UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CARACTERIZACIÓN DE UN ESQUEMA DE MODULACIÓN Y DEMODULACIÓN CAÓTICO PARA EL ESTÁNDAR LTE

ESTÉ JOSÉ
MENDOZA MOISÉS

Bárbula, 13 de Julio del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

CARACTERIZACIÓN DE UN ESQUEMA DE MODULACIÓN Y DEMODULACIÓN CAÓTICO PARA EL ESTÁNDAR LTE

TRABAJO ESPECIAL DE GRADO PRESENTADO ANTE LA ILUSTRE UNIVERSIDAD DE CARABOBO PARA OPTAR AL TÍTULO DE INGENIERO DE TELECOMUNICACIONES

ESTÉ JOSÉ
MENDOZA MOISÉS

Bárbula, 13 de Julio del 2015

UNIVERSIDAD DE CARABOBO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA DE
TELECOMUNICACIONES
DEPARTAMENTO DE SEÑALES Y SISTEMAS

AVAL DEL TUTOR

Quien suscribe Ahmad Osman, titular de la cédula de identidad 16.579.272, en mi carácter de TUTOR del Trabajo Especial de Grado titulado:

CARACTERIZACIÓN DE UN ESQUEMA DE MODULACIÓN Y DEMODULACIÓN CAÓTICO PARA EL ESTÁNDAR LTE

Y presentado por los bachilleres ESTÉ JOSÉ, cédula de identidad 20.313.186, MENDOZA MOISÉS, cédula de identidad 19.321.988, para optar al Título de Ingeniero de Telecomunicaciones, hago constar que dicho trabajo reúne los requisitos y méritos suficientes para ser sometido a la presentación pública y evaluación por parte del jurado examinador que se le designe

Firma
Prof. AHMAD OSMAN
CI: 16.579.272
TUTOR

Bárbula, 13 de Julio del 2015

Índice general

Índice de figuras

Acrónimos

2G	**2**da **G**eneracion
3G	**3**ra **G**eneracion
3GPP	**3**ra **G**eneration **P**artnership **P**roject
4G	**4**ta **G**eneracion
BER	**B**it **E**rror **R**atio
BW	**B**and **W**idth
CP	**C**yclic **P**refix
CS	**C**all **S**ervice
CDMA	**C**ode **D**ivision **M**ultiple **A**ccess
COFDM	**C**oded **O**rthogonal **F**requency **D**ivision **M**ultiplexing
DC	**D**irect **C**urrent
DCSK	**D**irect **C**haos **S**hift **K**eying
DFT	**D**iscrete **F**ourier **T**ransform
EPC	**E**volved **P**acket **C**ore
FEC	**F**orward **E**rror **C**orrection
GSM	**G**lobal **S**ystem for **M**obile **C**omunicaciones
HSPA	**H**igh **S**peed **P**acket **A**ccess
IMT	**I**nternational **M**obile **T**elecommunications
ISO	**I**nternational **O**rganization for **S**tandardization
IDFT	**I**nverse **D**iscrete **F**ourier **T**ransform
ISI	**I**nter**s**ymbol **I**nterference
ICI	**I**nter**c**arrier **I**nterference
IFFT	**I**nverse **F**ast **F**ourier **T**ransform
LMSC	**L**an **M**am **S**tandard **C**ommittee

IP	**I**nternet **P**rotocol
IEEE	**I**nstitute of **E**lectrical and **E**lectronics **E**ngineers
LAN	**L**ocal **A**rea **N**etwork
LTE	**L**ong **T**erm **E**volution
MIMO	**M**ultiple **I**nputs **M**ultiple **O**utputs
MAN	**M**etropolitan **A**rea **N**etwork
OFDM	**O**rthogonal **F**requency **D**ivision **M**ultiplexing
OFDMA	**O**rthogonal **F**requency **D**ivision **M**ultiplexing **A**ccess
OSI	**O**pen **S**ystem **I**nterconnection
PSK	**P**hase **S**hift **K**eying
PS	**P**acket **S**ervice
PAPR	**P**eak to **A**verage **P**ower **R**atio
QAM	**Q**uadrature **A**mplitude **M**odulation
SAE	**S**ystem **A**rchitecture **E**volution
SNR	**S**ignal to **N**oise **R**atio
UC	**U**niversidad de **C**arabobo
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
WIMAX	**W**orldwide **I**nteroperability for **M**icrowave **A**ccess
EC	**E**squema **C**onvencional
PC	**P**ortadora de **C**aótica
MC	**M**odulación **C**aótica
FC	**F**ull **C**aótica
UC	**U**niversidad de **C**arabobo

CARACTERIZACIÓN DE UN ESQUEMA DE MODULACIÓN Y DEMODULACIÓN CAÓTICO PARA EL ESTÁNDAR LTE

por

ESTÉ JOSÉ y MENDOZA MOISÉS

Presentado en el Departamento de Señales y Sistemas
de la Escuela de Ingeniería en Telecomunicaciones
el 13 de Julio del 2015 para optar al Título de
Ingeniero de Telecomunicaciones

RESUMEN

En este trabajo, se presenta un esquema novedoso de comunicación basado en la modulación caótica. Se caracterizó en GNU Radio un esquema de modulación y demodulación incluyendo la etapa de codificación y decodificación para el estándar LTE y así tener una referencia de resultados; los resultados de la simulación se dan bajo un canal AWGN y un canal de desvanecimiento plano Rayleigh típico del estándar LTE. Luego se hicieron unos cambios al estándar en cuanto al número de portadoras de datos y de referencia, así como también se modulo caóticamente dichas portadoras de referencia y se caracterizó el sistema. Posteriormente se procedió a crear en GNU Radio los bloques necesarios que permiten realizar una modulación y demodulación caótica en banda base para un único usuario para luego

agregar esta etapa en el esquema LTE convencional y medir su desempeño mediante los parámetros antes descritos. A continuación se realizó un nuevo esquema que contiene la nueva configuración del estándar y las portadoras de referencia moduladas caóticamente junto con el nuevo modulador y demodulador caótico creado. Una vez comprobados dichos esquemas se procedió a la comparación de los esquemas propuestos bajo el estándar LTE.

Palabras Claves: Caos, GNU Radio, LTE

Tutor: AHMAD OSMAN
Profesor del Departamento de Señales y Sistemas
Escuela de Telecomunicaciones. Facultad de Ingeniería adscrito al Laboratorio X

Capítulo I

Introducción

1.1. MOTIVACIÓN.

En un futuro cercano, habrá un gran número de dispositivos móviles que demandarán recursos en el limitado espectro de frecuencias. Es por ello, que el principal objetivo de las investigaciones en telefonía a nivel mundial ha sido resolver este problema adoptando un enfoque cada vez más novedoso. Lo anterior, implica un cambio de paradigma en régimen de periodicidad del espectro de frecuencias que dará lugar a lo que podríamos llamar el espectro caótico. La idea es utilizar el carácter caótico de algunas señales para desarrollar una nueva tecnología de comunicación unificada que vaya más allá de la multiplexación por división ortogonal de frecuencias. (OFDM). Esto último se fundamenta en estudios que han propuesto varios sistemas de comunicación caóticos, entre los cuales destacan: la modulación caótica, enmascaramiento caótico, modulación por desplazamiento de caos (CSK) y técnicas de espectro ensanchado [5] [6] [7] [8].

Por otro lado, LTE representa una tecnología que brinda una mayor velocidad de transmisión de datos, mediante la cual se puede acceder a una gran variedad de servicios eficientemente abriendo paso al desarrollo de nuevos productos que exigen mayores recursos [5].

Es por ello, que esta investigación pretende comparar un sistema de modulación y demodulación convencional bajo el estándar LTE respecto a un nuevo sistema dentro del mismo estandar incluyendo una modulación y demodulación caótica. En este sentido, se decide atacar el problema bajo un ambiente de simulación para observar el comportamiento de los esquemas y proveer información respecto al desempeño del sistema con portadora caótica para su futura implementación en tarjetas FPGA en tiempo real. [7].

Para llevar a cabo este proyecto, se hace uso del software GNU Radio ya que este es un software libre y cuenta con los bloques de procesamiento de señales digitales necesarios para la generación de los esquemas OFDM, así como también de la libertad de crear nuevos bloques en caso de ser necesario.

Por lo antes planteado y con la finalidad de encontrar una solución a estos problemas, se presenta este trabajo especial de grado, el cual se enfoca en la caracterización de un esquema de modulación y demodulación caótico para el estándar LTE con el objetivo de comparar su desempeño con respecto a los esquemas convencionales usados en la actualidad.

1.2. OBJETIVOS.

1.2.1. Objetivo General.

Caracterizar un esquema de modulación y demodulación caótico para el estándar LTE.

1.2.2. Objetivos Especificos.

- Revisar el material bibliográfico referente a los esquemas de modulación y demodulación convencionales para el estándar LTE y los esquemas caóticos aplicados en los sistemas de comunicaciones.

- Simular en software GNU Radio un esquema convencional OFDM bajo el estándar LTE.
- Simular en software GNU Radio un esquema OFDM caótico bajo el estándar LTE.
- Comparar el esquema OFDM convencional con el esquema OFDM caótico bajo el estándar LTE, esta comparación se realizara mediante el parámetro BER y el nivel de ruido AWGN soportado por la señal modulada.

1.3. ALCANCE.

Para cumplir los objetivos planteados en este trabajo especial de grado se caracterizó en GNU Radio un esquema de modulación y demodulación incluyendo la etapa de codificación y decodificación para el estándar LTE. En esta etapa del proyecto se tomaron las siguientes consideraciones:

- No se simuló la totalidad del estándar, solo se seleccionó una sección representativa.
- Los parámetros con los cuales se pudo comparar la eficiencia del sistema se limita al BER, el nivel de ruido que es capaz de soportar la señal modulada y a comparar la señal transmitida con la señal recibida en el caso de que la señal sea una foto o una canción.

Luego se hicieron unos cambios al estándar en cuanto al número de portadoras de datos y de referencia, del mismo modo se moduló caóticamente dichas portadoras de referencia.

Posteriormente se procedió a crear en GNU Radio los bloques necesarios para realizar una modulación y demodulación caótica en banda base para un único usuario, luego se agregó esta etapa en el esquema LTE convencional y posteriormente se midió su desempeño mediante los parámetros antes descritos.

A continuación, se realizó un nuevo esquema que contiene la nueva configuración del estándar y las portadoras de referencia moduladas caóticamente junto con el nuevo modulador y demodulador caótico creado, es decir, se plantea una nueva configuración del estándar donde las portadoras de referencia y la información están moduladas caóticamente.

Por último se procedió a la comparación de los esquemas desarrollados..

Capítulo II

Marco conceptual

2.1. LTE

Cuando la telefonía móvil se estrenó en los años 80's, los equipos solo servían para llamar y ofrecía velocidades de 14.4kbps, esa fue la primera generación de redes móviles (tecnologías como AMPS). La segunda generación, ya digital, aparece a inicios de los años 90 con las tecnologías GSM, TDMA y CDMA, permitiendo velocidades de datos cercanos a los 10kbps. Entre la segunda y tercera generación, aparecieron varias tecnologías como GPRS y EDGE, aumentando las velocidades de datos desde 144kbps hasta 384kbps por celda y permitiendo a los usuarios navegar por internet o descargar imágenes desde sus dispositivos móviles. Con la tercera generación (3G) es cuando realmente llega la banda ancha móvil con tecnologías como WCDMA y CDMA2000/EVDO, las personas desde sus teléfonos comienzan a descargar audio, video, imágenes, etc. La tercera generación siguió evolucionando con tecnologías como HSPA (3.5G o 3G+), permitiendo velocidades máximas por celda aún mayores entre 7.2 y 14.4 Mbps. Hoy en día, los operadores siguen actualizando sus redes con tecnologías HSPA+ para ofrecer velocidades máximas por celda de 21Mbps con una portadora ó 42Mbps con dos portadoras, e inclusive pudieran llegar a 84Mbps con doble sistema de antenas (MIMO2x2). En la práctica, la mayoría implementa solo los 21Mbps o hasta 42Mbps si tienen espectro suficiente. Con estas velocidades por celda, un usuario puede experimentar en promedio

entre 2 y 4Mbps, similares a las conexiones fijas que se tienen en su hogar u oficina [9].

LTE (Long Term Evolution) es un estándar especificado por el proyecto de cooperación en sistemas de tercera generación (3GPP) en el Release 8, cuyo objetivo principal fue crear una arquitectura mucho más plana y sencilla, definida por el 3GPP como Evolución de la Arquitectura del Sistema (SAE, System Architecture Evolution), la cual está soportada totalmente en el Protocolo de Internet (IP, Internet Protocol), con un número limitado de nodos e interfaces que permiten disminuir los tiempos de señalización y procesamiento entre nodos. Además la tecnología LTE cuenta con un planificador de paquetes (packet scheduler) que cumple con funciones de asignación dinámica de recursos, balanceo de carga, gestión de movilidad (mobility management), lo cual le permite adaptarse rápidamente a las condiciones variables del canal de radio por medio de la selección de diferentes esquemas de modulación y tasas de codificación [10].

LTE hace uso, en el enlace de bajada, de la técnica de Acceso Múltiple por División de Frecuencia Ortogonal (OFDMA, Orthogonal Frequency División Multiple Access) y en el enlace de subida de la técnica de Acceso Múltiple por División de Frecuencia de Portadora Única (SC-FDMA, Single Carrier Frequency División Multiple Access). Cabe destacar también que, LTE opera con anchos de banda variables que van desde 1.25 hasta 20 MHz y velocidades de trasmisión de datos teóricos de hasta 100 Mbps para el enlace de bajada y 50 Mbps en el enlace de subida en un ancho de banda de 20 MHz. LTE emplea: Modulación por Desplazamiento de Fase en Cuadratura (QPSK, Quadrature Phase Shift Keying), Modulación en Amplitud y Cuadratura de 16 niveles (16 QAM, 16 Quadrature Amplitude Modulation), Modulación en Amplitud y Cuadratura de 64 niveles (64 QAM, 64 Quadrature Amplitude Modulation), codificación turbo, y sistemas de antenas de Entradas Múltiples y Salidas Múltiples (MIMO, Multiple Input Multiple Output), logrando de esta manera incrementar la eficiencia espectral y la capacidad del sistema [10].

2.1.1. Trama Física LTE

Con la finalidad de entender la estructura física de una trama LTE la siguiente sección esta explicada en función de la última columna de la tabla(2.1)

En el dominio del tiempo, una trama de datos LTE tiene una duración de 10 ms ($T_{frame} = 307200 \cdot T_s$). Cada trama está dividida en diez subtramas de 1 ms ($T_{subframe} = 30720 \cdot T_s$). Esto aplica tanto para el enlace ascendente como para el descendente.

Cada subtrama consta de dos slots de 0,5 ms cada uno ($Tslot = 15360 \cdot Ts$). Cada slot a su vez consta de un número de símbolos OFDM que puede ser siete (prefijo cíclico normal) o seis (prefijo cíclico extendido). La Figura (2.1) muestra la estructura de una trama para LTE con un ancho de banda de 20 MHz.

El tiempo útil símbolo es $Tu = 2,048T_s \approx 66,7mS$. Para el modo normal, el primer símbolo tiene un prefijo cíclico de duración $T_{CP} = 160Ts \approx 5,2mS$. Los seis símbolos restantes tienen un prefijo cíclico de duracion $T_{CP} = 144 \approx Ts4,7mS$. La razón de diferentes longitudes de prefijo cíclico (CP) del primer símbolo es hacer que la longitud total ranura en términos de unidades de tiempo sea divisible por 15360.

Para el modo extendido, el prefijo cíclico es $TCP - e = 512 \cdot Ts \approx 16,7mS$. El CP es más largo que el prefijo cíclico normal por unos pocos microsegundos como se muestra en la Figura (2.2). Los prefijos cíclicos normales son utilizados en las células urbanas y aplicaciones de alta velocidad de datos, mientras que el prefijo cíclico extendido se utiliza en casos especiales como multi celdas de difusión y en células muy grandes (por ejemplo, las zonas rurales, las aplicaciones de baja velocidad de datos). El CP consume parte de la capacidad de la capa física: $7,5\,\%$ en el caso del prefijo cíclico normal. La separación de subportadora es $Df = \frac{1}{Tu} = 15kHz$ [1].

La transmisión de un trama LTE se realiza mediante bloques de recursos (RB) cada uno de los cuales consta de 12 subportadoras consecutivas y están contenidas en un slot (0,5 ms). Este valor es el máximo soportado por un slot para limitar la sobrecarga de señalización.

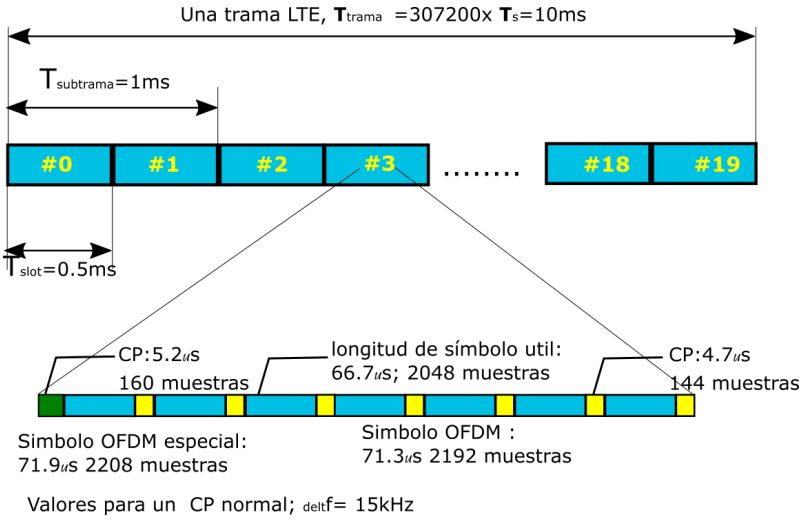

Figura 2.1: Estructura de una trama LTE [1].

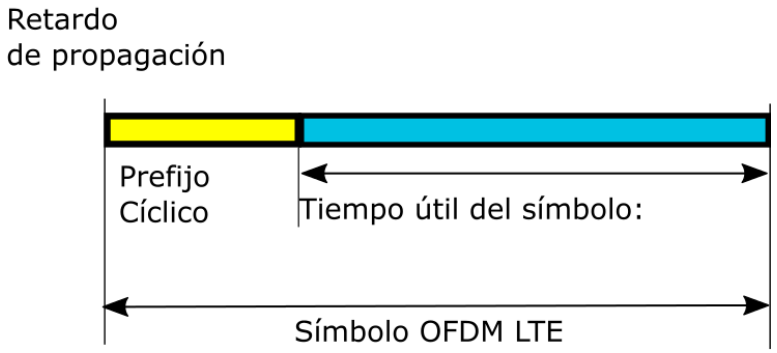

Figura 2.2: Estructura del símbolo LTE [1].

Un elemento de Recursos (RE) es la unidad más pequeña y consta de una subportadora OFDM durante un intervalo de símbolo OFDM. Cada bloque de recursos se compone de $12\ \mathtt{subportadoras} \cdot 7\ \mathtt{simbolos} = 84$ elementos de recursos en caso de prefijo cíclico normal (72 para CP extendido). La Figura (2.3) ilustra la definición de bloques de recursos y elementos de recursos.

Los elementos más importantes de la trama física LTE se ven en la tabla(2.1)

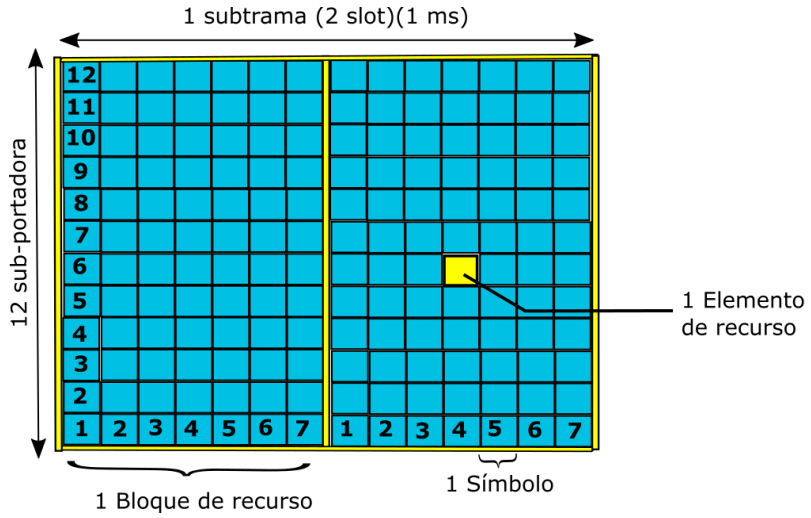

Figura 2.3: Bloque de recurso y elemento de recurso para un CP normal [1].

Tabla 2.1: Parámetros del estándar LTE

Ancho de banda de Transmision BW (MHz)	1.25	2.5	5	10	15	20
Espacio entre subportadoras (KHz)	15	15	15	15	15	15
Muestreo en Frecuencia Fs (MHz)	1.92	3.84	7.68	15.36	23.04	30.72
Longitud de la FFT	128	256	512	1024	1536	2048
Numero de subportadoras utiles sin DC(OT)	72	180	300	600	900	1200
Numero de subportadora de guarda	55	75	211	423	635	847
Numero de Subportadoras piloto	12	25	50	100	150	200
Resource Block	6	15	24	50	75	100
Prefijo Ciclico (CP) en muestas	9	18	36	72	108	144

2.1.2. Principio de la Técnica de Modulación OFDM

El principio básico de OFDM es dividir la secuencia de datos que debe ser transmitida a una velocidad de transmisión de Rs símbolos por segundo, en N sub-canales de datos paralelos, cada uno operando a una tasa de Rs/N símbolos

por segundo. Cada sub-canal, modula una sub-portadora de manera que la velocidad de transmisión total del sistema sea equivalente, a la de una sub-portadora. En general, las frecuencias de las subportadoras utilizadas para transmitir señales multiplexadas en el dominio de la frecuencia deben ser espaciadas un valor mayor que el ancho de banda de cada subportadora, o sea:[2]

$$\triangle f > \mathbf{BW_{sp}} \tag{2.1}$$

$$\triangle f > \frac{\mathbf{BW_s}}{N} \tag{2.2}$$

$$\triangle f > \mathbf{2R_m} \tag{2.3}$$

Donde BW_{sp} es el ancho de banda ocupada por una sub-portadora y R_m es la tasa de señalización de una sub-portadora. BW_s es definido como:

$$\mathbf{BW_s} = \frac{\mathbf{R_b}}{\mathbf{log_2(M)}}(\mathbf{1}+\alpha) = \mathbf{R_s}(\mathbf{1}+\alpha) \tag{2.4}$$

Donde R_B es la tasa de bit necesaria para garantizar la calidad de servicio del sistema, M es el orden de la modulación empleada, R_S es la velocidad de transmisión en la salida del modulador digital en fase y cuadratura y α es el factor de caída (roll-off) del filtro de Nyquist empleado [11] [12] [13] [2]

Para realizar el espaciamiento entre subportadoras, como fue presentado en la ecuación(2.2), es necesario que el ancho de banda total sea mucho mayor al ocupado por la señal modulada en una única portadora. Para evitar este problema, es necesario que las subportadoras sean sobrepuestas en el espectro de frecuencia sin introducir interferencia entre subportadoras ICI (Intercarrier Interference). Para esto, las subportadoras deben ser ortogonales entre sí, o sea [2]:

$$\int_0^T \mathbf{cos(w_i t)} \cdot \mathbf{cos(w_j t)dt} = 0 \quad \mathbf{i \neq j} \tag{2.5}$$

Donde $\mathbf{T} = \frac{\mathbf{1}}{\mathbf{R_m}} = \frac{1}{T_u}$ es la velocidad de transmisión de cada sub-portadora. T_u= duración útil del símbolo.

Cada sub portadora es ortogonal al resto, esto permite que la amplitud del espectro de una sub portadora dada, sea máxima cuando es cero el resto de las subportadoras. Esto hace que no exista interferencia, aumentado la eficiencia del uso del espectro debido a que no se utilizan bandas de separación entre subportadoras.

En la figura (2.4) se muestra un gráfico con señales multiplexadas en OFDM, las cuales se ven con colores diferentes, y se observa claramente la ortogonalidad entre cada una de ellas a la hora de ser transmitidas, por ejemplo, primero se transmite la sub portadora de color azul, luego ortogonalmente (desfasada 90 grados), se transmite la sub portadora de color rosado, así sucesivamente la verde, la naranja, morada, etc. Estas alcanzan su valor máximo en un punto adecuado para no interferir con las demás subportadoras.

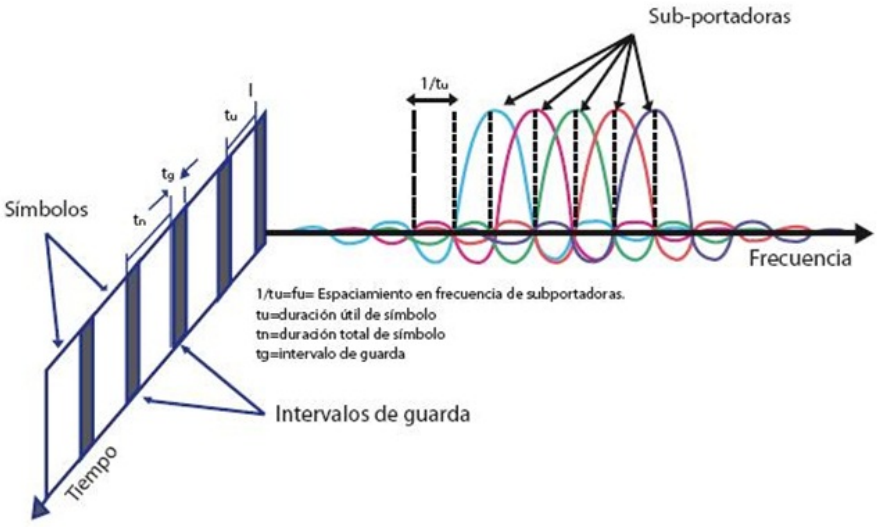

Figura 2.4: Espaciado Entre SubPortadora OFDM [2].

2.1.3. Generación y Recepción de señales OFDM.

El primer abordaje para la generación de señales OFDM consistía en utilizar un conversor serial – paralelo para separar la secuencia de entrada en N sub-canales

de datos. Cada uno de estos sub-canales modula una sub-portadora compleja, formada por un seno y un coseno en la misma frecuencia. La suma de todas las formas moduladas resulta en una señal OFDM. [2]

Una secuencia binaria de datos, $m(t)$, es convertida por un modulador digital de fase y cuadratura en una secuencia de símbolos complejos $cn = in + jqn$. La componente real del símbolo, in , que representa la señal digital en fase, modulada por la coseno de frecuencia ω_n , en cuanto que la componente imaginaria, q_n, que representa la componente en cuadratura, modulada por un seno también de frecuencia ω_n. De esta forma, el símbolo OFDM puede ser expresado por:

$$S(t) = \sum_{i=0}^{N-1} [i_n cos(\omega_n t) + q_n sin(\omega_n t)] \tag{2.6}$$

Como las funciones seno y coseno son ortogonales entre sí, entonces la señal OFDM puede ser detectada utilizando un banco de 2N correlacionadores, tal como se muestra en la figura 2.5

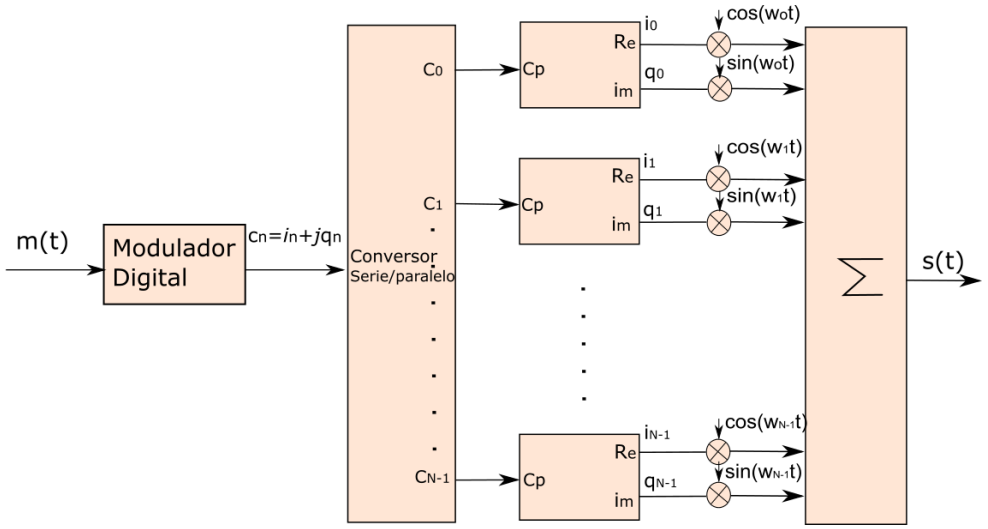

Figura 2.5: Diagrama de Bloques de un Transmisor OFDM. [2]

Suponiendo que, la señal recibida, r(t), sea igual a la señal transmitida, s(t); la información en la k-ésima portadora puede ser recuperada conforme a lo mostrado en.

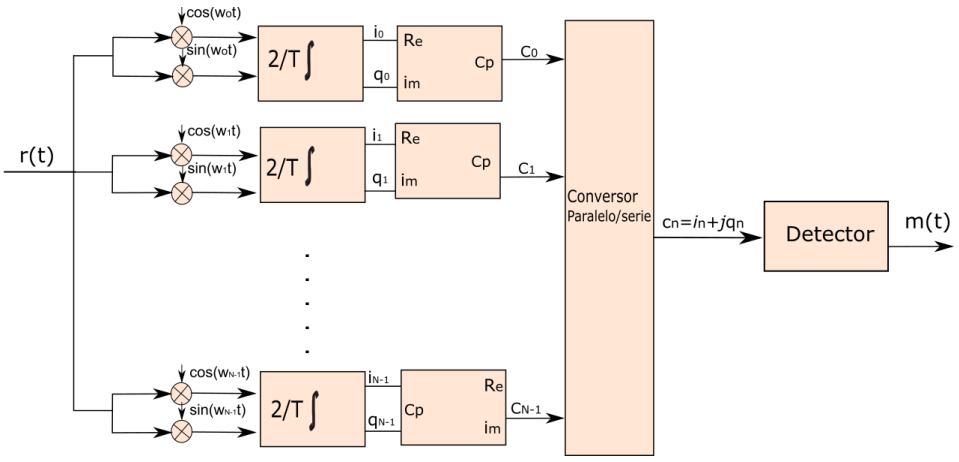

Figura 2.6: Diagrama de Bloques de un Receptor OFDM. [2]

$$i_k = \frac{2i_k}{T}\int_0^T \cos^2(\omega_k t)dt = i_k \tag{2.7}$$

Para que las subportadoras no interfieran entre sí, es necesario que todos los osciladores presentados en la figura (2.5) y figura (2.6) estén perfectamente espaciados de Rm (Hz) y perfectamente sincronizados. Por otro lado, para que OFDM presente ventajas relevantes sobre el sistema de portadora única, es necesario que el número de subportadoras sea elevado. En LTE, está previsto el uso de 128 o 2048 subportadoras. La implementación de este número de osciladores sincronizados, es inviable para fines comerciales. Alternativamente, es posible generar la señal OFDM de una manera más fácil, si la teoría de procesamiento digital de señales fuera aplicada. Analizando la ecuación (2.6), es posible concluir que la señal OFDM puede ser vista como una serie de Fourier limitada de N elementos, donde las componentes de fase y cuadratura son los coeficientes de esta serie. La ecuación (2.6) puede ser reescrita de la siguiente forma:

$$S(t) = \sum_{i=0}^{N-1} R[i_n cos(\omega_n t) - ji_n sin(\omega_n t) + jq_n cos(\omega_n t) + q_n sin(\omega_n t)] \tag{2.8}$$

Donde $R[i_n cos(\omega_n t) - j i_n sin(\omega_n t) + j q_n cos(\omega_n t) + q_n sin(\omega_n t)]$ representa la parte real de $s(t)$.

Muestreando la señal $s(t)$ presentada en (2.8), a una tasa de R_s muestras por segundo, es posible representar la señal OFDM como:

$$S(m) = R \sum_{i=0}^{N-1} [C_n e^{-j\frac{2\pi n m}{N}}] \tag{2.9}$$

Donde m es la posición temporal de las muestras de la señal OFDM.

La ecuación (2.9) muestra que la señal OFDM discreta, puede ser obtenida realizando la IDFT (InverseDiscrete Fourier Transform) de los símbolos c_n. Así, los símbolos c_n pueden ser vistos como el espectro de amplitud del símbolo OFDM.

Para demodular la señal OFDM solo se necesita aplicar la DFT, de la señal OFDM discreta. El tiempo necesario para que el procesador digital realice la IDFT en la transmisión, y la DFT en la recepción es de $T = 1/Rm$ segundos. Con el aumento del número de portadoras, el tiempo necesario para realizar las operaciones involucradas en la IDFT y en la DFT aumenta linealmente, por lo cual el tiempo total para realizar estas operaciones aumenta exponencialmente. Para un número elevado de portadoras, la velocidad de procesamiento necesaria puede no viabilizar, la generación y la recepción de la señal OFDM. Una manera de minimizar el tiempo de procesamiento es utilizar un algoritmo eficiente para el cálculo de la IDFT/DFT. Este algoritmo es denominado de transformada rápida de Fourier FFT (Fast Fourier Transform) y permite que el tiempo de generación/detección de señales OFDM sea reducido, cuando el número de portadoras empleado sea dado por:

$$\mathbf{N} = 2^p \tag{2.10}$$

Donde p es un número entero mayor que cero.

2.1.4. Prefijo Cíclico.

La ISI (Intersymbol Interference) introducida por las muestras pertenecientes al símbolo anteriormente transmitido puede degradar significativamente la transmisión debido a la quiebra de ortogonalidad de la señal, lo que resulta en ICI (Intercarrier Interference).

Una técnica utilizada para solucionar este problema es el uso del prefijo cíclico. El cual adiciona un prefijo antes o después del símbolo resultante de la IFFT. Este prefijo es constituido de la parte final del símbolo resultante de la IFFT, garantizando de esta manera la periodicidad dentro del nuevo símbolo. Debido a esta característica de mantener la periodicidad se da el nombre de prefijo cíclico CP (Cyclic Prefix). La Figura (2.7), muestra el efecto producido por el prefijo cíclico en la señal transmitida [2].

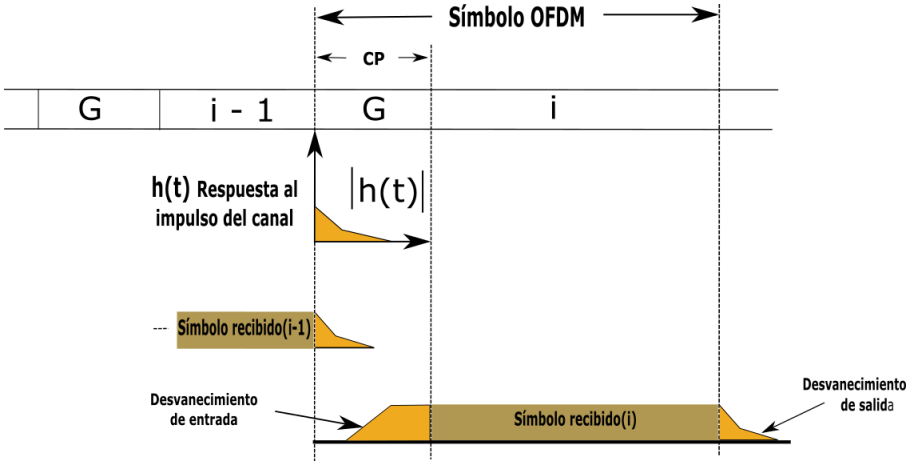

Figura 2.7: Efecto del prefijo cíclico en el símbolo OFDM recibido [2].

2.1.5. Estructura del símbolo OFDM.

La estructura del símbolo OFDM está compuesto por subportadoras las cuales pueden ser: de datos; pilotos que son usadas para estimación de canal; subportadoras nulas que son utilizadas como bandas de guarda; subportadoras no activas y DC. La figura (2.8), muestra la estructura de un símbolo OFDM [2].

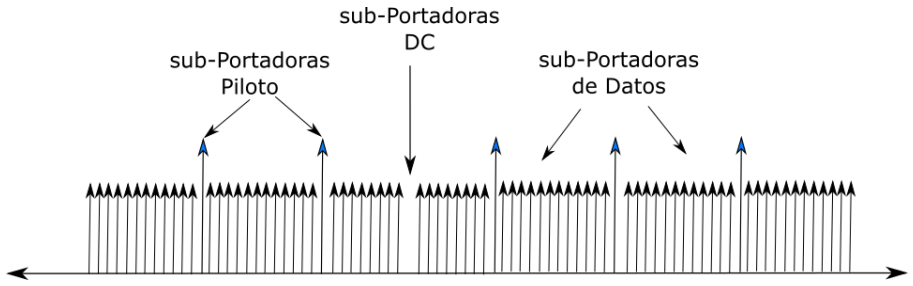

Figura 2.8: Estructura del símbolo OFDM [2].

En los sistemas coherentes, cuando el canal no es conocido en el receptor, son necesarias portadoras pilotos o de referencia para estimar el canal. El uso de bandas de guarda consiste en no transmitir información en las subportadoras OFDM de los extremos con el fin de reducir el espectro de transmisión. Esta técnica es muy útil si la limitación del canal está muy próxima al ancho de espectro, o si por el mismo canal se transmiten diferentes señales OFDM [2].

2.1.6. Diagrama de bloques de un sistema OFDM

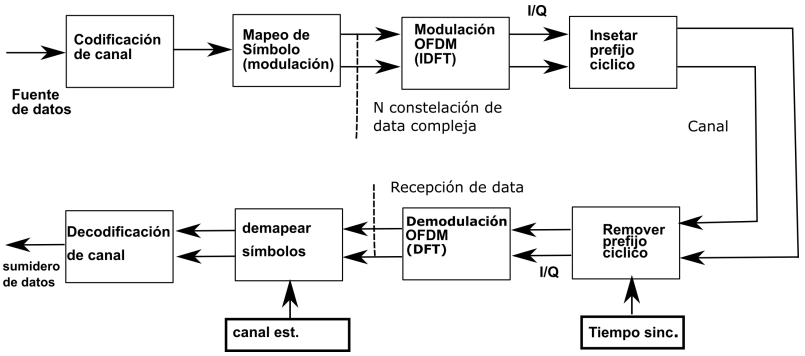

Figura 2.9: Diagrama de transmisión OFDM punto a punto. [3]

La figura (2.9) muestra el diagrama de bloques de un sistema de transmisión simple punto a punto utilizando OFDM y codificación de turbo códigos. Los tres principios fundamentales incorporados son los siguientes

- El IDFT y la DFT se utilizan, respectivamente, para modular y demodular las constelaciones de datos en las subportadoras ortogonales. Se tiene que tener en cuenta que en la entrada de la IDFT, N puntos están presentes como una representación de un diagrama de constelación que se vaya a usar, donde N es el número de puntos de la DFT. Estas constelaciones se pueden tomar de acuerdo con los puntos de una constelación PSK o QAM. Las N muestras en la salida de la IDFT representa la señal modulada que será enviada al canal, hay que destacar que no todas las subportadoras son usadas para transmitir datos, suele tomarse a ambos lados del espectro portadoras de guarda para protección de la señal. Generalmente N es tomado como un entero a la potencia de dos [3] [9].

- El segundo principio clave es la introducción de un prefijo cíclico como un intervalo de guarda, cuya longitud debe superar el exceso retardo máximo del canal de propagación multitrayectoria. Debido a la convolución cíclica, la conexión entre los bloques de transmisor, receptor y de datos es simplemente la multiplicación punto a punto por el canal en las frecuencias de las subportadoras. Así, las subportadoras permanecen independientes, y no se producen interferencia inter portadora (ICI). El uso del prefijo cíclico, como se menciona anteriormente, tiene la propiedad de transformar la convolución lineal con el canal en una convolución circular y, evitar la aparición de ICI. La ecualización (demapear los símbolos) requerido para la detección de las constelaciones de datos es un proceso que se realiza a la salida de la DFT, asi como también se realiza la estimación de canal que proporciona la frecuencia de desplazamiento que agrega el medio por el cual se transmite la informacion. Para esquemas de modulación de fase, la multiplicación por el conjugado complejo de la estimación de canal puede hacer la ecualización [3].

- Codificación de turbó códigos es la tercera idea fundamental aplicada. El canal de radio-frecuencia selectivo puede atenuar severamente a los símbolos de datos transmitidos en una o varias subportadoras, dando lugar a errores de bit. Un esquema de codificación eficiente puede corregir los bits erróneos

y por lo tanto explotar la diversidad de frecuencia del canal de banda ancha [3].

2.1.7. Sistemas de Comunicación MIMO

En concreto, MIMO refiere el número de antenas de transmisión y recepción implicadas en el intercambio de señales inalámbricas a través del canal de propagación o ruta. Así, por ejemplo, 2x2 MIMO indica la disponibilidad de dos antenas emisoras y otras dos en el extremo receptor [14].

MIMO abre una nueva dimensión, en los sistemas de comunicación que utilizan diversidad. El uso de MIMO en los estándares inalámbricos, incluyendo LTE, esta principalmente motivado para el incremento en la velocidad de transmisión, obtenida a través de la multiplexacion espacial (múltiples antenas) [14].

2.1.8. OFDMA

La modulación OFDM también se puede emplear como tecnología de acceso múltiple (OFDMA). En este caso, cada símbolo OFDM puede transmitir información hacia o desde varios usuarios utilizando un conjunto diferente de subportadoras (subcanales).Lo cual, no solo proporciona flexibilidad adicional para la asignación de recursos (aumentando la capacidad), sino que permite la optimización intercapa del uso del enlace radio eléctrico [15].

Este método de acceso permite asignar un número diferente de subportadoras a cada uno de los usuarios garantizando así, una diferente calidad de servicio (QoS) en función del ancho de banda asignado. Es decir, OFDMA permite establecer una velocidad de conexión y una probabilidad de error individualmente para cada usuario [15].

2.1.9. Modulación Digital

2.1.9.1. QPSK (Quadrature Phase-Shift Keying)

Este esquema de modulación es conocido también como Quaternary PSK (PSK Cuaternaria), Quadriphase PSK (PSK Cuadrafásica) o 4-QAM, pese a las diferencias existentes entre QAM y QPSK. Esta modulación digital es representada en el diagrama de constelación por cuatro puntos equidistantes del origen de coordenadas. Con cuatro fases, QPSK puede codificar dos bits por cada símbolo. La asignación de bits a cada símbolo suele hacerse mediante el código Gray, que consiste en que, entre dos símbolos adyacentes, los símbolos solo se diferencian en 1 bit, con lo que se logra minimizar la tasa de bits erróneos.

2.1.9.2. Modulación QAM

La Modulación de Amplitud en Cuadratura o QAM es una modulación digital en la que el mensaje está contenido tanto en la amplitud como en la fase de la señal transmitida. Se basa en la transmisión de dos mensajes independientes por un único camino. Esto se consigue modulando una misma portadora, desfasada 90º entre uno y otro mensaje. Esto supone la formación de dos canales ortogonales en el mismo ancho de banda, con lo cual se mejora en eficiencia de ancho de banda que se consigue con esta modulación.

QAM-16 esta modulación utiliza un alfabeto de 16 símbolos. Por lo tanto, usa palabras de cuatro bits.

QAM-64 esta modulación utiliza un alfabeto de 64 símbolos. Por lo tanto, usa palabras de seis bits.

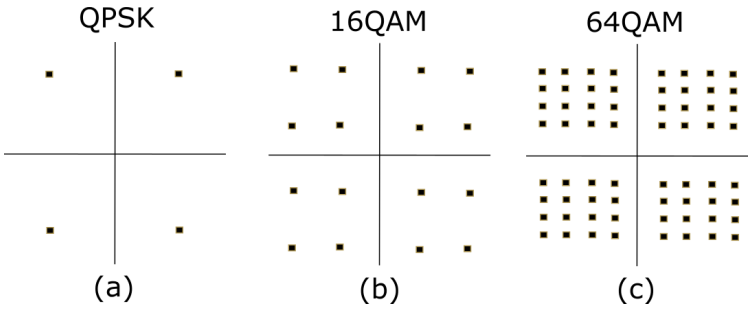

Figura 2.10: a)QPSK, b)16QAM, c)64QAM.[1]

2.1.10. Canales Multitrayectos

El multitrayecto provoca un patrón espacial de interferencias constructivas y destructivas en la señal recibida que varía en el orden de la longitud de onda de la transmisión. Dicho patrón se convierte en temporal debido al movimiento entre transmisor y receptor. Cuanto mayor sea la velocidad del receptor, más rápido se moverá por el patrón espacial y más rápidos serán los desvanecimientos. Estos desvanecimientos pueden ser profundos (de hasta -40 dB con respecto al nivel nominal de señal recibida) luego son perjudiciales pero corregibles mediante ecualización siempre que la forma de señal no se vea distorsionada por desvanecimientos variables dentro del tiempo de un solo símbolo y que dichos desvanecimientos no sean especialmente profundos [16].

El efecto de la variación rápida del nivel de potencia de la señal recibida en el dominio del tiempo se explica desde el punto de vista de la frecuencia, al producirse el conocido como efecto Doppler, cuando el transmisor y/o el receptor están en movimiento [16].

Para la evaluación y análisis de desempeño del enlace de bajada de LTE se emplean los modelos de canal multitrayecto EPA5, EVA70, ETU70, ETU300 [16].

La tabla (2.2) determina la velocidad de desplazamiento promedio para la frecuencia de operación definida y el corrimiento Doppler [16].

Tabla 2.2: Velocidad de desplazamiento promedio para la frecuencia de operación definida y el corrimiento Doppler.

Canal	Corrimiento Doppler [Hz]	Velocidad [Km/H]
EPA5	5	2.16
EVA70	70	30.24
ETU70	70	30.24
ETU300	300	129.6

2.2. Sistemas de comunicaciones caóticos.

Dada la naturaleza de banda ancha de señales caóticas y sus buenas propiedades de correlación, así como la extrema facilidad con la que puede ser generada, se han convertido en la opción preferida para su uso en sistemas de comunicación de espectro ensanchado [17].

Algunos de los generadores caóticos operan en modo continuo, como el circuito de Chua, mientras que otros son sistemas de tiempo discreto. Para los sistemas de comunicación caóticas, una fuente de señal caótica robusta es necesaria para una aplicación de radio en tiempo real. Muchos métodos de implementación de generadores caóticos se han estudiado y propuesto en la literatura [18] [19] [20].

Se han propuesto y evaluado en la literatura muchos sistemas de comunicación basados en el caos. A partir de esos sistemas, el DCSK (Differential Chaos Shift Keying) representa un esquema no coherente robusto en la que la señal caótica en el lado del receptor no tiene que ser conocida con exactitud. Además, el sistema DCSK es uno de los esquemas de comunicaciones basado en el caos más prometedor por la robustez contra las imperfecciones del canal [21].

Un sistema caótico se caracteriza por la dependencia sensible de las condiciones iniciales y la pseudoaleatoriedad. Estas dos características son las más importantes en los sistemas de comunicaciones DCSK, específicamente el generador caótico. Para generar la secuencia pseudo aleatoria se necesita establecer un mapeo que presente un comportamiento caótico los cuales se pueden parametrizar en tiempo

continuo o tiempo discreto. La idea de aplicar la teoría del caos a la aleatoriedad ha estado ligada al desarrollo de las comunicaciones.

Algunos de los mapas caóticos a considerar para ser implementado en este tipo de esquema fueron: Logisticmap, Duffingmap, Lorenz attractor y el mapeo polinomico de chebyshev.

Para esta investigación se decide emplear el polinomio de segundo orden de Chebyshev ya que posee las características para que el sistema sea caótico, las cuales son:

- Debe ser sensible a sus condiciones iníciales.
- Debe ser un sistema flexible y no lineal
- Debe presentar un comportamiento recursivo

2.2.1. Esquema de comunicación DCSK

Un diagrama de bloques de un sistema de comunicación DCSK para un único usuario se muestra en la Figura (2.11),

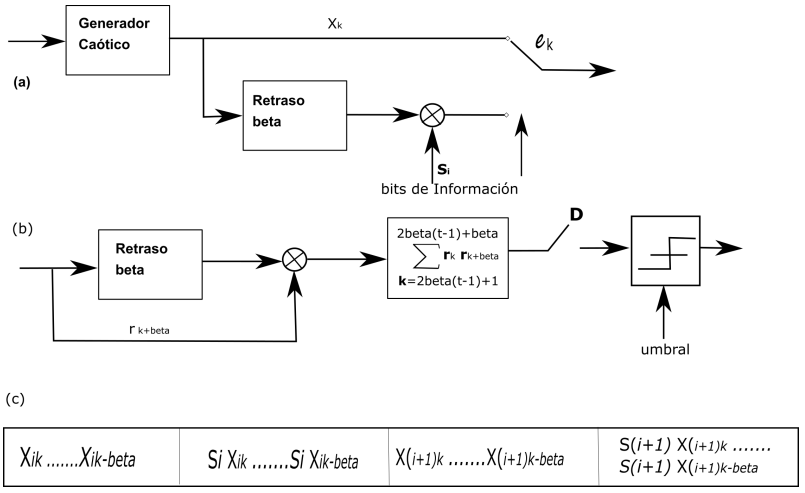

Figura 2.11: a)Transmisor DCSK, b)Receptor DCSK, c)Trama transmitida. [4]

Este sistema utiliza una portadora caótica para difundir la señal digital a través de una amplia banda de frecuencias. La figura (2.11a) representa el proceso de modulación que consta primero de un generador el cual va a producir una cantidad de símbolos caóticos creados por el valor inicial que se especifique en el generador y va a representar la primera mitad del bit modulado, el cual va a tener una duración de cero a 2β muestras. Luego de que el primer intervalo (de cero a β) de muestras caóticas es generada, un switch cambia de posición y comienza a modular la señal a través de una multiplicación de las muestras caóticas procedentes del generador y la información hasta completar la otra mitad del intervalo (de β a 2β) de muestras correspondiente a la totalidad del bit. La figura (2.11b) representa la demodulación de la información que se lleva a cabo mediante una función de correlación, y por último la figura (2.11c) muestra un ejemplo de trama modulada donde el primer slot van los símbolos de referencia y en el segundo slot la referencia por la multiplicación.

Durante la duración del bit i-ésimo, la salida del transmisor es ϵ_k :

$$\epsilon_k = \begin{cases} x_k & for \ 1 < k \leqslant \beta \\ s_i x_{k-\beta} & for \ \beta < k \leqslant 2\beta \end{cases} \tag{2.11}$$

Por otra parte, en el lado del receptor, la señal se contamina con ruido gaussiano blanco aditivo (AWGN) n_k. Finalmente, la señal recibida se modela como:

$$\mathbf{r_k} = \epsilon_k + n_k \tag{2.12}$$

La señal recibida pasa por un correlador donde se correlacionan las muestras de datos correspondientes y de referencia. A continuación, la señal de la salida del correlacionador se calcula para estimar el bit transmitido. Cabe simular que en una implementación práctica muchos parámetros como la sincronización y la corrección de tiempo de muestreo deben tenerse en cuenta para lograr correctamente la demodulación.

Para realizar el modulador caótico se necesita crear dos bloques nuevos, el cual es el generador caótico y el modulador que multiplica la secuencia caótica por la información.

Generador caótico: este bloque genera una secuencia caótica definida por una función polinómica Chebyshev de segundo orden.

$$\mathbf{X_{n+1}} = 1 - 2X_n^2 \tag{2.13}$$

donde:

$$\mathbf{X}\epsilon[-1;1] \quad \text{Es el valor inicial del generador} \tag{2.14}$$

Este mapeo es elegido por la facilidad con la que se genera secuencias caóticas y su buen desempeño [22].

Modulador caótico: posee dos entradas (secuencia caótica, bits de datos) una salida y un parámetro β, que representa el número de muestras caóticas las cuales serán usadas como señal de referencia para un símbolo, entonces un símbolo caótico tendrá un tamaño de 2β muestras.

Demodulador Caótico: La demodulación de un símbolo caótico se realiza desde el signo de la función de correlación seleccionado. La variable de decisión en la salida del correlador para un bit dado i es entonces:

$$\mathbf{D_i} = R\{\sum_{k=1}^{\beta} r_k r_k^* + \beta\} \tag{2.15}$$

Donde $R()$ y $*$ son el valor real y los operadores de complejos conjugados, respectivamente.

El algoritmo de sincronización intenta re sincronizar los símbolos mediante la búsqueda de la mejor relación de auto-correlación entre varias copias retardadas de la señal recibida. Debido a las buenas propiedades de auto correlación de la señal caótica, cualquier desalineación del símbolo (retardo) resultará en un valor muy bajo de correlación entre la referencia y los datos.

Con el fin de controlar la complejidad computacional requerida por este algoritmo de sincronización, se fija un parámetro que limita el número de copias retardadas. Como se muestra en la Figura (3.18) , después de la transposición de banda base, la señal recibida $r(t)$ se muestrea cada KT_C + T_d donde T_c es el tiempo de muestreo y T_d es el tiempo de desplazamiento que viene desde el reloj USRP local, (en este caso T_d es cero ya que no se utiliza un dispositivo USRP pero para una futura implementación en tarjetas FPGA es necesario tenerlo en cuenta. Este tiempo T_d no es constante y varía de un conjunto de bits a otro. Para un bit dado,$(2\beta + 2N)$ muestras de la señal muestreada r_k se almacenan en un acumulador. A continuación, un conjunto de 2N correladores calcula los valores de auto correlación entre cada conjunto de dos ranura de señal retardada con β muestras caóticas [1].

El parámetro N es una función del tiempo de desplazamiento T_d $(NT_C > T_d)$ donde el valor se establece para cubrir todos los retrasos posibles procedentes de la reloj y también el canal de comunicación. Por último, un circuito de decisión toma el máximo del valor de correlación absoluta para seleccionar el bloque de correlación correspondiente y luego calcula su signo para estimar el bit S_i transmitido. Como se muestra en la Figura 15, el almacenamiento de muestras caóticos para el siguiente bit $i+1$ comienza a partir de $(2\beta(i) + 1 + n - N)$ donde n es el índice de demora del bit i. Tenga en cuenta que el valor del índice de n se define a partir de la selección de los mejores bloques de correlación del bit i [1].

2.3. GNU Radio

Existen muchos programas que son usados para abordar el problema de realizar una caracterización de un modulador y demodulador LTE caótico.

De todas las opciones posibles se elige usar el software GNU Radio que es conjunto de herramientas de software de código abierto que ofrece una biblioteca de bloques de procesamiento de señal y la conectividad entre los mismos para construir y simular radio definida por software. Además permite la posibilidad de una

vez realizado el esquemático con los bloques poder compilar ese código y programa una tarjeta que sea compatible con el programa [23].

El software en GNU Radio está dividido en módulos. Estos módulos se pueden dividir en dos grandes grupos. Los módulos que se encargan del procesamiento de la señal necesaria en el sistema, se programan en C ++. Estos módulos de procesamiento de señales pueden ser filtros de señal, ecualizadores, módulos de FFT y así sucesivamente. El otro grupo de módulos incluye el software necesario para interconectar estos módulos de procesamiento de señal y configurarlos según fuera necesidades. Estos últimos se programan en Python y actúan como una especie de pegamento que hace que todo el sistema sea una unidad [24].

GNU Radio posee buffer de almacenamiento en los bloques de procesamiento que permiten alojar los datos antes de ser procesados, ellos se encuentran tanto en la entrada como en la salida del bloque y pueden compartir el tamaño con la entrada o la salida del bloque vecino, el tamaño máximo varía dependiendo el bloque y no puede ser incrementado.

Capítulo III

Procedimientos de la investigación

3.1. Fases de la investigación.

A continuación, se presentan los pasos para llevar a cabo la investigación,

3.1.1. Recopilación de Información.

Se recolectó información de libros, tesis, artículos científicos, etc, los cuales fueron usados para la realización de este proyecto.

Se investigó las características de un esquema de modulación y demodulación OFDM y se estudió el estándar LTE para adaptarlo al esquema OFDM; Las especificaciones más importantes están los tres tipo de modulación QPSK, 16QAM y 64QAM.

3.1.2. Realización de Esquemas Moduladores y Demoduladores para el Estándar LTE.

Se estudiaron las funciones principales del programa, así como también los bloques básicos y sus características para usarlo en el software GNU Radio. Debido

a los múltiples bloques OFDM que posee GNU Radio, se simularon esquemas de modulación y demodulación con todos los bloques que se encontraban disponibles en las librerías para comprobar cual presentaba mejor desempeño y se adaptara al estándar LTE, luego de haber realizado varias pruebas se escogió el esquema que presentó mejor desempeño.

Una vez elegido un esquema principal se buscó conocer los límites de procesamiento en cuanto a Bytes por trama, nivel de ruido AWGN del canal tolerable por la señal y el porcentaje de BER del esquema.

3.1.3. Realización de Esquemas de Modulación y Demodulación Caóticos Bajo el Estándar LTE.

Se crearon nuevos bloques en el software GNU Radio con el fin de realizar una modulación caótica en banda base. Luego se realizaron unos cambios al estándar y se modulo caóticamente la portadora de referencia. Posteriormente se realizó un sistema de comunicaciones LTE caótico integrando los bloques creados. Finalmente se utilizó la nueva configuración del estándar LTE y la portadora caótica junto con los nuevos bloques creados para la modulación caótica con el fin de construir un esquema de comunicaciones basado en la teoría del caos.

3.1.4. Análisis y Comparación de Resultados.

Luego de haber realizado todos los montajes, se simularon dichos esquemas bajo los efectos de un canal AWGN y un modelo dispersivo de tipo Rayleigh para su posterior análisis.

3.2. Etapas de la investigación

En este sentido, esta investigación se divide en cinco etapas principales como se muestra en la Figura (3.1) Cada una de ellas se explica de manera detallada en las siguientes secciones.

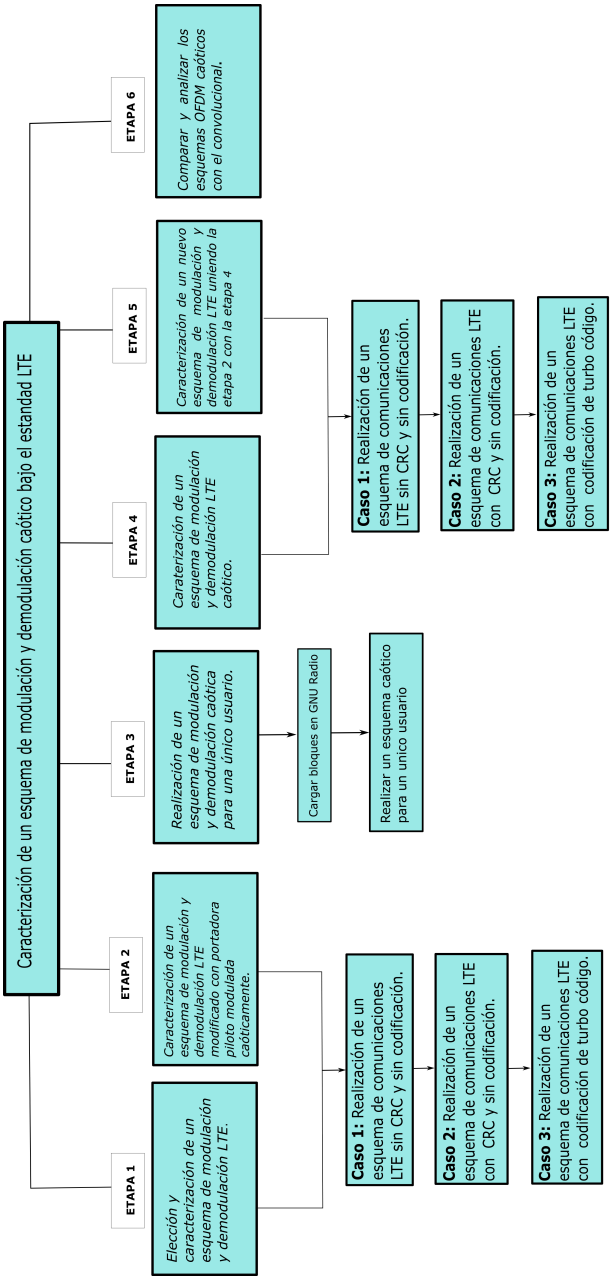

Figura 3.1: Etapas de la Investigación.

3.2.1. Etapa 1: Elección y caracterización de un esquema de modulación y demodulación LTE.

Se realizó mediante la documentación de las etapas que posee un sistema de comunicaciones LTE así como también las características del estándar a emplear en las simulaciones con el fin de comprender la estructura base de dicho sistema

El sistema de comunicaciones comprende los siguientes bloques fundamentales: fuente, codificador, transmisor, receptor y decodificador como lo ilustra la figura (3.2). El codificador para el cumplimento del estándar tiene que ser de turbo código o convolucional con una tasa de codificación de 1/3 y el tipo de modulación empleada por el transmisor es QPSK, 16QAM y 64 QAM

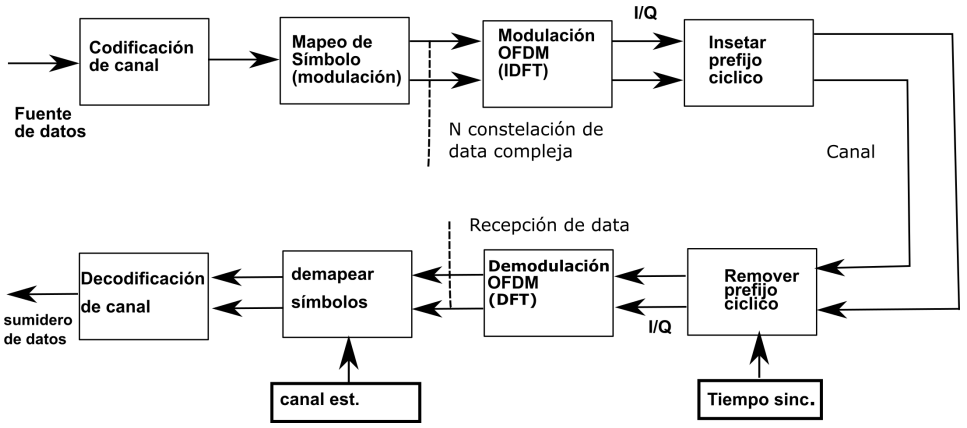

Figura 3.2: Esquema de comunicaciones LTE.

La tabla(2.1) resume los principales parámetros de la trama física para LTE usados en las simulaciones.

Luego de tener claro los parámetros del estándar y el esquema de modulación se procede a estudiar los diferentes bloques de modulación OFDM que posee las librerías de GNU Radio.

Así mismo, se debe señalar que se trabajó con GNU Radio Live SDR, el cual es una distribución de Ubuntu Linux con GNU Radio con *software* pre-instalado para ejecutar GNU Radio sin necesidad de tener previamente instalada alguna distribución de Linux.

El primer bloque analizado fue el OFDM Mod que a pesar de que es un bloque donde se pueden configurar muchas de las características de LTE, no se puede controlar la posición de las portadoras de datos ni tampoco especificar el número y posición de las portadoras piloto. En LTE por cada 6 subportadoras de datos va una portadora piloto. Además de que es un bloque que al contener tantas funciones en uno solo bloque su procesamiento es lento.

El siguiente bloque analizado fue el OFDM Transmitter que es un bloque más flexible y permite configurar todos los parámetros requeridos por el estándar, pero tiene la limitante de que solo puede modular en QPSK y debido a esto fue descartado, aunque posee un desempeño de procesamiento mucho mayor que el anterior.

Debido a estos inconvenientes se decidió estudiar a fondo el bloque OFDM Mod, es decir, abrirlo y ver que bloques comprendían el modulador,por lo que gracias a esta investigación, se logró desarrollar un esquema que aunque comprende numerosos bloques realiza los procesamiento de manera eficiente y rápida ya que al estar separadas las tareas se realizan una por una y no todas en un mismo bloque como sucedía con los dos bloques antes mencionados. Cabe destacar que también cumple con todos los requerimientos del estándar.

En la elaboración de los esquemas de esta etapa se analizaran tres casos por separados: Solo modulación y demodulación, Agregando Bits de redundancia cíclica (CRC) a la modulación y realizando una codificación de turbo códigos antes de la modulación. Con la finalidad de analizar el comportamiento del sistema cuando se van agregando mas bloques.

A continuación se explicará cómo se realizaron los tres esquemas de comunicaciones LTE con los bloques de la librería de GNU Radio.

3.2.1.1. Caso 1: Esquema de comunicaciones LTE sin CRC y sin codificación.

Este caso presenta un sistema de comunicaciones LTE como lo muestra el esquema de la figura (3.2) sin ningún tipo de codificación ni bits de redundancia. Este esquema está compuesto por los siguientes bloques:

Fuente: se empleó un bloque de nombre File Source que permite enviar un archivo, para este caso se utilizaron imágenes. Existen otros tipo de fuentes como lo son la Random Source que envían una secuencia aleatoria de muestras o la Vector Source que permite crear un vector de bits a transmitir, pero se eligió la File Source ya que se quiso hacer las simulaciones lo más real posible.

Transmisor: para la realización del transmisor se emplearon 3 bloques del esquema de la figura(3.2). Antes de entrar en detalles con el transmisor, se debe señalar como está estructurado un paquete de datos en GNU Radio, un paquete consta de los siguientes elementos:

- Un preamble. Este se utiliza para la detección, la sincronización (en tiempo y frecuencia) y la estimación posiblemente inicial del estado de canal.
- Una cabecera. Es de longitud fija y almacena información sobre el paquete, como su longitud (más importante), pero potencialmente otra información, tal como el número de paquete, su destinatario etc.
- Payload o carga útil.
- Una suma de comprobación, por lo general un valor CRC, para validar el contenido del paquete.

A continuación en la figura (3.3) se muestra de manera detallada los tres bloques que conforman el transmisor

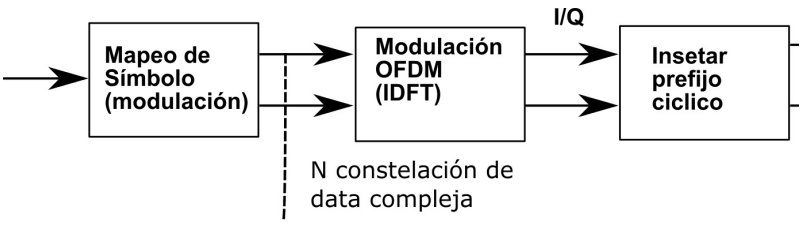

Figura 3.3: Transmisor OFDM.

Receptor: para la realización del receptor la señal a demodular requiere pasar por tres fases de la figura (3.2) entre las cuales están, quitar el prefijo cíclico, demodular la señal OFDM y retirar los símbolos mapeados en el transmisor (demodulación digital), como lo muestra la siguiente figura(3.4).

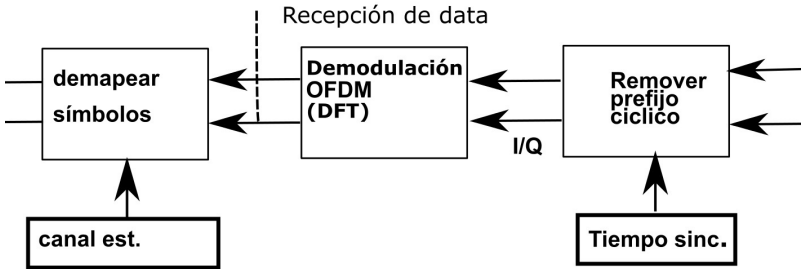

Figura 3.4: Receptor OFDM.

A continuación se explica detalladamente los bloques que se usaron en GNU Radio para el transmisor.

Mapeo de símbolos: En esta sección corresponde el mapeo o modulación digital tanto de la trama de datos como la de la cabecera para luego ser multiplexada. En la figura (3.5) se muestra los bloques empleados en GNU Radio.

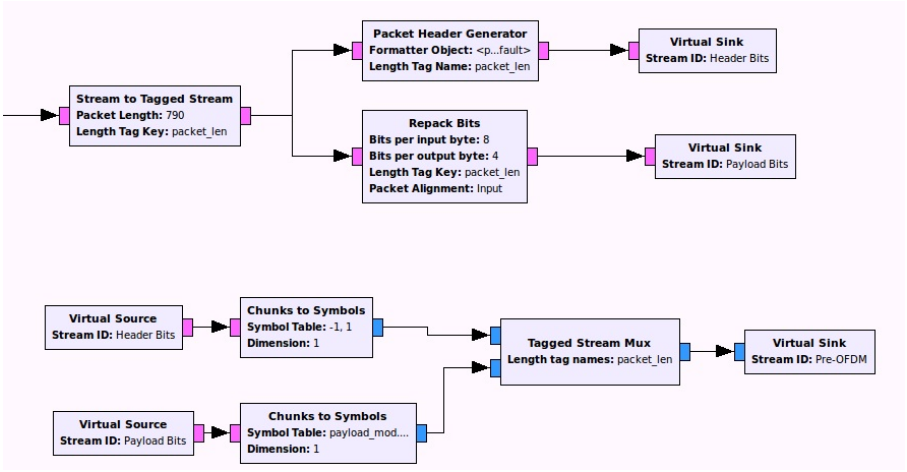

Figura 3.5: Mapeo de símbolos.

En la figura (3.5) los Bytes proveniente de la fuente se etiquetan en N cantidad de Bytes y se les asigna un nombre o clave para que el sistema posteriormente

conozca el *stream* de datos al cual le realizara las operaciones, a continuación se genera una cabecera (*header*) y se separan los N bits para el mapeo de la *payload* (N depende de la modulación a usar). Esta *header* tiene una longitud fija y se puede modular a conveniencia y tiene como función almacenar la información sobre el paquete como lo es la longitud del paquete y el número del paquete. Para posteriormente ser mapeado tanto la cabecera como la *payload* y luego se unen mediante un multiplexdor.

Bloques empleados:

- *Stream to Tagged Stream*: convierte un flujo regular de datos en un flujo de datos etiquetado. Todo este bloque pertime agregar etiquetas de longitud en intervalos regulares. La longitud se asigna en Bytes. Un paquete de datos consta de N bytes. Sin embargo, en los bloques de GNU Radio, si se envía un tren de N bytes a un bloque, no hay manera de saber el límite de paquetes. El modulador tiene que anteponer una palabra de sincronización antes de cada paquete, o añadir un CRC. Así, mientras que algunos bloques no se preocupan por límites del paquete, otros bloques si lo hacen.
- *Packet header generator*: genera una cabecera de tamaño 29 bits, los bits 0-11 contiene el tamaño de los N bytes etiquetados, los bits 12-27 tiene la cantidad bits totales de la cabecera y el bit 28 es un bit de paridad.
- *Repack bits*: empaqueta bits de un flujo de entrada en flujo de bits de salida. Ningún bit es descartado en este bloque, el rango de bits que se puede empaquetar está en el intervalo [1-8], se lee desde el bit menos significativo y se empieza a copiar también desde el menos significativo. Si una etiqueta de datos es detectada, el bloque asume una secuencia marcada, si el número de bits relevantes a la entrada de este bloque no es múltiplo del número de bits de entrada, se procede a rellenar con ceros hasta completar. Para el caso del estándar LTE por cada 8 bits de entrada salen 2, 4 o 6 bits a la salida dependiendo de qué modulación se vaya a usar QPSK, 16QAM o 64QAM.
- *Virtual Sink*: funciona como un nodo de finalización o sumidero, se le proporciona un nombre al bloque.

- *Virtual Source*: funciona como una fuente, los datos de esta fuente provienen del Virtual Sink que contenga el mismo nombre.
- *Chunks to Symbols*: mapea un flujo de datos de acuerdo a la constelación indicada, para esta investigación se utiliza la constelación de Grey para QPSK, 16QAM y 64QAM y la dimensión (D) tiene que ser 1.
- *Tagged Stream Mux*: combina flujo de datos etiquetados. Toma N flujos como entrada. Para este montaje la primera entrada se coloca la cabecera y la segunda entrada la *payload*. A la salida se tiene un único flujo de datos el cual está compuesto primero por la cabecera y luego la *payload*.

Modulación OFDM: en esta sección se toma los símbolos escalares modulados y se transforman en vectores, los cuales serán la entrada para una transformada rápida inversa de Fourier. También se asignara la cantidad y posición de las portadoras de datos y piloto. En la figura (3.6) se muestra los bloques empleados en GNU Radio.

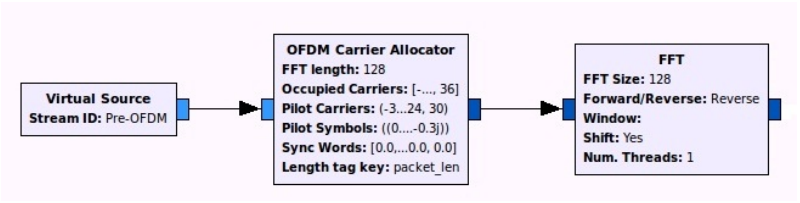

Figura 3.6: Modulación OFDM.

La trama (*header* + *payload*) es modulada por un esquema OFDM bajo el estándar LTE el cual asignará dependiendo el ancho de banda la cantidad de portadoras disponíbles (FFT), las portadoras que llevarán la información (*Occupied Carrier*), las portadoras pilotos y como se modularán esas portadoras pilotos (*Pilot Carriers* and *Pilot symbols*) por último se creará un preámbulo (*preamble*), el cual se utiliza para la detección, la sincronización y la posible estimación del canal, este preámbulo viene dado por dos *Sync Words* cuya longitud debe ser del mismo tamaño de la FFT y su construcción viene dada por el trabajo [25].

La sincronización se basa en la búsqueda de un símbolo de entrenamiento con dos mitades idénticas en el dominio del tiempo, que permanecerá igual después de pasar a través del canal, excepto que habrá una diferencia de fase entre ellas causadas por el desplazamiento de frecuencia de la portadora. Las dos mitades del símbolo de entrenamiento (*Trainning Symbol*), en nomenclatura de OFDM un símbolo se refiere a todas las subportadoras (FFT), se construyen igual (en orden de tiempo), mediante la transmisión de una secuencia de pseudoruido (PN), en las frecuencias pares, mientras que los ceros se usan en las frecuencias impares. Esto quiere decir que en cada frecuencia por uno de los puntos de una constelación QPSK es transmitido. Con el fin de mantener la energía de la señal aproximadamente constante para cada símbolo de los componentes de la frecuencia de este símbolo de entrenamiento se multiplica por raíz de dos en el transmisor, o los cuatro puntos de la constelación QPSK se toman de una constelación mayor, como la 64 QAM, de manera que se puedan utilizar los puntos con mayor energía. Los datos transmitidos no serán confundidos con el inicio de la trama ya que los datos reales deben contener frecuencias impares.

Bloques empleados:

- *OFDM Carrier Allocator*: crea símbolos OFDM en el dominio de la frecuencia de valores complejos y añade portadoras pilotos. Este bloque convierte símbolos complejos modulados en vectores los cuales son la entrada para aplicar la IFFT en un transmisor OFDM. También soporta la posibilidad de ubicar símbolos pilotos en las portadoras. Las portadoras pueden ser ubicadas libremente, si no son ubicadas se colocan como cero. Los parámetros a especificar en el bloque son: el número de la FFT (*FFT Length*), la posición de las portadoras con datos a utilizar (*Occupied Carriers*), el lugar donde se ubican las portadoras pilotos (*Pilot Carriers*), la cantidad de símbolos con que se van a modular las portadoras pilotos (*Pilot Symbols*). El número de símbolos debe coincidir con el número de portadoras pilotos, las palabras de sincronización a usar (*Sync Word*) normalmente son 2 y cada una es del tamaño de la FFT. Por último se debe especificar el nombre del flujo de datos al cual se le vaya a aplicar este proceso (*Lenght Tag Key*)

- FFT: aplica la transformada inversa de Fourier a un flujo de vectores. Para lograr esto se tiene que especificar en el bloque la opción *Reverse*.

Insertar Prefijo Cíclico: Para completar la modulación se agregara la longitud del prefijo cíclico que viene dada por el estándar. En la figura (3.7) se muestra los bloques empleados en GNU Radio.

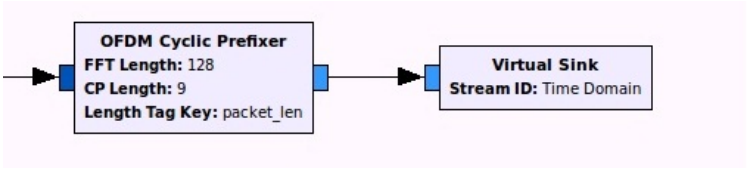

Figura 3.7: Prefijo Cíclico.

Bloques empleados:

- *OFDM Cyclic Prefixer*: Añade un prefijo cíclico y realiza conformación de impulsos de símbolos OFDM. A la entrada se tienen símbolos OFDM en el tiempo luego de aplicar la IFFT. Se puede aplicar el prefijo cíclico a un flujo de datos etiquetados, se debe especifica dicha etiqueta

Canal: La información ya modulada se pasara por un canal que simula el AWGN y pasar por un simulador de modelo de desvanecimiento tipo Rayleigh que pondrá la trama modulada bajo los efectos de la frecuencia Doppler que puede presentar el medio, dicho simulador de desvanecimiento esta modelado por el siguiente trabajo [26]. En la figura (3.8) se muestra los bloques empleados en GNU Radio.

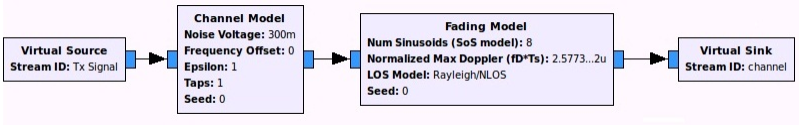

Figura 3.8: Canal.

Bloques empleados:

- Channel Model: Este bloque implementa un simulador básico que modela un canal que puede ser utilizado para ayudar a evaluar, diseñar y probar varias señales, formas de onda, y algoritmos. Este modelo permite al usuario ajustar la tensión de una fuente de ruido AWGN.
- Fading Model: Este algoritmo implementa el método descrito en el [27] para simular los efectos de desvanecimientos Rayleigh.

A continuación se muestran los bloques usados en GNU Radio para el receptor.

Remover Prefijo Cíclico y Sincronización: Para la demodulación se inicia con el proceso de sincronización el cual detecta primero las *Sync Word* para mandar una señal de reloj al demultíplexor que junto con la trama modulada y otra señal que indica el tamaño de la cabecera procede a separar la cabecera de la *payload*. En la figura (3.9) se muestra los bloques empleados en GNU Radio.

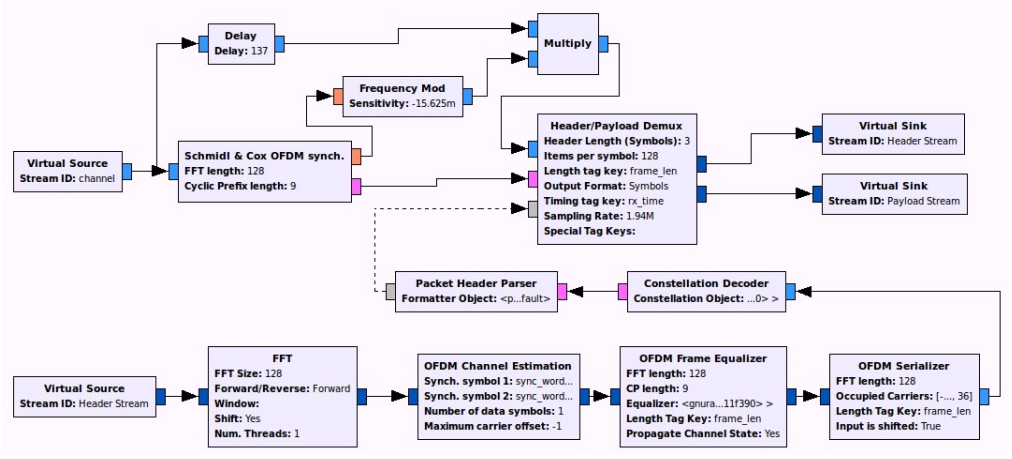

Figura 3.9: Sincronización, Detección y Demultiplexación.

Bloques empleados:

- *Delay*: retrasa la entrada cierto número de muestras. Un retraso positivo inserta ceros al principio del flujo de datos y un retraso negativo descarta muestras. Para esta investigación el *delay* tienes que ser igual al número de la FFT mas el número de muestras de prefijo ciclico (CP), ya que si no se usa esta cantidad de muestras para el retraso, se producen errores de sincronización.

- *Schmidl and Cox Synchronisation for OFDM*: a la entrada se tienen muestras complejas. Se tiene dos salidas, la salida 0 (morado) se tiene el desplazamiento en frecuencia, escalado por la duración del símbolo OFDM. La salida 1 (naranja) se tiene el comienzo del primer símbolo OFDM sin las palabras de sincronización.

- *Frequency Mod*: multiplica señales, desplaza en frecuencia.

- *Header/Payload Demux*: separa la cabecera y la *payload* del flujo modulado. En la primera entrada (azul) recibe la señal modulada en flujo de muestras de datos que provienen del dispositivo receptor. Este bloque descarta todas las muestras hasta que llegue la señal del trigger que es la segunda entrada (morado), cuando la señal del trigger, es detectada, el demultiplezador copia la *header* y el *preambler* en la primera salida (*header*). El bloque entonces espera hasta recibir un mensaje en la tercera entrada (Gris). Este mensaje debe ser del tipo *PMT dictionary* luego la *payload* se copia en la segunda salida (*payload*).

Demodulación OFDM: Luego que la señal es separada, se aplica el proceso inverso a la modulación. Para esta parte hay que demodular la cabecera y la trama por separado. En la figura (3.10) se muestra los bloques empleados en GNU Radio.

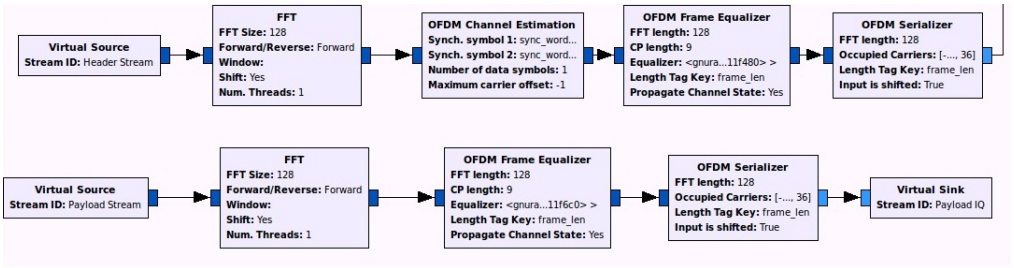

Figura 3.10: Demodulación OFDM.

Bloques empleados:

- *OFDM Channel Stimation*: estima el canal y la frecuencia de desplazamiento. A la entrada tiene símbolos OFDM (en el dominio de la frecuencia). El primero

o los primeros dos símbolos son esperados para la sincronización de símbolos, los cuales son usados para estimar la frecuencia de desplazamiento. A la salida se tiene el flujo de bits sin la palabra de sincronización.

- *OFDM Frame Equalizer*: elimina la frecuencia de desplazamiento. Realiza la ecualización de una trama OFDM etiquetada. A la entrada de este bloque se conecta un flujo de símbolos OFDM etiquetados, a la salida se tiene el mismo flujo de entrada pero ecualizado y con la frecuencia corregida, la etiqueta que contiene el desplazamiento en frecuencia no se elimina.
- *OFDM Serializer*: realiza la operación inversa al OFDM *CarrierAllocator*, remueve los símbolos pilotos, a su salida un flujo de datos etiquetado. Se puede especificar dos diferentes etiquetas, para este montaje una para la cabecera o a la *payload* o las dos al mismo tiempo.
- *Constellation Decoder*: decodifica puntos complejos de una constelación de una trama de bits, basado en el mapeo de Grey.
- *Packet Header Parser*: es el bloque inverso al Packet Header Generator. La diferencia es que la cabecera se analiza como un diccionario y no como un flujo de bits.

Demapeo del símbolo: se demodula la trama de bits de informacion. En la figura (3.11) se muestra los bloques empleados en GNU Radio.

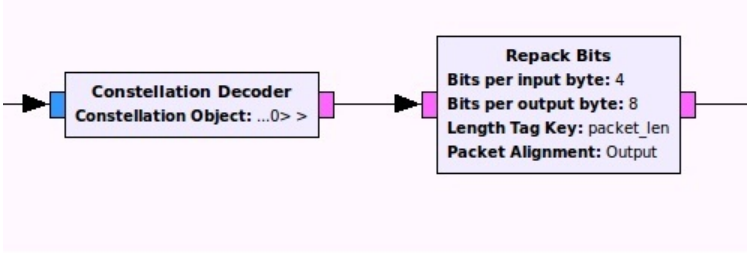

Figura 3.11: Demapeo del símbolo.

Bloques empleados:

- *Constellation Decoder*: decodifica puntos complejos de una constelación de una trama de bits, basado en el mapeo de Grey.
- *Repack K bits*: se encarga de empaquetar K bits para este montaje se usa este empaquetador de bits, este bloque permite que salgan 8 bits por Byte, para la entrada el número de bits depende de la modulación empleada. Ejemplo si entran dos Bytes y k = 8 Entrada = [0xf5, 0x08] Salida = [1,1,1,1, 0,1,0,1, 0,0,0,0, 1,0,0,0].

Para evaluar el desempeño se usó el bloque *Bit Error Rate* el cual calcula el BER entre dos señales. Hay que especificar cada cuantas muestras se quiere realizar el cálculo del BER (*Windows Size*) y el número de bits por simbolos de las señales.

3.2.1.2. Caso 2: Esquema de comunicaciones LTE con CRC y sin codificación.

Para este caso, se mantendrá el mismo esquema modulador y demodulador anterior, solo se agregaron bits de redundancia cíclica, como se muestra en la figura (3.12), el cual es un código de detección de errores usado frecuentemente en redes digitales y en dispositivos de almacenamiento para detectar cambios accidentales en los datos. Trabaja al nivel de mensaje, agregando varios caracteres de control al final, siendo lo más común 2 o 4 Bytes de control. Para este caso se usara el de 4 Bytes.

El esquema no varía mucho en cuanto al de la figura (3.2) , solo cambiar el codificador por un bloque que genere el CRC y el decodificador por un bloque que verifique el CRC. En la figura (3.13) se muestra los bloques empleados en GNU Radio.

El generador de CRC se compone por tres bloques en GNU Radio los cuales son:

- *Tagged Stream to PDU*: convierte el *stream* de data en PDUs. El mensaje enviado es un par PMT. El primer elemento es un diccionario que contiene todas las etiquetas. El segundo es un vector que contiene la data

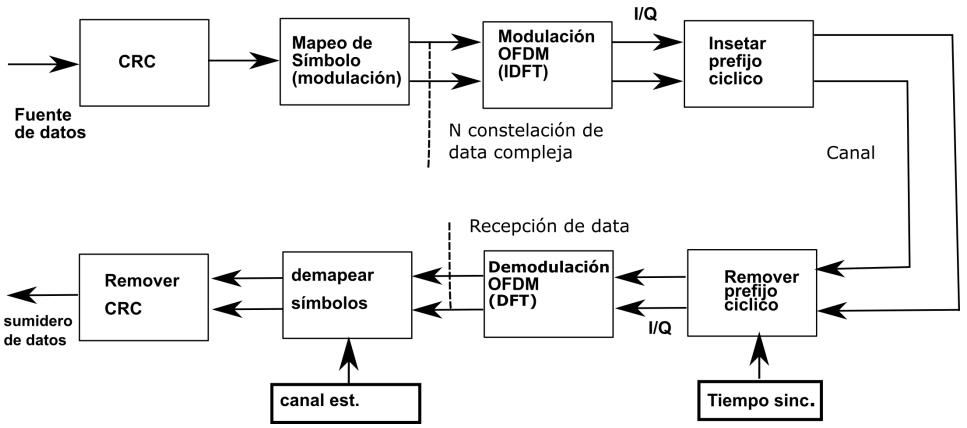

Figura 3.12: Esquema de comunicaciones LTE con CRC.

- *Async CRC32*: procesa paquetes de datos para CRC32. A la salida del bloque se tiene la misma data solo que será 4 Bytes o 32 bits más larga
- *PDU to Tagged Stream*: realiza la operación inversa al *Tagged Stream to PDU*. Convierte el PDU en un *stream* de data etiquetada.

Cabe destacar que el nombre de la etiqueta en cada uno de los bloques tiene que ser igual.

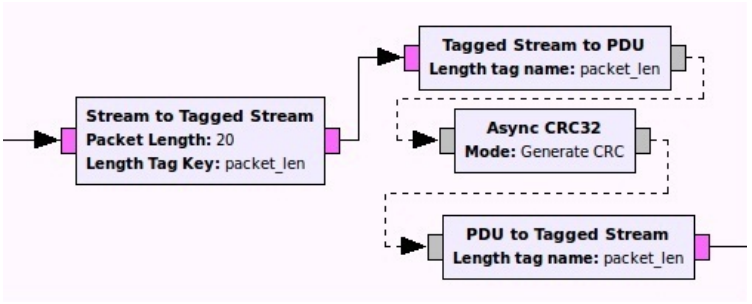

Figura 3.13: CRC.

Para chequear y remover los bits de redundancia cíclica solo se necesita un bloque en GNU radio, el cual es siguiente:

- *Stream CRC32*: remueve los bits de redundancia cíclica creada al principio de la transmisión. Para que realice esta operación hay que seleccionar la opción $\ll$ *Check* $\gg$.

La finalidad de emplear bits de redundancia cíclica en un esquema LTE, a pesar de que el estándar no especifica el uso del mismo, es netamente experimental ya que se quiere observar y luego comparar con la codificación turbo el desempeño de agregar bits al final de la trama en vez de mezclarlo.

3.2.1.3. Caso 3: Esquema de comunicaciones LTE con codificación de turbo código.

Para este caso, se realizó una codificación de canal típica al estándar al sistema de comunicaciones LTE convencional como se ve en la figura(3.2), existen dos tipos la convolucional y la de turbo códigos, se decidió emplear la de turbo código ya que tuvo mejor desempeño en las simulaciones con tasa de codificación de 1/3. Al final del demodulador, se agrega el bloque que decodifica la señal. En la figura (3.14) se muestra los bloques empleados en GNU Radio.

El codificador está compuesto por 4 bloques en GNU Radio los cuales son los siguientes:

- *Throttle*: es un regulador de muestras para que tasa de muestras promedio no exceda las muestras por segundo.
- *Stream to Tagged Stream*: etiqueta una cantidad de bytes. Es necesario crear esta etiqueta ya que el siguiente bloque necesita conocer la cantidad de bytes con que va a trabajar.
- *Repack bits*: es el bloque fundamental para que el codificador funcione. Ya que regula la cantidad de bits. Al trabajar con un codificador 1/3 se necesita que a su entrada entre 1 bit, por lo mismo se configura para que a la salida de este bloque salga 1 bit por Byte, si la cantidad de bits es mayor la simulación se cancelará.

- *PCC Encoder*: bloque encargado de realizar la codificación. Este bloque está compuesto por dos bloques convolucionales en paralelo con tasa de 1/2.

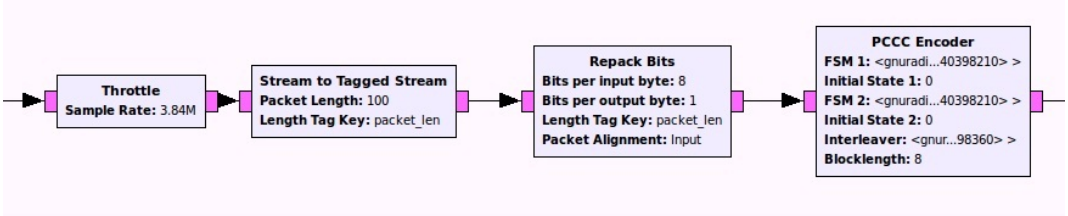

Figura 3.14: Codificador Turbo Código.

Para la decodificación de la señal, solo se necesitan 3 bloques en GNU Radio.

- *Char to Float*: es un cambio de variable de tipo `char` a tipo `float` para poder enviar la señal al decodificador
- *PCC Decoder Combo*: realiza todo el proceso de decodificar la señal
- *Repack bits*: realiza la función inversa al bloque *Repack bits* del codificador, para que a la salida se tenga un Byte de 8 bits.

En la figura (3.15) se muestra los bloques empleados en GNU Radio

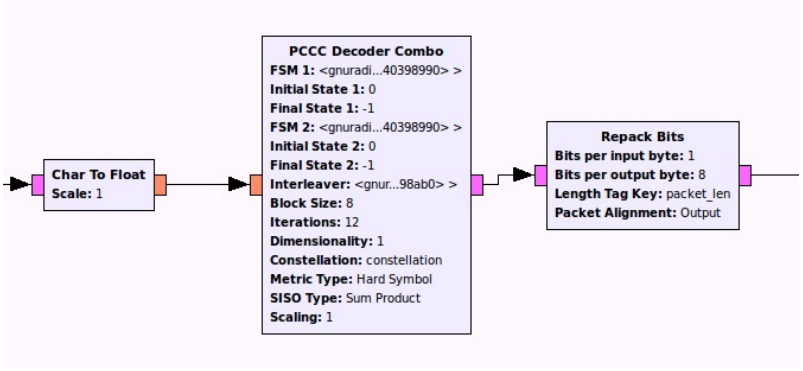

Figura 3.15: Decodificador Turbo código.

3.2.2. Etapa 2: Simulación y evaluación, de un esquema de modulación y demodulación LTE modificado con portadora piloto modulada caóticamente.

En esta etapa, primero se presentó un nuevo diseño, el cual es un esquema LTE modificado con portadoras piloto moduladas caóticamente. En el lado transmisor, todas las subportadoras ocupadas están agrupadas en varios grupos (L grupos). En cada grupo, una subportadora se asigna para transmitir el slot de referencia, mientras que las otras subportadoras (M subportadoras) lleva las subportadoras de datos. Este diseño aumenta la velocidad de datos y ahorra la energía de bit transmitido porque una referencia caótica se utiliza para transmitir una cantidad de bits mayor a lo especificado por el estándar. Luego, se analizó el desempeño a través del BER bajo un canal AWGN y un modelo de dispersión Rayleigh.

De acuerdo con el informe técnico 3GPP [28], para un ancho de banda de transmisión dada (B_w), el número de subportadoras ocupadas (N_{sub}) debe ser menor que el tamaño de la FFT (N_{fft}) para OFDM. Siendo N_{pilot} el número de subportadoras piloto (subportadoras de señal de referencia en este trabajo), y N_{data} el número de subportadoras de datos $(N_{pilott} + N_{data} = N_{sub})$. Inspirado en el modelo de distribución de portadoras de OFDM, se propone un nuevo diseño para sistema de comunicación caótico, nombrado como OFDM DCSK, con el fin de aumentar la velocidad de datos. Luego, se transmitieron las señales de referencia moduladas caóticamente en lugar de los símbolos piloto convencionales usados en LTE.

Para el esquema propuesto, primero se dividió las N_{sub} subportadoras ocupadas en L grupos, cada grupo tiene M + 1 subportadoras $(N_{sub} = L \cdot (M + 1))$. Para cada grupo, el centro de la subportadora se asignó a la señal de referencia caótica, mientras que las otras M subportadoras llevarán las señales de información.

Un ejemplo se da en la Figura (3.16) a fin de mostrar el formato de la señal transmitida con claridad. Supongamos que los datos [1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1] se transmiten. Definiendo $N_{sub} = 68, M = 16, L = 4, \beta = 16$. El primer

grupo del nuevo esquema de comunicaciones modficado se muestra en la Figura (3.16).

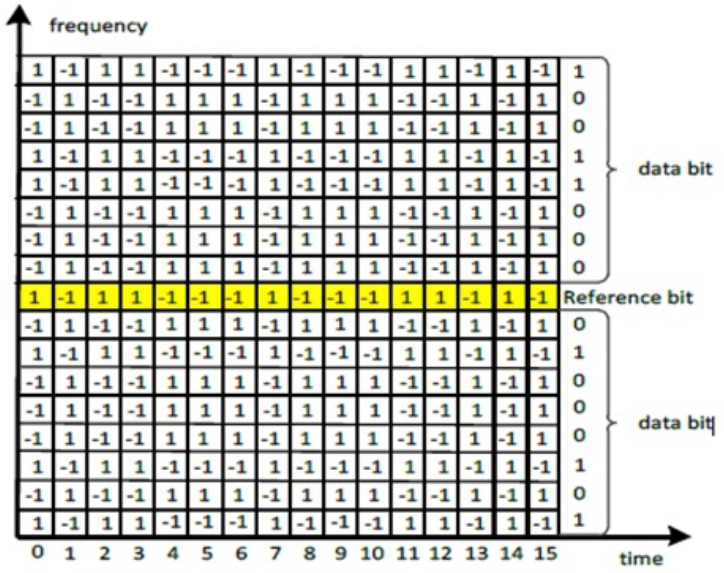

Figura 3.16: Ejemplo del formato de la señal transmitida.[4]

Para la generación de la secuencia caótica se usó una función polinómica Chebyshev de segundo orden definida por la ecuacion(2.13).

La nueva configuración con la cual se trabajó en las simulaciones se pueden ver en la tabla (3.1):

Tabla 3.1: Parámetros para la Simulación de la Etapa 2

Bandwidth(MHz)	1.25	2.5	5	10	15	20
FFT size	128	256	512	1024	1536	2048
N_{sub}(3GPP)	76	151	301	601	901	1201
N_{sub}(this work)	68	153	306	595	901	1190
N_{data}	64	144	288	560	848	1120
N_{pilot}	4	9	18	35	53	70

Para la simulación en el software GNU Radio se simularon los mismos casos que se hicieron en la etapa 1, primero solo el modulador y demodulador, luego agregando bits de redundancia cíclica y finalmente agregando el codificador turbo código, con los mismos bloques anteriormente explicados. Con el fin de hacer una

comparación y analizar su desempeño con respecto a la configuración convencional del estándar LTE.

3.2.3. Etapa 3: Realización de un esquema de modulación y demodulación caótico para un único usuario.

Para la realizar el esquema de modulación y demodulación caótico para un único usuario, primero se debe construir y añadir los bloques caóticos, ya que la librería de GNU Radio no cuenta con dichos bloques caóticos.

El proyecto de GNU Radio permite el desarrollo de bloques de procesamiento de señales que pueden ser escritos tanto en lenguaje de programación Python o C++. Este tipo de módulos son conocidos como *out of tree*, ya que, aunque los módulos serán integrados dentro del catálogo de bloques de GNU Radio no se van a integrar al proyecto para su distribución.

3.2.3.1. Procedimiento para la creación de bloques de procesamiento de señales en GNU Radio

Para crear un módulo *out-of-tree* se ejecuta desde una terminal de linux el siguiente comando:

```
$ gr_modtool create
Name of the new module:Caos
Creating out-of-tree module in ./gr-Caos... Done.
Use 'gr_modtool add' to add a new block to this currently empty module.
```

El nombre que se indique para el módulo es con el que va a aparecer en el catálogo de GNU Radio

Para crear los bloques que integrarán el módulo se utiliza la herramienta `gr_modtool` en la carpeta raíz del proyecto. Quedando en la consola de la siguiente forma:

```
Enter code type: general
Language: C++
Enter name of block/code (without module name prefix): Chaos_Modulator
Block/code identifier: Chaos_Modulator
Enter valid argument list, including default arguments: Chaos_Modulator
Add Python QA code? [Y/n] Y
Add C++ QA code? [y/N] y
Adding file 'lib/Chaos_Modulator_impl.h'...
Adding file 'lib/Chaos_Modulator_impl.cc'...
Adding file 'include/caos/Chaos_Modulator.h'...
Adding file 'lib/qa_Chaos_Modulator.cc'...
Adding file 'lib/qa_Chaos_Modulator.h'...
Editing swig/caos_swig.i...
Adding file 'python/qa_Chaos_Modulator.py'...
Editing python/CMakeLists.txt...
Adding file 'grc/caos_Chaos_Modulator.xml'...
Editing grc/CMakeLists.txt...
ubuntu@ubuntu:~/gr-modtool/gr-caos$
```

De esta manera se repetirá el paso anterior el mismo número de veces como módulos nuevos se requieran crear. Ejemplo si se necesitan dos bloques hay que realizar el paso anterior dos veces. En el ejemplo anterior el nombre del módulo es ≪Chaos_Modulator≫.

Los archivos donde se realizará la programación en código c++ se ubican en la carpeta ≪ lib ≫, dentro de la carpeta ≪ gr_caos ≫ y se distinguen porque son archivos con extensión .cc

Dentro del archivo .cc hay que indicar las librerías que se utilizaran y la cantidad de flujos de datos que manejará el bloque de procesamiento, Luego, se realizó el código que hace el procesamiento. El código final se encuentra en el Apendice [A] .

El archivo .h es el archivo cabecera o header del modulo, es donde se definen las variables que se utilizaran en el archivo .cc

Para la interfaz del bloque en GNU Radio Companion se desarrolla en archivos .xml que se ubica en la carpeta $\ll$ grc $\gg$, dentro de la carpeta $\ll$ gr_caos $\gg$. Dentro del archivo .xml hay varios parámetros que pueden ser personalizados, tal como el nombre del bloque, este parámetro se define en la línea correspondiente a $<$ name $>$.

Ejemplo: $<$ name $>$ ChaosMod $<$ /name $>$.

En la parte de $<$ param $>$ se indica la variable de entrada del bloque

```
<param>
<name>Vector size</name>
<key>vec_size</key>
<type>int</type>
</param>
```

El conector de entrada del bloque de procesamiento se define en $<$ sink $>$. Se puede declarar cualquier tipo de variable de entrada en este ejemplo será float, pero puede de tipo complex, char, short o integer.

```
<sink>
<name>in</name>
<type>float</type>
</sink>
```

El conector de salida en $<$ source $>$. Al igual que la entrada el tipo de salida será de tipo float, pero puede ser cualquiera de las variables que maneja GNU Radio.

```
<source>
<name>out</name>
<type>float</type>
</source>
```

De igual manera el código .xml se podrá ver en el anexo A

Compilación de los bloques

Una vez ya realizado todos los procesos referentes al código de programación de los bloques de procesamiento se procede a integrarlos en el proyecto de GNU Radio. Esto se realiza por medio de una serie de comandos desde una terminal de linux en la raíz del proyecto. En caso de no tener una carpeta de build, se crea por medio de comandos de Linux

```
$ mkdirbuild
$ cdbuild
```

Una vez dentro de la carpeta de build, se procede a compilar el proyecto

```
$ cmake ../
$ make
$ sudo make install
$ sudo ldconfig
```

De esa manera queda integrado el módulo *out-of-tree* dentro de GNU Radio. En el apendice (A) se encuentran los codigos de los nuevos bloques creados.

Una vez finalizado el procedimiento de creación de un bloque se explicara los nuevos bloques caoticos.

En la figura (3.17) se muestra la configuración en GNU Radio del esquema de modulación caótico.

Bloques empleados:

- Generador caótico: genera una secuencia caotica antes explicada. Se debe dar el valor inicial con la cual se comenzara la secuencia, debe estar comprendida entre [-1, +1]
- *Deinterleave*: la salida se tiene la misma secuencia caótica pero intercalada.

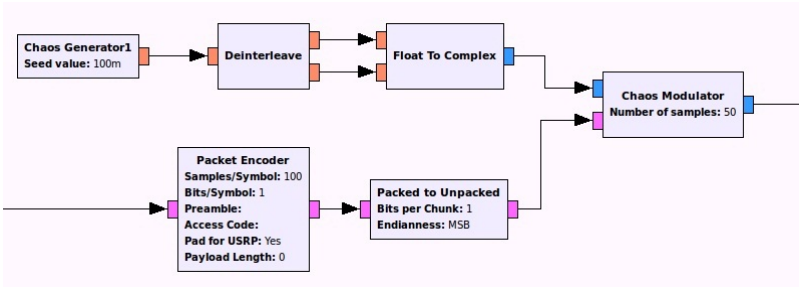

Figura 3.17: Modulador Caótico.

- `FloattoComplex`: cambio de variable de tipo `Float` a tipo `Complex`.
- *Packet Encoder*: en este bloque se inserta el número de muestras a trabajar (2β) y la cantidad de bits por símbolo. Se requiere que se un bit por símbolo ya que se está modulando bit a bit.
- *Packed to Unpacked*: convierte un flujo de Bytes empaquetados para transmitir en bytes descomprimidos. Es decir, agrupa una cantidad de bits para luego transmitirlos a la salida del bloque. Esa cantidad de bits se especifica en el bloque. Para este tipo de modulación esa cantidad es uno ya que se quiere enviar al modulador de bit en bit.
- *Chaos mod*: modula la información como antes fue explicado.

En el caso del demodulador caótico solo se necesita crear un solo bloque que se encargue de hacer la demodulación.

En la figura (3.18) se muestra los bloques necesarios para demodular la señal caótica en GNU Radio

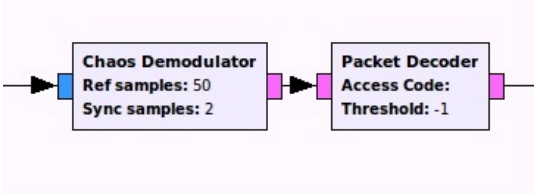

Figura 3.18: Demodulador Caótico.

Bloques empleados:

Demodulador caótico: se encarga de demodular la señal mediante una función de correlación. Hay que especificar el parámetro beta y el parámetro N.

Packet Decoder: realiza la función inversa al Packet encoder.

3.2.4. Etapa 4: Simulación y evaluación de un esquema de modulación y demodulación LTE caótico.

Se decide emplear la modulación caótica después de la fuente y no sustituir la modulación convencional (QPSK, 16QAM y 64QAM) por dicha modulación caótica ya que crea conflictos con el esquema de modulación desarrollado en GNU Radio. Este problema se debe principalmente por tres bloques, el *OFDM frame equalizer*, el *OFDM serializer* y *constellation decoder* debido a que estos bloques fueron creados y configurados para procesar símbolos de modulaciones digitales típicas como lo son: la 8PSK, BPSK, QPSK, 16QAM, 64QAM y 256QAM. Y si se utilizan en un mapeo con símbolos caóticos, el receptor no identificará dicho símbolo y producirán una mala recepción en el receptor.

Para esta etapa, se tomó el modulador caótico y se colocó antes de la modulación OFDM, por último se colocó el demodulador caótico para recuperar la señal original tal como se muestra en la figura (3.19).

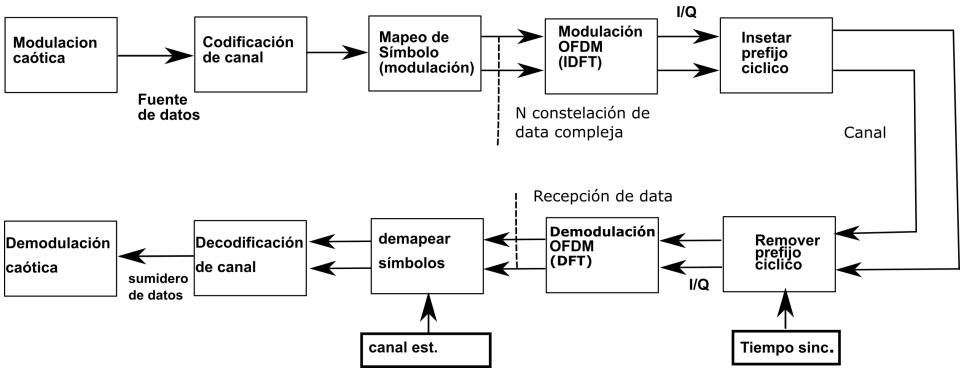

Figura 3.19: Sistema de Comunicación LTE Caótico.

Para las simulaciones, se aplicaron los tres casos, el OFDM más modulación caótica, agregando CRC y por último realizando la codificación de turbo código.

3.2.5. Etapa 5: Caracterización de un nuevo esquema de modulación y demodulación LTE uniendo la etapa 2 con la etapa 4

En la realización de esta etapa se mezcló la nueva configuración para LTE y la portadora piloto caótica de la etapa 2 con la modulación caótica de la información de la etapa 4 con el fin de estudiar el desempeño de un esquema de comunicaciones que emplea la teoría del caos tanto en los bits como en la modulación. La figura(3.20) muestra el esquema caótico propuesto.

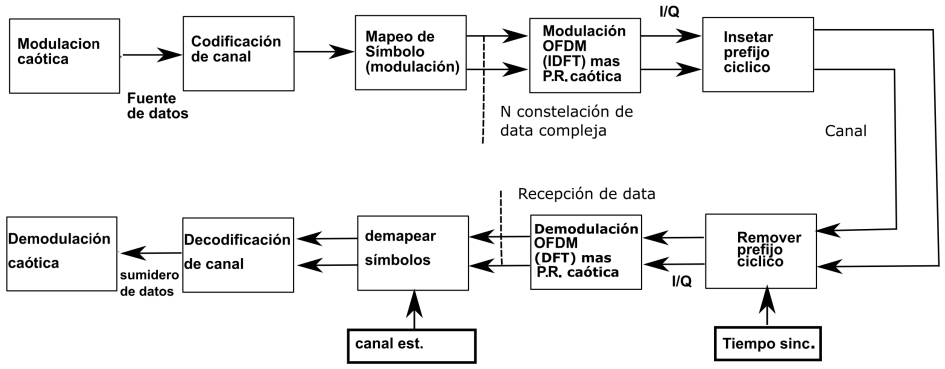

Figura 3.20: Sistema de Comunicación LTE Caótico.

Para las simulaciones se aplicaran los tres casos, el OFDM mas modulación caótica, agregando CRC y por ultimo realizando la codificación de turbo código.

3.2.6. Etapa 6: Comparar y analizar los esquemas OFDM caóticos con el convencional.

Luego de haber realizado y analizado todas las simulaciones se hizo una comparación para observar el desempeño de los esquemas caóticos en comparación al convencional. Dicha comparación se lleva a cabo más adelante.

A continuación, se presentan las tablas con los resutlados de los diferentes montajes estudiados

Capítulo IV

Análisis, interpretación y presentación de los resultados

4.1. Resultados de las simulaciones en GNU Radio .

A continuación se muestran los resultados en lo que respecta al voltaje y la cantidad de bytes máximo. El voltaje es expresado en voltios ya que el bloque que simula el canal está especificado en voltios. Se usó como fuente una imagen de tipo .jpg

Tabla 4.1: Tabla comparativa entre LTE convencional y LTE caótico para una FFT de 128

FFT: 128	LTE convencional			LTE Caótico		
Modulación	QPSK	16QAM	64QAM	QPAK	16QAM	64QAM
AWGN (V)	1,1	0,2	0,01	1,1	0,2	0,01
PacketLenght	40	50	75	40	50	75

Tabla 4.2: Tabla comparativa entre LTE convencional y LTE caótico para una FFT de 256

FFT: 256	LTE convencional			LTE Caótico		
Modulación	QPSK	16QAM	64QAM	QPAK	16QAM	64QAM
AWGN (V)	0,01	0,01	0,01	0,01	0,01	0,01
PacketLenght	50	100	150	50	100	150

Tabla 4.3: Tabla comparativa entre LTE convencional y LTE caótico para una FFT de 512

FFT: 512	LTE convencional			LTE Caótico		
Modulación	QPSK	16QAM	64QAM	QPAK	16QAM	64QAM
AWGN (V)	3,5	0,9	0,1	1,5	0,9	0,1
PacketLenght	31	60	75	30	60	75

Tabla 4.4: Tabla comparativa entre LTE convencional y LTE caótico para una FFT de 1024

FFT: 1024	LTE convencional			LTE Caótico		
Modulación	QPSK	16QAM	64QAM	QPAK	16QAM	64QAM
AWGN (V)	1,7	1,9	0,5	1,7	1,5	0,1
PacketLenght	10	30	27	15	30	27

Los valores expresados en las tablas son aquellos valores máximos con los cuales se obtiene una recepción sin errores es decir con un BER de 0 %, para valores mayores el BER es diferente de 0 %. En el caso de LTE convencional se usó un modelo de canal dispersivo con una frecuencia Doppler de 5 HZ el cual aplica para las FFT de 128, 256 y 512 mientras que para la FFT de 1024 no se usa ya que presenta errores de sincronización el tiempo promedio de simulación es entre 9 y 18 minutos, a medida que se aumenta el valor de FFT, el tiempo de simulación es mayor. Para el caso del sistema LTE caótico no se usó modelo de canal de desvanecimiento ya que se presentan errores de sincronización en la recepción, el tiempo promedio de simulación es entre 15 y 20 minutos.

4.2. Gráficas de BER vs SNR

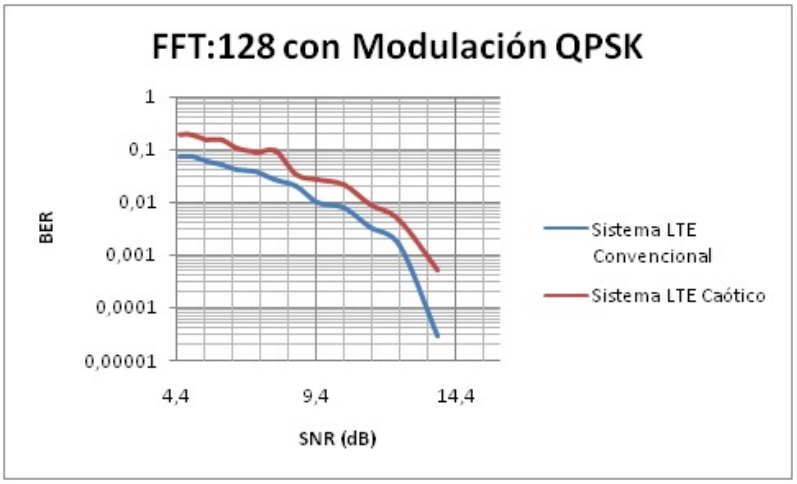

Figura 4.1: Gráfica de BER vs SNR con modulación QPSK y una FFT: 128.

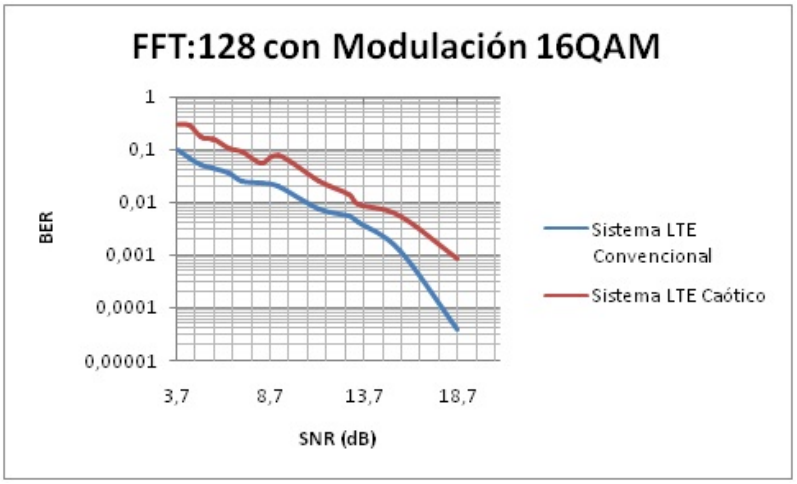

Figura 4.2: Gráfica de BER vs SNR con modulación 16QAM y una FFT: 128.

Al estudiar las tres modulaciones para la FFT de 128 se observa que el sistema LTE convencional presenta un mejor desempeño que el sistema LTE caótico. En la figura (4.1) se gráfica el BER para un rango de SNR comprendido entre 4,4 dB y 14,4 dB ya que a partir de 4,4 el BER es diferente de cero. Se observa que entre 7,2 dB y 7,8 dB el BER aumenta para el sistema LTE Caótico mientras que el sistema convencional disminuye. La figura (4.2) tiene un rango de SNR comprendido entre 3,7 dB

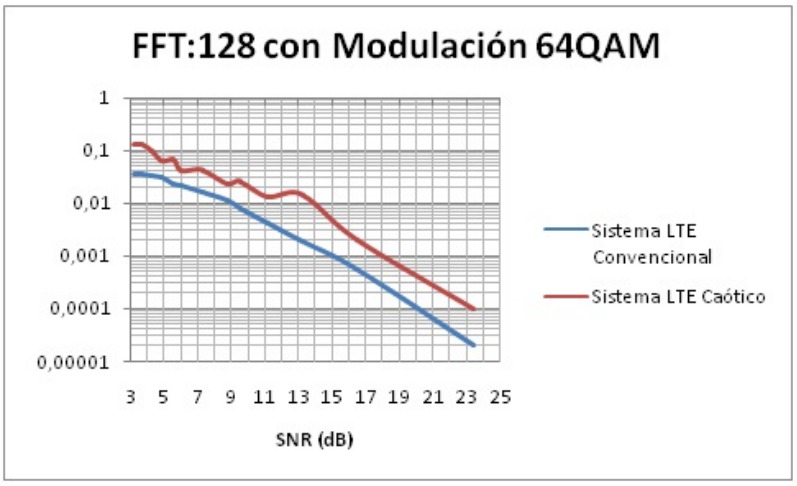

Figura 4.3: Gráfica de BER vs SNR con modulación 64QAM y una FFT: 128.

y 18,7 dB, la curva de SNR vs BER del sistema convencional presenta un aumento no deseado en 9,09 dB. Para el caso de la figura (4.3) el rango estudiado es de 3 dB a 24 dB presentando el sistema caótico un aumento de BER en 13 dB mientras que el sistema convencional en ese punto disminuye el BER. La modulación 64 QAM es la más sensible al canal por lo tanto es la que presenta un rango de estudio más extenso.

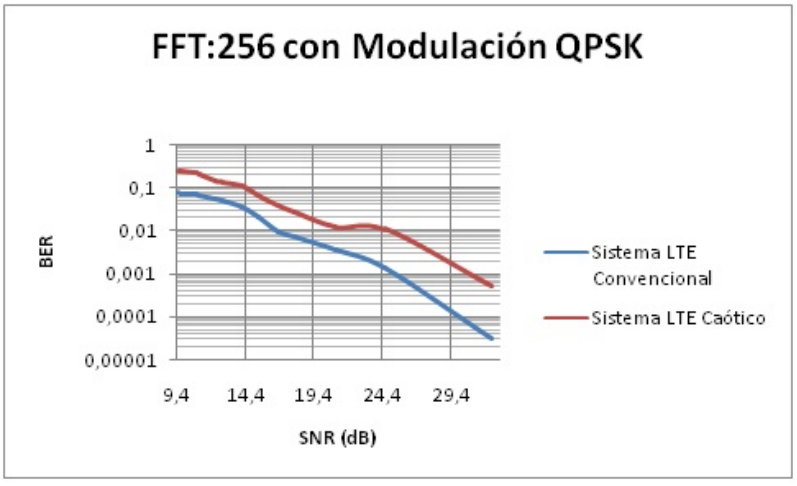

Figura 4.4: Gráfica de BER vs SNR con modulación QPSK y una FFT: 256.

Con una FFT de 256 y las tres modulaciones del estándar LTE se observa que el

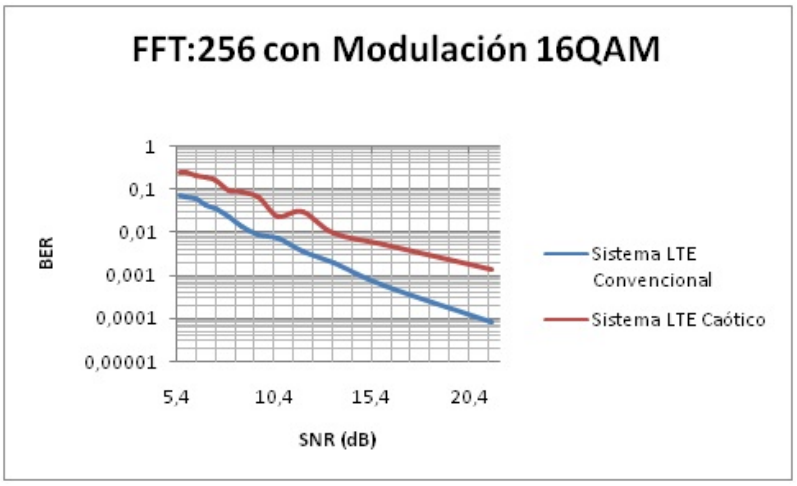

Figura 4.5: Gráfica de BER vs SNR con modulación 16QAM y una FFT: 256.

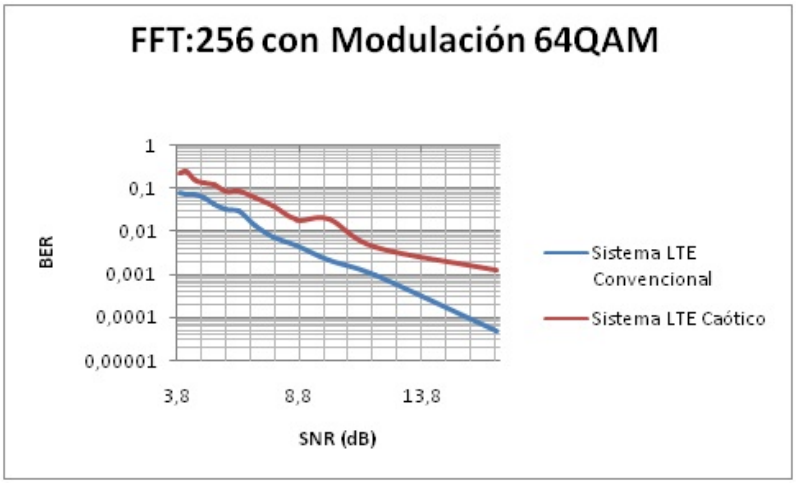

Figura 4.6: Gráfica de BER vs SNR con modulación 64QAM y una FFT: 256.

comportamiento del sistema convencional es superior al de sistema caótico ya que en las tres gráficas anteriores la curva del sistema caótico siempre está por encima de la curva del sistema convencional en cada una de las tres modulaciones. Los rangos de SNR de las gráficas es cuando el BER deja de ser cero. En la figura (4.4) se tiene un aumento de BER en 20,971 dB, en la figura (4.5) tiene un aumento en 11, 861 dB y la figura (4.6) en 9.979 dB todos estos aumentos de BER son para el caso del sistema caótico, mientras que en el sistema convencional para los valores

de SNR antes mencionados el BER disminuye.

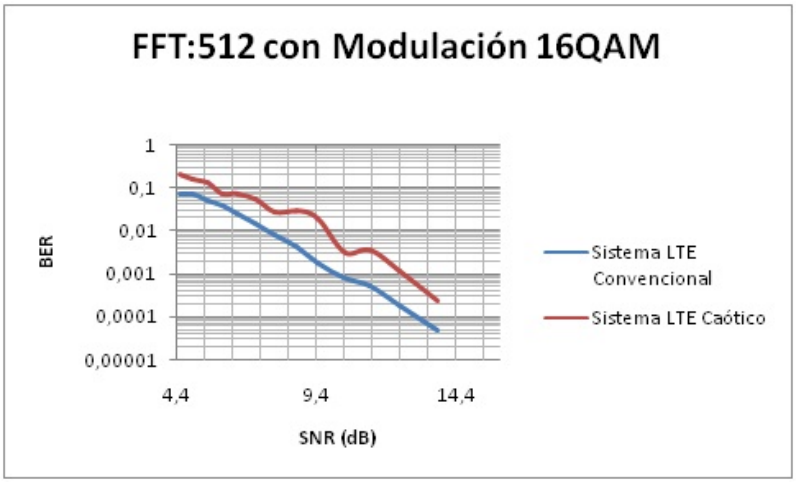

Figura 4.7: Gráfica de BER vs SNR con modulación 16QAM y una FFT: 512.

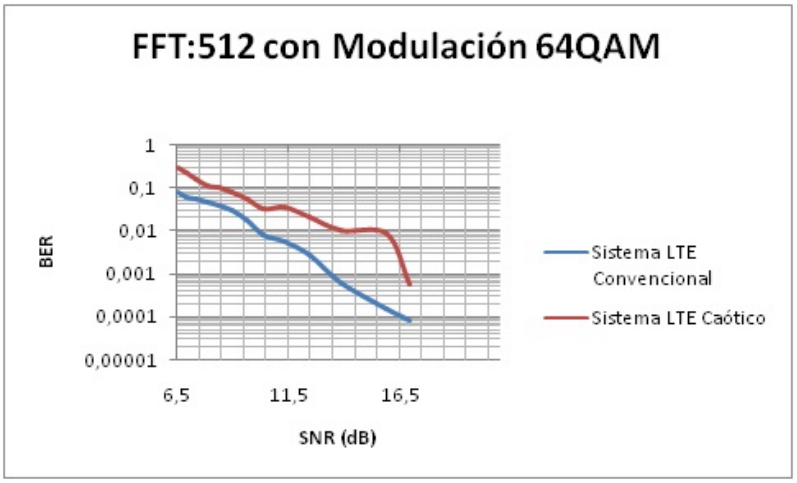

Figura 4.8: Gráfica de BER vs SNR con modulación 64QAM y una FFT: 512.

Las gráficas de BER vs SNR anteriores muestran el comportamiento del sistema convencional LTE y el sistema Caótico para las FFT 512 y 1024 con modulación QPSK, 16 QAM y 64 QAM. En cada caso el sistema LTE convencional muestra mejor desempeño que el caótico. Para la modulación QPSK y FFT de 512 no se realizó gráfica debido a que el sistema caótico luego de superar el nivel de SNR donde el BER da diferente de cero no proporciona las muestras suficientes mientras que el

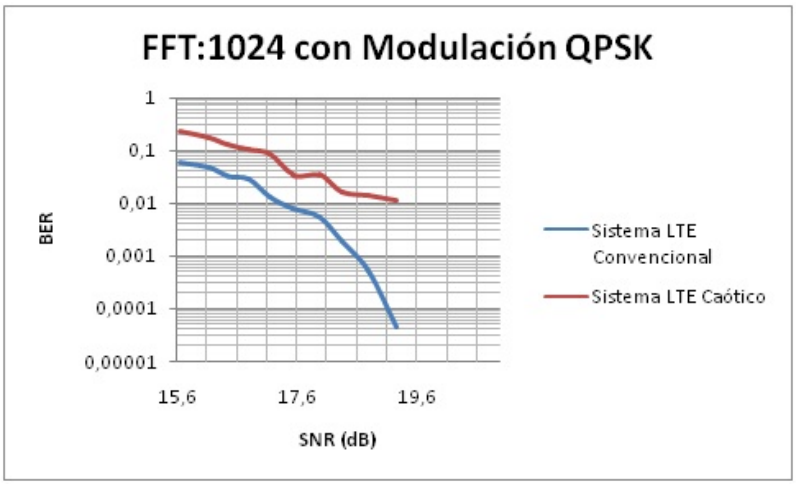

Figura 4.9: Gráfica de BER vs SNR con modulación QPSK y una FFT: 1024.

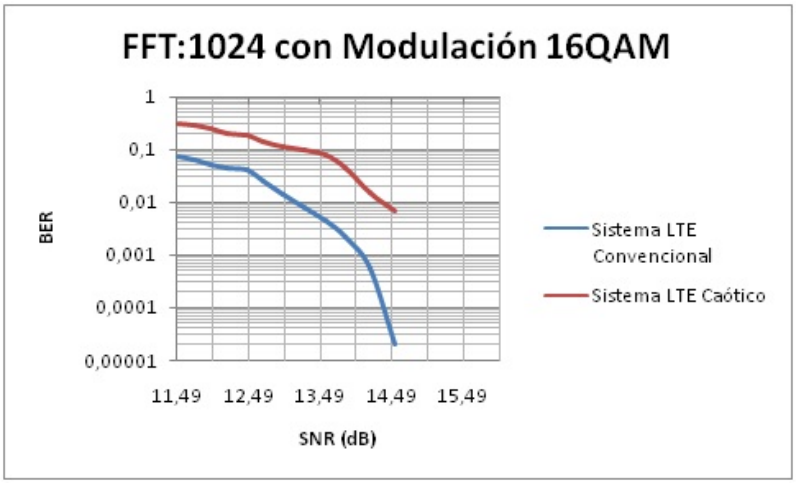

Figura 4.10: Gráfica de BER vs SNR con modulación 16QAM y una FFT: 1024.

sistema convencional si por lo tanto se concluye que el sistema convencional presenta mejores resultados en este caso en particular. Todas las curvas del sistema LTE caótico están por encima del sistema LTE convencional por lo tanto para mismos valores de SNR el sistema convencional presenta menos bits erróneos en la recepción que el caótico.

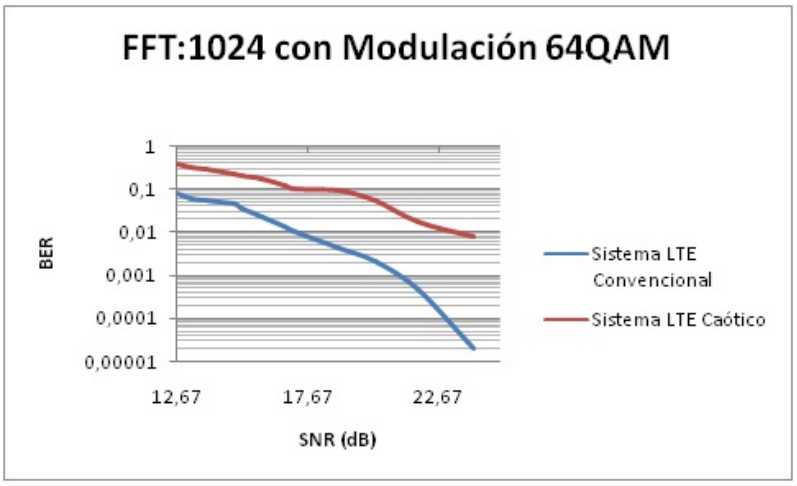

Figura 4.11: Gráfica de BER vs SNR con modulación 64QAM y una FFT: 1024.

4.3. Gráficas obtenidas en GNU Radio de los esquemas realizados

4.3.1. Modulación OFDM convencional

A continuación, se muestra la señal modulada en el espectro en banda base para una FFT de 256, modulación 16 QAM y una frecuencia de muestreo de 3,84 MHz. El ancho de banda es 2.5 MHz como se aprecia en la figura(4.12) correspondiente al estándar.

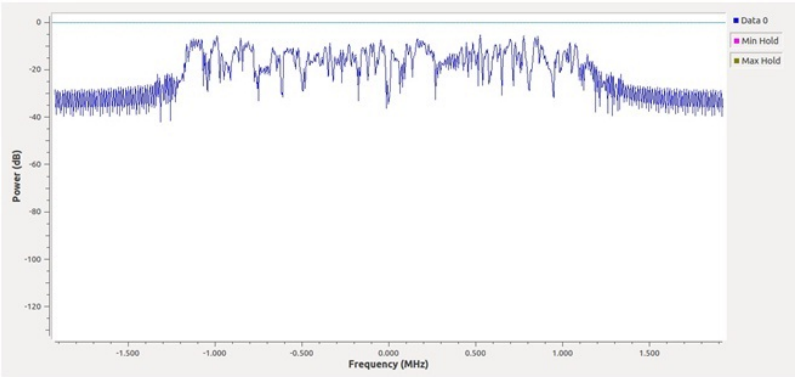

Figura 4.12: Modulación OFDM convencional.

4.3.2. Modulación Caótica

En esta sección se muestra como se ve la señal modulada caóticamente para un β igual 100 y un valor inicial del generador de 0.1 sin agregar el esquema OFDM en banda base. En la figura (4.13) se encuentra la señal en el espectro y su comportamiento es parecido al ruido y no se aprecia ningún tipo de patrón conocido por las modulaciones convencionales, en la figura (4.14) se muestra en el tiempo y también se ve un comportamiento semejante al ruido.

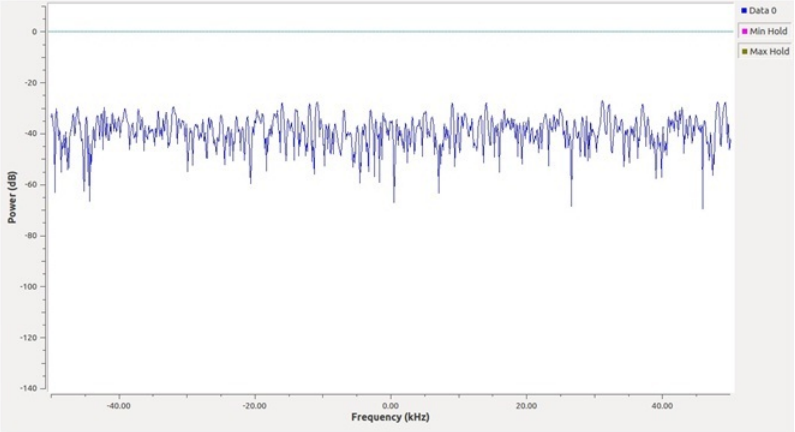

Figura 4.13: Modulación caótica en la frecuencia.

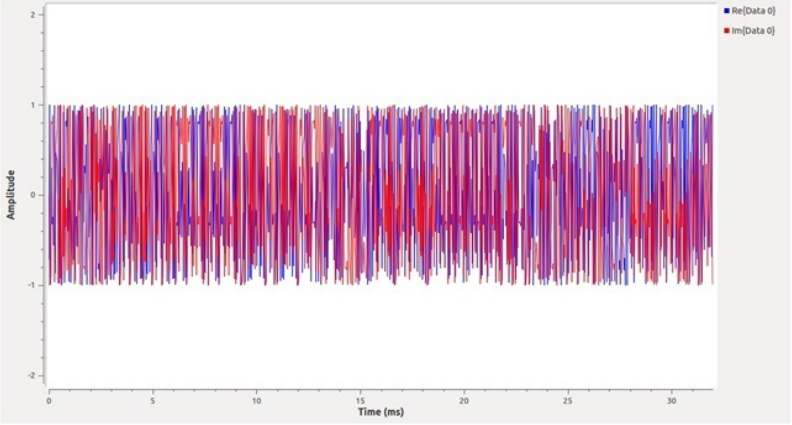

Figura 4.14: Modulación caótica en el tiempo.

4.3.3. Modulación caótica y esquema LTE modificado con portadora caótica de referencia

A continuación en la figura(4.15) se muestra la señal en el espectro modulada en banda base con un β igual 100 y un valor inicial del generador de 0.1, FFT de 256, modulación 16 QAM y frecuencia de muestreo de 3,84 MH. Esta señal en la frecuencia tiene un comportamiento igual a la modulada convencionalmente en LTE. Presenta el mismo ancho de banda de 2.5 MHz.

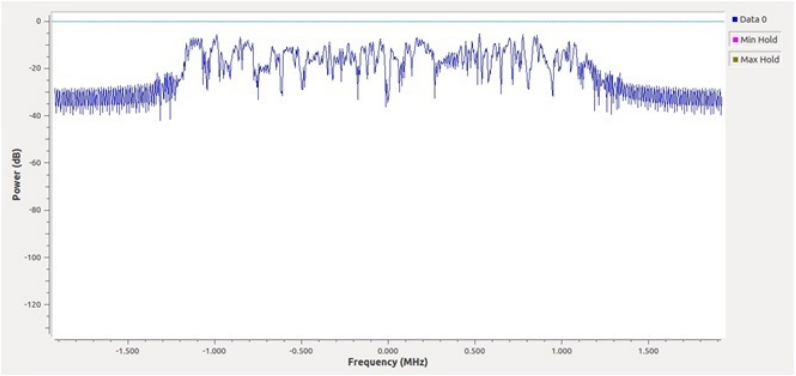

Figura 4.15: Modulación caótica y esquema LTE modificado con portadora caótica de referencia en la frecuencia.

4.4. Resultados de la imagen transmitida y recibida

A continuación en la figura (4.16) se muestra la imagen que se coloca en la fuente para ser transmitida con un tamaño de 68,4 KB con la finalidad de poder comparar el comportamiento del sistema convencional y caótico cuando se presentan errores en la recepción o un BER diferente de 0 %.

Al utilizar el esquema convencional LTE con una modulación 16QAM y una FFT de 512 agregando al canal un AWGN de 1 V y un SNR de 13,74 dB se obtiene en la recepción la imagen de la figura (4.17) la cual tiene un BER de 0,005 %.

Figura 4.16: Archivo a transmitir en la fuente.

Figura 4.17: Archivo demodulado y decodificado para un sistema LTE convencional con una FFT de 512 y modulación 16 QAM.

Por último se cambia el sistema convencional LTE por el sistema LTE caótico dejando los mismos parámetros de FFT, modulación y canal mencionados anteriormente, para obtener en la recepción la imagen mostrada en la figura (4.18) la cual tiene un BER de 0.023 %

Se puede observar los cambios producidos por los bits erróneos en la recepción para ambos sistemas. Para un mismo voltaje y SNR el sistema convencional presenta menos errores que el caótico.

Figura 4.18: Archivo demodulado y decodificado para un sistema LTE caótico con una FFT de 512 y modulación 16QAM.

4.5. Análisis de las limitaciones en la caracterización de un esquema de comunicaciones LTE caótico en el software GNU Radio.

El principal problema en la caracterización en el software GNU Radio, se concentra en la capacidad de almacenamiento de bytes en los buffer de algunos bloques esenciales de la librería de GNU Radio, como lo es el OFDM Cyclic Prefixer y el OFDM Carrier Allocator. Debido a esto se decide simular los primeros cuatro canales del estándar hasta la FFT de 1024. Una solución a este problema es cambiando GR FIXED BUFFER SIZE de 32k a 128k pero a costa de una gran desaceleración en la simulación. Pero recientes declaraciones de los creadores del software recomiendan no hacer este tipo de cambios ya que esto involucraría demasiados cambios en la arquitectura del programa.

También hay que tener en cuenta, como se vio en el capítulo (III) que los esquemas realizados están constituidos por una gran cantidad de bloques que son dependientes uno de otros y por lo tanto son sensibles a cambios que no sean los que ya están configurados en sus códigos de procesamiento. Debido a esto el esquema de modulación y demodulación no se aplicaron cambios significativos, por el contrario para la elaboración del caos se decide crear bloques fuera de la librería

del software de tal manera que se puedan conectar antes del modulador y después del demodulador.

En el caso de la referencia caótica, este cambio es dentro del esquema del modulador y demodulador que es viable debido a que las portadoras de referencia en el demodulador son eliminadas y no demodulada.

4.6. Errores comunes en GNU Radio

A continuación se elabora una lista de los errores más comunes que surgieron durante las simulaciones:

- Detected an invalid packet at item INFO: Parser returned: Este error ocurre porque hay una desincronización en el receptor y se puede deber a que no se construyeron bien las palabras de sincronización o en el receptor se aplico un retraso más largo o corto de lo debido.
- thread [thread per block[1]: block OFDM Cyclic Prefixer (1)]Buffer too small for min noutput items: Este error se debe a que el buffer del bloque es insuficiente, se puede solucionar quitando la etiqueta pero no se sabría con qué cantidad de bytes estaría trabajando el bloque.
- thread [thread per block[1]: block OFDM Carrier Allocator (1)] Buffer too small for min noutput items: Error similar al anterior pero con la diferencia que para correguir este problema se necesita disminuir la cantidad de FFT a usar.
- thread [thread per block[1]: block Tagged Stream Mux(1)] Buffer too small for min noutput items: Este error se debe a que el buffer del bloque es insuficiente, se puede solucionar bajando la cantidad de Bytes del Packet Length.
- thread [thread per block[1]: block Tagged Stream Mux(1)] Missing Length Tag: Este error es de los más comunes y se debe a que en algún bloque falto colocar el nombre de la etiqueta o se coloco un nombre diferente, se recomienda siempre usar una misma etiqueta.

- CPU CONGESTION: Este error ocurre cuando hace falta un throttle en la simulación.
- 'gr top block sptr' object has no attribute 'set: ocurre cuando se utilize una opcion de generacion de gráfica diferente a la configurada en el Top Block, es decir si se está trabajando con gráficas de tipo QT y el Top Block está configurado como WX aparecerá este error, se tiene que tener el mismo modelo para que la simulación corra.

Capítulo V

Conclusiones y recomendaciones

5.1. Conclusiones

El reto de los sistemas de comunicaciones móviles es aumentar la velocidad de transmisión de archivos, así como ofrecer una mayor seguridad. Optar por un sistema de modulación y demodulación caótica es, en efecto, una de las soluciones más prometedoras y de fácil implementación en el mundo de las comunicaciones y fue a la que se abocó esta investigación. Ya que permite aumentar la velocidad de transferencia de archivos y crear una señal más segura.

- Para la elaboración del caos en sistemas de comunicaciones existen varios métodos, en este trabajo se selecciono trabajar con secuencias pseudoaleatorias ya que además de eficientes son de fácil implementación.
- Dichas secuencias caóticas y los esquemas de comunicaciones fueron simulados en el software GNU Radio, el cual es un software libre que posee librerías para la realización de este proyecto y brinda la facilidad de poder crear un bloque si así requiere el trabajo.

- A pesar de que el software presenta un buen desempeño, existen algunas restricciones que no permiten simular la totalidad del sistema, pero si lo suficiente como para obtener datos e información necesaria acerca de las modulaciones caóticas.

- El software permitio simular el desempeño bajo condiciones de ruido blanco Gaussiano y efectos de dispersión con el objetivo de realizar una simulación lo más real posible.

- Al proponer la nueva configuración con portadora piloto modulada caóticamente, se demostró la vialidad de este esquema, ya que permite modular más información que el estándar convencional. Aunque pueda ser susceptible a canales con efectos dispersivos, así como también errores de sincronización para modulaciones de alto orden como lo es la de 64QAM.

- La utilización y uso del bloque CRC es para estudiar el comportamiento del sistema cuando se le agregaban bytes al final de la trama, aunque este bloque no es parte del estándar LTE es un bloque recomendado por los autores del software cuando se utilizan bloques de sistemas OFDM. Los resultados obtenidos son muy parecidos al caso anterior, solo presenta algunos errores recepción para el caso de la modulación 64 QAM, que es una modulación bastante sensible a las perturbaciones.

- El codificador de canal empleado fue un turbo codificador ya que de los dos modelos estipulados por el estándar LTE (convolucional y turbo códigos) fue el que mejor desempeño mostro con los esquemas OFDM a pesar de que la implementación del mismo amerita una carga computacional mayor a la hora de realizar las simulaciones.

- En la creación del bloque de modulación y demodulación caótica se siguieron los pasos estipulados por la pagina oficial de GNU Radio en la sección Out Of Tree para integrar los nuevos bloques a la librería del programa, los tres archivos mas importantes para la creación de un nuevo bloque son el .cc, .h y

.xml ya que estos representan la función que va a realizar el bloque y la interfaz gráfica. La plantilla de estos tres archivos son descargados al introducir el comando en la consola.

- Al integrar el modulador y demodulador caótico al ya extenso sistema de comunicaciones LTE, la finalidad es demostrar que la Teoría del Caos es un método viable en las comunicaciones actuales y futuras, a pesar de que las simulaciones tardan un extenso periodo de tiempo comparada con el estándar convencional. El tiempo promedio de las simulaciones es 15 minutos y se reciben 512 bytes, pero estos bytes se reciben sin ninguna distorsión lo que demuestra que el caos es un método viable para sistemas de comunicaciones OFDM y puede seguir siendo objeto de estudio con el fin de optimizar el tiempo de transmisión y recepción de datos.

- En lo que respecta a la unión de los esquemas con modulación caótica, tanto en la portadora de referencia como en la información se puede observar el desempeño del mismo, así como también queda demostrado que aplicar la teoría del caos en los sistemas de comunicaciones actuales es una posibilidad real que ofrece una seguridad extra que tanto se demanda en la actualidad.

- La seguridad que ofrece este esquema de modulación caótica es que su espectro en el rango de frecuencias se ve como ruido, esto permite que no se observe la frecuencia a la cual se esta enviando la información.

- Gracias al estudio de las secuencias caoticas se pueden construir un número casi infinito de códigos caóticos que pueden ser asignados a un número casi infinito de direcciones mediante la tecnología OFDMA. Si el futuro es el Internet de las Cosas significa que varios centenares de miles de millones de cosas estarán conectadas inalámbricamente, por lo tanto se necesitará ese tipo de tecnología de multiplexión con un número potencialmente casi infinito de códigos de ensanchamiento ortogonales.

- A lo largo de las pruebas realizadas para el sistema LTE convencional y LTE caótico se demuestra que el sistema convencional presenta mejor desempeño que el caótico en el software GNU Radio.

5.2. Recomendaciones

- Ampliar el *Buffer* del bloque *OFDM Carrier Allocator* con el fin de poder utilizar este quema de modulación y demodulación con estándares que requieran mayor cantidad de FFT como lo es la televisión digital.
- Implementar los esquemas desarrollados en este trabajo en tarjetas FPGA compatibles con el programa GNU Radio.
- Realizar una interfaz grafica donde el usuario pueda controlar los parámetros del estándar LTE y del caos sin tener que abrir los bloques de GNU Radio
- Estudiar otro tipo de generación de caos que se adapte a un sistema de comunicaciones LTE.
- Sustituir la modulación convencional (QPSK, 16QAM y 64 QAM) por una modulación con mapeo de secuencias pseudoaleatorias.
- Incorporar esta y otros tipos de modulaciones caóticas en el generador vectorial de ondas arbitrarias basado en tecnología SDR.

Apéndice A

Codigos en C++ y .xml de los nuevos Bloques caóticos

1.1. Generador Caótico .cc

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "gen_logi_map_f_impl.h"

namespace gr {
  namespace chaos {

    gen_logi_map_f::sptr
    gen_logi_map_f::make(float seed)
     //valor inicial de la secuencia caotica que
     //tiene que establecer el usuario
    {
```

```
  return gnuradio::get_initial_sptr
    (new gen_logi_map_f_impl(seed));
}

/*
 * The private constructor
 */
gen_logi_map_f_impl::gen_logi_map_f_impl(float seed)
//declaracion de la salida y el tipo de salida
  : gr::sync_block("gen_logi_map_f",
                   gr::io_signature::make(0, 0, 0),
                   gr::io_signature::make(1, 1, sizeof(float))),
    d_seed(seed)
{
  next_val = d_seed;
  // valor inicial con que empieza la secuencia pseudoaletoria
}

/*
 * Our virtual destructor.
 */
gen_logi_map_f_impl::~gen_logi_map_f_impl()
{
}

float
gen_logi_map_f_impl::gen_next_chaos()
{
  float val;

  val = next_val;
  next_val = 1.0 - 2.0 * (val * val);
```

```
      //funcion que genera la secuencia pseudoaleatoria

      return (val);
    }

    int
    gen_logi_map_f_impl::work(int noutput_items,
                              gr_vector_const_void_star &input_items,
                              gr_vector_void_star &output_items)
    {
      float *out_samples = (float *) output_items[0];
      int i;

      for (i = 0; i < noutput_items; i++) {
        out_samples[i] = gen_next_chaos();
      }

      return (noutput_items);
    }

  } /* namespace chaos */
} /* namespace gr */
```

1.2. Modulador Caótico .cc

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif

#include <gnuradio/io_signature.h>
#include "dcsk_mod_cbc_impl.h"
```

```
namespace gr {
  namespace chaos {

    dcsk_mod_cbc::sptr
    dcsk_mod_cbc::make(int n_samples)
    {
      return gnuradio::get_initial_sptr
        (new dcsk_mod_cbc_impl(n_samples));
    }

    /*
     * The private constructor
     */
    dcsk_mod_cbc_impl::dcsk_mod_cbc_impl(int n_samples)
      : gr::block("dcsk_mod_cbc",
 gr::io_signature::make2(2, 2, sizeof (gr_complex), sizeof (unsigned char)),
 gr::io_signature::make(1, 1, sizeof(gr_complex))),
        d_n_samples (n_samples)
    {
        set_n_samples(n_samples);
    }

    /*
     * Our virtual destructor.
     */
    dcsk_mod_cbc_impl::~dcsk_mod_cbc_impl()
    {
    }

    void
    dcsk_mod_cbc_impl::set_n_samples (int n_samples)
```

```
    {
      if (n_samples < 1) {
           n_samples = 1;
      }

      d_n_samples = n_samples;

      set_relative_rate((double)(2 * n_samples));
      set_output_multiple (2 * n_samples);
    }

    void
dcsk_mod_cbc_impl::forecast (int noutput_items, gr_vector_int
 &ninput_items_required)
    {
        int output_bits;

        output_bits = noutput_items / (2 * d_n_samples);
        ninput_items_required[0] = output_bits * d_n_samples;
        ninput_items_required[1] = output_bits;
    }

    // Función que verifica si tenemos en la entrada suficientes muestras
    //caoticas, bits de datos y suficientes bits disponibles
// en la salida para modular la señal caotica completamente.
//La función devuelve el número máximo de bits que se pueden calcular.
    unsigned int
    dcsk_mod_cbc_impl::verification(unsigned int n_input_chaos,
                                    unsigned int data_bits,
                                    int n_output_items)
    {
        unsigned int chaos_bits, output_bits, n_bits;
```

```
        chaos_bits = n_input_chaos / d_n_samples;
        output_bits = n_output_items / (2 * d_n_samples);

        if (chaos_bits < output_bits) {
          if(chaos_bits < data_bits) {
            n_bits = chaos_bits;
          }
          else {
            n_bits = data_bits;
          }
        }
        else {
          if(output_bits < data_bits) {
            n_bits = output_bits;
          }
          else {
            n_bits = data_bits;
          }
        }
        return n_bits;
    }

    int
    dcsk_mod_cbc_impl::general_work (int noutput_items,
                       gr_vector_int &ninput_items,
                       gr_vector_const_void_star &input_items,
                       gr_vector_void_star &output_items)
    {
       // numero de entradas totales (2:caos,data)
 const gr_complex *in_chaos = (const gr_complex *) input_items[0];
  // entrada de la secuencia caotica
```

```
 const unsigned char *in_data = (const unsigned char *) input_items[1];
 // entrada de los datos
 unsigned int ninput_chaos = ninput_items[0];
 // numero de muestras caoticas disponibles
 unsigned int ninput_data = ninput_items[1];
 // number of data bits available
gr_complex *out_signal = (gr_complex *) output_items[0];
// señal de salida

      int i,j,nbits_data;
      unsigned int chaos_consumed, out_produced;



      nbits_data = verification(ninput_chaos,ninput_data,noutput_items);

      for(i=0;i<nbits_data;i++)
 // hasta que todos los bits pasen a la salida
        {
          for(j=0;j<d_n_samples;j++)
 // 1 bit para 2*'x' muestras caoticas
            {
              out_signal[(2*d_n_samples*i)+j] = in_chaos[(d_n_samples*i)+j];
               // referencia

              if(in_data[i])
// si data=1, data = referencia
                {
   out_signal[((2*d_n_samples*i)+d_n_samples)+j] = in_chaos[(d_n_samples*i)+j];
      // data
                }
              else
// si data=0, data = -referencia
```

```
                {
    out_signal[((2*d_n_samples*i)+d_n_samples)+j] = (gr_complex) -1 *
     (in_chaos[(d_n_samples*i)+j]);
 // data
                }
            }
          j=0;
        }
      i=0;

      chaos_consumed = nbits_data*d_n_samples;
      out_produced = nbits_data*(2*d_n_samples);

      consume(0, chaos_consumed);
      consume(1, nbits_data);
 //std::cout << "chaos:" << chaos_consumed
 //<< " data:" << nbits_data << " out:" << out_produced << std::endl;

// Le dice al sistema de ejecución cuántos elementos de salida se producen.
      return out_produced;
    }

  } /* namespace chaos */
} /* namespace gr */
```

1.3. Demodulador Caótico .cc

```
#ifdef HAVE_CONFIG_H
#include "config.h"
#endif
```

```
// #define DCSK_DEBUG

#ifdef DCSK_DEBUG
#include <iostream>
#endif

#include <gnuradio/io_signature.h>
#include <math.h>
#include "dcsk_demod_cb_impl.h"

namespace gr {
  namespace chaos {

    dcsk_demod_cb::sptr
    dcsk_demod_cb::make(int n_samples, int n_sync)
    {
      return gnuradio::get_initial_sptr
        (new dcsk_demod_cb_impl(n_samples, n_sync));
    }

    /*
     * The private constructor
     */
    dcsk_demod_cb_impl::dcsk_demod_cb_impl(int n_samples, int n_sync)
//declaracion de las variables N_sync y numero muestras (beta) que son
 suministradas por el usuario
      : gr::block("dcsk_demod_cb",
 gr::io_signature::make(1, 1, sizeof(gr_complex)),
 //variable de entrada del bloque tipo complejo
 gr::io_signature::make(1, 1, sizeof(unsigned char))),
 // variable de salida del bloque tipo char
```

```
    d_n_sync(n_sync)
{
  set_n_samples(n_samples);
}

/*
 * Our virtual destructor.
 */
dcsk_demod_cb_impl::~dcsk_demod_cb_impl()
{
}

void
dcsk_demod_cb_impl::set_n_samples (int n_samples)
{
  if (n_samples < 1)
    n_samples = 1;

  d_n_samples = n_samples;

  set_relative_rate(1.0 / (double)(2 * n_samples));
}

void
dcsk_demod_cb_impl::forecast (int noutput_items,
                              gr_vector_int &ninput_items_required)
{
  ninput_items_required[0] = noutput_items * 2 * d_n_samples;
}

gr_complex
dcsk_demod_cb_impl::cross_corr (const gr_complex * chaos_ref,
```

```
                                    const gr_complex * chaos_data)
    {
      int i;
      gr_complex correlation(0.0, 0.0);

      for (i = 0; i < d_n_samples; i++)
        correlation += (chaos_ref[i] * conj(chaos_data[i]));

      return correlation;
    }

    int
    dcsk_demod_cb_impl::general_work (int noutput_items,
                      gr_vector_int &ninput_items,
                      gr_vector_const_void_star &input_items,
                      gr_vector_void_star &output_items)
    {
      const unsigned int needed_smp = 2 * d_n_samples + 2 * d_n_sync;
      const gr_complex *in_signal = (const gr_complex *) input_items[0];
      unsigned int ninput_signal = ninput_items[0];
      unsigned char *out_info = (unsigned char *) output_items[0];
      gr_complex cor(0.0, 0.0);
      gr_complex ref(0.0, 0.0);
      int out_bits = 0;
      gr_complex best_cor(0.0, 0.0);
      int best_id = 0;
      unsigned int pos = 0;
      int i;

      // Verificar que se cumpla: n_sync < n_sample/2

#ifdef DCSK_DEBUG
```

```
      std::cout << "----------------------" << std::endl;
      std::cout << "Inicio: muestras de entrada: " << ninput_signal
           << " muestras de salida: " << noutput_items
           << std::endl;
#endif

      while ((out_bits < noutput_items) &&
             ((pos + needed_smp) <= ninput_signal)) {

        // Busqueda de la mejor correlacion, out of d_n_sync * 2 + 1 tries.

        // calculo de la correlacion inicial
        cor = cross_corr(&in_signal[pos],
                         &in_signal[pos + d_n_samples]);
        best_cor = cor;

#ifdef DCSK_DEBUG
 std::cout << "bucle pos=" << pos << std::endl;
 std::cout << "A partir de la muestra: " << in_signal[pos] << std::endl;
 std::cout << "correlación inicial: " << cor << " mag=" << abs(cor)
 << std::endl;
#endif

        for (i = 1; i < d_n_sync * 2 + 1; i++) {
          //cor = cross_corr(&in_signal[pos + i],
          //                 &in_signal[pos + i + d_n_samples]);
          // Optimización: en vez de recalcular toda la correlación,
          // Se desliza la correlación por una muestra a la derecha:
          // resta las muestras salientes y añade los entrantes.

          cor +=
            ( in_signal[pos + d_n_samples + i] *
```

```
              conj(in_signal[pos + 2 * d_n_samples + i]))
            -
            in_signal[pos + i] * conj(in_signal[pos + d_n_samples + i]);

          if (abs(cor) > abs(best_cor)) {
            best_cor = cor;
            best_id = i;
#ifdef DCSK_DEBUG
            std::cout << "encontrar la mejor cor=" <<
              abs(best_cor) << " at " << best_id << std::endl;
#endif
          }
        }

        if ( std::signbit(best_cor.real() ) == 0 )
          out_info[out_bits] = 1;
        else
          out_info[out_bits] = 0;

#ifdef DCSK_DEBUG
        std::cout << "Mejor desplazamiento es " << best_id << " (" <<
          (best_id - d_n_sync) << ")." << std::endl;
        std::cout << "Referencia auto corr es: " << ref << std::endl;
        std::cout << "La decisión de salida es:" << (unsigned int)out_info[i]
        << std::endl;
#endif

        pos += 2 * d_n_samples + best_id - d_n_sync;
        out_bits++;
      }

#ifdef DCSK_DEBUG
```

```
      std::cout << "consumido" << pos << " muestras" << std::endl;
      std::cout << "producido " << out_bits << " bits" << std::endl << std::endl;
#endif

      consume_each(pos);

      return (out_bits);
    }

  } /* namespace chaos */
} /* namespace gr */
```

1.4. Modulador Caótico .xml

```
- <block>
  <name>Chaos Modulator</name>
  <key>chaos_dcsk_mod_cbc</key>
  <category>Chaos</category>
  <import>import chaos</import>
  <make>chaos.dcsk_mod_cbc($n_samples)</make>
  <callback>set_n_samples($n_samples)</callback>
- <param>
  <name>Number of samples</name>
  <key>n_samples</key>
  <value>50</value>
  <type>int</type>
  </param>
  <check>$n_samples > 0</check>
- <sink>
  <name>chaos</name>
```

```
  <type>complex</type>
  </sink>
- <sink>
  <name>data</name>
  <type>byte</type>
  </sink>
- <source>
  <name>out</name>
  <type>complex</type>
  </source>
  </block>
```

1.5. Demodulador Caótico .xml

```
- <block>
  <name>Chaos Demodulator</name>
  <key>chaos_dcsk_demod_cb</key>
  <category>Chaos</category>
  <import>import chaos</import>
  <make>chaos.dcsk_demod_cb($n_samples, $n_sync)</make>
  <callback>set_n_samples($n_samples)</callback>
- <param>
  <name>Ref samples</name>
  <key>n_samples</key>
  <value>50</value>
  <type>int</type>
  </param>
- <param>
  <name>Sync samples</name>
  <key>n_sync</key>
```

```
  <value>5</value>
  <type>int</type>
  </param>
  <check>$n_samples > 0</check>
- <sink>
  <name>in</name>
  <type>complex</type>
  </sink>
- <source>
  <name>out</name>
  <type>byte</type>
  </source>
  </block>
```

1.6. Generador Caótico .xml

```
- <block>
  <name>Chaos Generator1</name>
  <key>chaos_gen_logi_map_f</key>
  <category>Chaos</category>
  <import>import chaos</import>
  <make>chaos.gen_logi_map_f($seed)</make>
- <param>
  <name>Seed value</name>
  <key>seed</key>
  <value>0.1</value>
  <type>real</type>
  </param>
- <source>
  <name>out</name>
  <type>float</type>
```

```
</source>
</block>
```

Referencias Bibliográficas

[1] Erik Dahlman, Stefan Parkvall, and Johan Skold. *4G: LTE/LTE-advanced for mobile broadband*. Academic press, 2013.

[2] Carlos F. da Rocha Cesar V. Vargas, Wilson E. Lopez. *Sistemas de Comunicación Inalámbrica MIMO - OFDM*. GPqCom - UFSC, 2007.

[3] Ramjee Prasad. *OFDM Wireless Communications Systems*. Universal Personal Communications, 2004.

[4] Shuying Li, Yaqin Zhao, and Zhilu Wu. Design and analysis of an ofdm-based differential chaos shift keying communication system. *Journal of Communications*, 10(3), 2015.

[5] K. Umeno and M. H. Kao. *Chaos Theory as the answer to limited spectrum?* ITU News, No.10, 2013.

[6] Angel Rendón Pedro Castillo. *Implementación de una Herramienta de Software para la Caracterización de Series de Tiempo de Dinámica no Lineal y Comportamiento caótico*. Universidad de Carabobo, 2014.

[7] Michael Kennedy, Riccardo Rovatti, and Gianluca Setti. *Chaotic electronics in telecommunications*. CRC press, 2000.

[8] John AC Bingham. Multicarrier modulation for data transmission: An idea whose time has come. *Communications Magazine, IEEE*, 28(5):5–14, 1990.

[9] Product Development and Test Challenges. Agilent: 3gpp long term evolution - system overview. 2009.

[10] Carlos Alberto Serra Jiménez and Francisco Reinerio Marante Rizo. Arquitectura general del sistema lte. *Revista Telem@ tica*, 12(2):81–90, 2013.

[11] A Goldsmith. *Wireless Communications. 1a ed. Cambridge*. Cambridge University Press, 2005.

[12] B Sklar. *Digital Communication Fundamental and Applications*. Prentice Hall, 1988.

[13] S Haykin. *Communication System*. 4th Ed. John Wiley, 2001.

[14] Lajos Hanzo, Yosef Jos Akhtman, Li Wang, and Ming Jiang. Ofdm standards. *MIMO-OFDM for LTE, Wi-Fi and WiMAX*, pages 37–60, 2011.

[15] Hongwei Yang. A road to future broadband wireless access: Mimo-ofdm-based air interface. *Communications Magazine, IEEE*, 43(1):53–60, 2005.

[16] Telesystem Innovations. Lte in a nutshell. *White Paper*, 2010.

[17] F. C. M. Lau and C. K. Tse. *Chaos-Based Digital CommunicationSystems*. Springer-Verlag, 2003.

[18] M. Delgado-Restituto and A. Rodriguez-Vazquez. *Mixed-signalmapconfigurableintegrated chaos generator for chaotic communications*. Circuits and Systems I: Fundamental Theory and Applications, IEEETransactionson, vol. 48, no. 12, pp. 1462 –1474, dec. 2001.

[19] M. Delgado-Restituto and A. Rodriguez-Vazquez. *Integrated chaos generators*. Proceedings of the IEEE, vol. 90, no. 5, pp. 747 –767, may 2002.

[20] T. S. Parker and L. O. Chua. *Practical Numerical Algorithms for chaotic systems*. Springer-Verlag, 1989.

[21] G. Kis M. P. Kennedy, G. Kolumb´an and Z. J´ak´o. *Performanceevaluation of FM-DCSK modulation in multipathenvironments*. IEEETrans. Circuits and Systems, vol. 47, pp. 1702–1711, 2000.

[22] Systems Birkh¨auser, Circuits and pp. 925–944 SignalProcess., vol. 28. *A methodology for bit error rate prediction in chaos-based communication systems*. 2009.

[23] Yong Geng, Si Long Wu, and Fang Kun Jia. The research of chirp signal based on gnu radio and usrp. In *Applied Mechanics and Materials*, volume 336, pages 1765–1770. Trans Tech Publ, 2013.

[24] Eric Blossom. Gnu radio: Tools for exploring the rf spectrum. *Linux Journal*, 122, 2004.

[25] Timothy M. Schmidl and Donald C. Cox. *Robust Frequency and Timing Synchronization for OFDM*.

[26] Christian Schlegel. Saeed F. Fard, Bruce F. Cockburn. *A Compact Rayleigh and Rician Fading Simulator Basedon Random Walk Processes Amirhosse in Alimohammad*.

[27] Amirhossein Alimohammad, Saeed Fouladi Fard, Bruce F Cockburn, and Christian Schlegel. Compact rayleigh and rician fading simulator based on random walk processes. *Communications, IET*, 3(8):1333–1342, 2009.

[28] Tech. Rep. V700 3rd Generation Partnership Project. *Technical specification group radio access network; Physical layer aspects for evolved universal terrestrial radio access (UTRA)*. June 2006.