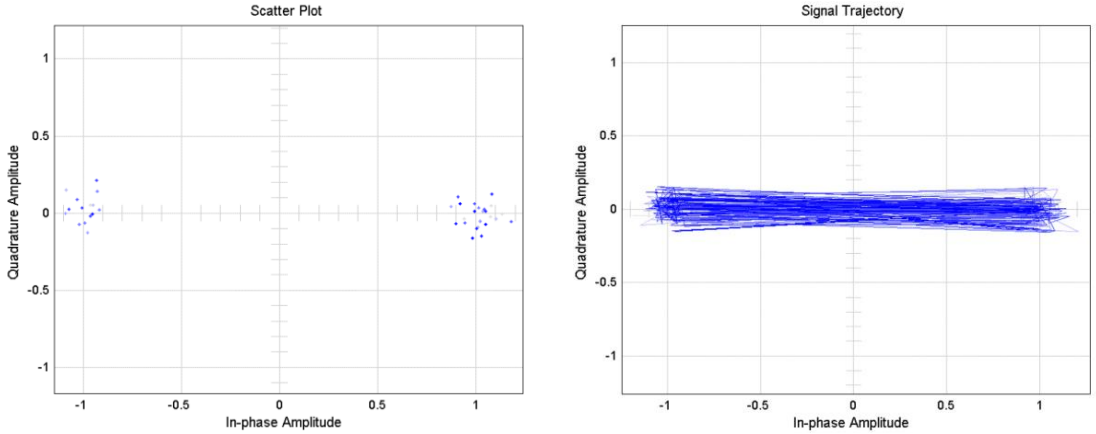


Figura C.13. Diagramas en el tiempo generados por Simulink para BPSK con $E_b/N_o = 21dB$.

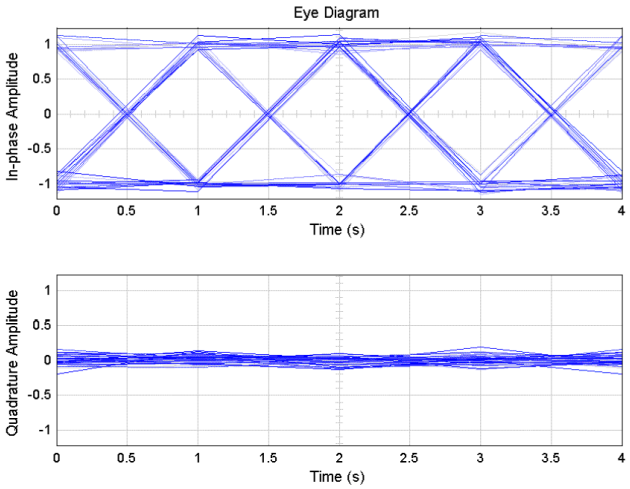


Figura C.14. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

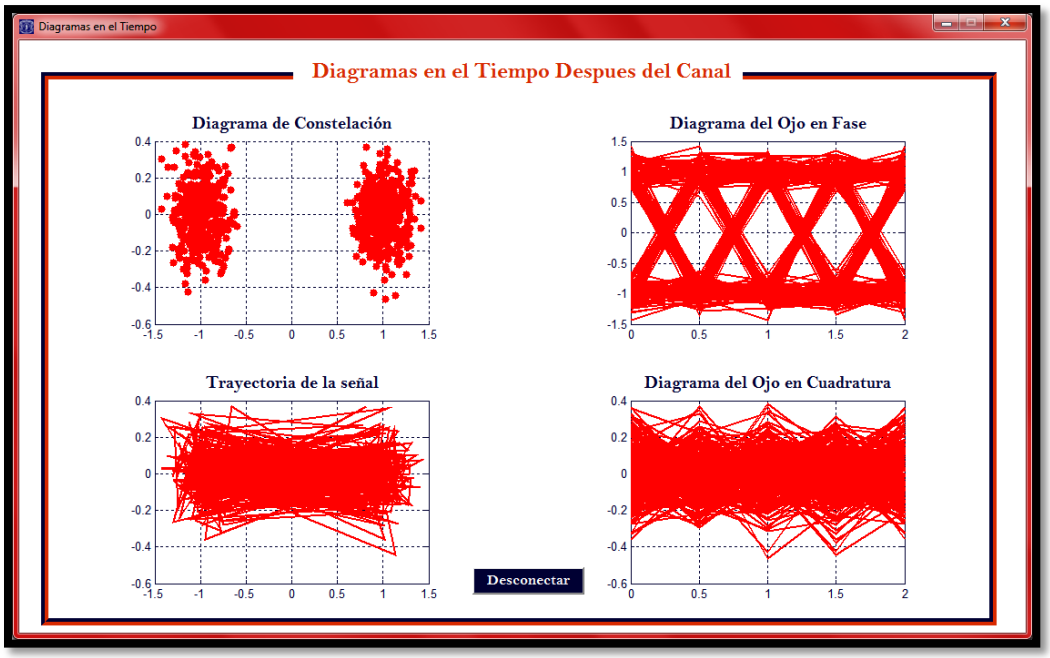


Figura C.15. Diagramas en el tiempo generados por la interfaz gráfica para BPSK con $E_b/N_o = 14dB$.

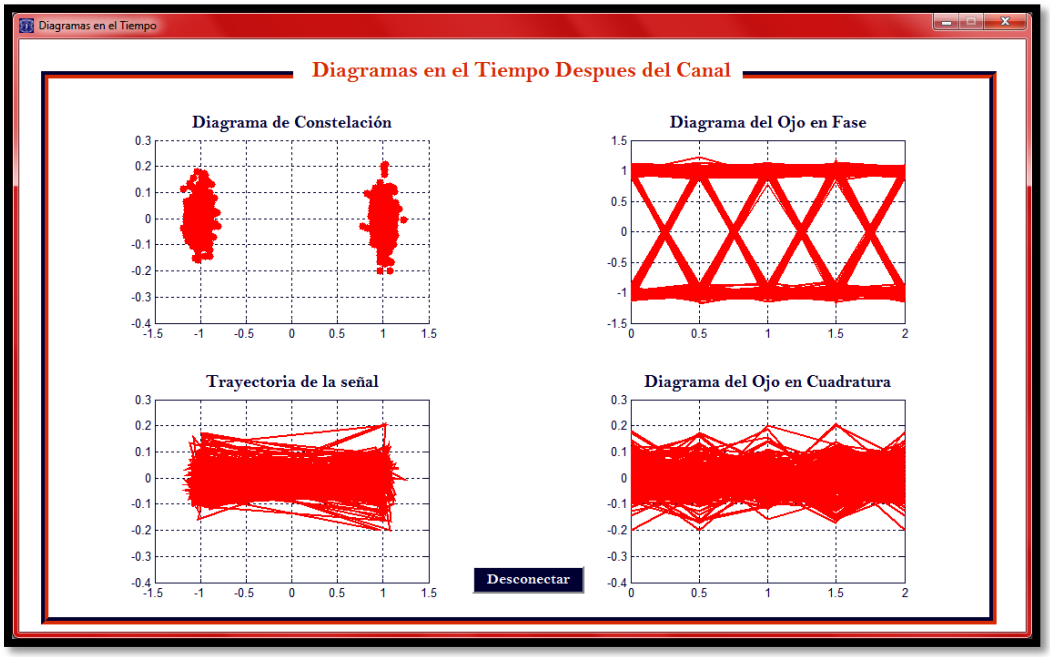


Figura C.16. Diagramas en el tiempo generados por la interfaz gráfica para BPSK con $E_b/N_o = 21dB$.

C.1.3 Modulación por desplazamiento de fase en cuadratura compensada

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

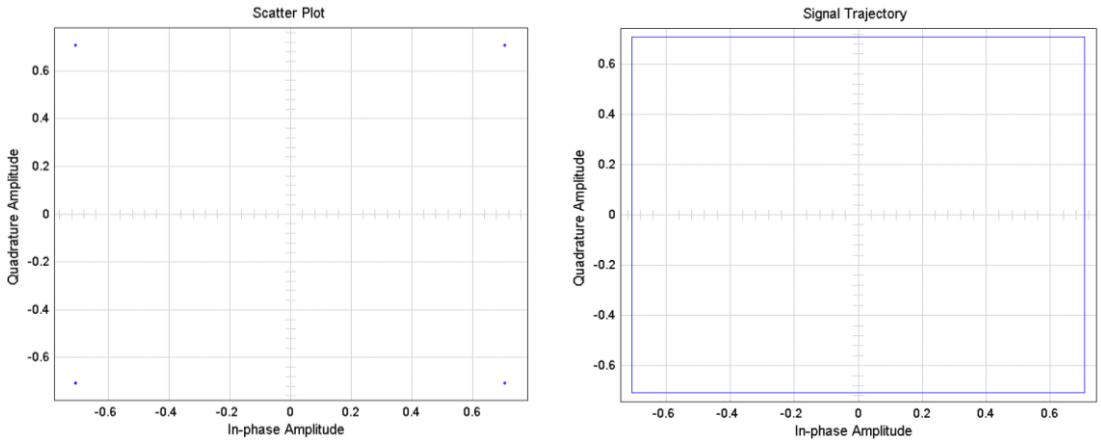


Figura C.17. Diagramas en el tiempo generados por Simulink para OQPSK.

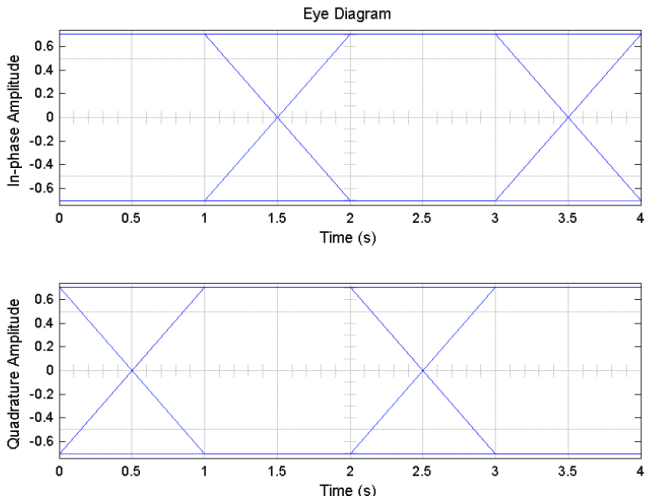


Figura C.18. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

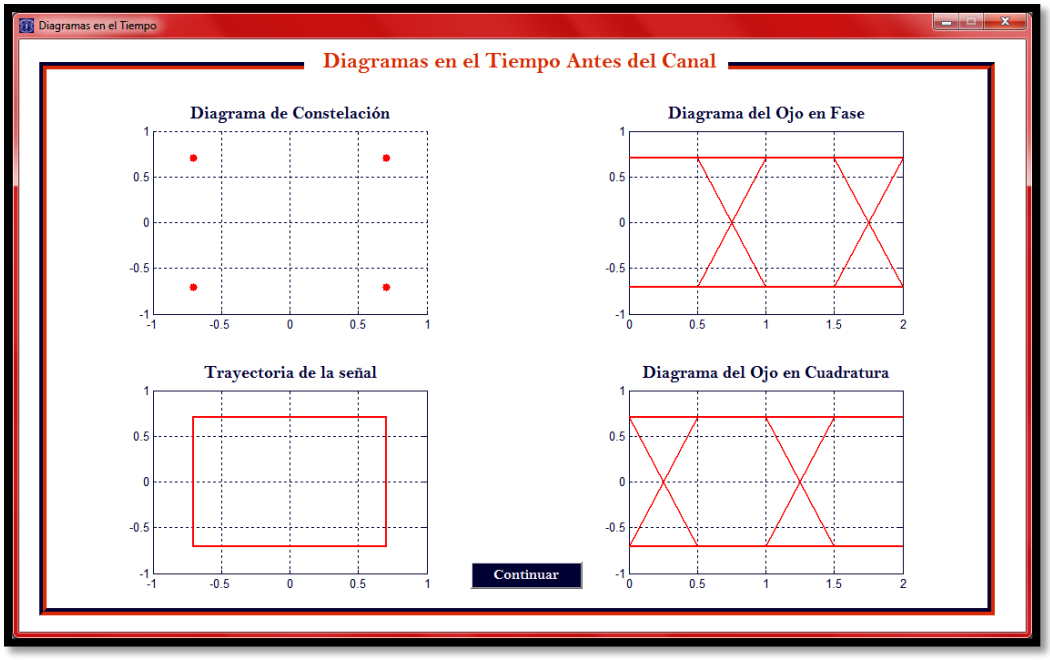


Figura C.19. Diagramas en el tiempo generados por la interfaz gráfica para OQPSK.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

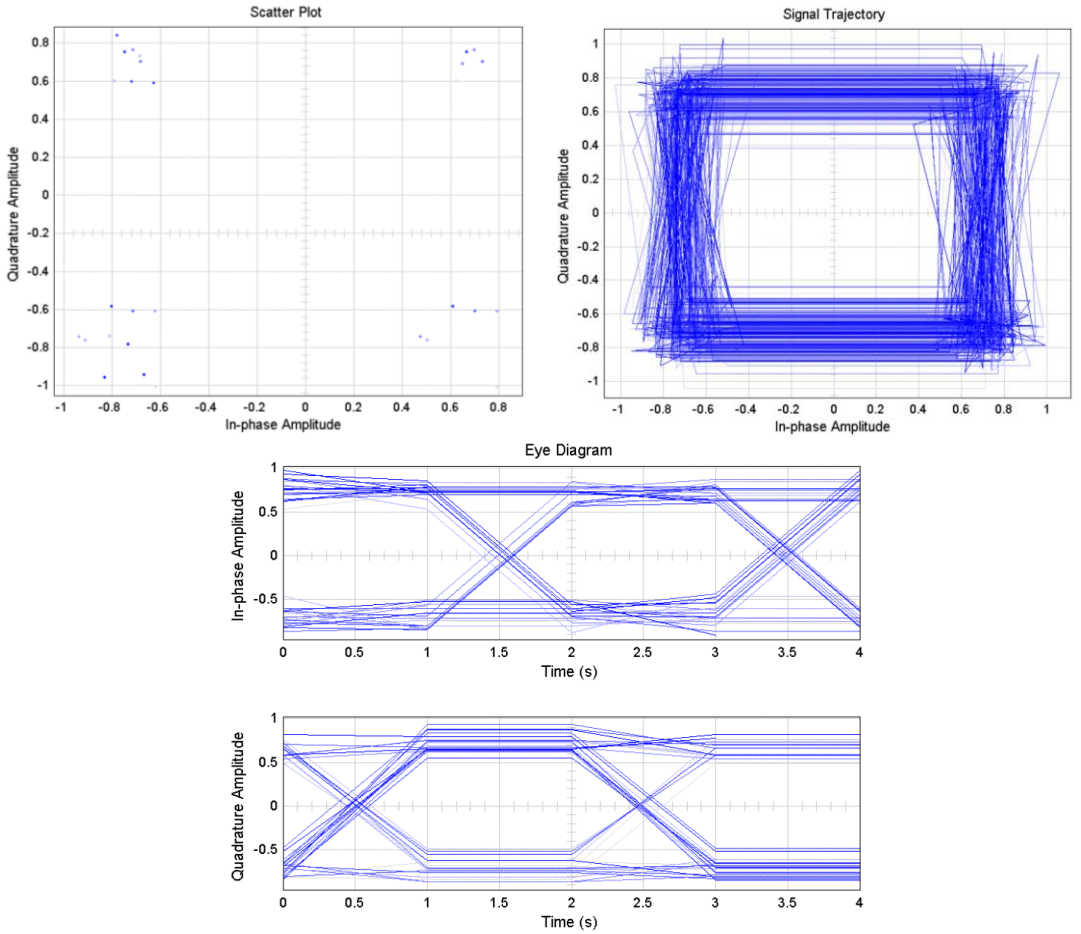


Figura C.20. Diagramas en el tiempo generados por Simulink para OQPSK con $E_b/N_o = 14dB$.

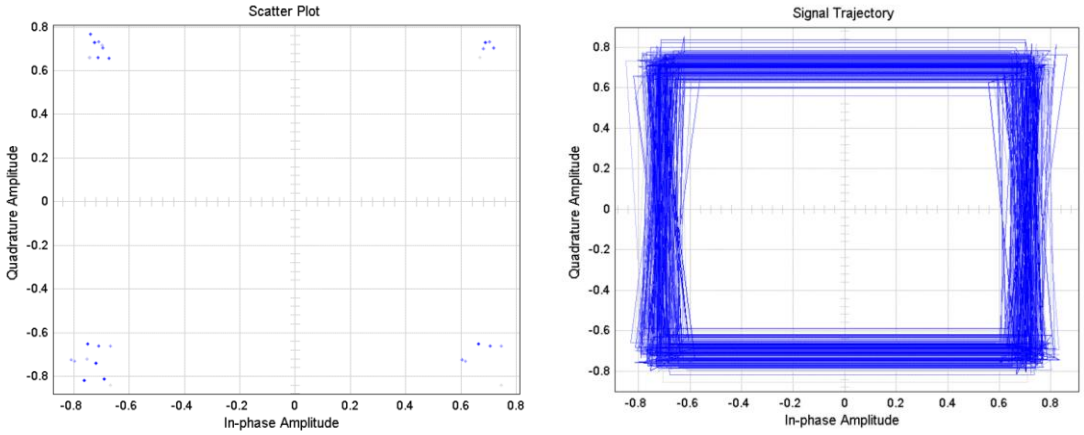


Figura C.21. Diagramas en el tiempo generados por Simulink para OQPSK con $E_b/N_o = 21dB$.

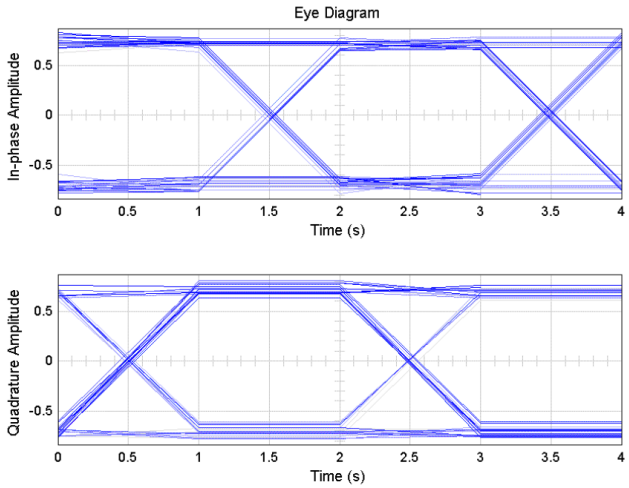


Figura C.22. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

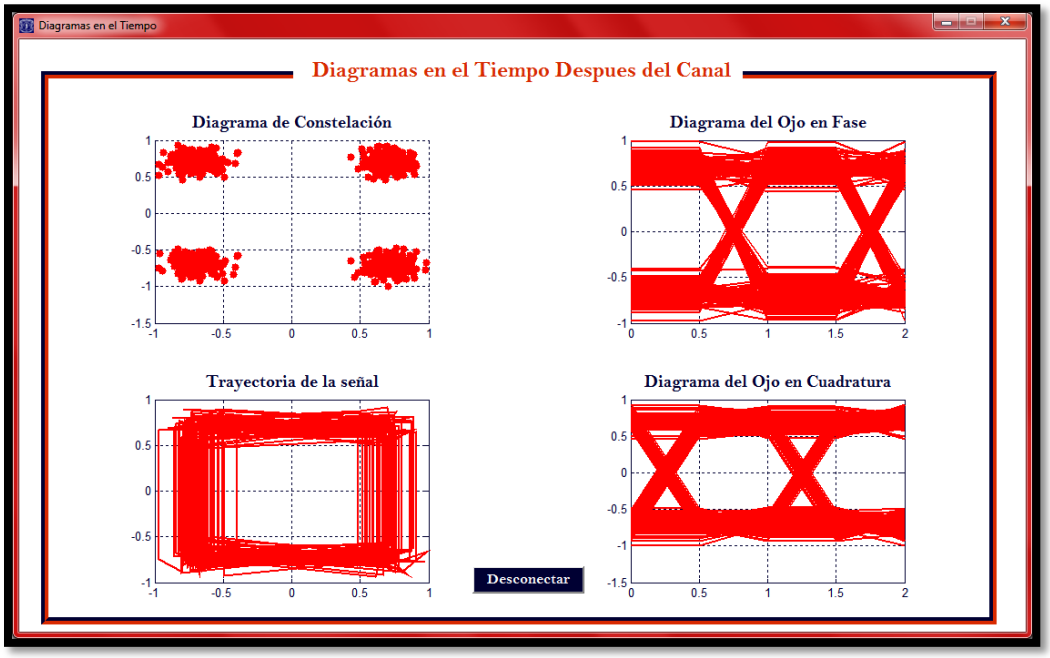


Figura C.23. Diagramas en el tiempo generados por la interfaz gráfica para OQPSK con $E_b/N_o = 14dB$.

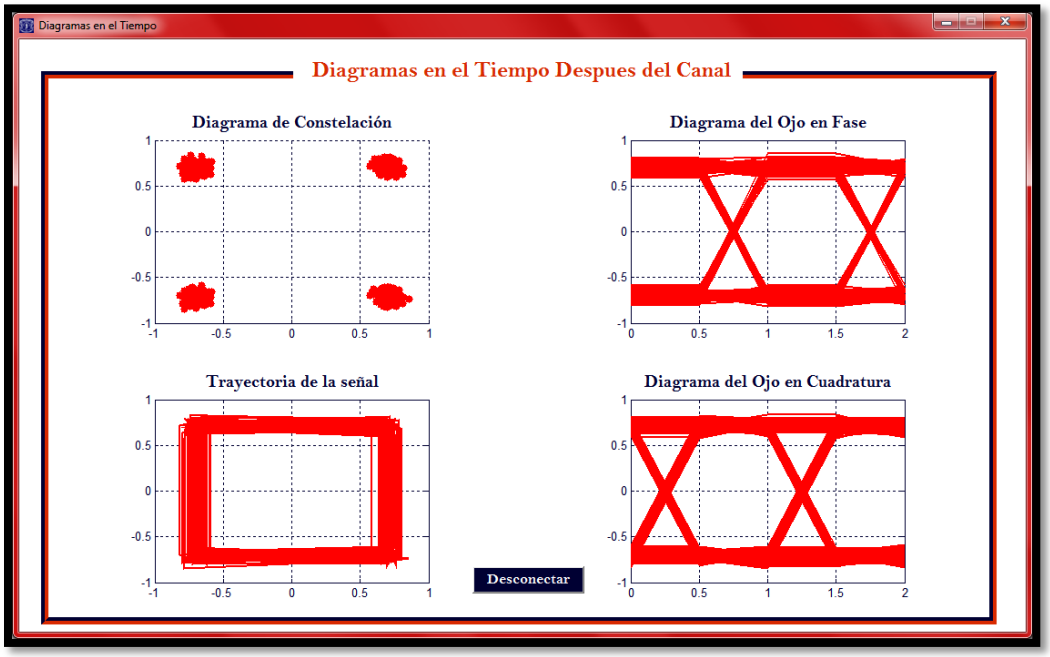


Figura C.24. Diagramas en el tiempo generados por la interfaz gráfica para OQPSK con $E_b/N_o = 21dB$.

C.1.4 Modulación por desplazamiento de fase en cuadratura diferencial

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

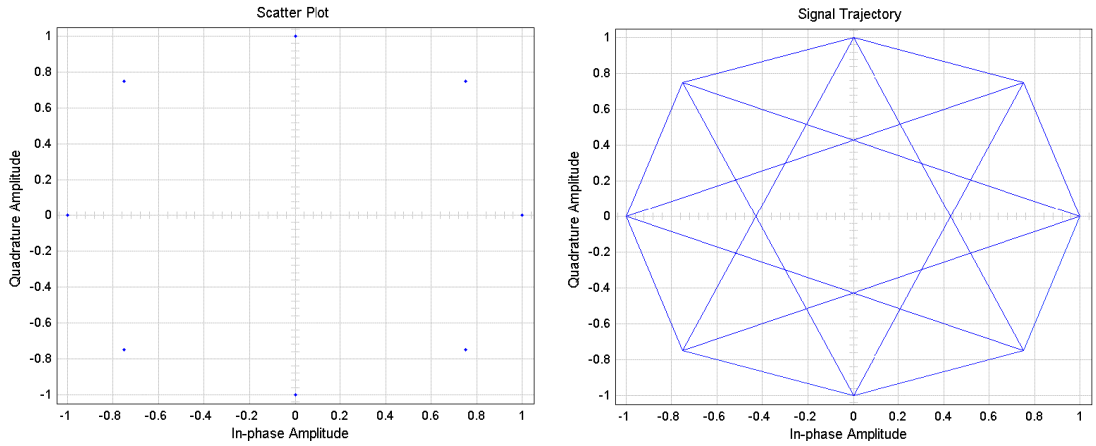


Figura C.25. Diagramas en el tiempo generados por Simulink para DQPSK.

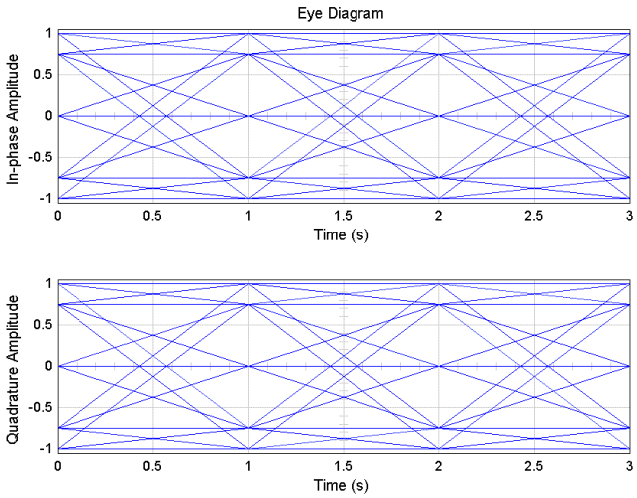


Figura C.26. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

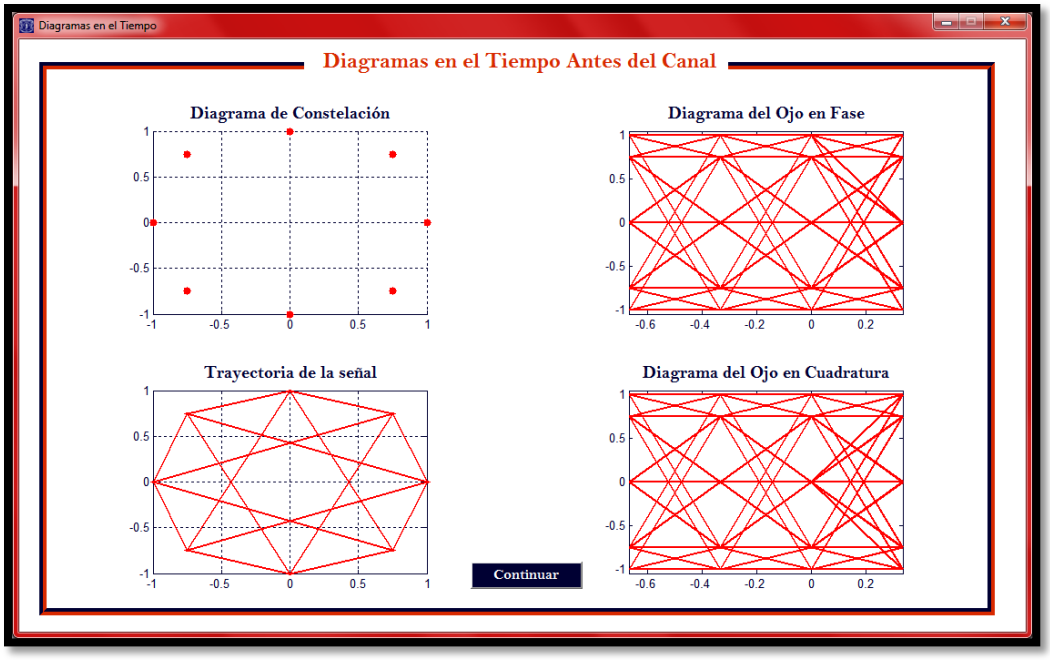


Figura C.27. Diagramas en el tiempo generados por la interfaz gráfica para DQPSK.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

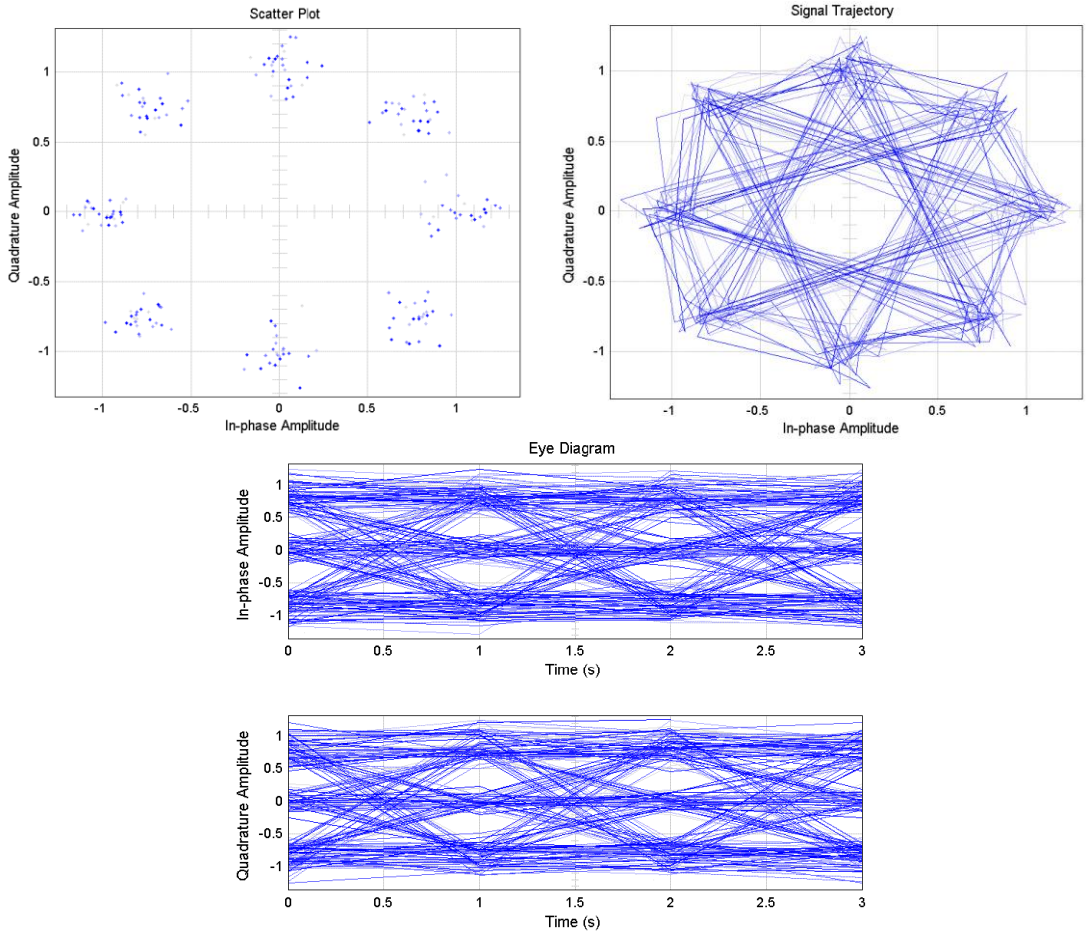


Figura C.28. Diagramas en el tiempo generados por Simulink para DQPSK con $E_b/N_o = 14dB$.

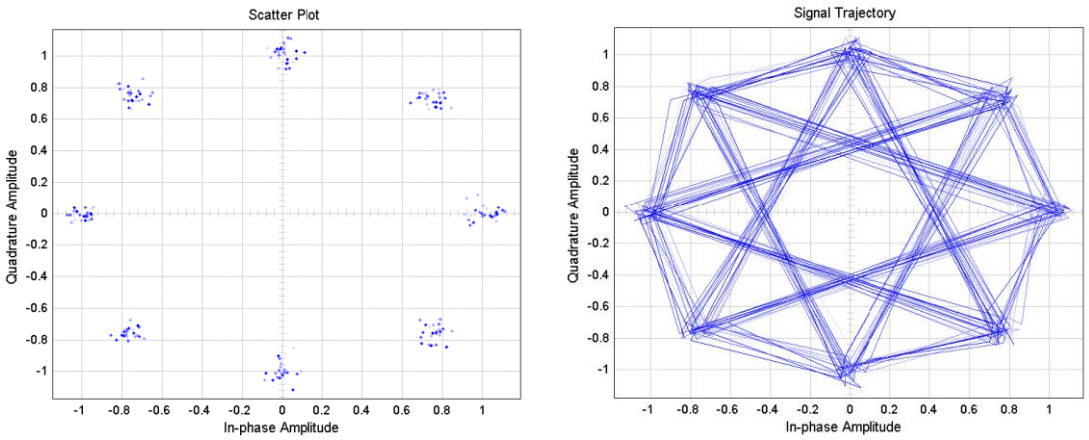


Figura C.29. Diagramas en el tiempo generados por Simulink para DQPSK con $E_b/N_o = 21dB$.

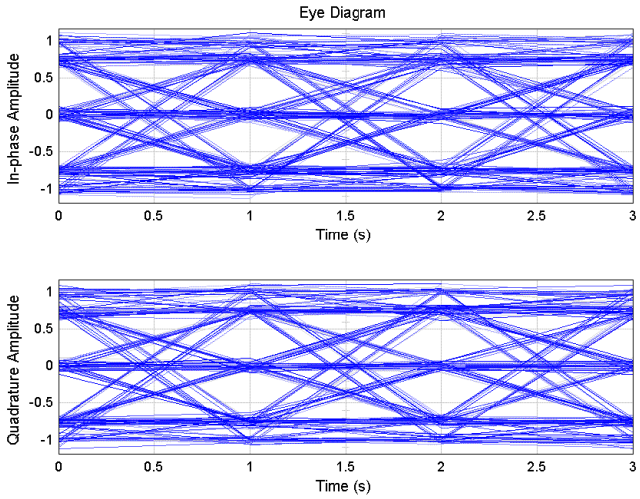


Figura C.30. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

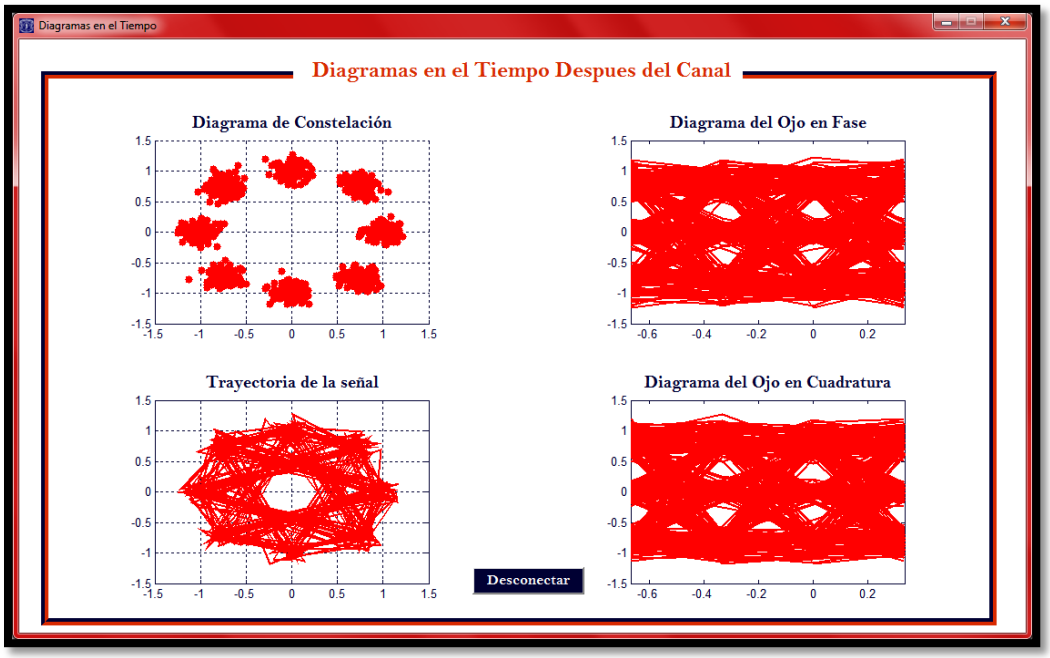


Figura C.31. Diagramas en el tiempo generados por la interfaz gráfica para DQPSK con $E_b/N_o = 14dB$.

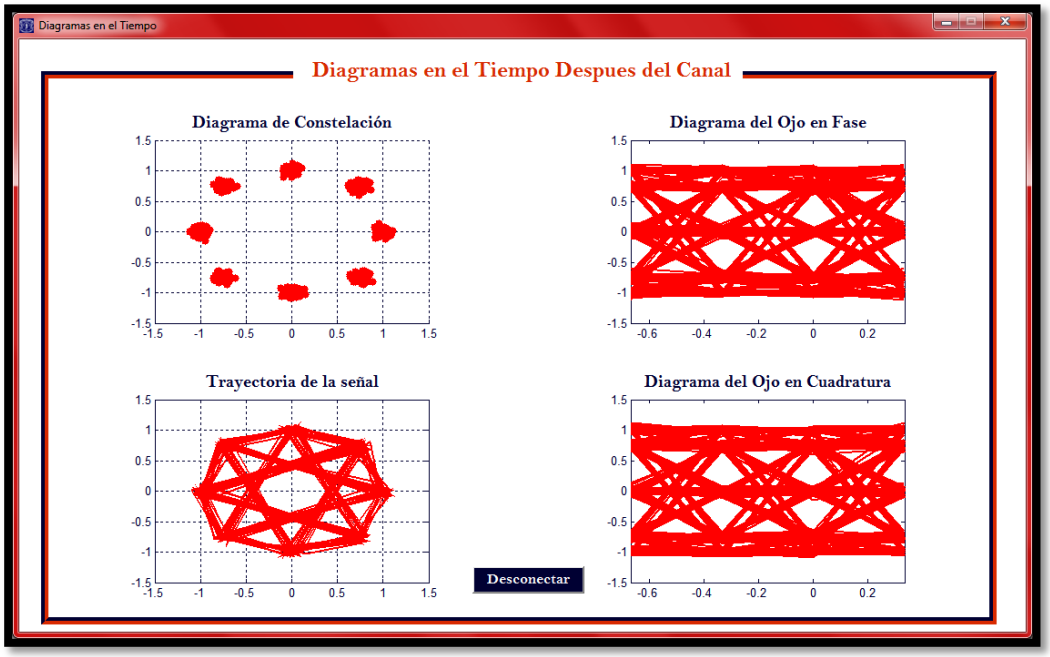


Figura C.32. Diagramas en el tiempo generados por la interfaz gráfica para DQPSK con $E_b/N_o = 21dB$.

C.1.5 Modulación por desplazamiento de fase binaria diferencial

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

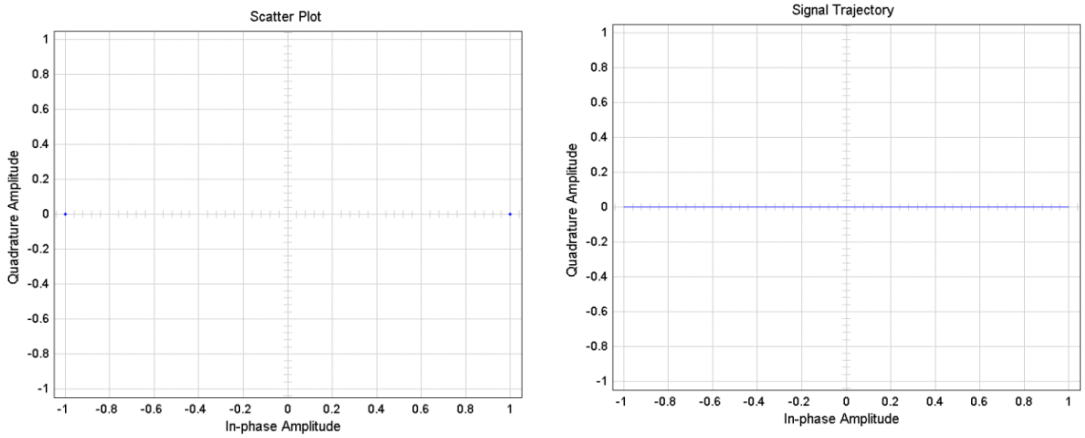


Figura C.33. Diagramas en el tiempo generados por Simulink para DBPSK.

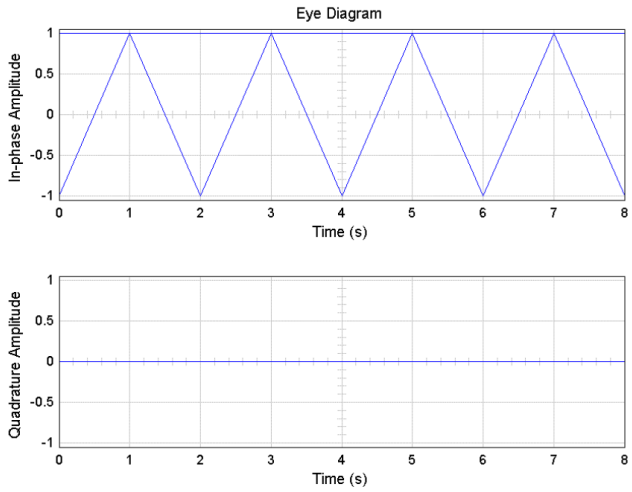


Figura C.34. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

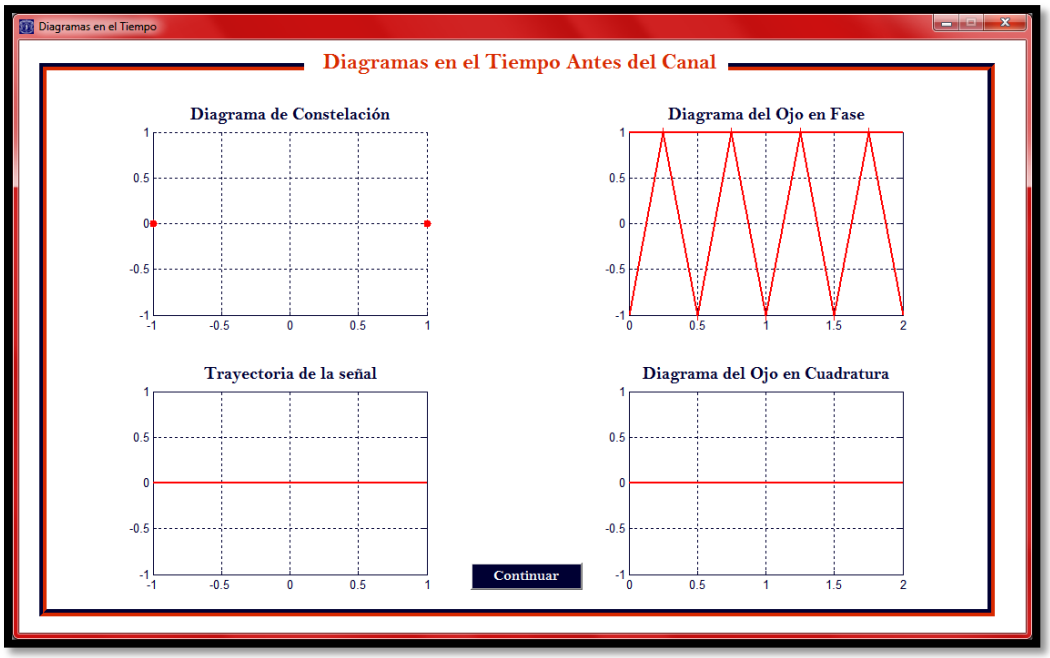


Figura C.35. Diagramas en el tiempo generados por la interfaz gráfica para DBPSK

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

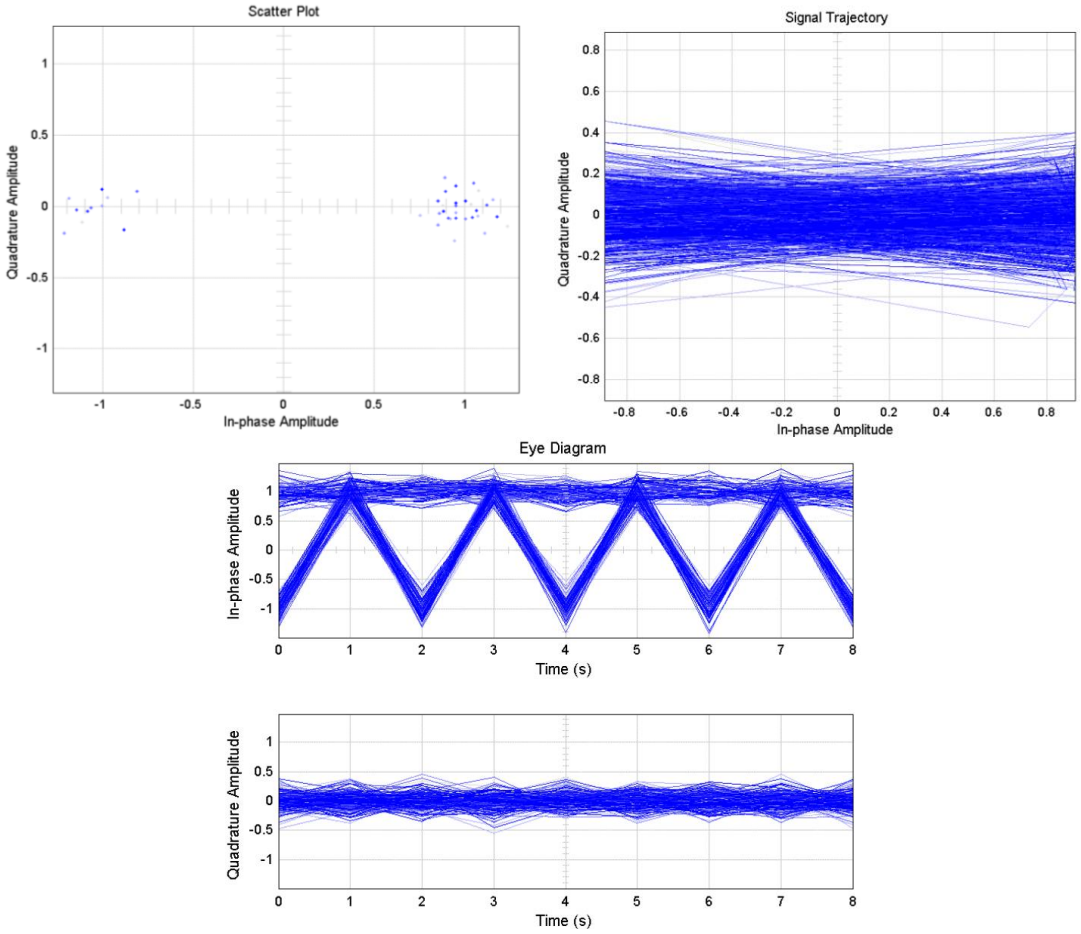


Figura C.36. Diagramas en el tiempo generados por Simulink para DBPSK con $E_b/N_o = 14dB$.

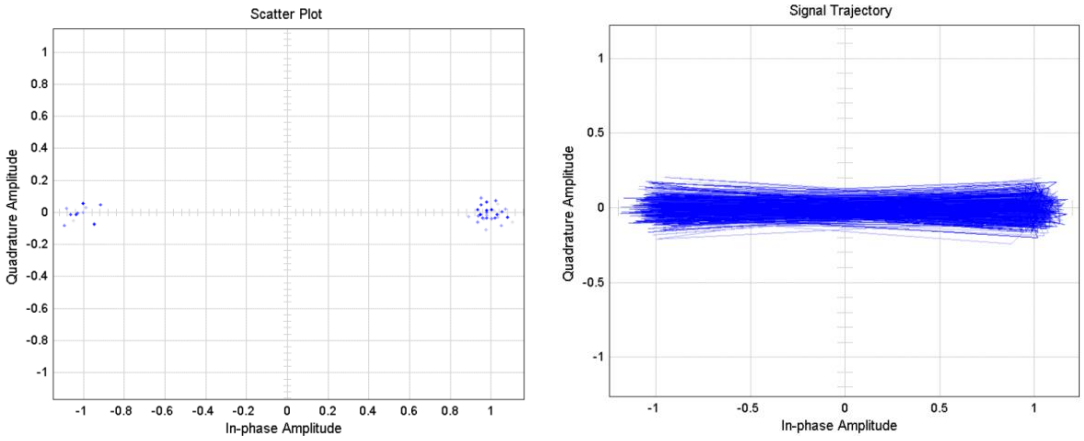


Figura C.37. Diagramas en el tiempo generados por Simulink para DBPSK con $E_b/N_o = 21dB$.

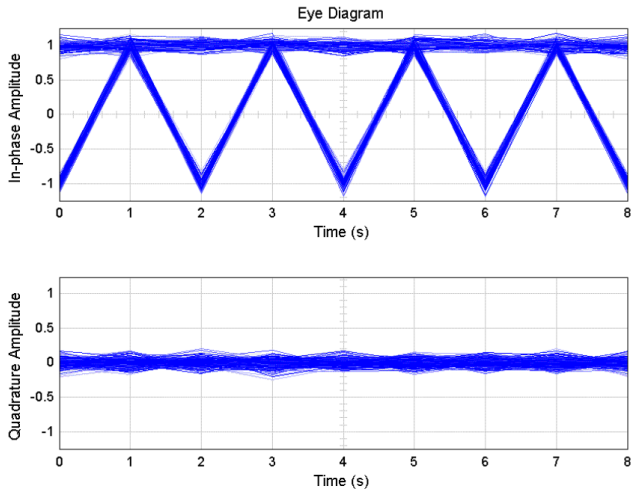


Figura C.38. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

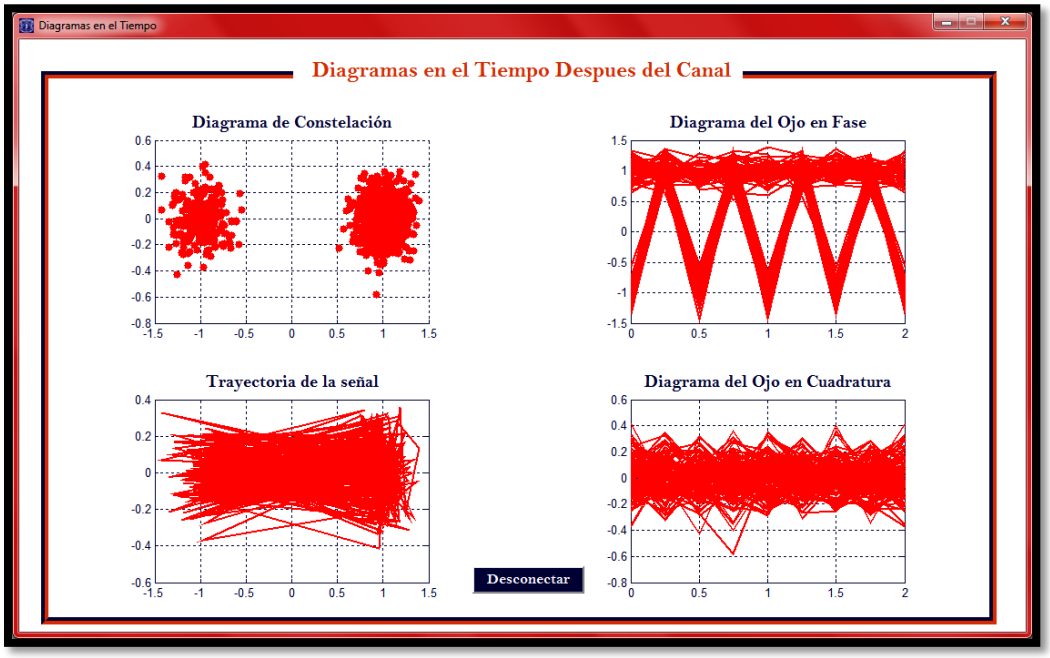


Figura C.39. Diagramas en el tiempo generados por la interfaz gráfica para DBPSK con $E_b/N_o = 14dB$.

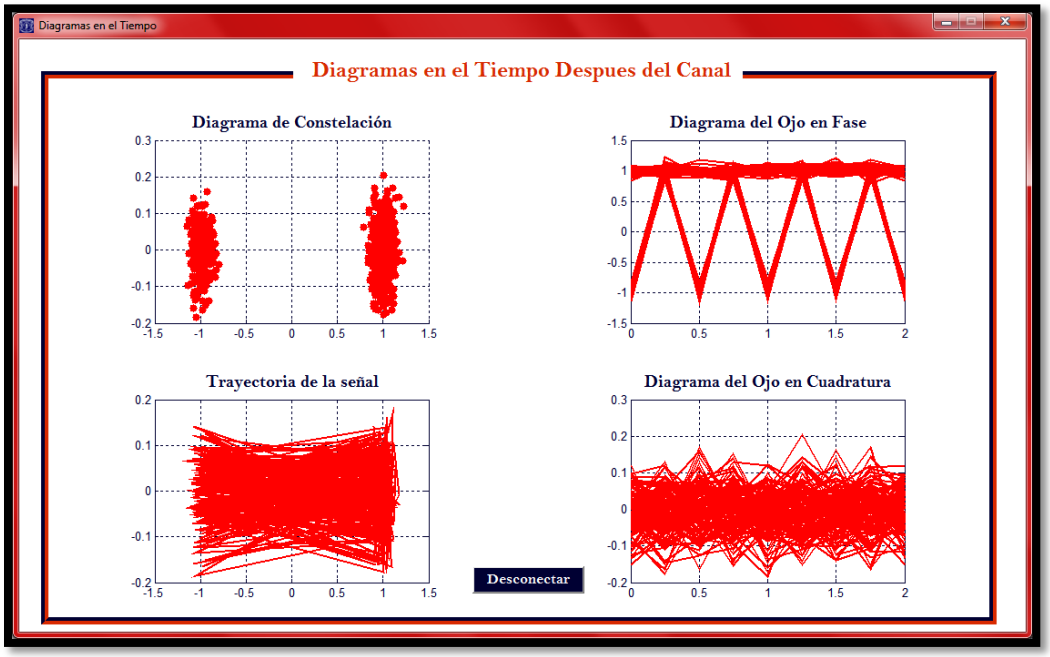


Figura C.40. Diagramas en el tiempo generados por la interfaz gráfica para DBPSK con $E_b/N_o = 21dB$.

C.2. Modulación de Amplitud en Cuadratura

C.2.1. Modulación de amplitud en cuadratura con M = 16

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

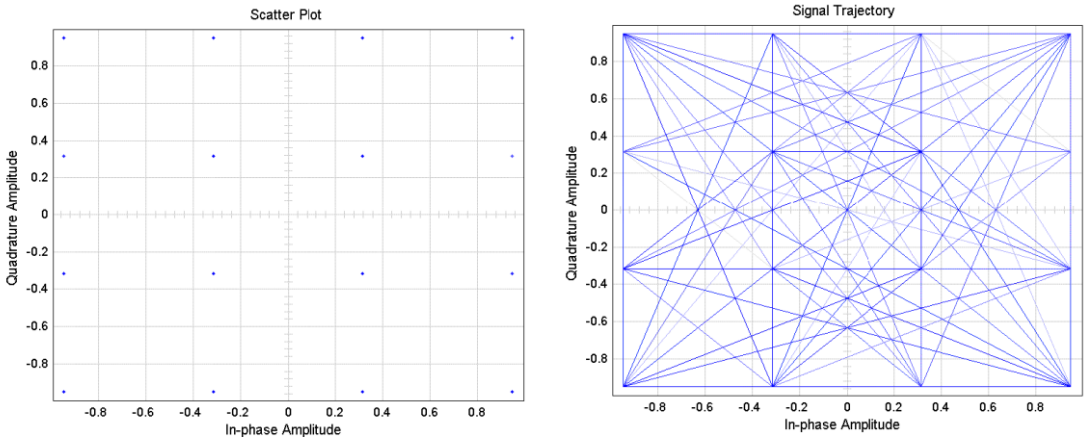


Figura C.41. Diagramas en el tiempo generados por Simulink para 16QAM.

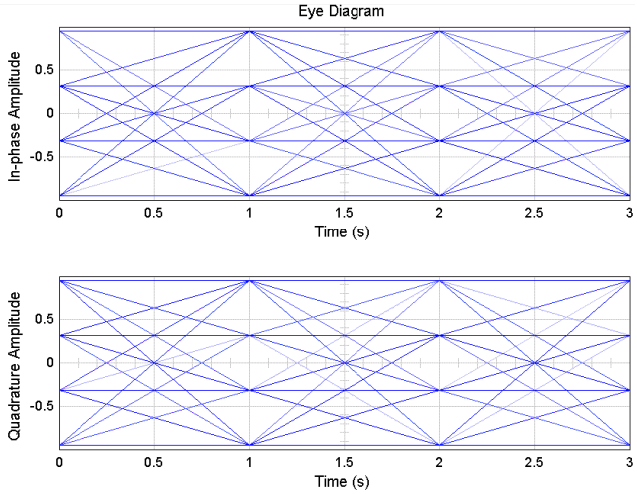


Figura C.42. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

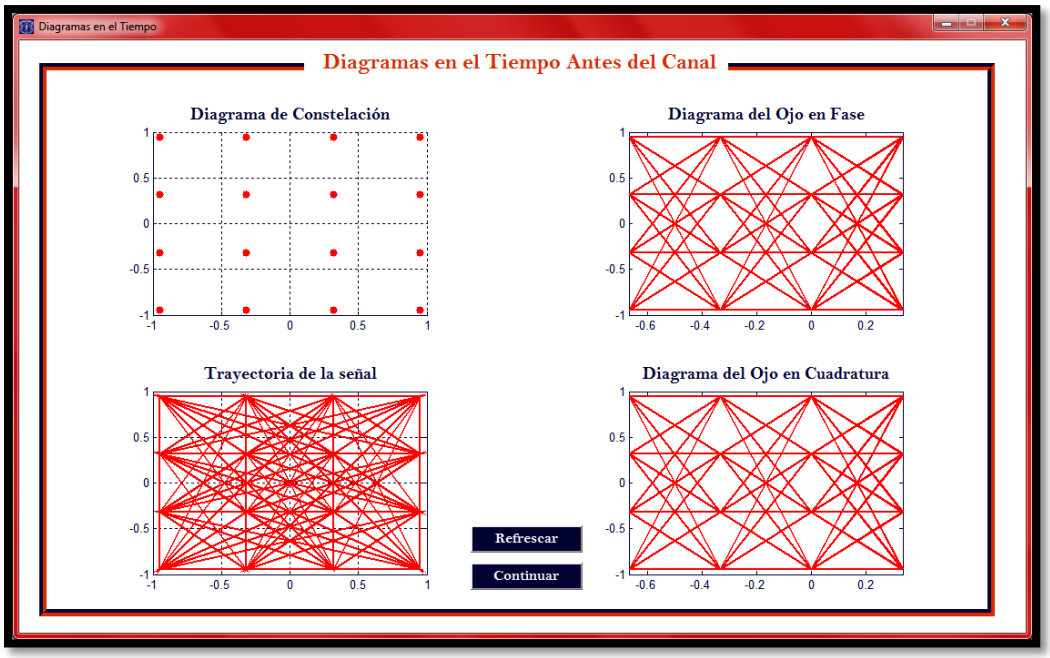


Figura C.43. Diagramas en el tiempo generados por la interfaz gráfica para 16QAM.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

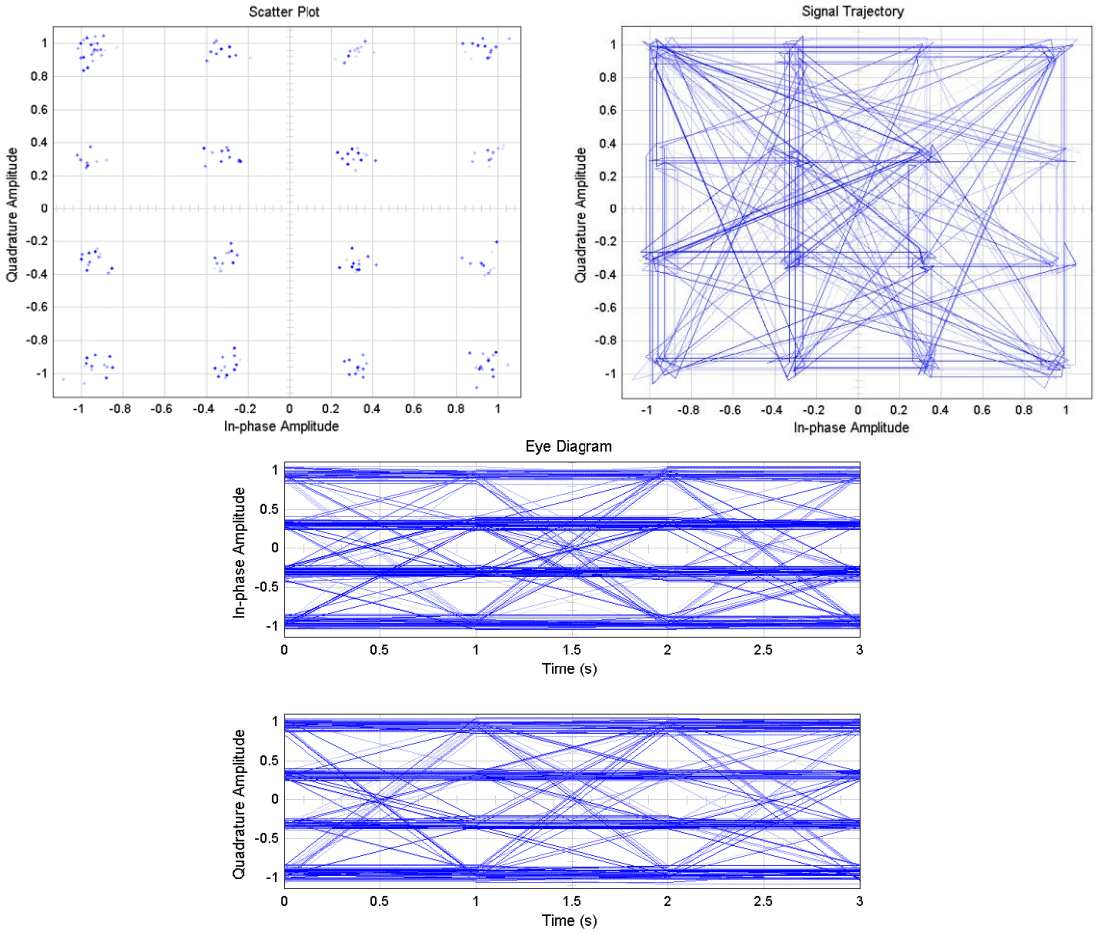


Figura C.44. Diagramas en el tiempo generados por Simulink para 16QAM con $E_b/N_o = 17dB$.

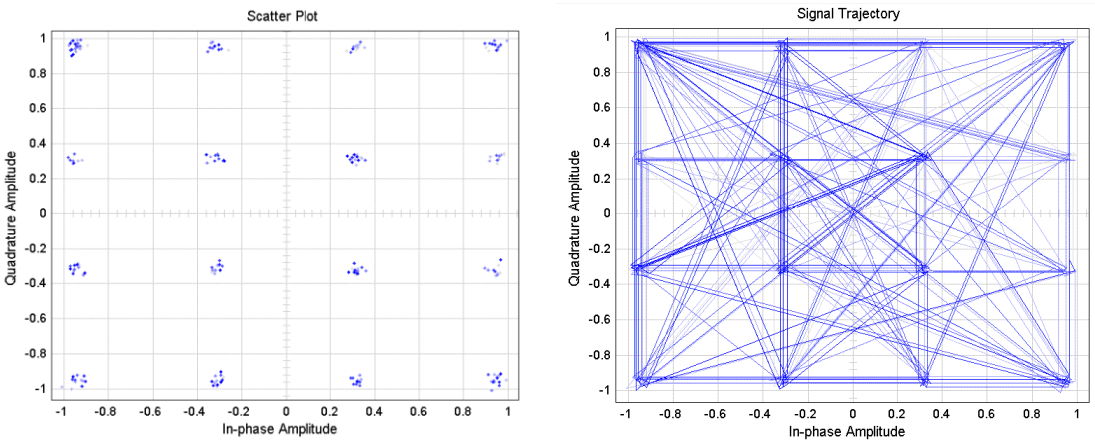


Figura C.45. Diagramas en el tiempo generados por Simulink para 16QAM con $E_b/N_o = 24dB$.

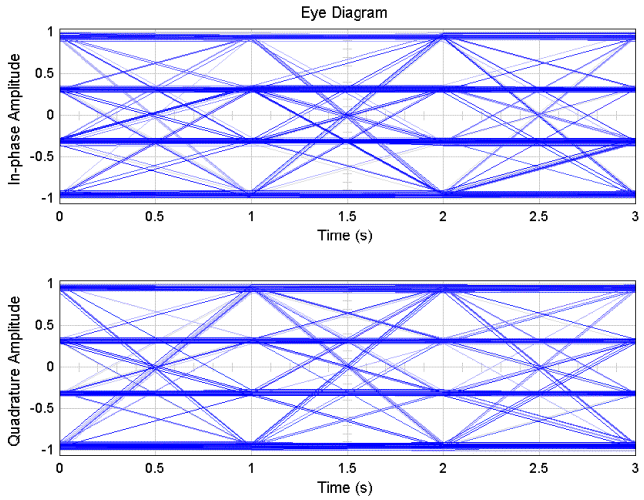


Figura C.46. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

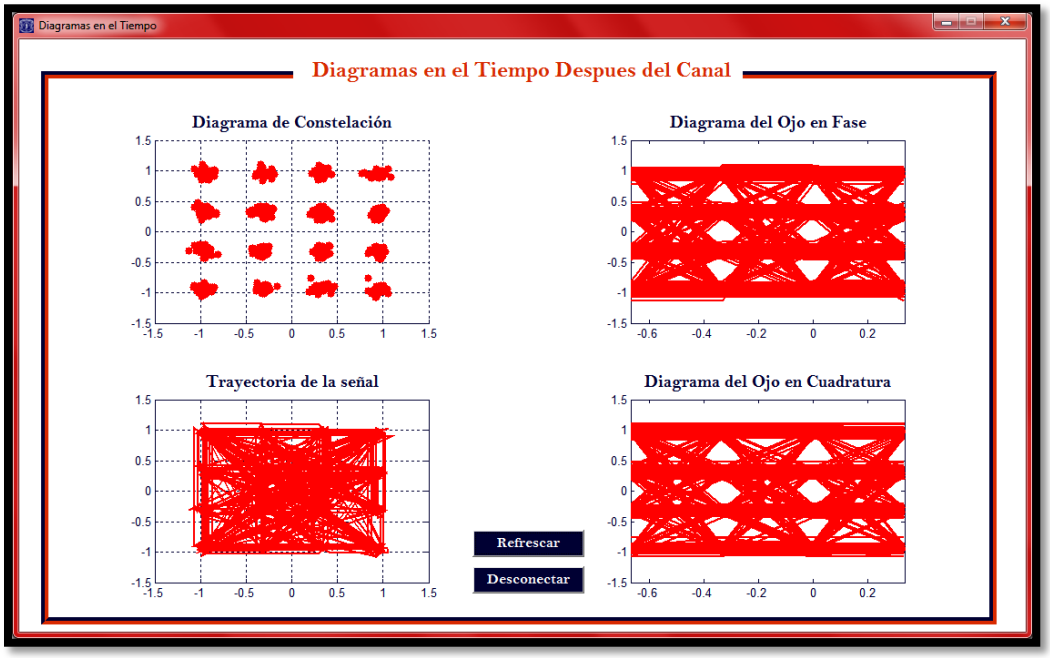


Figura C.47. Diagramas en el tiempo generados por la interfaz gráfica para 16QAM con $E_b/N_o = 17dB$.

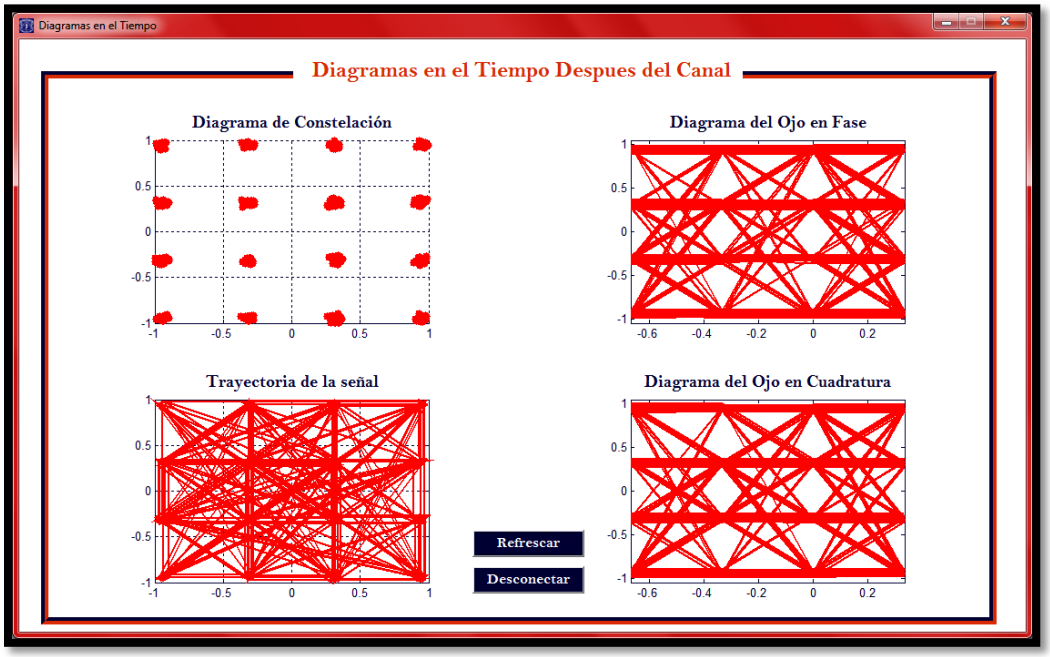


Figura C.48. Diagramas en el tiempo generados por la interfaz gráfica para 16QAM con $E_b/N_o = 24dB$.

C.2.2. Modulación de amplitud en cuadratura con M = 64

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

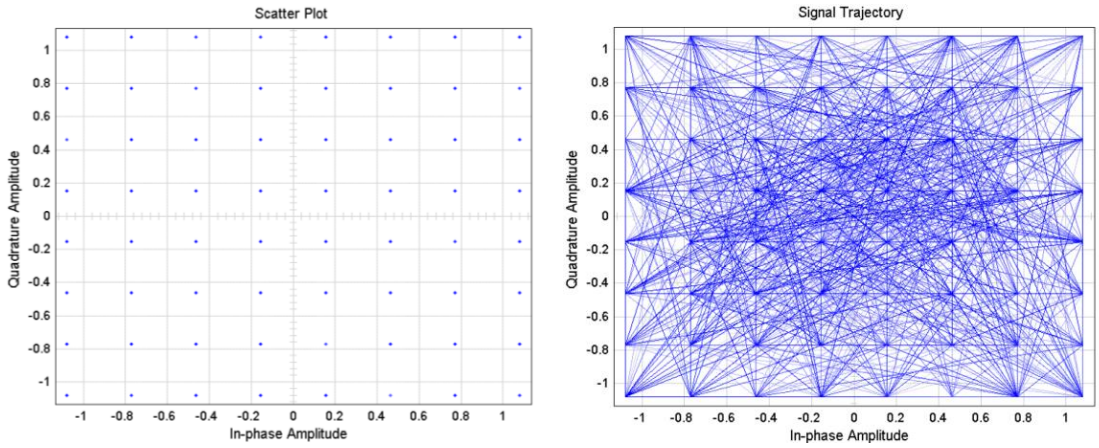


Figura C.49. Diagramas en el tiempo generados por Simulink para 64QAM.

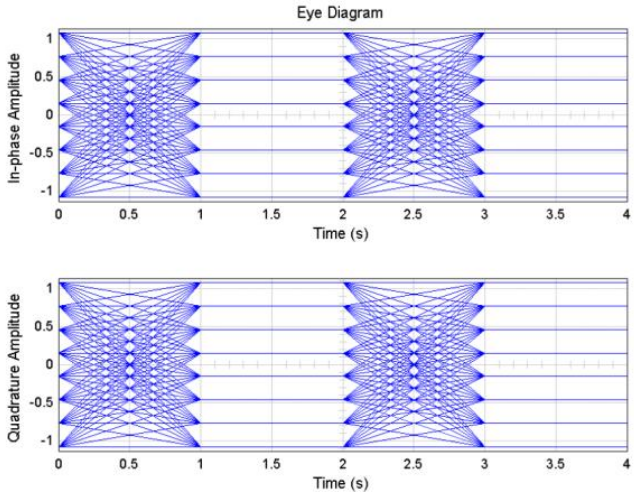


Figura C.50. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

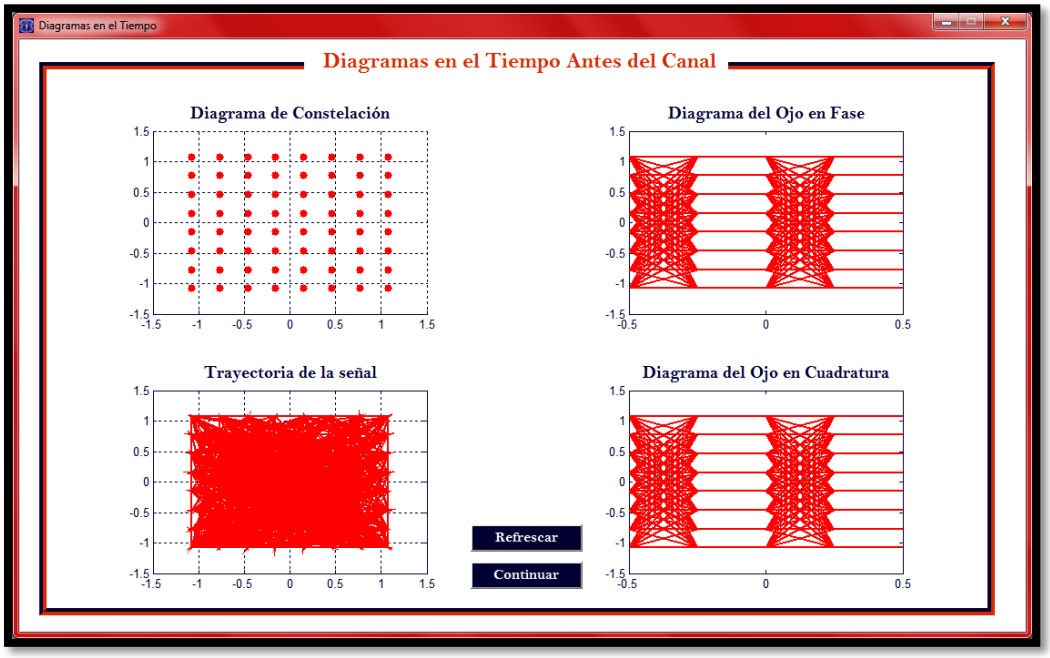


Figura C.51. Diagramas en el tiempo generados por la interfaz gráfica para 64QAM.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

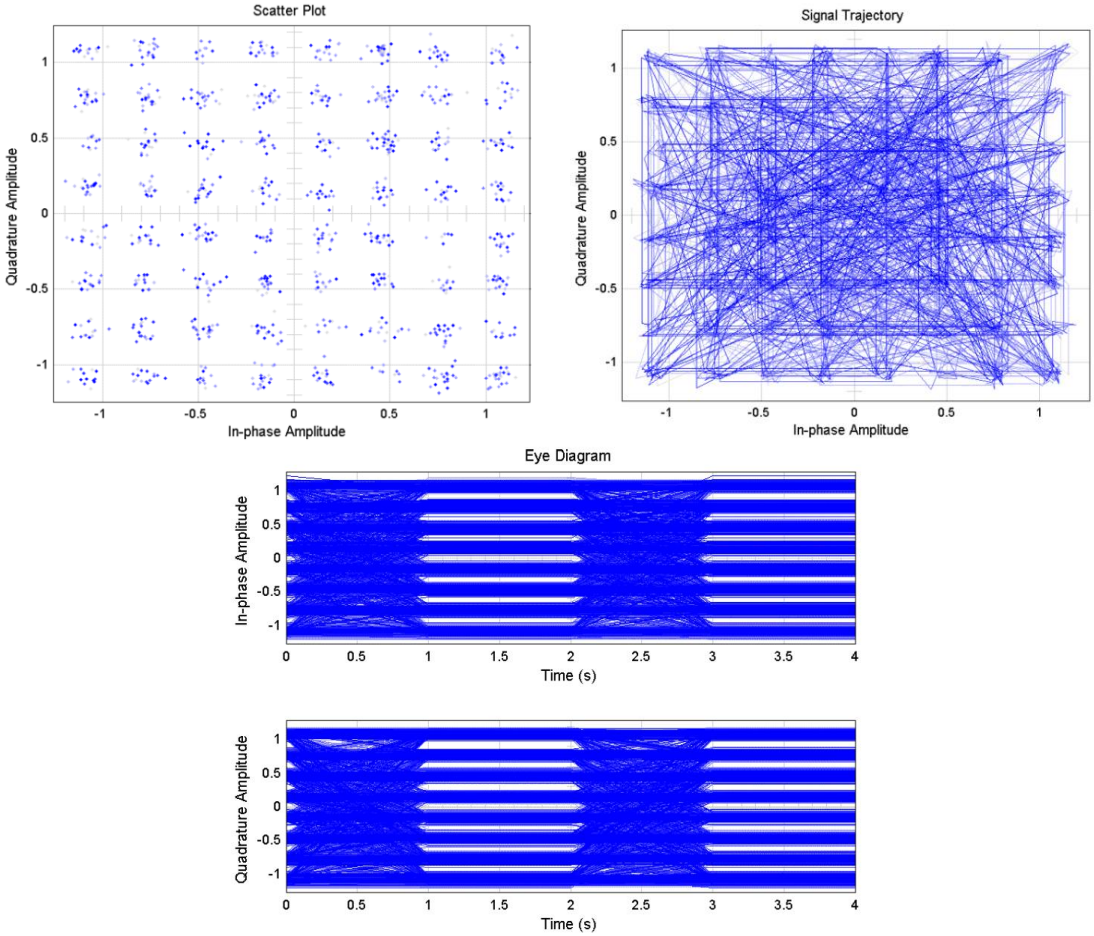


Figura C.52. Diagramas en el tiempo generados por Simulink para 64QAM con $E_b/N_o = 17dB$.

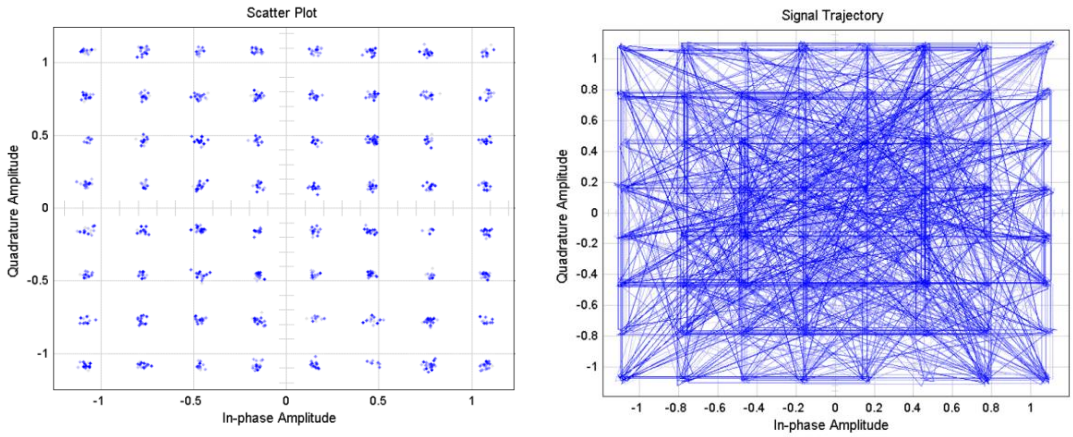


Figura C.53. Diagramas en el tiempo generados por Simulink para 64QAM con $E_b/N_o = 24dB$.

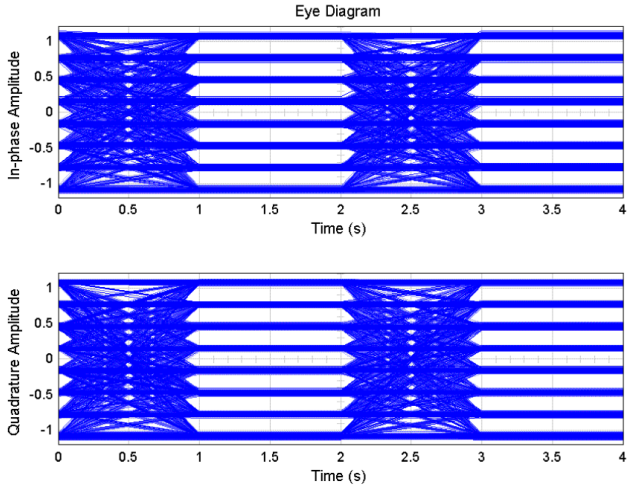


Figura C.54. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

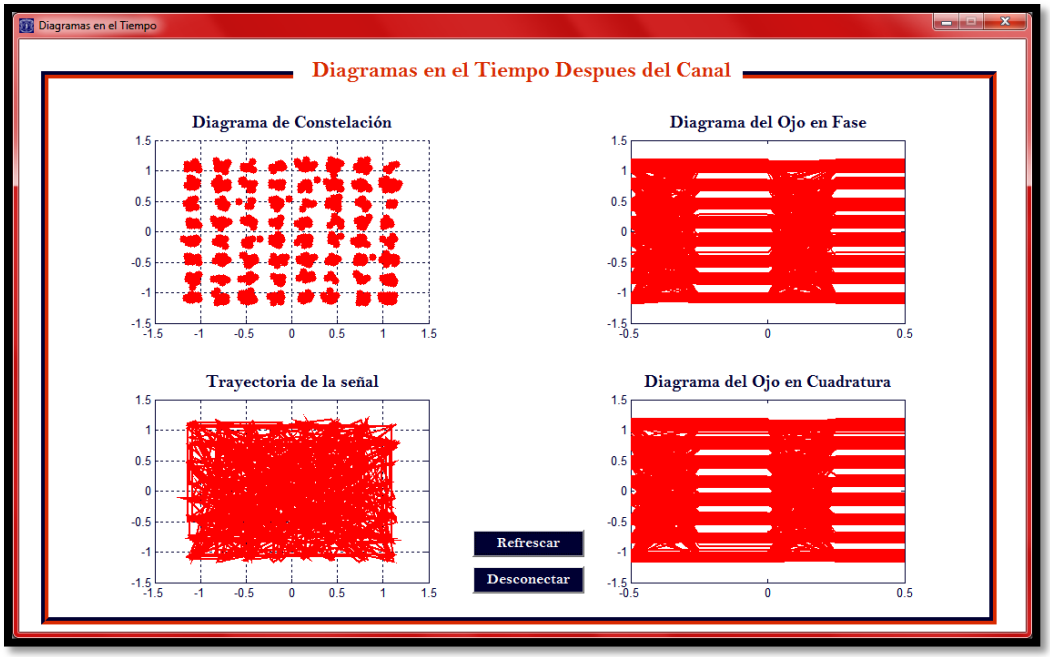


Figura C.55. Diagramas en el tiempo generados por la interfaz gráfica para 64QAM con $E_b/N_o = 17dB$.

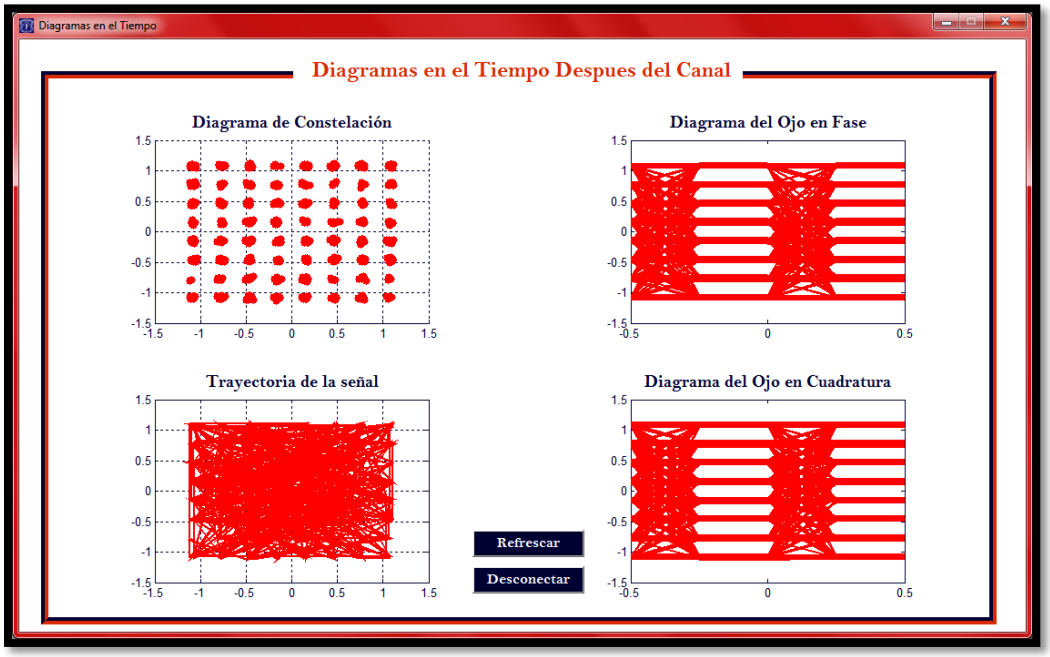


Figura C.56. Diagramas en el tiempo generados por la interfaz gráfica para 64QAM con $E_b/N_o = 24dB$.

C.3. Modulación por Mínimo Desplazamiento de Frecuencia

- **Figuras obtenidas a partir de los bloques de Simulink antes del canal**

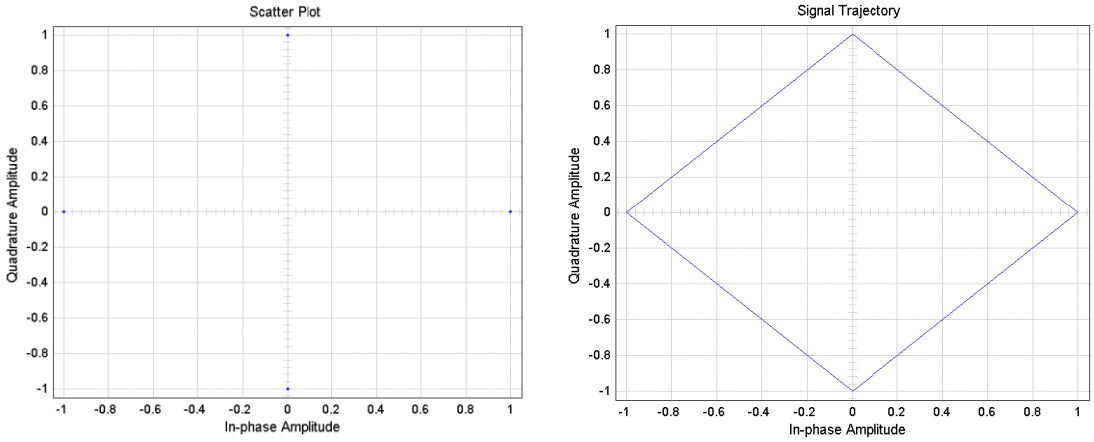


Figura C.57. Diagramas en el tiempo generados por Simulink para MSK.

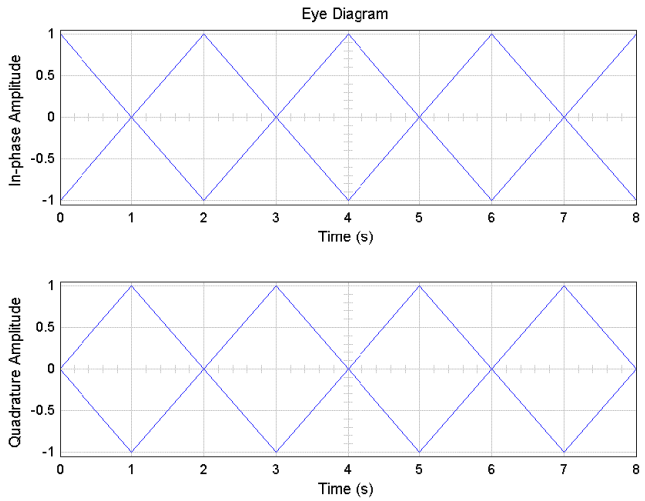


Figura C.58. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica antes del canal**

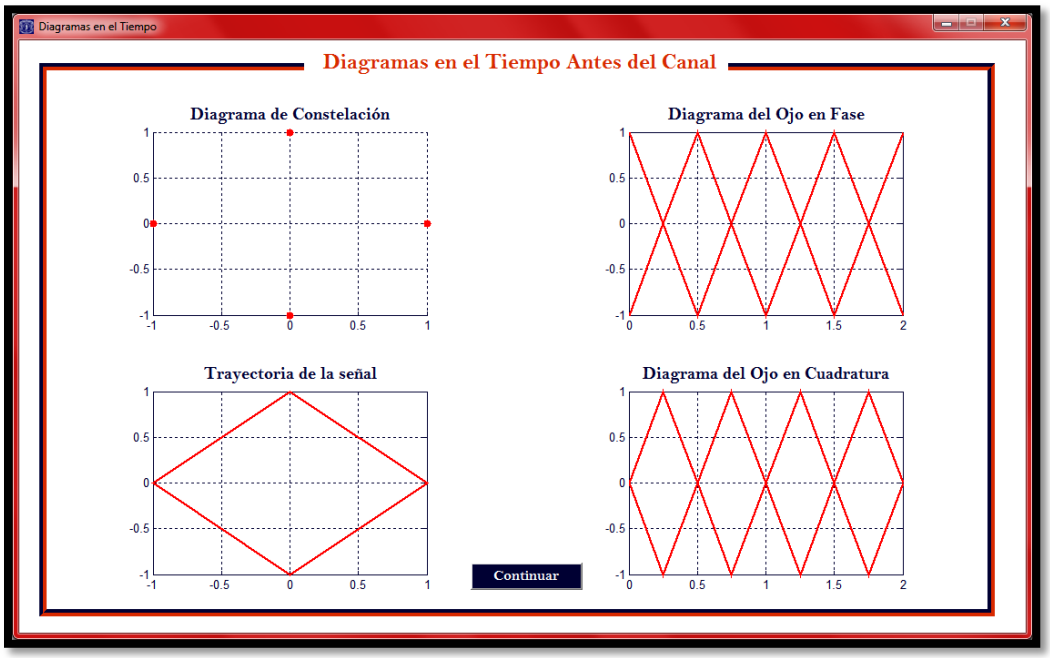


Figura C.59. Diagramas en el tiempo generados por la interfaz gráfica para MSK.

- **Figuras obtenidas a partir de los bloques de Simulink después del canal**

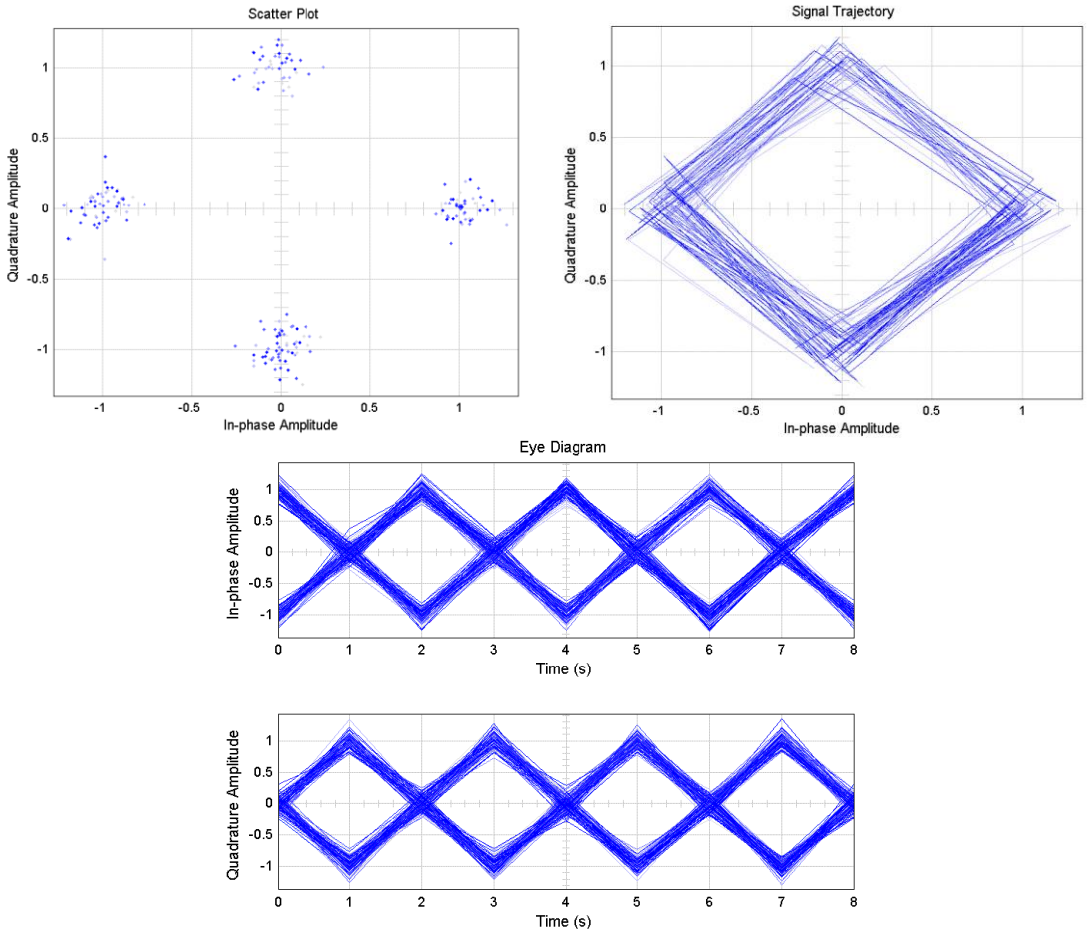


Figura C.60. Diagramas en el tiempo generados por Simulink para 16QAM con $E_b/N_o = 17dB$.

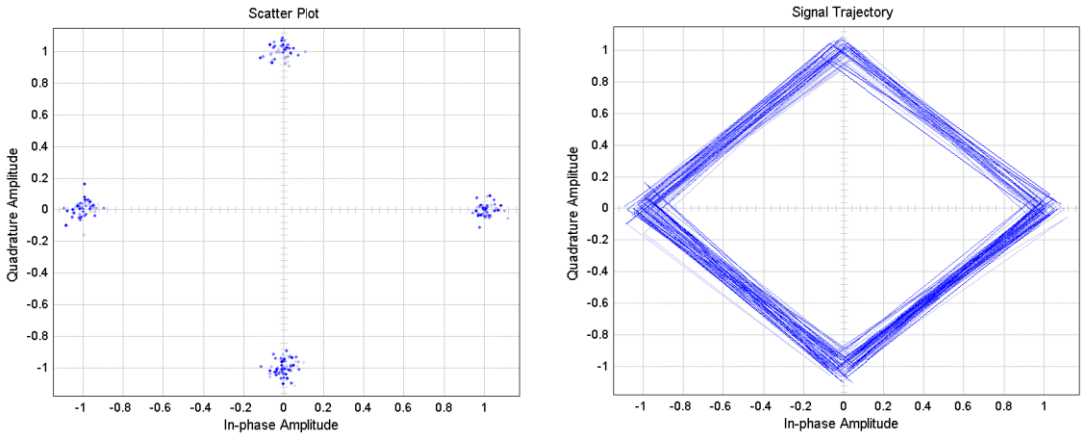


Figura C.61. Diagramas en el tiempo generados por Simulink para 16QAM con $E_b/N_o = 24dB$.

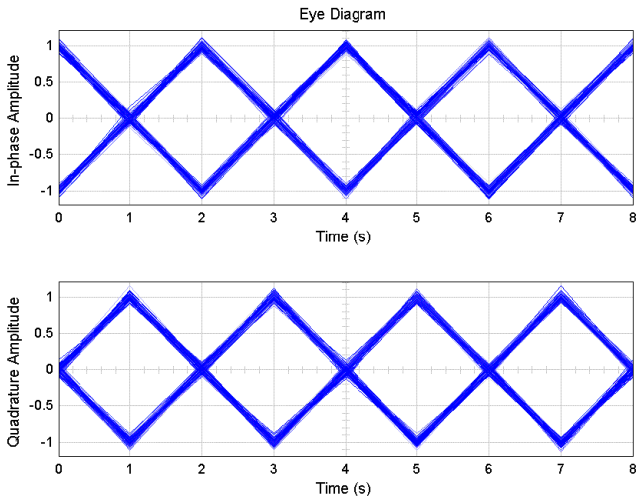


Figura C.62. Continuación.

- **Figuras obtenidas a partir de la Interfaz Gráfica después del canal**

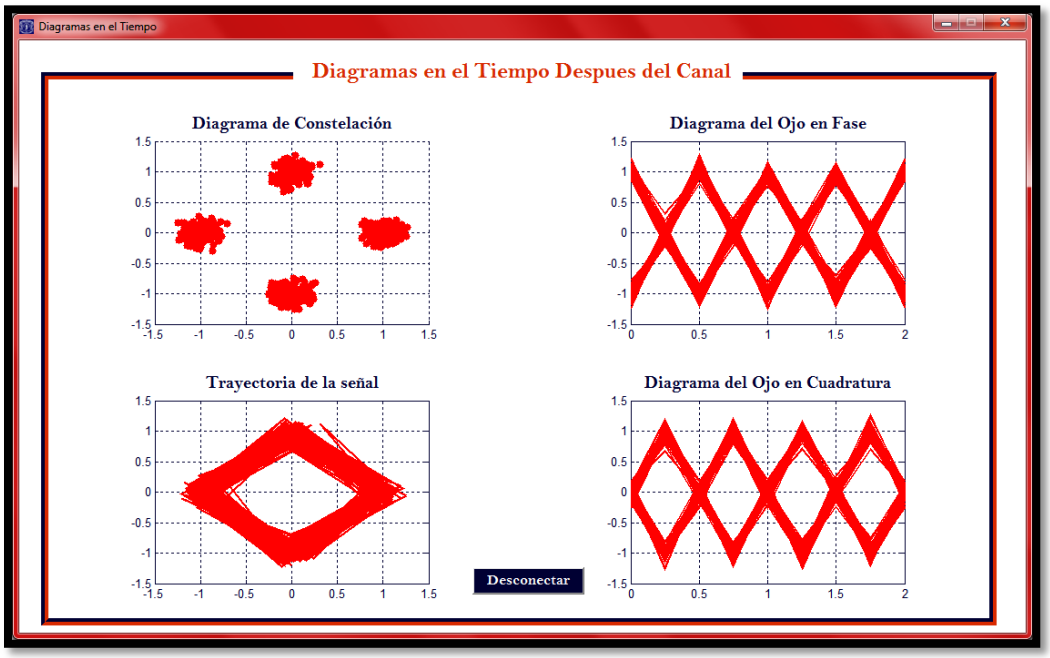


Figura C.63. Diagramas en el tiempo generados por la interfaz gráfica para MSK con $E_b/N_o = 17dB$.

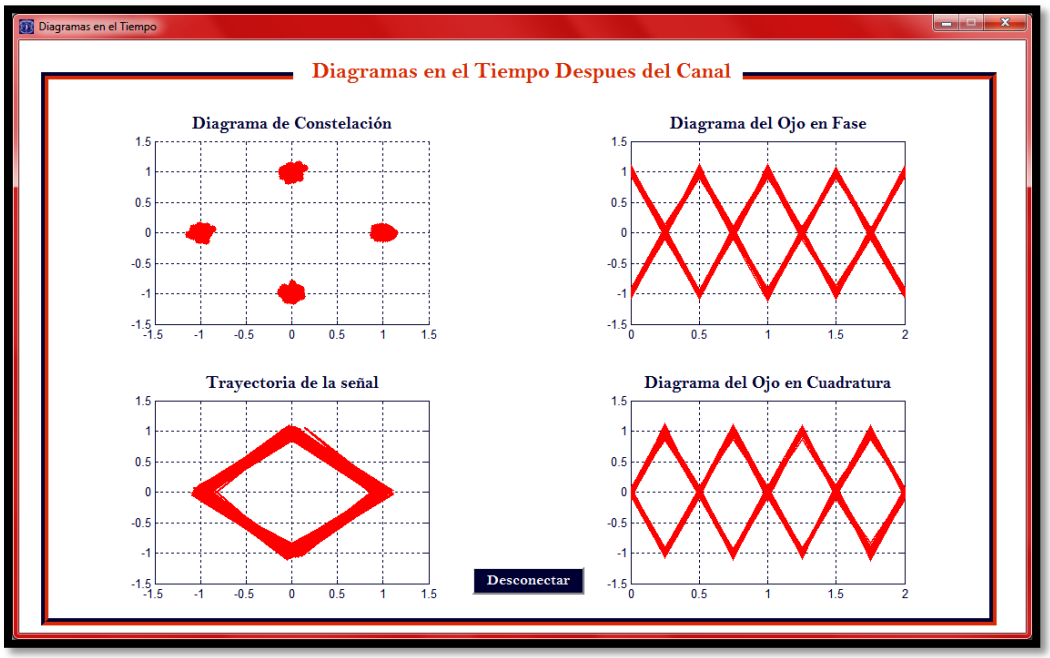


Figura C.64. Diagramas en el tiempo generados por la interfaz gráfica para MSK con $E_b/N_o = 24dB$.

APENDICE D

MANUAL DE USUARIO

D.1. Banco de Trabajo

Para implementar correctamente el actual proyecto son necesarios ciertos elementos específicos para cumplir funciones determinadas, razón por lo cual serán resaltados a continuación.

- FPGA Spartan-3A kit board.
- Cable QPCOM convertidor de serial a USB.
- Computador.

Figura D.1. Banco de trabajo.

D.2. Requerimientos Hardware y Software del Sistema

Antes de proceder a generar o visualizar cualquier diagrama de tiempo en la interfaz gráfica, se debe tener en cuenta los requerimientos mínimos del sistema a nivel hardware y software.

D.2.1. Requerimientos hardware

- Procesador: Pentium Dual-Core @ 1,87 GHz
- Memoria instalada (RAM): 2,00 GB.
- Disco duro: 18 GB de espacio disponible.

D.2.2. Requerimientos software

- Sistema operativo: Windows XP Professional, Windows 7 Professional, soportado para 32-bit y 64-bit.
- ISE Design Suite 14.4 System Edition.
- MATLAB R2011a, R2011b, 2012a and 2012b.

D.3. Conexiones

Con todos los elementos y requerimientos a disposición, se procede con las siguientes conexiones del dispositivo Spartan-3A:

1. Conectar el cable de alimentación del FPGA a la red eléctrica.
2. Conectar el cable de programación de la FPGA a un puerto USB del computador.
3. Por último el cable QPCOM, debe conectarse el extremo serial al puerto DB9 DCE hembra de la Spartan-3A y el otro extremo a cualquiera de los puertos USB del computador tal y como se observa en la Figura D.2.

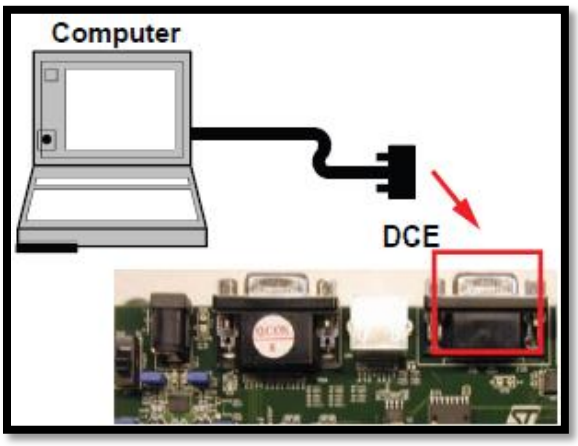


Figura D.2. Conexión al puerto serial del FPGA.

Es importante realizar estos pasos antes de empezar la configuración y programación del dispositivo debido que al momento de esto, se requiere que el computador identifique a el FPGA SPARTAN-3A para continuar con el proceso.

D.4. Procesos de Configuración Software

Una vez diseñado el modelo con la herramienta Simulink con *System Generator* de XILINX®, se debe como primer paso configurar el sistema, lo cual se logra en el menú que se despliega al hacer doble clic en el "token" de *System Generator* (Figura D.3).

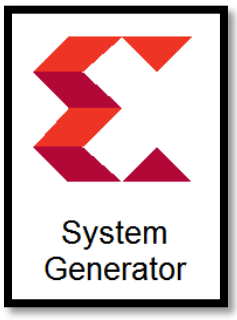
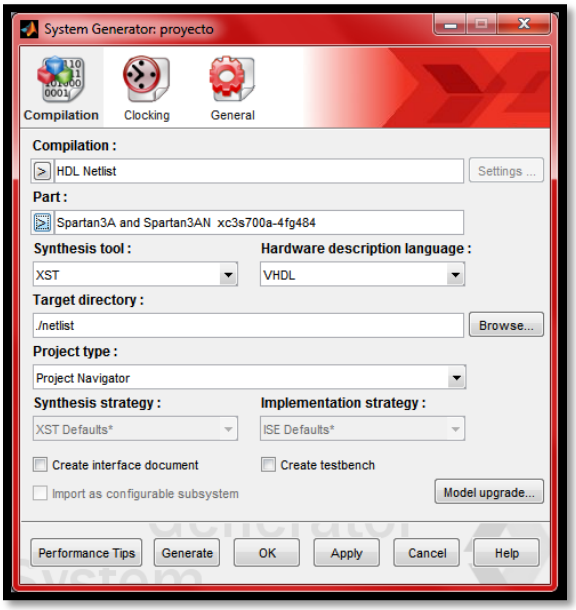


Figura D.3. Token de *System Generator*.

En el menú desplegado (Figura D.5 (a)) se escoge el dispositivo FPGA con el que se está trabajando y por último se genera el sistema presionando el botón "Generate", esta acción puede tomar algunos minutos pero una vez terminado mostrara un mensaje advirtiendo que se ha generado completamente (Figura D.5 (c)).



(a)

Figura D.4. (a) Menú desplegado.

(b) **(c)**

Figura D.5. (b) Proceso de generación del archivo .xise. **(c)** Generación completa y libre de errores.

Al finalizar este proceso, dentro de la carpeta del proyecto se habrá creado otra carpeta llamada “netlist" en la cual se encontrara un archivo de formato .xise, el cual se abre con el programa ISE Design Suite 14.4 desde la opción “Open Project” (Figura D.6).

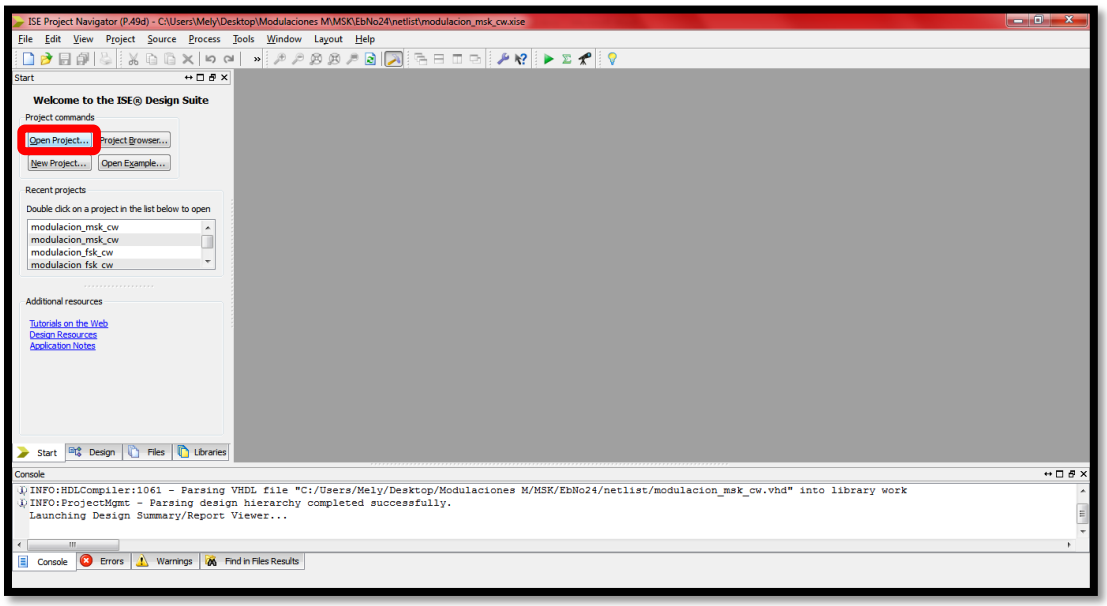


Figura D.6. Ventana de inicio del programa ISE Design Suite 14.4.

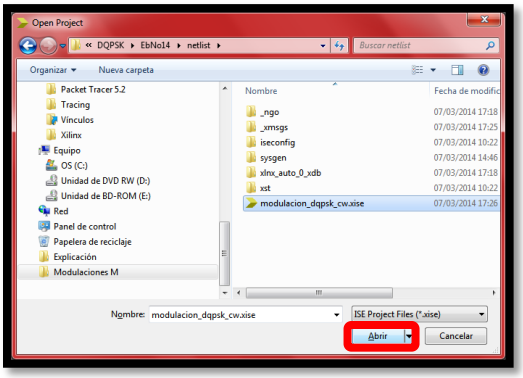


Figura D.7. Ventana desplegada al oprimir Open Project.

La Figura D.8 muestra la ventana que se despliega al abrir el archivo corresponde al diseño del proyecto, la cual permitirá sintetizarlo para ser programado en el FPGA, para esto se hace clic derecho sobre la opción Generate Programming File y luego clic en Rerun All, a partir de este momento el programa sintetiza, traduce y mapea el archivo, produciendo que esta acción tome algún tiempo.

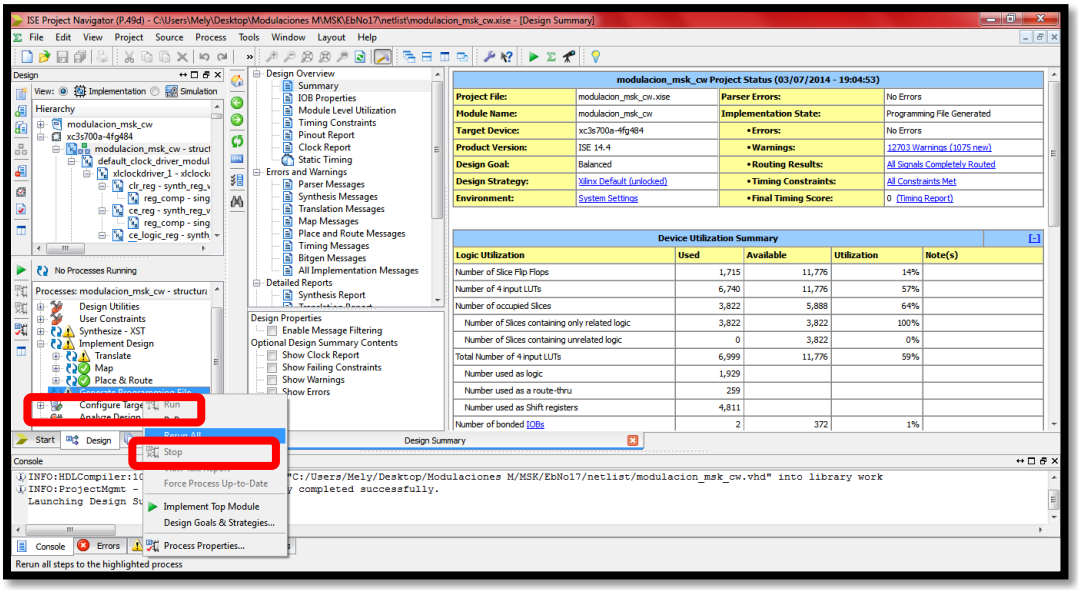


Figura D.8. Proceso para generar el archivo de programación.

Terminado este proceso, se ha creado nuevamente en la carpeta netlist un archivo de formato .bit, necesario para programar el FPGA mediante el programa IMPACT ubicado en la carpeta ISE Design Tools, una vez abierto se da clic sobre el icono que se resalta en la Figura D.9.

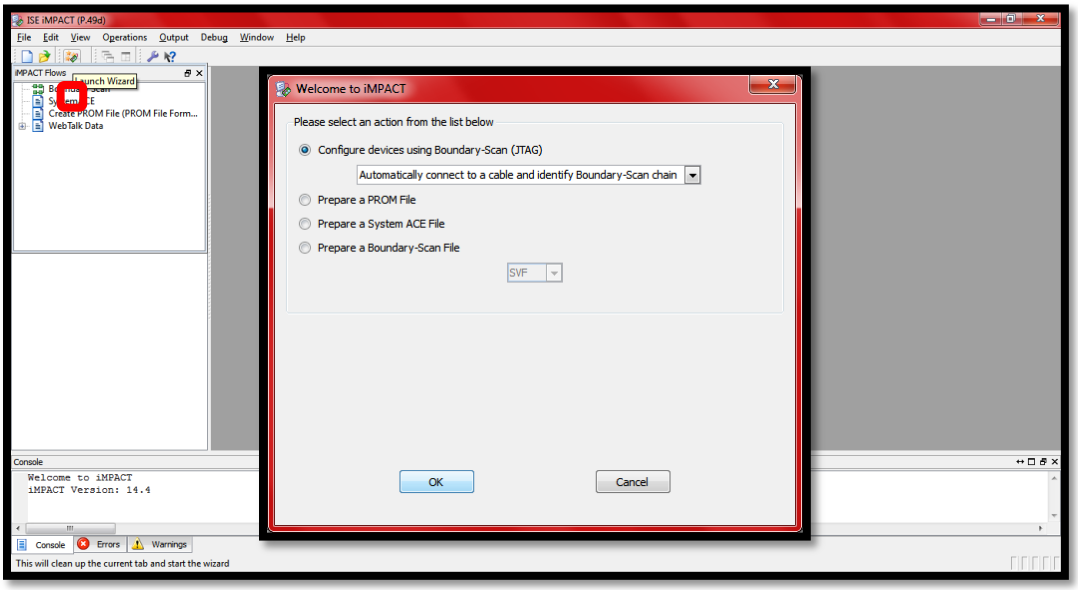


Figura D.9. Ventana principal del programa IMPACT.

A continuación se muestra una ventana de bienvenida la cual se cierra al presionar “OK”, de este modo el programa detectara el FPGA SPARTAN-3A y mostrara las dos memorias con las que cuenta el dispositivo en mención, como se observa en la Figura D.10.

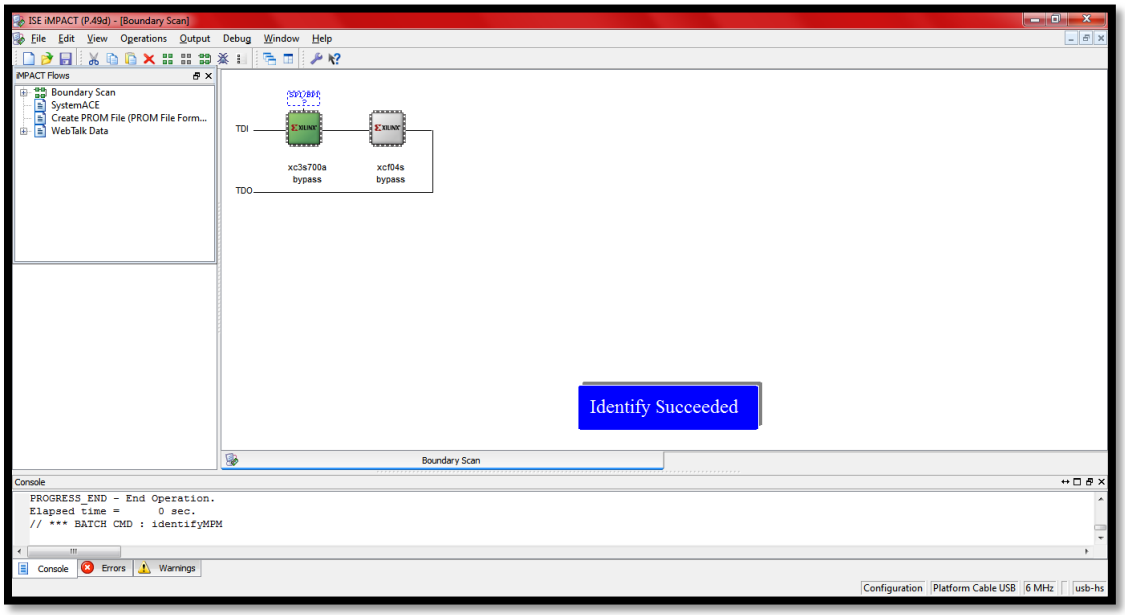


Figura D.10. Identificación del dispositivo satisfactoria.

Sobre la memoria xc3s700a se da doble clic, con lo que se abre una nueva ventana que permitirá cargar el archivo .bit previamente generado (Figura D.11), al terminar esta acción se mostrara una advertencia preguntando si se quiere programar la memoria Flash PROMs, por lo que cabe aclarar que este proyecto no hará uso de esta memoria, así que se da clic en no (Figura D.12).

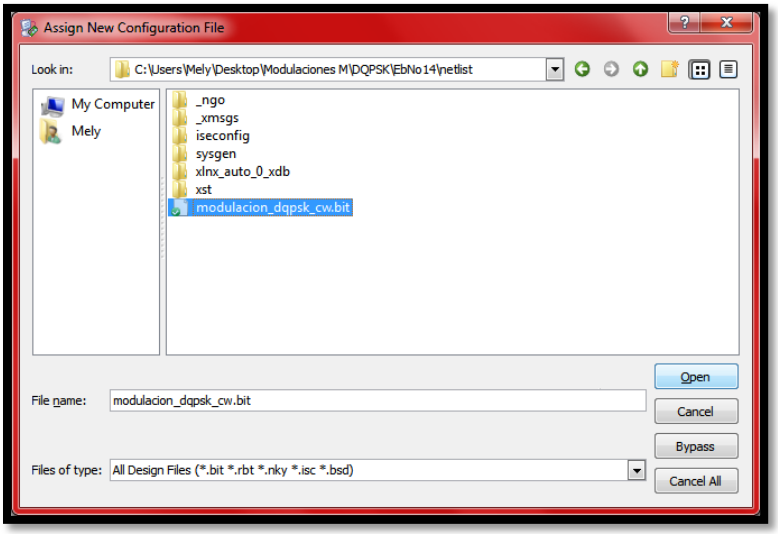


Figura D.11. Asignación del archivo .bit a la memoria seleccionada.

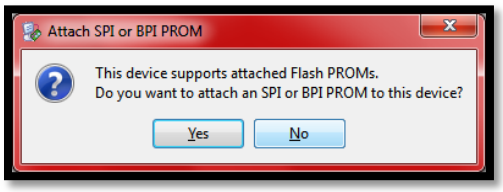


Figura D.12. Ventana emergente.

Una vez el archivo está cargado sobre la memoria se da doble clic sobre esta y se escoge la opción “Program” (Figura D.13), obteniendo como resultado la programación del FPGA y evidenciándose con el mensaje azul Program Succeeded mostrado en la Figura D.14, dando como finalizado el proceso de configuración software.

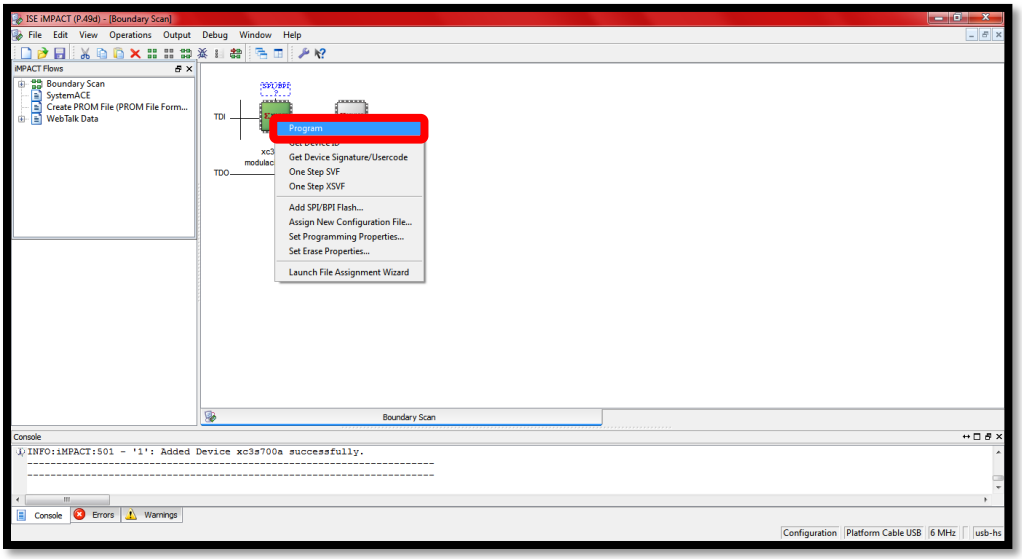


Figura D.13. Programación del FPGA.

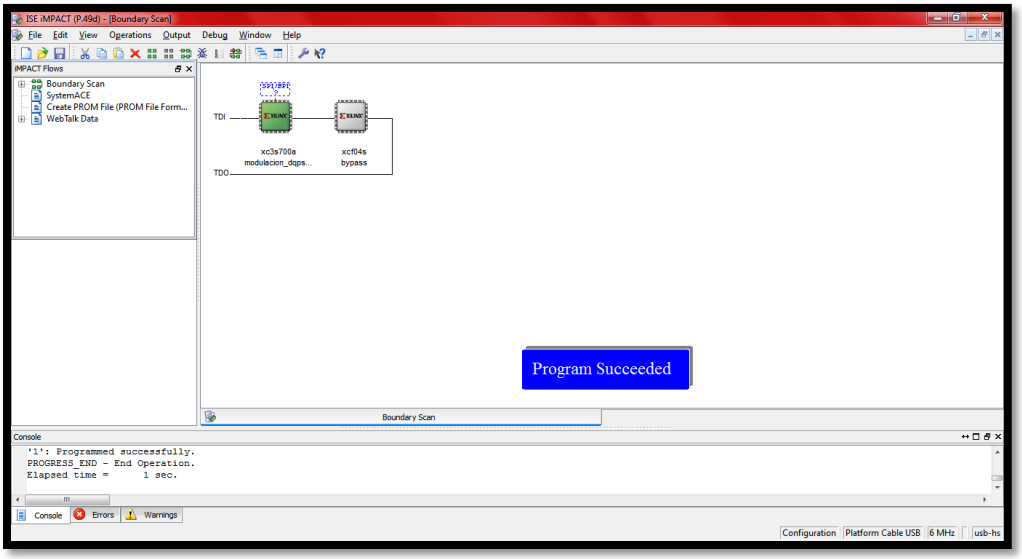


Figura D.14. Programación satisfactoria.

D.5. Manejo y Visualización de la Interfaz Gráfica

Para el acceso a las ventanas de la interfaz gráfica, es necesario realizar una serie de pasos que permitirán iniciar el proceso de visualización.

1. Abrir MATLAB R2012a.
2. En la ventana de comandos escribir guide.
3. Seleccionar la pestaña abrir GUI existente.
4. Buscar la carpeta Rx+ Interfaz.
5. Escoger la modulación que desea visualizar.
6. Abrir el archivo inicio.fig.
7. Oprimir el botón Run.

La interfaz gráfica está conformada por un grupo de 3 ventanas, las cuales serán detalladas a continuación:

D.5.1. Presentación

La ventana de presentación del proyecto (Figura D.15) cuenta con el botón iniciar, el cual al oprimirlo permite desplegar la ventana de diagramas en el tiempo antes del canal.

Figura D.15. Ventana de presentación de la interfaz gráfica.

D.5.2. Diagramas en el tiempo antes y después del canal

Al momento de abrirse esta ventana comienza el proceso de lectura y gráfica de datos, en esta se cuenta con cuatro gráficos que hacen alusión a él diagrama de constelación, la

trayectoria de la señal, el diagrama de ojo en fase y en cuadratura, tal como se muestra en la Figura D.16.

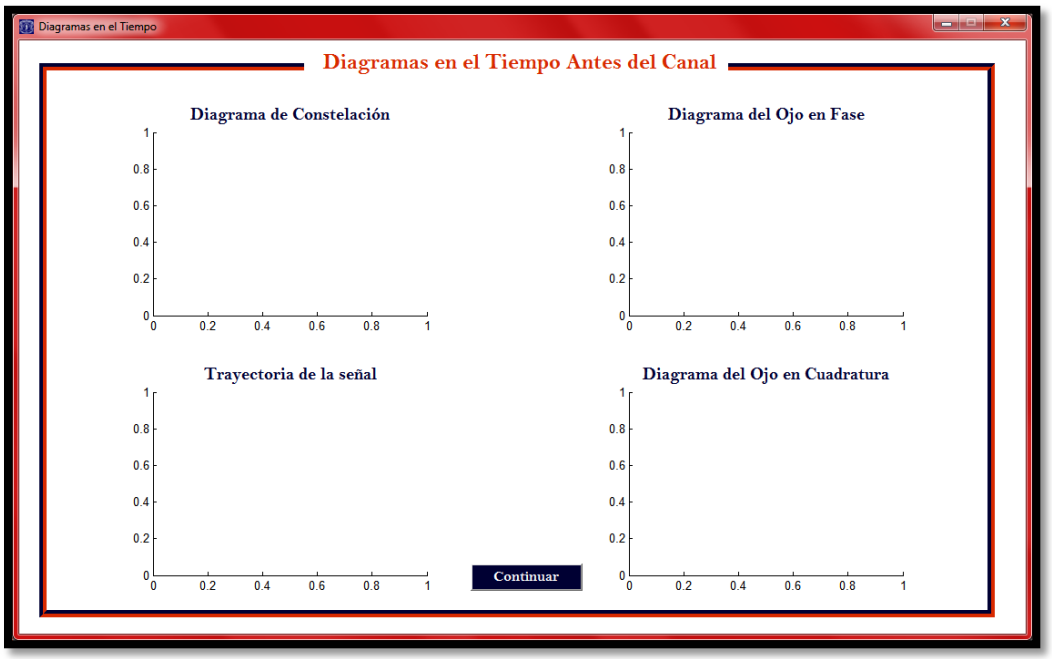


Figura D.16. Ventana para visualizar los diagramas antes del canal.

El botón continuar da acceso a la ventana de la Figura D.17, en la cual se obtendrán los diagramas restantes a partir de los datos provenientes después del canal.

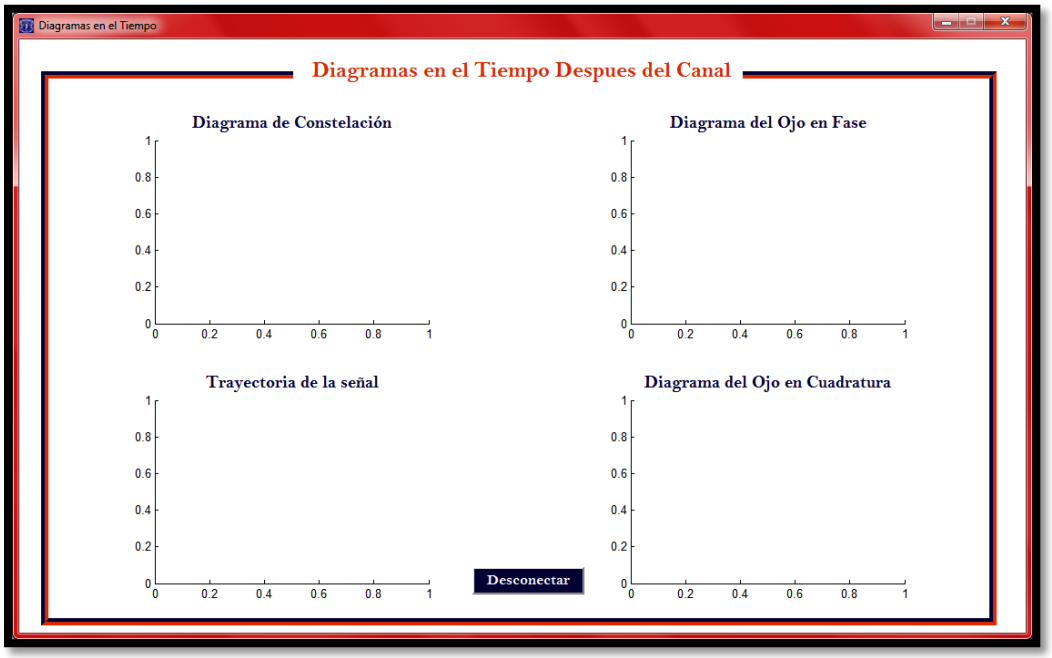


Figura D.17. Ventana para visualizar los diagramas después del canal.

La ventana de la Figura D.17 esta compuesta exactamente por los mismos diagramas que la ventana de la Figura D.16, con la diferencia que los datos recibidos vienen contaminados con ruido, produciendo diagramas más distorsionados. Al ser la última ventana cuenta con el botón desconectar que permite cerrar la interfaz gráfica y terminar el proceso de recepción y visualización.

Los datos en las ventanas de los diagramas antes y después del canal son refrescados automáticamente permitiendo al usuario tener una visualización muy aproximada al tiempo real, logrando amenizar el uso de la interfaz gráfica.