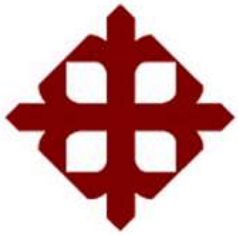


UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE EDUCACIÓN TÉCNICA
PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TEMA:

Diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab

AUTOR:
Barros Tapia, Luis Fernando

Componente práctico del examen complexivo previo a la obtención del grado de **INGENIERO EN TELECOMUNICACIONES**

REVISOR:
M. Sc. Pacheco Bohórquez, Héctor Ignacio

Guayaquil, Ecuador
22 de marzo del 2019

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

CERTIFICACIÓN

Certificamos que el presente **componente práctico del examen complexivo**, fue realizado en su totalidad por **Barros Tapia, Luis Fernando** como requerimiento para la obtención del título de **INGENIERO EN TELECOMUNICACIONES**.

REVISOR

_____

M. Sc. Pacheco Bohórquez, Héctor Ignacio

DIRECTOR DE CARRERA

_____

M. Sc. Heras Sánchez, Miguel Armando

Guayaquil, 22 de marzo del 2019

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

DECLARACIÓN DE RESPONSABILIDAD

Yo, **Barros Tapia, Luis Fernando**

DECLARÓ QUE:

El **componente práctico del examen complexivo, Diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab**, ha sido desarrollado respetando derechos intelectuales de terceros conforme las citas que constan en el documento, cuyas fuentes se incorporan en las referencias o bibliografías. Consecuentemente este trabajo es de mi total autoría.

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

Guayaquil, 22 de marzo del 2019

EL AUTOR

_____

BARROS TAPIA, LUIS FERNANDO

UNIVERSIDAD CATÓLICA
DE SANTIAGO DE GUAYAQUIL
FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO
CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

AUTORIZACIÓN

Yo, **Barros Tapia, Luis Fernando**

Autorizó a la Universidad Católica de Santiago de Guayaquil a la **publicación** en la biblioteca de la institución del **componente práctico del examen complexivo, Diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab** cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y total autoría.

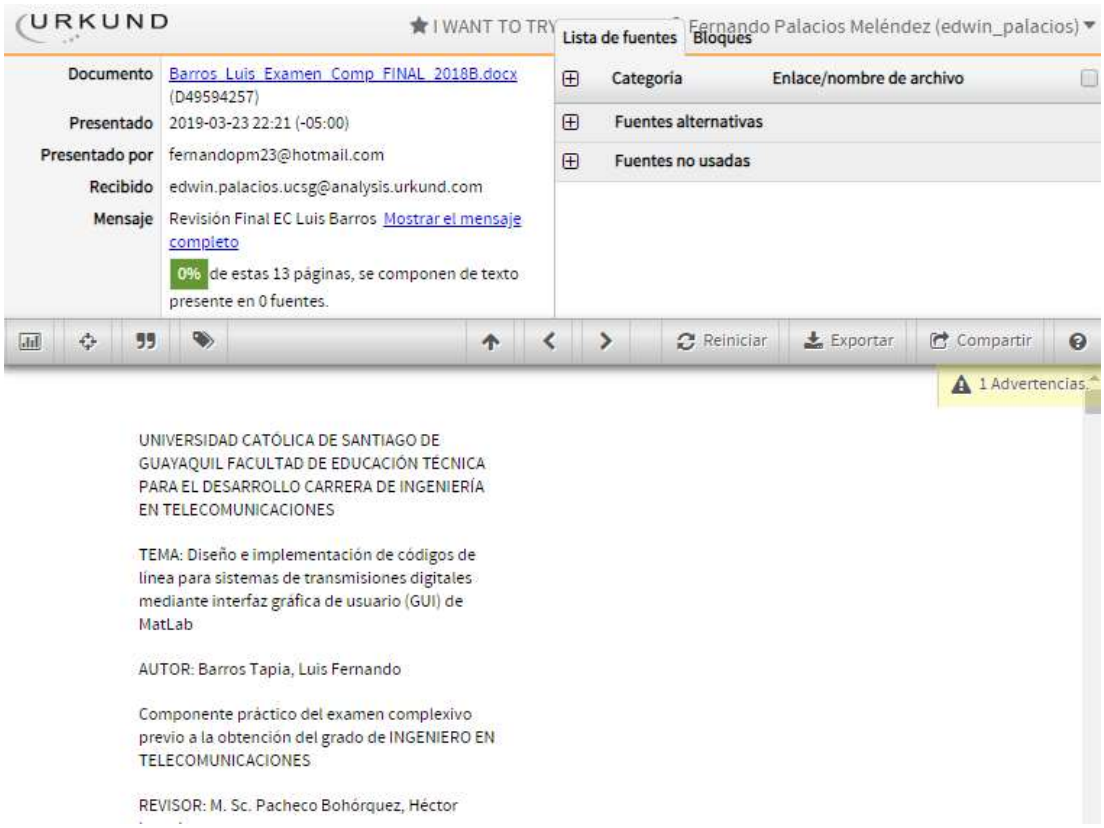
Guayaquil, 22 de marzo del 2019

EL AUTOR

_____

BARROS TAPIA, LUIS FERNANDO

REPORTE DE URKUND



DEDICATORIA

"Una persona buena edifica sus sueños, pero una persona sabia a más de eso contribuye a edificar el de los demás"

El presente trabajo de titulación, me permito con altísimo honor, ponerlo en manos de mi Señora Madre, mis hermanas, mis abuelos y mi familia que han conocido y recorrido de mi mano este trayecto, que más de una vez se volvió insostenible, sin embargo, ustedes jamás me soltaron.

EL AUTOR

BARROS TAPIA, LUIS FERNANDO

AGRADECIMIENTO

"Ser grato es una virtud de almas nobles"

En la recta final de esta etapa formativa quiero evocar mi agradecimiento a quienes han sido participes, en diversas formas, de este arduo y productivo proceso. Como creyente de un ser supremo, es menester agradecer primeramente a Dios, artífice de la vida y regidor de cada paso nuestro, que me ha permitido llegar al culmino de mi carrera con sabiduría y humildad.
A mi querida madre quien con su amor y ejemplo de perseverancia y dedicación ha guiado mi vida convirtiéndose en mi motivación a lo largo de estos años de estudio, de igual forma a mis hermanas, abuelos y familia cercana quienes con sus consejos me han alentado para seguir adelante en mis metas. Gracias Familia por el sacrificio volcado en mí, porque con su esfuerzo yo he tenido el privilegio de poder estudiar esta importante carrera en esta prestigiosa universidad.
A mis amigos en el ámbito civil y policial que pusieron su voto de confianza en mi dándome su apoyo en diferentes momentos para así seguir adelante con mi objetivo final, y de manera muy especial con un sentimiento de enorme gratitud quiero reconocer a mi Teniente Coronel VILLAVICENCIO SALVADOR y mi Mayor CRISTHIAN GONZALES por su incondicionalidad en los permisos necesarios para mi asistencia a mis actividades académicas.
Finalmente y sin restar importancia por ello, a mis estimados docentes por su entrega ilimitada y apertura al brindarme sus conocimientos y experiencias durante mi formación, siendo en esta parte una pieza clave mi tutor el Sr. Ing. Msc. FERNANDO PALACIOS quien de forma activa me ha permitido consolidarme en un futuro ingeniero de mi país.
A todos y cada uno de los mencionados, quiero decirles GRACIAS, sepan que en cada logro, meta y éxito que conseguiré a lo largo de mi vida, ustedes serán parte de ellos, porque son parte ahora mismo de los cimientos de este sueño que hoy se vuelve tangible y real.

EL AUTOR

BARROS TAPIA, LUIS FERNANDO

UNIVERSIDAD CATÓLICA

DE SANTIAGO DE GUAYAQUIL

FACULTAD DE EDUCACIÓN TÉCNICA PARA EL DESARROLLO

CARRERA DE INGENIERÍA EN TELECOMUNICACIONES

TRIBUNAL DE SUSTENTACIÓN

f._____

M. Sc. ROMERO PAZ, MANUEL DE JESUS

DECANO

f._____

M. Sc. HERAS SÁNCHEZ, MIGUEL ARMANDO

DIRECTOR DE CARRERA

f._____

M. Sc. PALACIOS MELÉNDEZ, EDWIN FERNANDO

OPONENTE

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

Capítulo 2:

Capítulo 3:

ÍNDICE DE TABLAS

Capítulo 2:

Resumen

Para el desarrollo del componente práctico del examen complexivo se tuvieron que revisar diferentes plataformas de simulación, se evaluó la funcionalidad de cada uno. Entre las plataformas analizadas se escogió el software MatLab. Posteriormente, se investigaron trabajos relacionados al uso de la codificación de línea y a través de tutoriales y libros pude comprender el uso de MatLab. El presente trabajo consistió en realizar el diseño de una interfaz gráfica de la codificación de línea muy utilizados en sistemas de transmisiones digitales. En el capítulo 1, se describe una breve introducción relacionada al diseño asistido por computadoras (CAD), objetivo general y objetivos específicos del componente práctico. En el capítulo 2, se describen los fundamentos teóricos de las comunicaciones digitales. En el capítulo 3, se realiza el diseño de dos interfaces gráficas, una para el procesamiento de códigos de líneas y la otra para visualizar la generación de la densidad espectral de potencia para cada código de línea.

Palabras claves: DATOS, COMUNICACIONES, CODIFICACIÓN, ESPECTRO, TRANSMISIONES, MATLAB.

CAPÍTULO 1: DESCRIPCIÓN DEL COMPONENTE PRÁCTICO

1.1. Introducción.

Hoy en día, las herramientas computacionales son indispensables en el diseño de aplicaciones de ingeniería, por ejemplo, en aplicaciones de circuitos electrónicos debido al aumento en la complejidad y la necesidad de administrar grandes cantidades de datos relacionados con el diseño. El desarrollo de nuevas metodologías y herramientas se ha convertido en un área estratégica en el desarrollo de nuevas tecnologías, en particular en el desarrollo de herramientas de diseño asistido por computadora *(Computer Aided Design, CAD)*. Las herramientas CAD se pueden entender mejor como sistemas de gestión de información de diseño, junto con la creación de entradas basadas en gráficos y se crean simulaciones de los diseños. Estas simulaciones se pueden utilizar, compartir, publicar, volver a publicar y reutilizar en diferentes formatos, escalas y niveles de detalle.

En aplicaciones de telecomunicaciones, por ejemplo, la señal portadora de información generalmente se convierte a datos digitales en bits binarios y luego los bits se codifican en impulsos eléctricos o formas de onda para transmitir información a través del canal. El procedimiento para elegir un par particular de formas de onda y convertir los datos digitales en formas de onda eléctricas se denomina comúnmente codificación de línea, que se usa ampliamente en varias señales para telecomunicaciones, telemetría y telemando. (Gupta & Singh, 2016)

Es bien sabido que existen dos categorías representativas de códigos de línea: retorno a cero (RZ) y no retorno a cero (NRZ). En señales codificadas de RZ, las formas de onda vuelven a un nivel de voltaje cero en algún punto, normalmente la mitad de un intervalo de bits; esto no ocurre en señales codificadas de NRZ. (Zapata Y. & Jurado P., 2012)

Los esquemas de codificación de línea se clasifican además como códigos unipolares, polares, bipolares y Manchester de acuerdo con la forma de asignación de los niveles de voltaje a los datos binarios. Estos códigos se

aplican selectivamente para maximizar la velocidad de bits para un canal determinado, para minimizar la potencia de transmisión, para recuperar la sincronización de la señal recibida o para reducir el valor de CC en varios sistemas de comunicación digital. (Sklar, 2011)

Dado que cada esquema de codificación de línea tiene su formato de forma de onda único y rasgos característicos en función de la polaridad de pulsos, valor de CC, número de 1 o 0 consecutivos, ocupación espectral y potencia promedio, por lo general se considera más de un criterio al seleccionar un esquema. Hay que tener en cuenta que la característica de potencia promedio dada, lo que es importante en el algoritmo que se realizará en el capítulo 3 cuando se realice el diseño de la interfaz gráfica. (Latha & Pradesh, 2011)

1.2. Objetivo General.

Realizar el diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab.

1.3. Objetivos Específicos.

a. Describir los fundamentos teóricos de sistemas de transmisiones digitales.
b. Diseñar las interfaces gráficas en GUI de Matlab para cada escenario de codificación de línea utilizado en sistemas de comunicaciones digitales.
c. Evaluar la interfaz gráfica de usuario del sistema de códigos de línea propuesto y demostración de la funcionalidad del GUI desarrollado.

CAPÍTULO 2: Fundamentos teóricos de sistemas de transmisión digital.

2.1. Introducción.

La información está presente en la vida cotidiana en muchas formas diferentes: voz, música, video, archivos de datos, páginas web, y se transfiere de un punto a otro de múltiples maneras y con diferentes velocidades según las necesidades. Igualmente, importante, aunque menos tradicional, es la necesidad de preservar la información a lo largo del tiempo, almacenando y recuperando datos de medios magnéticos y ópticos.

La información transmitida se asocia generalmente con una señal eléctrica variable en el tiempo y, por lo tanto, puede representarse mediante una función de tiempo (t). Es oportuno subrayar cómo hoy es posible transformar cualquier información en una señal eléctrica, por lo que la transmisión de información se debe considerar como una señal eléctrica no reductiva ni limitante. En cambio, es particularmente interesante considerar la forma en que se puede representar la información, específicamente, en forma analógica o en forma digital (numérica).

La representación es analógica cuando la información está asociada con un parámetro de la señal que toma valores en un conjunto continuo: por ejemplo, la señal que sale de un micrófono es analógica, ya que puede asumir todos los valores que pertenecen a un intervalo específico de valores. La información es, en cambio, digital o numérica cuando se asocia con parámetros que pueden asumir solo un número finito de valores, que pertenece a un conjunto discreto.

Por ejemplo, si se considera la información contenida en una secuencia de 4 bits y la asociamos con las posibles configuraciones 00-01-10-11 las amplitudes -3A, -A, A, *3A de una forma de onda sinusoidal se obtiene un conjunto de señales numéricas. De manera similar, la información asociada con el lanzamiento de un dado está formada por el conjunto de números del uno al seis, y esta información puede asociarse con una señal constante (t)

cuyos tramos pueden tomar seis valores posibles o a una sinusoide cuya frecuencia puede asumir seis valores posibles.

Los últimos 20 años han visto un reemplazo progresivo de todos los equipos de transmisión analógicos con sistemas de transmisión digital; las razones de esto se pueden resumir en los siguientes puntos:

1) La información analógica se puede convertir en forma numérica mediante operaciones de muestreo y cuantificación. Este proceso de conversión implica una pérdida irreversible de información, de modo que ya no es posible reconstruir la señal inicial s(t) a partir de su versión numérica. Sin embargo, el proceso de conversión de analógico/digital introduce una pérdida de información que puede reducirse o determinarse de otro modo a través de las especificaciones de diseño. Por lo tanto, los sistemas de transmisión numérica se pueden usar para transmitir información originalmente analógica que ha sido sometida a un proceso de conversión de analógico/digital. La red de telefonía fija, inicialmente diseñada y construida como un sistema de comunicación analógica y hoy casi universalmente se convierte en un sistema de transmisión digital; en particular, la señal de voz se convierte en un flujo de datos numérico con una velocidad de bits de 64 Kbps. Incluso la red antigua de telefonía celular GSM se basaba en el uso de modulación numérica con una velocidad de bits de 13 Kbps.
2) Las fuentes analógicas de diferente naturaleza se pueden convertir indistintamente en secuencias de bits, de modo que las diferentes fuentes de información se pueden transmitir simultáneamente (por ejemplo, en la multiplexación por división de tiempo, los bits que pertenecen a diferentes mensajes se alternan antes del proceso de modulación).
3) El hecho de que cada fuente analógica se puede convertir en una secuencia simbólica, junto con el hecho de que los símbolos binarios son las unidades de información elemental utilizadas por las computadoras, significa que muchas de las operaciones realizadas en una cadena de transmisión y recepción de un sistema de

transmisión digital se puede realizar utilizando sistemas de procesamiento de datos. Además, estos últimos siempre se realizan con más frecuencia mediante sistemas de hardware de consumo programables (DSP y FPGA). Como ejemplo se tiene el trabajo realizado por Checa R., Velásquez C., & Álvarez R., (2012).

4) En presencia de ruido, los sistemas de transmisión digital pueden ser mucho más confiables que los sistemas de comunicación analógicos.

2.2. Modelo de un sistema de transmisión digital o numérico.

En la figura 2.1 se ilustra el diagrama de bloques de un sistema de transmisión digital. A continuación, se describen cada uno de los bloques que dispone todo sistema de comunicación digital.

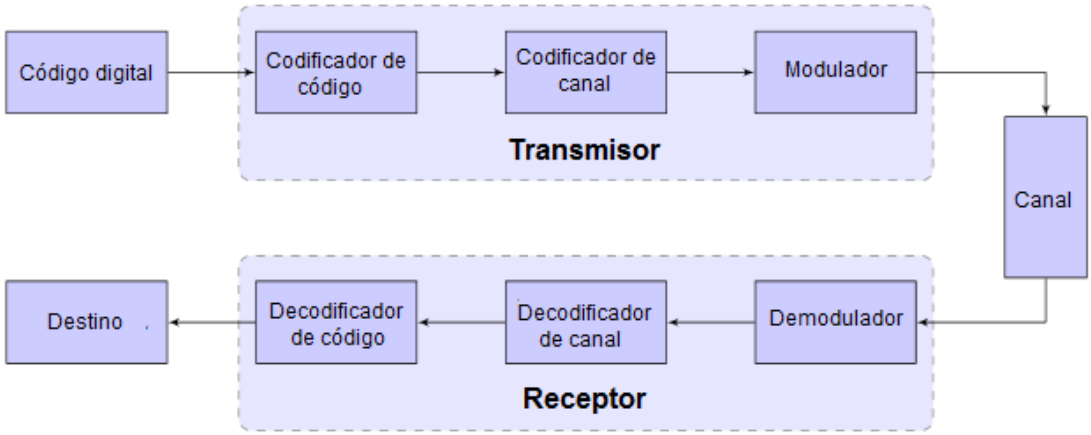


Figura 2. 1: Diagrama de bloques de un sistema de comunicación digital.
Fuente: (Cumbajín V. & Rivadeneira E., 2016)

2.2.1. Código digital:

El código digital es la fuente numérica que representa la fuente de la información a transmitir. Los símbolos emitidos por la fuente pueden representar la información en formados (por ejemplo, un archivo de datos), o pueden provenir de una fuente analógica que se haya convertido en una forma numérica. Los símbolos emitidos por la fuente generalmente pertenecen a un alfabeto binario. (Pérez Vega, 2015)

2.2.2. Codificador de código:

Tiene la tarea de representar la secuencia de símbolos emitidos por la fuente a través de una nueva secuencia de símbolos que tienen la longitud

mínima posible. En otras palabras, el codificador de origen tiene la tarea de eliminar la redundancia presente en la fuente de información, de modo que se pueda transmitir con la máxima eficiencia y sin pérdida de información. Para aclarar las ideas, suponga que la fuente emite cuatro símbolos posibles con diferentes probabilidades: los símbolos (00 01 10 11) se emiten, respectivamente, con probabilidad (1/2 1/4 1/8 1/8).

Evidentemente, es conveniente asociar los símbolos 00 y 01 con secuencias más cortas (0 y 10 por ejemplo) y los símbolos 10 y 11 con secuencias más largas (110 y 111 por ejemplo). Esto permite reducir la longitud promedio de la palabra de código, dada por la suma de las longitudes de las palabras de código ponderadas por su probabilidad de ser emitida, que en el primer caso es igual a 2×0.5 + 2×0.25 + 2×0.125 + 2×0.125 = 2 y en el segundo caso a 1×0.5 + 2×0.25 + 3×0.125 + 3×0.125 = 1.75 (compresión sin pérdida).

Cuando es posible tolerar una pérdida de información, se habla de compresión con distorsión o pérdida. Este tipo de compresión permite obtener un mayor nivel de compactación en el caso en el que el usuario final no puede distinguir o puede aceptar un cierto nivel de degradación de la información. Esto ocurre comúnmente en el caso de fuentes musicales o fuentes de video. (Dalwadi, Goradiya, Solanki, & Holia, 2011)

2.2.3. Codificador de canal:

El codificador de canal agrega a la secuencia de bits de salida el codificador de códigos de bits de redundancia para hacer que la información sea menos vulnerable a los errores que pueden ocurrir durante la fase de recepción de información llevada a cabo en el destino. Un ejemplo de codificación de canal ocurre con la simple adición de bits de paridad. Agregar bits de paridad de tal manera que las secuencias de bits de longitud adecuada tengan un número de valores “1”, ayuda a identificar las secuencias de bits que se ven afectadas por errores debidos a la transmisión de la señal en el canal. (Torres Salamea, 2014)

Si bien puede parecer contradictorio eliminar la redundancia a través de la codificación de código de bits y luego agregarla a través de la codificación del canal, se debe tener en cuenta que el codificador agrega una redundancia conocida al receptor, que puede usar este conocimiento para detectar errores y, en muchos casos, corregirlos. Finalmente, el conjunto de bloques "código", "codificador de código" y "codificador de canal" constituye un código binario equivalente, tal como se ilustra en la figura 2.2.

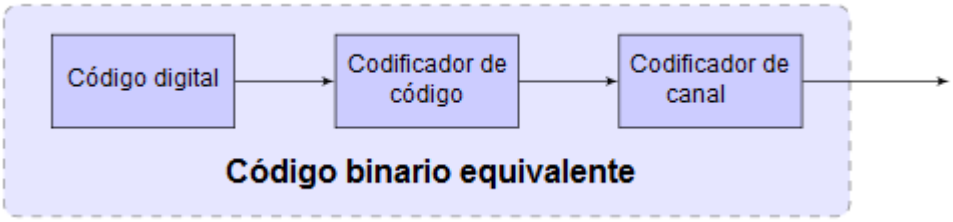


Figura 2. 2: Diagrama de bloques del código binario equivalente.
Elaborado por: Autor.

2.2.4. Modulador

En su forma más simple, el modulador es un dispositivo que genera una secuencia de formas de onda con cadencia T, que se asocia a cada una de las posibles secuencias $M = 2^k$, de k símbolos binarios con M formas de onda posibles de duración T. El modulador está definido por los siguientes parámetros:

- ✓ El intervalo del símbolo (Ts), es el tiempo que transcurre entre la emisión de un símbolo binario y el siguiente. La inversa de (Ts) es la tasa de símbolos (symbol-rate) indicada por Rs, que mide el número de símbolos emitidos en la unidad de tiempo; medición en símbolos por segundo o incluso en baudios. Para k=1 se obtiene la simple emisión de símbolos binarios. El tiempo entre la emisión de dos bits se denomina intervalo de bits y se indica con Tb, mientras que su inversa y comúnmente llamada velocidad de bits se indica con Rb. La velocidad de bits expresa el número de bits por segundo emitidos por la fuente.
- ✓ La cardinalidad M, es un número entero, generalmente una potencia de 2, $M=2^k$. Si M=2 el modulador se llama binario, de lo contrario sería un modulador M-ario.

- ✓ Un conjunto M de formas de onda S = {$s_1(t)$, $s_2(t)$, ..., $s_M(t)$}. Se asumirá que las formas de onda tienen una duración limitada, es decir, no son nulas solo porque pertenecen al intervalo [0, T], y que son de suma cuadrada, o son señales de energía.
- ✓ Una función biunívoca que se asocia a secuencias binarias de longitud $k = \log_2 M$ de una de las formas de onda M disponibles para el modulador.

Para que el sistema de transmisión funcione continuamente, es necesario que la duración T de las formas de onda del modulador (generalmente indicado por el término de intervalo de señalización) no sea mayor que el tiempo que el codificador de canal tarda en producir los símbolos binarios. De esta manera, cuando en la salida del codificador de canal hay una secuencia de k símbolos binarios, el modulador envía al canal una de las formas de onda disponibles, y esta transmisión termina antes de que una nueva secuencia esté nuevamente disponible.

Cabe señalar que entre el modulador y el canal generalmente hay bloques adicionales, como por ejemplo amplificadores, convertidores de radiofrecuencia y, en el caso de transmisión en el canal de radio, una antena transmisora. A continuación, se dará por sentada la presencia de dichos dispositivos cuya definición y diseño no se tratarán en el presente documento. En su lugar, debe mencionarse que el modulador numérico tiene una función de adaptación de la forma de onda generada al canal de transmisión; esta operación se realiza modificando los parámetros de una señal sinusoidal, llamada portadora.

Existen numerosas razones que hacen que la modulación por medio de un portador sea oportuna o necesaria:

1) Para irradiar las señales de manera eficiente, es necesario que las antenas tengan dimensiones comparables con la longitud de onda de la radiación a transmitir; por lo tanto, para señales de paso bajo (por ejemplo, la señal de voz cuya banda oscila entre 300 Hz y 3400 Hz), se requerirán antenas de dimensiones que no sean físicamente

factibles. Con la modulación, el espectro de la señal de información se traduce alrededor de la frecuencia portadora; la operación de modulación, por lo tanto, permite el uso de antenas eficientes y físicamente viables.

2) El ancho de banda de los dispositivos utilizados en un sistema de comunicación (amplificadores, filtros, canal de comunicación, etc.) debe contener la banda de la señal a procesar. Por otro lado, la implementación práctica de un sistema de amplificación requiere que sea de banda estrecha. Por lo tanto, para transmitir en bandas anchas sin violar la hipótesis de banda estrecha del sistema, es necesario que la frecuencia portadora fc sea suficientemente alta.
3) La modulación permite compartir el canal entre varios usuarios; diferentes estaciones de radio pueden, por ejemplo, cubrir la misma área geográfica si transmiten utilizando diferentes partes del espectro electromagnético; las señales se pueden separar en la recepción con la ayuda de filtros especiales. Más en general, las modulaciones analógicas y digitales permiten transmitir en el mismo canal (y posiblemente con el mismo transmisor) más señales simultáneamente (si se usa el mismo transmisor, se construye una señal múltiplex).

2.2.5. Canal de transmisión.

El canal de transmisión es el medio físico que realiza la conexión entre la fuente y el destino. Se puede distinguir entre los canales de tipo cableado (como la línea de dos hilos o trenzada, el cable coaxial, la guía de onda y la fibra óptica) y los canales inalámbricos (atmósfera o espacio libre). En el caso del canal inalámbrico, la señal eléctrica se convierte en una radiación electromagnética de alta frecuencia mediante una antena transmisora y otra antena, en la recepción, realiza la operación inversa.

Las dos antenas generalmente se consideran parte integral del canal. La mayoría de los canales de transmisión se pueden usar en rangos de frecuencia asignados previamente. La tabla 2.1 muestra algunos ejemplos de canales que llevan a cabo la propagación guiada de ondas electromagnéticas

y las frecuencias a las que se usan comúnmente. Hay que tener en cuenta que, para las fibras ópticas, es habitual referirse a longitudes de onda en lugar de frecuencias (en el espacio libre, una longitud de onda de 3 μm corresponde a una frecuencia de 100 THz = 10^{14} Hz).

Tabla 2. 1: Medios de transmisión que realizan la propagación guiada de ondas electromagnéticas y rangos de frecuencias.

Medio de transmisión	Banda
Cable trenzado	1 – 300 kHz
Cable coaxial	300 kHz – 1 GHz
Guía de onda	1 – 300 GHz
Fibra óptica	0.6 – 1.6 um

Elaborado por: Autor

2.2.6. Demodulador

El demodulador procesa la señal $r(t)$ del receptor para identificar cuál de las formas de onda M se ha transmitido en el canal. Incluso en el caso del canal AWGN, la adición de ruido térmico cambia inesperadamente la señal transmitida y es razonable esperar que el demodulador se vea afectado por un cierto número de errores. Los errores corresponden a decisiones erróneas, en las que el demodulador decide por una señal $s_j(t)$, diferente de la que realmente se transmite. El problema de identificar el "demodulador óptimo", donde el criterio de optimalidad es la minimización de la probabilidad de error.

2.2.7. Decodificador de canal.

Debido a los errores que se introducirán inevitablemente en el proceso de demodulación, la secuencia de bits que entran en el decodificador de canal puede diferir de la producida por el codificador de canal. La tarea del decodificador de canal es utilizar la redundancia introducida por el codificador de canal para identificar y corregir parte de los errores producidos en la fase de decisión.

2.2.8. Decodificador de código.

El decodificador de origen realiza la operación inversa a la realizada por el codificador de origen. A partir de los datos comprimidos, reintroduce la redundancia estadística para devolver el significado semántico a los datos y

luego hacerlos utilizables para el destinatario. Tenga en cuenta que la cascada de bloques de modulador, canal y demodulador, como se muestra en la figura 2.3, es un sistema que acepta símbolos binarios en la entrada y produce símbolos binarios en la salida que, con cierta probabilidad de error, pueden ser diferentes de símbolos de entrada correspondientes.

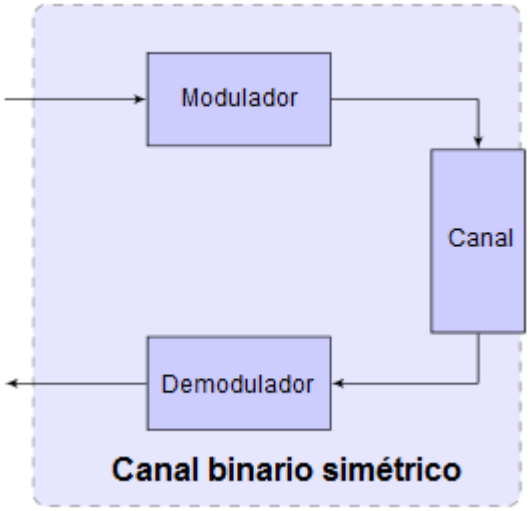


Figura 2. 3: Diagrama de bloque del canal binario simétrico.
Elaborado por: Autor.

Capítulo 3: Diseño e Implementación de la Interfaz Gráfica de Usuario

3.1. Introducción a la interfaz gráfica de usuario – GUI.

GUI es una interfaz gráfica de usuario, en otras palabras, es una visualización gráfica en una o más ventanas, las mismas disponen de controles, conocidos como componentes, lo que permite a los usuarios realizar tareas interactivas. El usuario de la interfaz gráfica de usuario no tiene que crear un script o escribir comandos en la línea de comandos para realizar las tareas. A diferencia de los programas de codificación para realizar las tareas, el usuario de una interfaz gráfica de usuario no necesita entender los detalles de cómo se realizan las tareas. Los componentes de los GUIs incluyen menús (pantalla principal), barras de herramientas, botones, botones de opción, cuadros de lista y deslizadores, sólo para nombrar unos pocos.

Los GUIs se desarrollan utilizando la plataforma MatLab, en la cual permite realizar cualquier clase de cálculo, leer y escribir archivos de datos, así como comunicarse con otras interfaces gráficas de usuario, y mostrar datos como tablas o como parcelas. En la figura 3.1 se ilustra una interfaz gráfica de usuario simple que los usuarios (estudiantes, docentes, investigadores, etc.) pueden construir fácilmente.

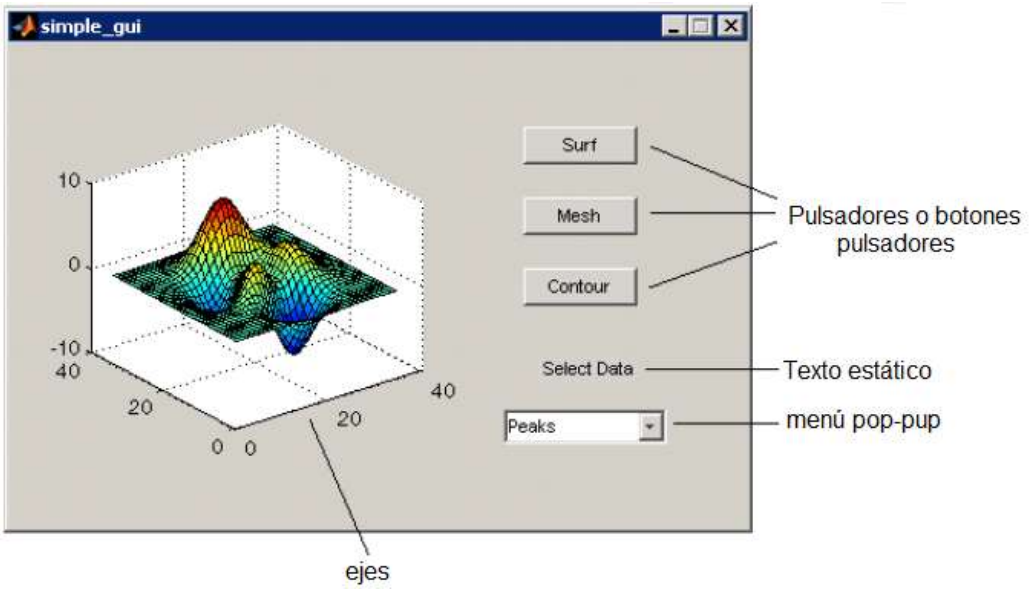


Figura 3. 1: Aplicación realizada en el GUI de MatLab.
Fuente:

La interfaz gráfica de usuario contiene:

a. Un componente para los ejes ya sea 2-D o 3-D.
b. Un menú pop-up, en la cual lista tres conjuntos de datos que corresponden a las funciones de MATLAB.
c. Un componente de texto estático para etiquetar el menú pop-up.
d. Tres botones que proporcionan diferentes tipos de parcelas: superficie, malla, y nivel de curvas.

3.2. Creación de interfaz GUI sencilla.

En este apartado, se mostrará el desarrollo para la crear interfaces GUIs, tal como se observó la figura 3.1. Es decir, que se guiará en el proceso de creación de GUIs. Si se prefiere ver y ejecutar el código que se ha creado la GUI, establezca una carpeta a uno a los que tiene acceso de escritura. Copiamos el código de ejemplo y abrimos en el editor el siguiente comando en MATLAB:

```
copyfile(fullfile(docroot, 'techdoc','creating_guis',...
  'examples','simple_gui2*.*')), fileattrib('simple_gui2*.*', '+w');
edit simple_gui2.m
```

3.2.1. Creación de archivos de código de programación de GUIs.

Crear un archivo de función (a diferencia de archivos script, que contiene una secuencia de comandos de MatLab, pero no define funciones):

a. En MATLAB, escribir edit.
b. Escribir la siguiente instrucción en la primera línea del editor:

```
function simple_gui2
```

c. A raíz de la declaración de función, escriba estos comentarios, que termina con una línea en blanco.

```
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.
(Leave a blank line here)
```

d. Al final del archivo, después de la línea en blanco, agregar una declaración final.
e. Guardar el archivo en la carpeta actual o en un lugar que está guardada en la ruta de MATLAB.

3.2.2. Creación de figuras de interfaz gráfica de usuario simple.

Añada las siguientes líneas antes de la declaración final de su archivo para crear una figura y colocarla en la pantalla. (En el software MATLAB, una interfaz gráfica de usuario es una figura.)

```
%  Create and then hide the GUI as it is being constructed.
f = figure('Visible','off','Position',[360,500,450,285]);
```

Las llamadas funciones de la figura 3.1, se utiliza dos pares de propiedad o valor, que son:

i. La propiedad *"Visible"*, hace que la GUI sea invisible para que el usuario GUI no pueda ver los componentes que se agregan o se inicializan.

ii. Cuando la interfaz GUIs tiene todos sus componentes y se inicializa, el ejemplo hace que sea visible. Las propiedades *"Position"* es un vector de cuatro elementos que especifica la ubicación y tamaño de la GUI en la pantalla, bajo la siguiente estructura: [distancia izquierda, la distancia inferior, ancho, alto]. Por defecto las unidades son píxeles.

3.2.3. Agregar componentes de la interfaz gráfica de usuario simple.

En esta sección, se podrá agregar los botones (pulsadores), texto estático, menú pop-up, y los ejes de componentes rectangulares, cilíndricas o esféricas para una interfaz gráfica (GUI), y se debe seguir los siguientes pasos:

1. Para añadir estas declaraciones a su archivo de código, creamos tres componentes de botón pulsador.

```
% Construct the components.
hsurf    = uicontrol('Style','pushbutton',...
              'String','Surf','Position',[315,220,70,25]);
hmesh    = uicontrol('Style','pushbutton',...
              'String','Mesh','Position',[315,180,70,25]);
hcontour = uicontrol('Style','pushbutton',...
              'String','Countour','Position',[315,135,70,25]);
```

Cada instrucción utiliza una serie de pares (propiedad/valor) de ***uicontrol***, para definir un botón o pulsador:

- La propiedad de estilo especifica que el ***uicontrol*** es un botón pulsador.

- La propiedad ***String*** especifica la etiqueta de cada botón: *Surf*, *Mesh*, y *Countour*.
- La propiedad de posición especifica la ubicación y tamaño de cada botón dentro de la GUI en la pantalla, bajo la siguiente estructura: [distancia izquierda, la distancia inferior, ancho, alto]. Las unidades por defecto para pulsadores son píxeles.

2. Añadir el menú pop-up y su etiqueta de texto estática al GUI, mediante la adición de declaraciones en el archivo de código, la siguiente definición del pulsador. La primera sentencia crea un menú emergente y la segunda sentencia crea un componente de texto que sirve como una etiqueta para el menú pop-pup.

```
hpopup = uicontrol('Style','popupmenu',...
           'String',{'Peaks','Membrane','Sinc'},...
           'Position',[300,50,100,25]);
htext  = uicontrol('Style','text','String','Select Data',...
           'Position',[325,90,60,15]);
```

   La propiedad *String* del componente menú pop-pup, utiliza una matriz de celdas para especificar los tres elementos en el menú pop-pup, tal como: *Peaks*, *Membrane* y *Sinc*. La componente del texto estático, la propiedad String especifica instrucciones para el usuario GUI. Para ambos componentes, la propiedad de posición especifica la ubicación y tamaño de cada componente dentro de la GUI, bajo la siguiente estructura: [distancia izquierda, la distancia parte inferior, ancho, alto]. Unidades por defecto para los componentes son píxeles.

3. Crear los ejes a la interfaz gráfica de usuario mediante la adición de esta declaración en el fichero de código.

```
ha = axes('Units','pixels','Position',[50,60,200,185]);
```

   La propiedad *"Units"* especifica las unidades en píxeles, para que los ejes tengan las mismas unidades que los otros componentes.

4. A raíz de todas las definiciones de componentes, agregamos esta línea al archivo de código para alinear todos los componentes, excepto los ejes, a lo largo de sus centros.

```
align([hsurf,hmesh,hcontour,htext,hpopup],'Center','None');
```

5. Añadir este comando después del comando de alineación.

```
%Make the GUI visible.
set(f,'Visible','on')
```

Su archivo de código debería tener este aspecto:

```
function simple_gui2
% SIMPLE_GUI2 Select a data set from the pop-up menu, then
% click one of the plot-type push buttons. Clicking the button
% plots the selected data in the axes.

   %  Create and then hide the GUI as it is being constructed.
   f = figure('Visible','off','Position',[360,500,450,285]);

   %  Construct the components.
   hsurf = uicontrol('Style','pushbutton','String','Surf',...
           'Position',[315,220,70,25]);
   hmesh = uicontrol('Style','pushbutton','String','Mesh',...
           'Position',[315,180,70,25]);
   hcontour = uicontrol('Style','pushbutton',...
           'String','Countour',...
           'Position',[315,135,70,25]);
   htext = uicontrol('Style','text','String','Select Data',...
           'Position',[325,90,60,15]);
   hpopup = uicontrol('Style','popupmenu',...
           'String',{'Peaks','Membrane','Sinc'},...
           'Position',[300,50,100,25]);
   ha = axes('Units','Pixels','Position',[50,60,200,185]);
   align([hsurf,hmesh,hcontour,htext,hpopup],'Center','None');

   %Make the GUI visible.
   set(f,'Visible','on')

end
```

6. Se ejecuta el código escribiendo ***simple_gui2*** en el *prompt* de MatLab, y se mostrará (véase figura 3.2) la GUI creada.

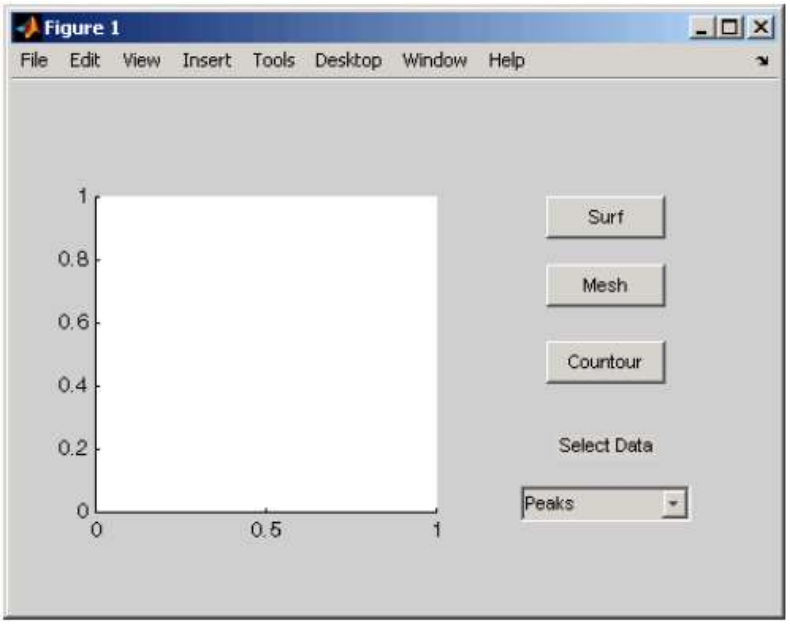


Figura 3. 2: Ventana final desarrollada en GUI de MatLab.
Fuente: http://www.mathworks.com/products/matlab/

Se selecciona un conjunto de datos en el menú pop-pup, y dando clic en los pulsadores (botoneras), sin pasar nada. Esto se debe a que no existe el código callback (devolución de llamada) en el archivo al servicio en el menú pop-up o de la botonera.

3.3. Código de programación de un GUI.

El código de programación del menu pop-pup, permite a los usuarios seleccionar datos a la trama. Cuando se selecciona uno de los tres conjuntos de datos del GUI en el pop-pup, MatLab establece la propiedad ***Value***, para el índice de la cadena seleccionada. El menú pop-pup callback, lee la propiedad ***Value***, y así determinar qué elemento se está mostrando actualmente y por consecuencia establecido por current_data.

```
%  Pop-up menu callback. Read the pop-up menu Value property to
%  determine which item is currently displayed and make it the
%  current data. This callback automatically has access to
%  current_data because this function is nested at a lower level.
   function popup_menu_Callback(source,eventdata)
      % Determine the selected data set.
      str = get(source, 'String');
      val = get(source,'Value');
      % Set current data to the selected data set.
```

```
    switch str{val};
    case 'Peaks' % User selects Peaks.
       current_data = peaks_data;
    case 'Membrane' % User selects Membrane.
       current_data = membrane_data;
    case 'Sinc' % User selects Sinc.
       current_data = sinc_data;
    end
end
```

3.4. Diseño de códigos de línea.

En esta sección se desarrollará una herramienta de interfaz gráfica de usuario (GUI o GUIDE) en la cual se programarán cada uno de los siguientes códigos de línea: Unipolares NRZ y RZ, Polar NRZ, Bipolar RZ, AMI NRZ, AMI RZ y Manchester NRZ; con la única finalidad de simular. Además, se podrá generar aleatoriamente 10 bits y que el usuario elija la codificación, posteriormente se podrá mostrar las densidades espectrales de potencia. En la figura 3.3 se muestran algunos de los códigos de líneas que serán programadas dentro de la GUI.

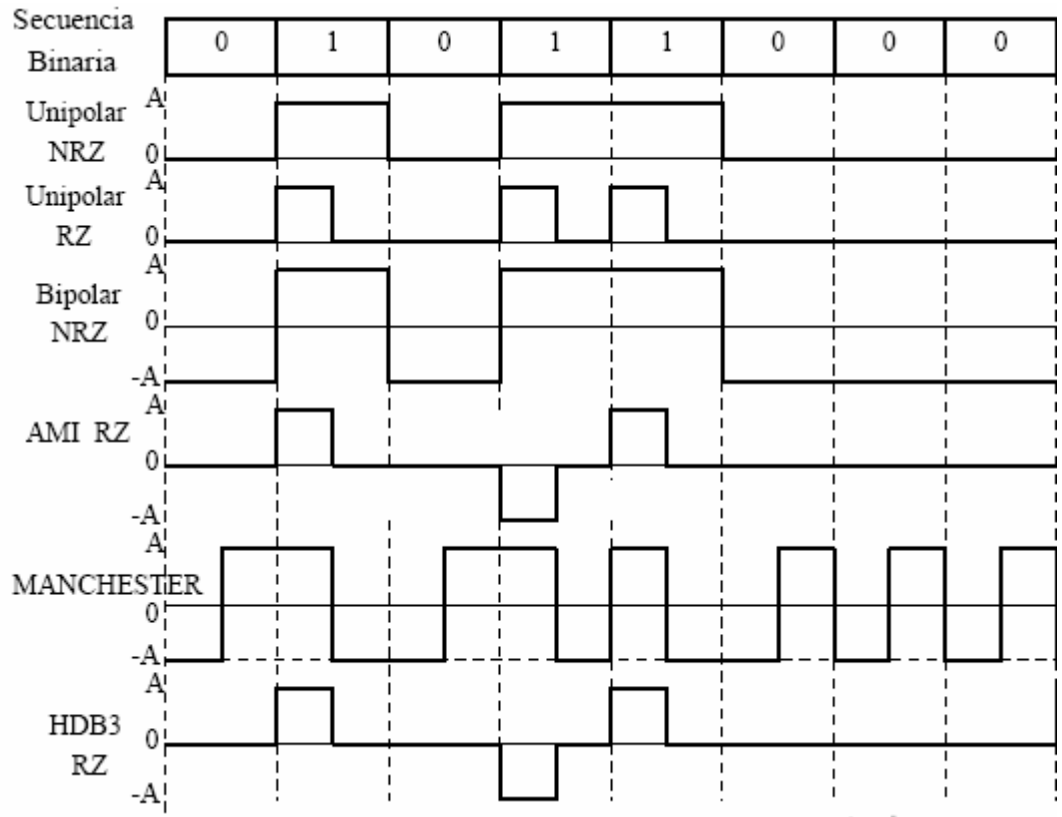


Figura 3. 3: Códigos de línea a simular excepto HDB3 RZ.
Elaborado por: Autor.

En los sistemas de comunicaciones digitales se emplean códigos de línea, entre los cuales se destacan los de no retorno a cero (NRZ), retorno a cero (RZ), Manchester y HDB3 de retorno a cero. En las subsecciones 3.4.1 y 3.4.2 se muestran los diseños de las interfaces gráficas de los códigos de línea y de la densidad de potencia espectral.

3.4.1. Diseño de la GUI principal para los códigos de línea.

En la figura 3.4 se muestra el diseño de la ventana (pantalla) principal, en la cual se observan los datos binarios aleatorios, la selección de los códigos de líneas (ver en Unipolar NRZ), el *axes1* que permitirá graficar el código seleccionado y el botón que permitirá obtener la señal de la densidad espectral de potencia de cada código.

c

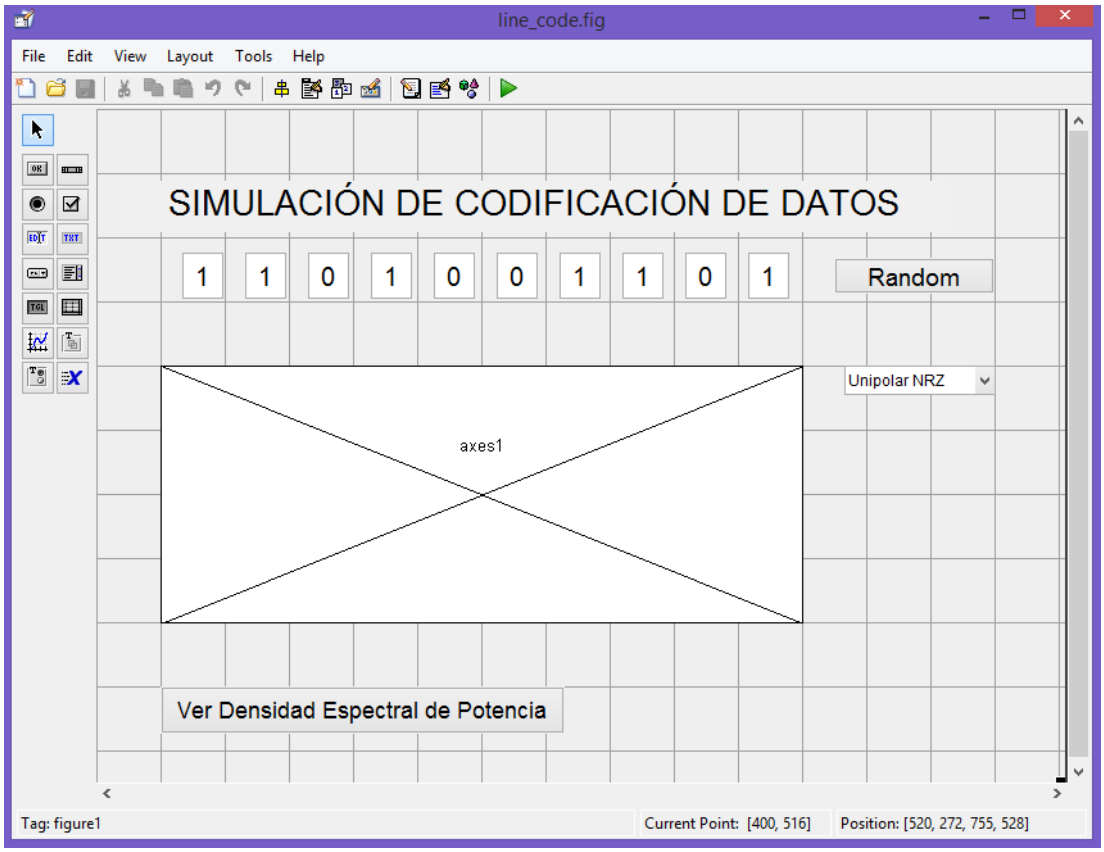


Figura 3. 4: Ventana GUI para códigos de línea.
Elaborado por: Autor.

3.4.2. Diseño de la GUI para generar la densidad de potencia espectral.

En la figura 3.5 se muestra el diseño de la ventana (pantalla) que permitirá obtener la señal de densidad espectral de potencia, en la misma se

observan los códigos de líneas que serán seleccionados para así visualizar dichas señales espectrales.

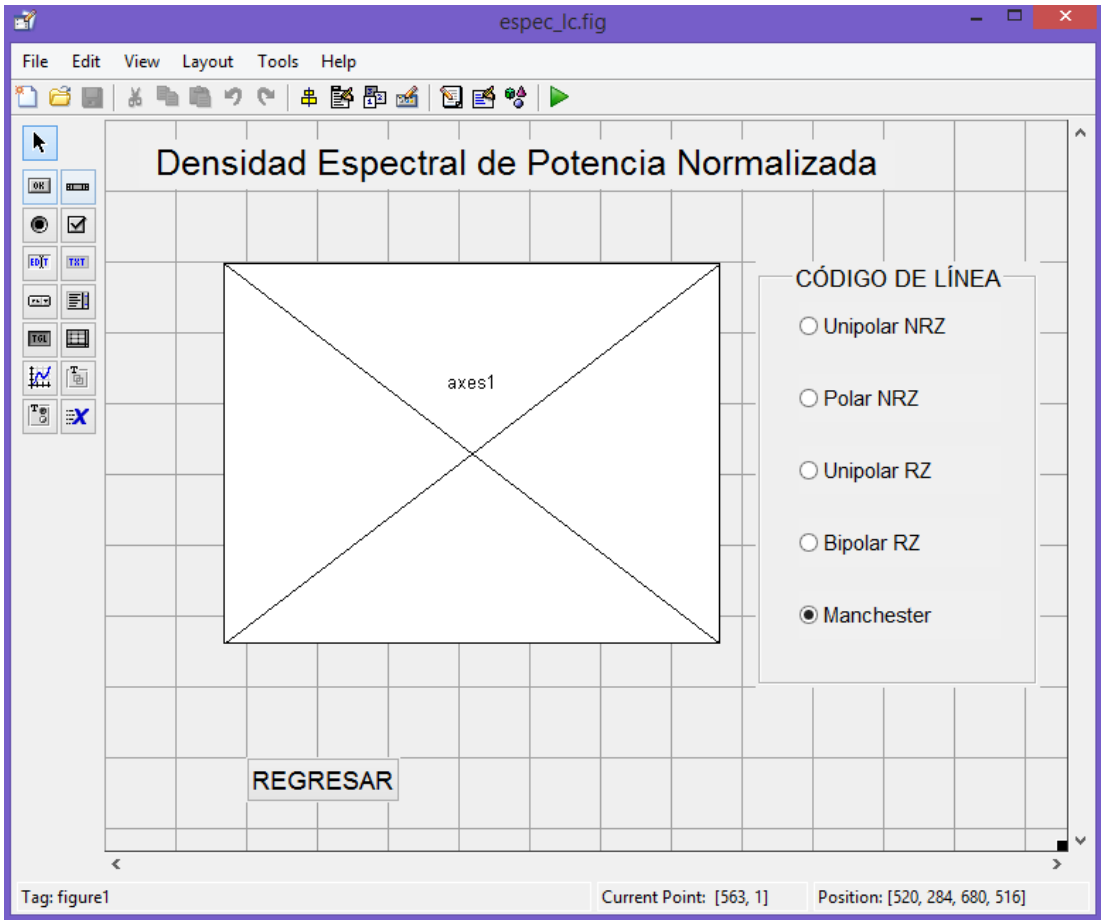


Figura 3. 5: Ventana GUI para generar la densidad de potencia espectral.
Elaborado por: Autor.

3.5. Programación de las GUIs – Códigos de Línea y PSD.

En esta sección se desarrollará los algoritmos de programación de cada interfaz gráfica (GUI) que se mostraron en las figuras 3.4 y 3.5. Es decir, que se realizará la programación en lenguaje de alto nivel (Matlab) de cómo generar las secuencias de bits de manera aleatoria, así como la generación de los códigos de líneas escogidos para ser simulados y la obtención de cada una de las señales de densidad de potencia espectral.

3.5.1. Programación para generar los Bits aleatorios.

La parte inicial del modelado de códigos de línea es la generación de secuencias de bits (1010101000) de manera aleatoria (random). A continuación, se muestra el código de programación que permite generar aleatoriamente los 10 bits.

```
function line_code_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to line_code (see VARARGIN)

hold off;
        h=[1 1 0 1 0 0 1 1 0 1];
        n=1;
        h(11)=1;
        while n<=10;
            t=n-1:0.001:n;
        if h(n) == 0
            if h(n+1)==0
                y=(t>n);
            else
                y=(t==n);
            end
            d=plot(t,y);title('Codificación UNIPOLAR NRZ');grid on
            set(d,'LineWidth',2.5);
            hold on;
            axis([0 10 -1.5 1.5]);
        else
            if h(n+1)==0
                y=(t<n)-0*(t==n);
            else
                y=(t<n)+1*(t==n);
            end
            d=plot(t,y);title('Code UNIPOLAR NRZ');grid on;
            set(d,'LineWidth',2.5);
            hold on;
            axis([0 10 -1.5 1.5]);
        end
        n=n+1;
        end
```

Posteriormente, se muestra el código de programación que activará el botón o pulsador *Random* (ver figura 3.4) lo que posteriormente llamará a la selección de la línea de código y mostrarán las gráficas de cada uno de los códigos de líneas escogidas en el presente componente práctico del examen complexivo. Se puede observar en las líneas del código de programación que se debe considerar una secuencia de 10 bits (n<=10) y para cada caso se visualizará el código de línea que se escoja en la interfaz gráfica principal

```
% --- Executes on button press in random.
function random_Callback(hObject, eventdata, handles)
% hObject    handle to random (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
a=round(rand(1,1));
b=round(rand(1,1));
c=round(rand(1,1));
d=round(rand(1,1));
e=round(rand(1,1));
f=round(rand(1,1));
g=round(rand(1,1));
h=round(rand(1,1));
i0=round(rand(1,1));
j0=round(rand(1,1));
ran=[a,b,c,d,e,f,g,h,i0,j0];
set(handles.uno,'String',ran(1));
set(handles.dos,'String',ran(2));
set(handles.tres,'String',ran(3));
set(handles.cuatro,'String',ran(4));
set(handles.cinco,'String',ran(5));
set(handles.seis,'String',ran(6));
set(handles.siete,'String',ran(7));
set(handles.ocho,'String',ran(8));
set(handles.nueve,'String',ran(9));
set(handles.diez,'String',ran(10));

%-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-
handles.bits=[a,b,c,d,e,f,g,h,i0,j0];
cod=get(handles.select_code,'Value');
switch cod
    case 1
        hold off;
        h=handles.bits;
        n=1;
        h(11)=1;
        while n<=10;
            t=n-1:0.001:n;
        if h(n) == 0
            if h(n+1)==0
                y=(t>n);
            else
                y=(t==n);
            end
            d=plot(t,y);title('Code UNIPOLAR NRZ');grid on
            set(d,'LineWidth',2.5);
```

```
        hold on;
        axis([0 10 -1.5 1.5]);
    end
    n=n+1;
    end
case 2
    hold off;
    h =handles.bits;
    n=1;
    h(11)=1;
    while n<=10;
        t=n-1:0.001:n;
    if h(n) == 0
        if h(n+1)==0
            y=-(t<n)-(t==n);
        else
            y=-(t<n)+(t==n);
        end
        d=plot(t,y);title('Code POLAR NRZ');grid on
        set(d,'LineWidth',2.5);
        hold on;
        axis([0 10 -1.5 1.5]);
    else
        if h(n+1)==0
            y=(t<n)-1*(t==n);
        else
            y=(t<n)+1*(t==n);
        end
        d=plot(t,y);title('Code POLAR NRZ');grid on;
        set(d,'LineWidth',2.5);
        hold on;
        axis([0 10 -1.5 1.5]);
    end
    n=n+1;
    end

case 3
    hold off;
    h =handles.bits;
    n=1;
    h(11)=1;
    while n<=10;
        t=n-1:0.001:n;
    %Graficación de los CEROS (0)
        if h(n) == 0
            if h(n+1)==0
                y=(t>n);
```

```
            else
                y=(t==n);
            end
        d=plot(t,y);title('Code UNIPOLAR RZ');grid on
        set(d,'LineWidth',2.5);
        hold on;
        axis([0 10 -1.5 1.5]);
    %Graficación de los UNOS (1)
        else
        if h(n+1)==0
            y=(t<n-0.5);
        else
            y=(t<n-0.5)+1*(t==n);
        end
        d=plot(t,y);title('Code UNIPOLAR RZ');grid on;
        set(d,'LineWidth',2.5);
        hold on;
        axis([0 10 -1.5 1.5]);
        end
    n=n+1;

    end
```

Solo se han mostrado las configuraciones de los tres primeros códigos que dispone el GUI, los otros son similares, pero con otras características. Finalmente se configura el botón para obtener la PSD.

```
% --- Executes on button press in espectro.
function espectro_Callback(hObject, eventdata, handles)
% hObject    handle to espectro (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close line_code;
espec_lc
```

3.5.2. Programación para la generación de las señales PSD.

En esta sección se desarrollará el programa que permite obtener las señales de la densidad espectral de potencia. A continuación, se muestra el código que permite la llamada desde la ventana principal (ver figura 3.5)

```
function varargout = espec_lc(varargin)
% Simulación de Códigos de Línea
% Autor: Luis Fernando Barros Tapia
% luferbata@gmail.com

% Comienza código de inicialización - NO EDITAR
```

```
% Comienza código de inicialización - NO EDITAR
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @espec_lc_OpeningFcn, ...
                   'gui_OutputFcn',  @espec_lc_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
% Final del código de inicialización - NO EDITAR

% --- Executes just before espec_lc is made visible.
function espec_lc_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to espec_lc (see VARARGIN)
    hold off;
    A=1;
    Tb=1.5;
    R=1/Tb;
    L=2*R;
    f=0:L/50:L;
    P=(A.^2*Tb)*(sinc(f*Tb/2)).^2.*(sin(pi*f*Tb/2)).^2;
    g=plot(f,P);
    title('DENSIDAD ESPECTRAL: MANCHESTER NRZ');
    hold on;xlabel('Frecuencia');ylabel('Potencia Normalizda');
    axis([0 L 0 1.1*Tb]);set(g,'LineWidth',2.5);
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{['R'];['2R']})
    set(gca,'YTickLabel',{['0.5*Tb'];['Tb']})
% Choose default command line output for espec_lc
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes espec_lc wait for user response (see UIRESUME)
% uiwait(handles.figure1);
```

A continuación, se muestran las líneas de códigos de programación para obtener las densidades de potencias espectrales de cada uno de los códigos de líneas propuestos en el presente trabajo de titulación:

```
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
% hObject    handle to uipanel1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if (hObject==handles.Unipolar_NRZ)
    hold off;
    A=sqrt(2);
    Tb=1.5;
    R=1/Tb;
    L=2*R;
    f=0:L/50:L;
    del=0;
    P=(A.^2*Tb)/4*(sinc(f*Tb)).^2*(1+(1/Tb)*del);
    g=plot(f,P);
    title('DENSIDAD ESPECTRAL: UNIPOLAR NRZ');
    hold on;xlabel('Frecuencia');ylabel('Potencia Normalizada');
    axis([0 L 0 1.1*Tb]);set(g,'LineWidth',2.5);
    stem(0,(A.^2*Tb)/2,'LineWidth',2.5);hold off;
    axis([0 L 0 1.09*Tb]);set(g,'LineWidth',2.5);
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{'R =';'2R'})
    set(gca,'YTickLabel',{'0.5*Tb';'Tb'})

elseif(hObject==handles.Polar_NRZ)
    hold off;
    A=1;
    Tb=1.5;
    R=1/Tb;
    L=2*R;
    f=0:L/50:L;
    del=0;
    P=(A.^2*Tb)*(sinc(f*Tb)).^2;
    g=plot(f,P);hold on;xlabel('Frecuencia');
    ylabel('Potencia Normalizada');
    title('DENSIDAD ESPECTRAL: POLAR NRZ')
    axis([0 L 0 1.01*Tb]);set(g,'LineWidth',2.5);
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{['R'];['2R']})
    set(gca,'YTickLabel',{['0.5*Tb'];['Tb']})

elseif(hObject==handles.Unipolar_RZ)
    hold off;
    A=2;
    Tb=1;
    R=1/Tb;
```

```
    L=2*R;
    f=0:L/50:L;
    del=0;
    P=(A.^2*Tb)/16*(sinc(f*Tb/2)).^2;
    g=plot(f,P);
    title('DENSIDAD ESPECTRAL: UNIPOLAR NRZ');
    hold on;xlabel('Frecuencia');ylabel('Potencia Normalizada');
    axis([0 L 0 1.1*Tb]);set(g,'LineWidth',2.5);
    stem([0 R],[(A.^2*Tb)/8 P(26)+0.1],'LineWidth',2.5);hold off;
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{['R'];['2R']})
    set(gca,'YTickLabel',{['0.5*Tb'];['Tb']})

elseif(hObject==handles.Bipolar_RZ)
    hold off;
    A=2;
    Tb=1.5;
    R=1/Tb;
    L=2*R;
    f=0:L/50:L;
    P=(A.^2*Tb)/8*(sinc(f*Tb/2)).^2.*(1-cos(2*pi*f*Tb));
    g=plot(f,P);
    title('DENSIDAD ESPECTRAL: BIPOLAR RZ');

    hold on;xlabel('Frecuencia');ylabel('Potencia Normalizada');
    axis([0 L 0 1.1*Tb]);set(g,'LineWidth',2.5);
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{['R'];['2R']})
    set(gca,'YTickLabel',{['0.5*Tb'];['Tb']})

else
    hold off;
    A=1;
    Tb=1.5;
    R=1/Tb;
    L=2*R;
    f=0:L/50:L;
    P=(A.^2*Tb)*(sinc(f*Tb/2)).^2.*(sin(pi*f*Tb/2)).^2;
    g=plot(f,P);
    title('DENSIDAD ESPECTRAL: MANCHESTER NRZ');
    hold on;xlabel('Frecuencia');ylabel('Potencia Normalizada');
    axis([0 L 0 1.1*Tb]);set(g,'LineWidth',2.5);
    set(gca,'XTickMode','manual','XTick',[R,2*R]);grid on;
    set(gca,'YTickMode','manual','YTick',[0.5*Tb,Tb]);
    set(gca,'XTickLabel',{['R'];['2R']})
    set(gca,'YTickLabel',{['0.5*Tb'];['Tb']})

end
```

3.6. Resultados obtenidos de la simulación de códigos de línea.

Aquí se muestran los resultados que se obtienen de la simulación de los códigos de línea mencionados anteriormente y se obtienen las densidades espectrales de potencia. En la figura 3.6 se muestra la señal del código Unipolar NRZ de los bits aleatorios.

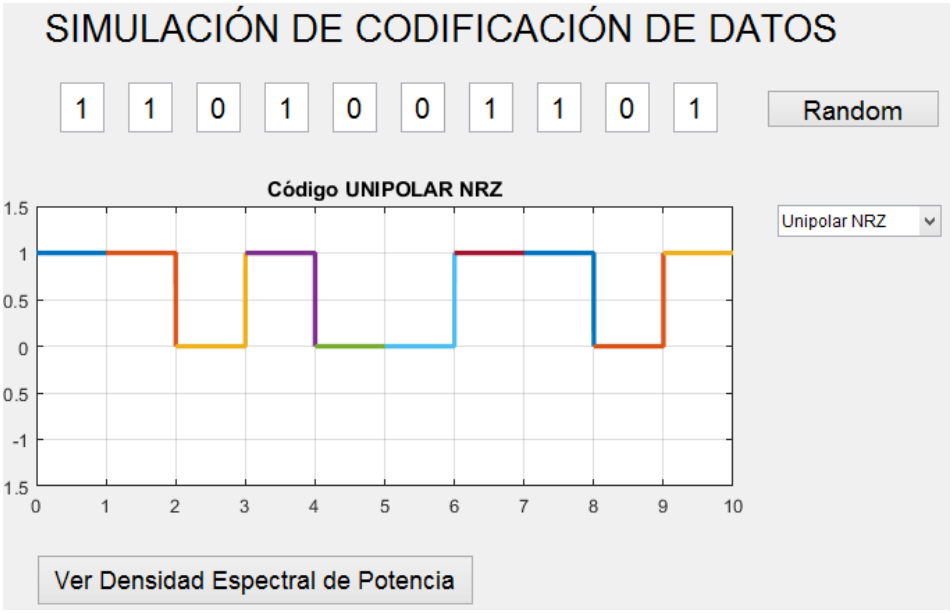


Figura 3. 6: Generación de bits del código Unipolar NRZ.
Elaborado por el Autor.

Una vez generado el código, se debe pulsar *See Spectrum*, para mostrar la densidad espectral de la potencia (véase la figura 3.7).

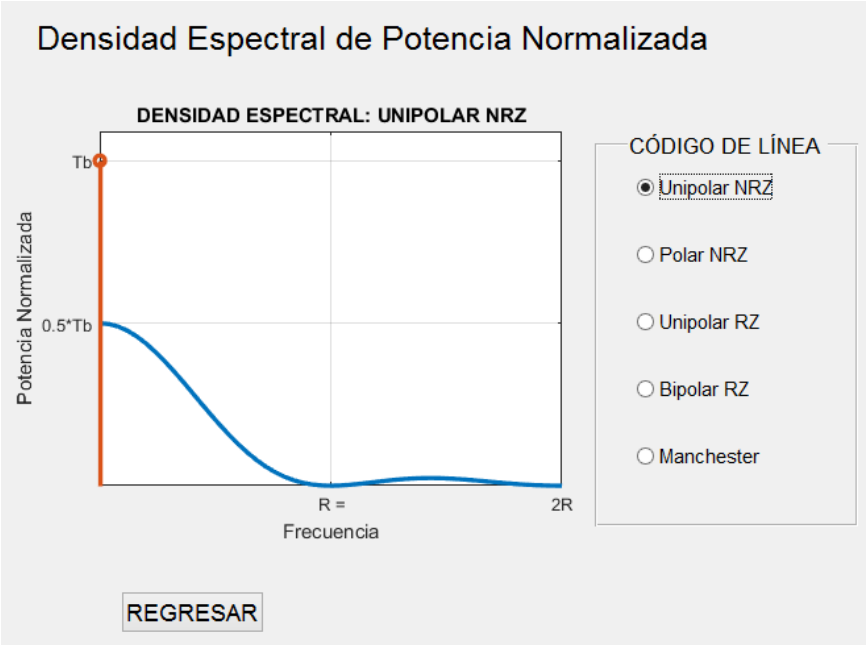


Figura 3. 7: Densidad espectral de potencia para Unipolar NRZ.
Elaborado por el Autor.

En la figura 3.8 se muestra la señal de bits para el código Polar NRZ obtenida aleatoriamente. En la figura 3.9 se muestra la densidad espectral de potencia del código Polar NRZ.

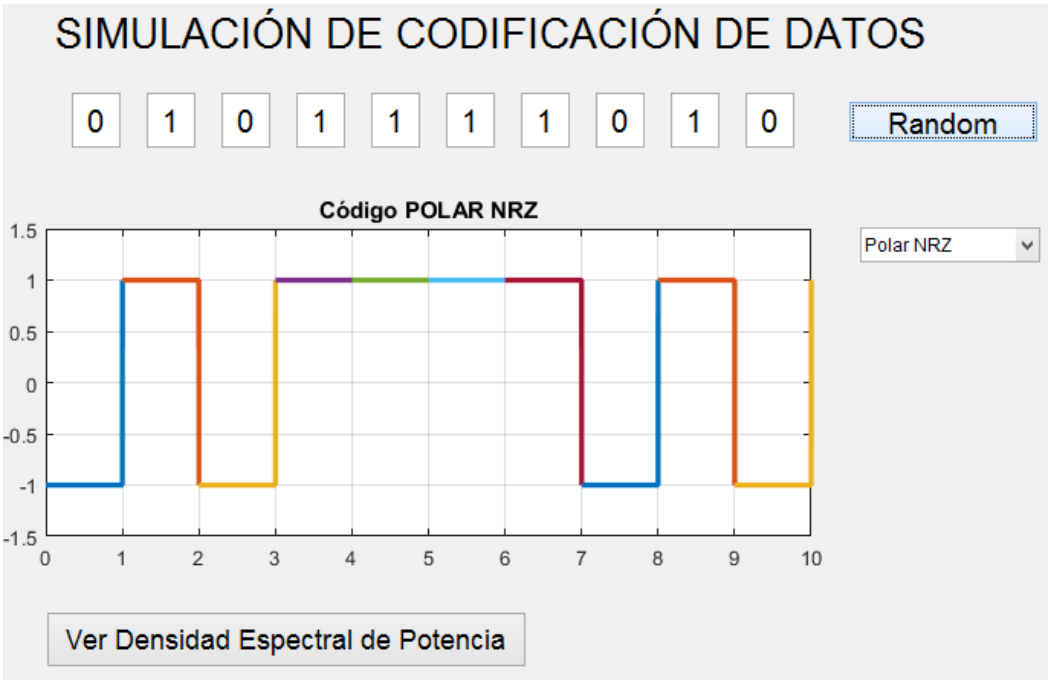


Figura 3. 8: Generación de bits del código Polar NRZ.
Elaborado por el Autor.

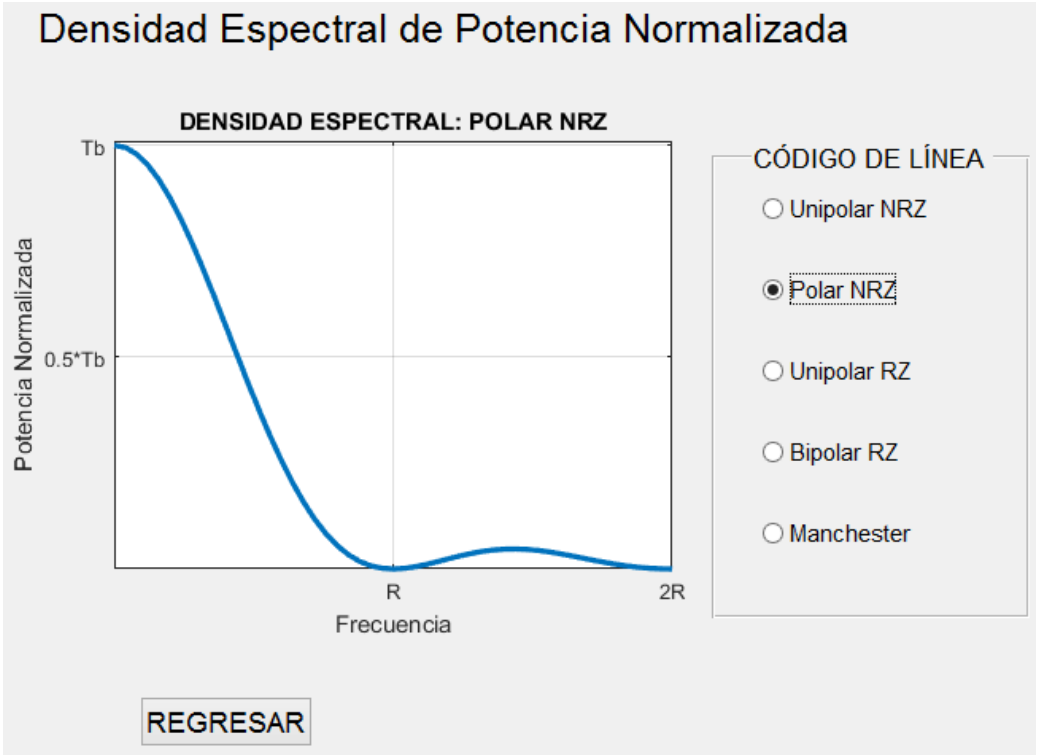


Figura 3. 9: Densidad espectral de potencia para Unipolar NRZ.
Elaborado por el Autor.

La generación de bits de los códigos de líneas restantes se puede visualizar en la figura 3.10 (Unipolar RZ), figura 3.11 (Bipolar RZ), figura 3.12 (AMI NRZ), figura 3.13 (AMI RZ) y figura 3.14 (Manchester).

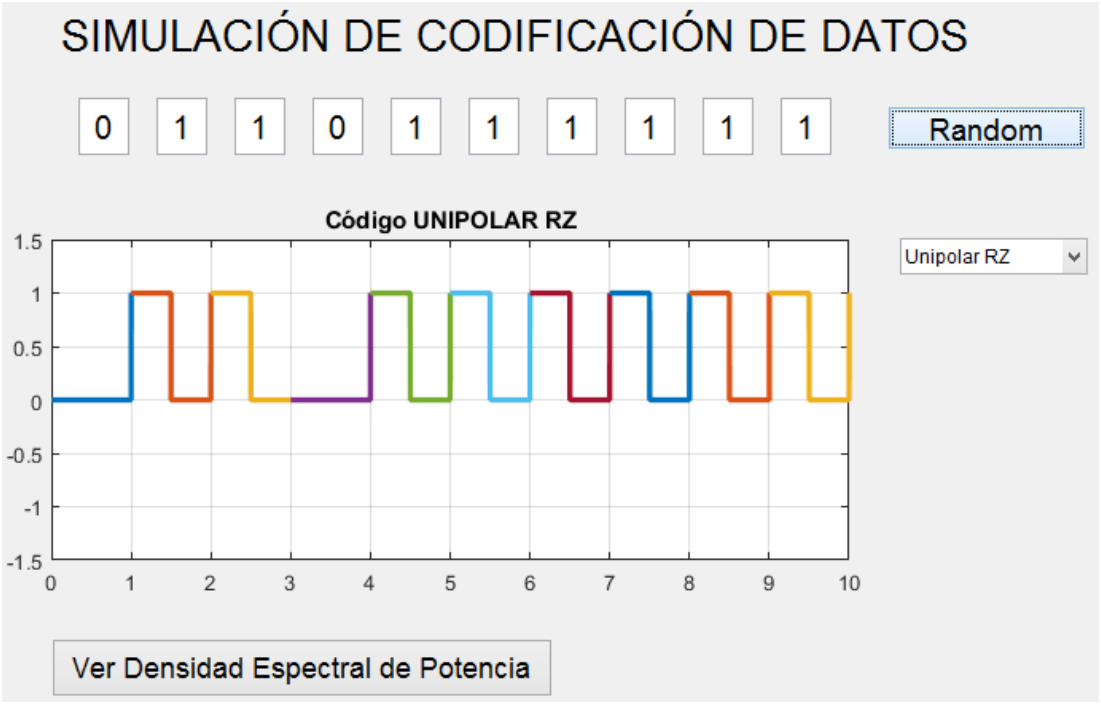


Figura 3. 10: Generación de bits del código Unipolar RZ.
Elaborado por el Autor.

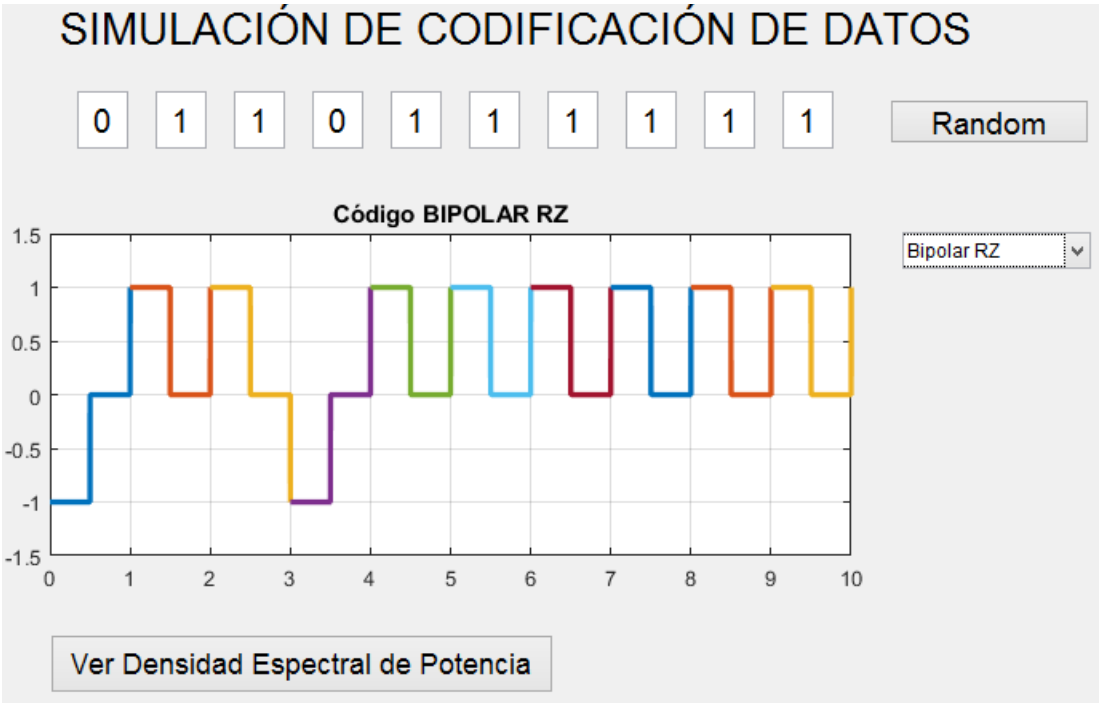


Figura 3. 11: Generación de bits del código Bipolar RZ.
Elaborado por el Autor.

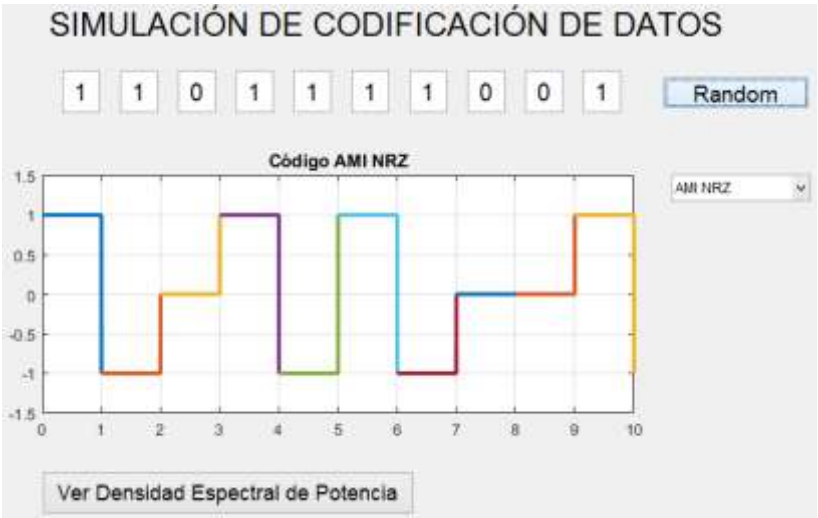


Figura 3. 12: Generación de bits del código AMI NRZ.
Elaborado por el Autor.

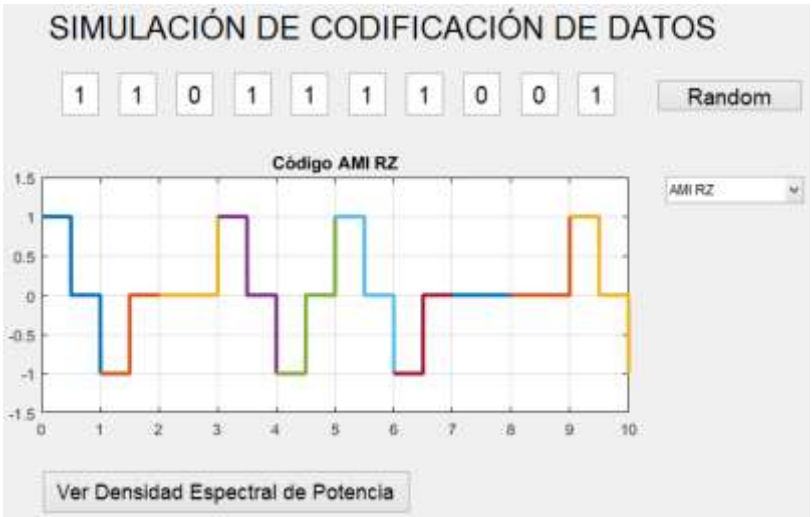


Figura 3. 13: Generación de bits del código AMI RZ.
Elaborado por el Autor.

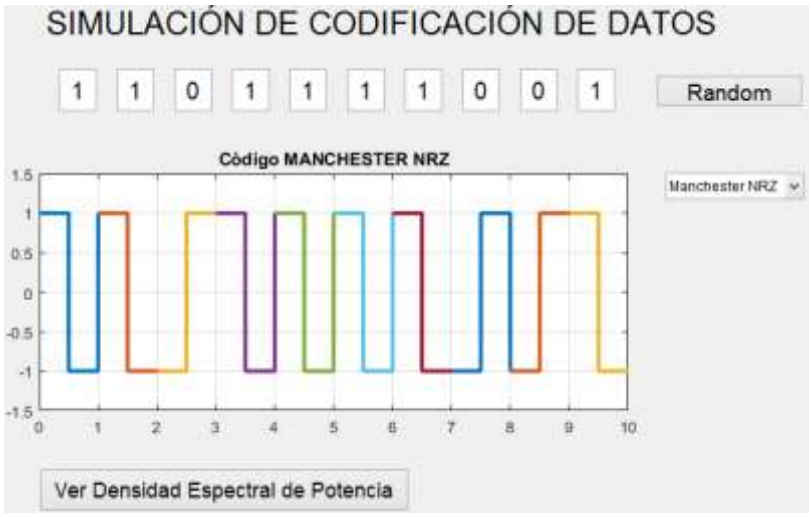


Figura 3. 14: Generación de bits del código Manchester NRZ.
Elaborado por el Autor.

Como sucedió en la figura 3.7, aquí se obtienen las densidades espectrales de potencia tal como se ilustran en las figuras 3.15, 3.16 y 3.17.

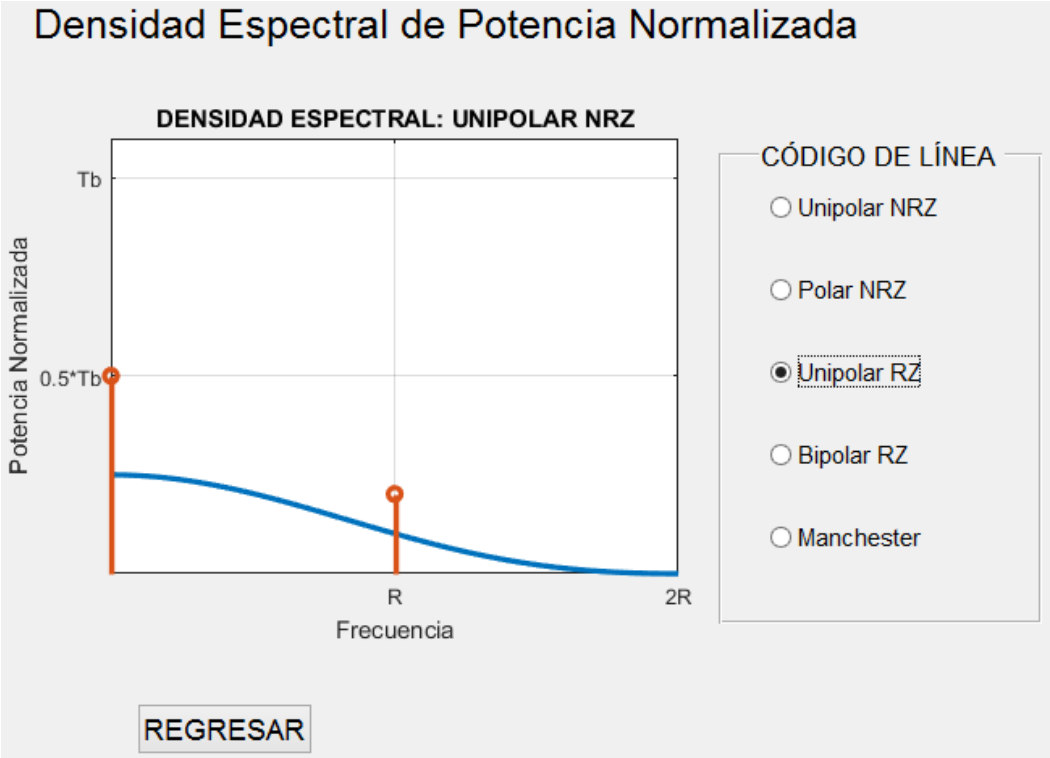


Figura 3. 15: Densidad espectral de potencia para Unipolar RZ.
Elaborado por el Autor.

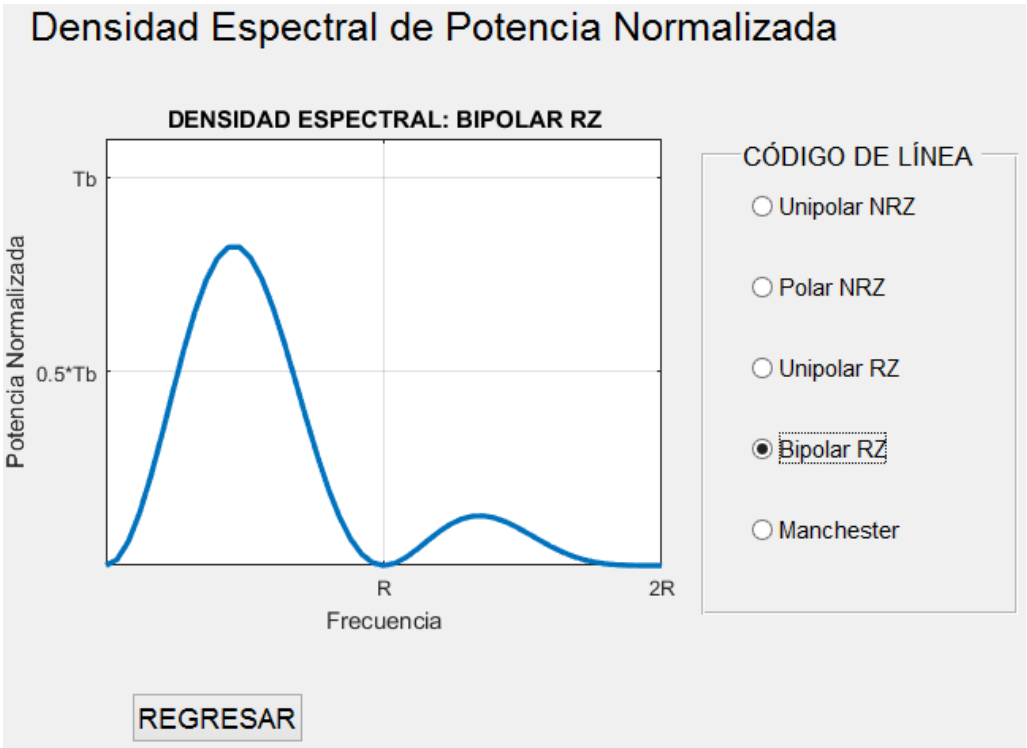


Figura 3. 16: Densidad espectral de potencia para Bipolar RZ.
Elaborado por el Autor.

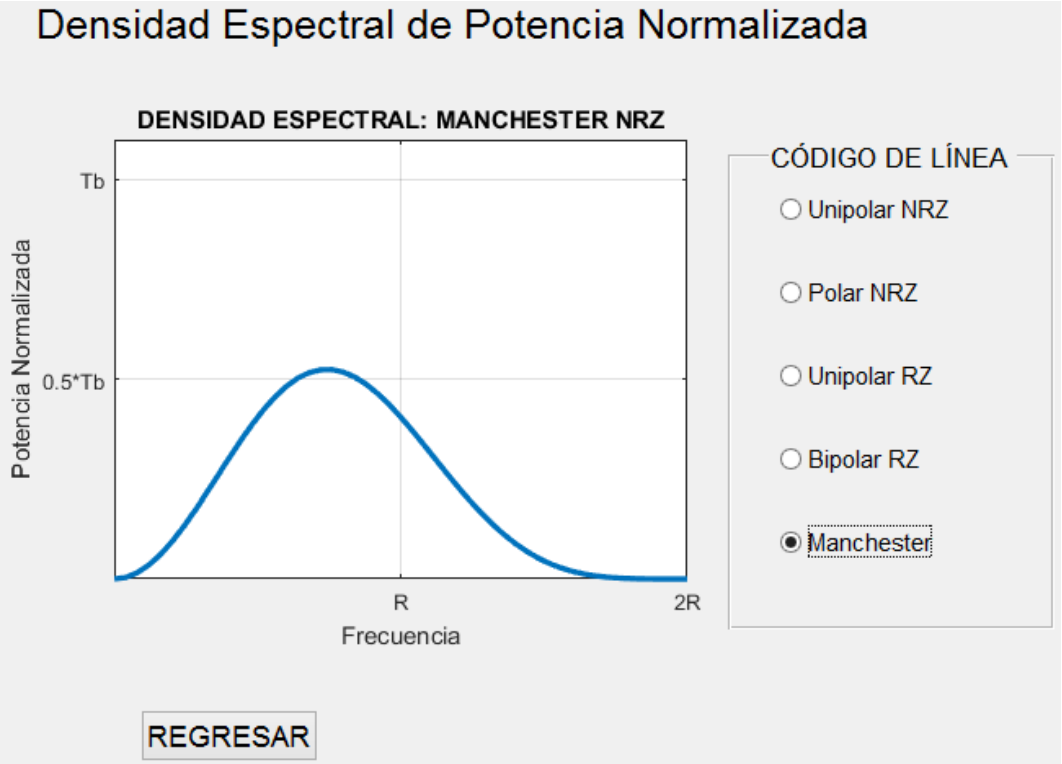


Figura 3. 17: Densidad espectral de potencia para AMI y Manchester NRZ. Elaborado por el Autor.

Conclusiones

- Una presentación detallada de codificación de línea, particularmente aplicable a la telefonía, se ha incluido en el presente trabajo de titulación. En la cual en el estado del arte se examinaron las características más deseables de códigos de línea. En la cual se presentaron cinco códigos de línea comunes y ocho códigos de línea alternos. Cada línea de código se ilustra con una forma de onda de ejemplo. En la mayoría de los casos se dieron y se representan expresiones para el PSD y la probabilidad de error.

- Se examina brevemente la plataforma GUI-MatLab que es la plataforma de programación gráfica, muy utilizada para diferentes aplicaciones en el campo de la Ingeniería en Telecomunicaciones.

- La parte más importante del presente trabajo de titulación fue diseñar una interfaz de programación gráfica GUI, la misma que fue creada para verificar que las técnicas de codificación y la densidad espectral de potencian escogidas sean lo más parecidos a lo descrito en la parte del estado del arte.

Recomendaciones.

Durante el desarrollo del componente práctico del examen complexivo se utilizó MatLab y GUI. Para lo cual, está plataforma de simulación MatLab dispone de herramientas como Simulink y GUI, las que permiten el desarrollo de aplicaciones en ingeniería en especial de Telecomunicaciones, por lo tanto, se recomienda:

- Que la Universidad Católica de Santiago de Guayaquil adquiera la licencia académica de MatLab 2019 incluyendo las herramientas Simulink y GUI, y esto servirá como ayuda académica para mejorar el aprendizaje en la mayoría de las asignaturas disponibles en la nueva malla curricular del rediseño de la Carrera de Telecomunicaciones.

- A los Docentes de la FETD deben realizar cursos de capacitación o certificarse en el manejo de MatLab, para que no solamente quede como obligación de los estudiantes investigar el uso de MatLab, es decir, que los estudiantes requieren de un profesional capacitado para emplear dicha herramienta.

Bibliografía.

Checa R., V. E., Velásquez C., J. D., & Álvarez R., R. (2012). *Implementación de códigos de línea en una tarjeta de entrenamiento basada en un FPGA* (Publicaciones y Memorias de Eventos). Escuela Politécnica Nacional, Quito. Recuperado de http://bibdigital.epn.edu.ec/handle/15000/4907

Cumbajín V., J. V., & Rivadeneira E., P. A. (2016). *Desarrollo e implementación del algoritmo de codificación de línea BnZS para optimizar el uso de canal de transmisión de Light Fidelity (Li-Fi)* (Tesis de Grado). Universidad Politécnica Salesiana, Quito. Recuperado de http://dspace.ups.edu.ec/handle/123456789/13688

Dalwadi, D. C., Goradiya, B. C., Solanki, M. M., & Holia, M. S. (2011). Performance evaluation of Power Spectral Density of different line coding technique. *National Conference on Recent Trends in Engineering & Technology*. Recuperado de http://www.bvmengineering.ac.in/misc/docs/published-20papers/etel/etel/401065.pdf

Gupta, A., & Singh, G. (2016). Implementation and Analysis of Different Line Coding Schemes using Verilog. *International Journal of Science, Engineering and Technology Research*, *5*(2), 395–401.

Latha, S., & Pradesh, A. (2011). Performance evaluation of different line codes. *Indian Journal of Computer Science and Engineering*, *2*(4), 575–588.

Pérez Vega, C. (2015). Codificación de canal y modulación. Recuperado el 25 de marzo de 2019, de https://personales.unican.es/perezvr/

Sklar, B. (2011). *Digital communications: fundamentals and applications* (2nd ed). Upper Saddle River, N.J: Prentice-Hall PTR.

Torres Salamea, H. M. (2014). *Análisis, modelado y simulación de algoritmos sobre técnicas de comunicación tolerante a fallas aplicadas en sistemas industriales* (Tesis de Maestría). Universidad del Azuay, Cuenca. Recuperado de http://dspace.uazuay.edu.ec/bitstream/datos/3345/1/10111.pdf

Zapata Y., C. F., & Jurado P., F. X. (2012). *Diseño e implementación de un módulo de laboratorio para los códigos de línea, empleando microcontroladores con una interfaz USB de conexión a LABVIEW para diseño, control y pruebas de codificación y decodificación de datos* (Tesis de Grado). Universidad Politécnica Salesiana, Quito.

DECLARACIÓN Y AUTORIZACIÓN

Yo, **BARROS TAPIA, LUIS FERNANDO** con C.C: # 110472399-2 autor del componente práctico de examen complexivo: **Diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab,** previo a la obtención del título de **INGENIERO EN TELECOMUNICACIONES** en la Universidad Católica de Santiago de Guayaquil.

1.- Declaro tener pleno conocimiento de la obligación que tienen las instituciones de educación superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación para que sea integrado al Sistema Nacional de Información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.

2.- Autorizo a la SENESCYT a tener una copia del referido trabajo de titulación, con el propósito de generar un repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Guayaquil, 22 de marzo del 2019

f. _____

Nombre: BARROS TAPIA, LUIS FERNANDO

C.C: 110472399-2

REPOSITORIO NACIONAL EN CIENCIA Y TECNOLOGÍA

FICHA DE REGISTRO DE TESIS/TRABAJO DE TITULACIÓN

TÍTULO Y SUBTÍTULO:	**Diseño e implementación de códigos de línea para sistemas de transmisiones digitales mediante interfaz gráfica de usuario (GUI) de MatLab.**		
AUTOR(ES)	BARROS TAPIA, LUIS FERNANDO		
REVISOR(ES)/TUTOR(ES)	M. Sc. PACHECO BOHÓRQUEZ, HÉCTOR IGNACIO		
INSTITUCIÓN:	Universidad Católica de Santiago de Guayaquil		
FACULTAD:	Facultad de Educación Técnica para el Desarrollo		
CARRERA:	Ingeniería en Telecomunicaciones		
TITULO OBTENIDO:	Ingeniero en Telecomunicaciones		
FECHA DE PUBLICACIÓN:	22 de marzo del 2019	**No. DE PÁGINAS:**	38
ÁREAS TEMÁTICAS:	Fundamentos de comunicación, comunicaciones inalámbricas		
PALABRAS CLAVES/ KEYWORDS:	DATOS, COMUNICACIONES, CODIFICACIÓN, ESPECTRO, TRANSMISIONES, MATLAB.		

RESUMEN/ABSTRACT:

Para el desarrollo del componente práctico del examen complexivo se tuvieron que revisar diferentes plataformas de simulación, se evaluó la funcionalidad de cada uno. Entre las plataformas analizadas se escogió el software MatLab. Posteriormente, se investigaron trabajos relacionados al uso de la codificación de línea y a través de tutoriales y libros pude comprender el uso de MatLab. El presente trabajo consistió en realizar el diseño de una interfaz gráfica de la codificación de línea muy utilizados en sistemas de transmisiones digitales. En el capítulo 1, se describe una breve introducción relacionada al diseño asistido por computadoras (CAD), objetivo general y objetivos específicos del componente práctico. En el capítulo 2, se describen los fundamentos teóricos de las comunicaciones digitales. En el capítulo 3, se realiza el diseño de dos interfaces gráficas, una para el procesamiento de códigos de líneas y la otra para visualizar la generación de la densidad espectral de potencia para cada código de línea.

ADJUNTO PDF:	☒ SI	☐ NO
CONTACTO CON AUTOR/ES:	**Teléfono:** +593-9-91536141	E-mail: luferbata333@gmail.com
CONTACTO CON LA INSTITUCIÓN: COORDINADOR DEL PROCESO DE UTE	**Nombre:** Palacios Meléndez Edwin Fernando	
	Teléfono: +593-9-67608298	
	E-mail: edwin.palacios@cu.ucsg.edu.ec	

SECCIÓN PARA USO DE BIBLIOTECA	
Nº. DE REGISTRO (en base a datos):	
Nº. DE CLASIFICACIÓN:	
DIRECCIÓN URL (tesis en la web):	