UNIVERSIDAD POLITECNICA DE VALENCIA
ESCUELA POLITECNICA SUPERIOR DE GANDIA
I.T. TELECOMUNICACIÓN (IMAGEN Y SONIDO)

UNIVERSIDAD
POLITECNICA
DE VALENCIA

ESCUELA POLITECNICA
SUPERIOR DE GANDIA

"Evaluación de arquitecturas Radio-Sobre-Fibra para la transmisión de señales radio en coexistencia empleando la modulación OFDM."

TRABAJO FINAL DE CARRERA
Autor/es:
Alexandra Valentines Vallés
Director/es:
Dr. Roberto Llorente
GANDIA, 2012

Dedicado a mis padres por confiar siempre en mi
A mis amigos, que me lo han hecho más fácil
Al Dr. Roberto Llorente Saéz por haber sido mi tutor
A Maria Moratán Peréz por toda su ayuda.

Resumen

Actualmente las redes y los sistemas de telecomunicaciones se enfrentan a la necesidad de proporcionar elevados regímenes binarios para poder abarcar la gran demanda ya que nuestra sociedad cada día está más interconectada.

Con el objetivo de transmitir una señal con una alta tasa binaria, utilizaremos el sistema de comunicación UWB, basada en la definición dada por el estándar ECMA368.

En este proyecte nos centraremos en evaluar el funcionamiento de distintos sistemas Radio-Sobre-Fibra mediante simulaciones realizadas con el software *Virtual Photonics Inc.*

El objetivo principal para este proyecto es el estudio de distintos sistemas RoF para la distribución de señales en coexistencia a fin de proporcionar servicios inalámbricos de última de generación como la señal UWB.

Estudiaremos los requisitos de los sistemas MB-OFDM UWB en entornos de redes inalámbricas personales.

Lista De Tablas:

Lista De Figuras:

Lista De Acrónimos

BER	*Bit Error Rate*
BG	*Band Group*
BM	*Burst Mode*
BPM	*Biphase Pulse Modulation*
CPE	*Customer Premises Equipment*
CRC	*Cyclic Redundancy Check*
CS	*Central Station*
CW	*Continuous Wave*
DCM	*Dual Carrier Modulation*
DMT	*Tono Multi Discreto*
DS-UWB	*Direct Sequence UWB*
ECMA	*European Computer Manufacturers Association*
EVM	*Error Vector Magnitude*
FCC	*Federal Communications Comission*
FCS	*Frame Check Sequence*
FDS	*Frequency-Domain Spreading*
FEC	*Forward Error Correction*
FFI	*Fixed-Frequency Interleaved*
FFT	*Fast Fourier Transform*
FIVER	*Full-Converged Quintuple-Play Integrated Optical-Wirless Access Architectures*
GSM	*Global System for Mobile communications*
HCS	*Header Check Sequence*
IDFT	*Inverse Discrete Fourier Transform*
IEEE	*Institute of Electrical and Electronics Engineers*
IR-UWB	*Impulse Radio UWB*
ISO	*Organización Internacional de Normalización*
ITU-R	*International Telecommunication Union*
LSB	*Least-Significant Bit*
LTE	*Long Term Evolution*

MAC	*Medium Access Control*
MBOA	*Multiband OFDM Alliance*
MBOA	*Multiband OFDM Alliance*
MB-OFDM	*Multi-Band Orthogonal-Frequency Division-Multiplexing*
MSB	*Most-Significant Bit*
MZ	*Mach-Zehnder*
MZM	*Modulador Mach-Zehnder*
OFDM	*Orthogonal Frecuency Division Multiplexing*
OOK	*On-Off keying*
OPM	*Orthogonal Pulse Modulation*
PAM	*Pulse Amplitud Modulation*
PAN	*Personal area network*
PHY	*Physical High Rate*
PLCP	*Physical Layer Convergence Protocol*
PPDU	*PLCP Protocol Data Unit*
PRBS	*Pseudo-Random Binary Sequence*
PSDU	*PHY Service Data Unit*
PSK	*Phase Shift Keying*
PT	*Preamble Type*
QAM	*Quadrature Amplitude Modulation*
QPSK	*Quadrature-Phase Shift Keying*
RF	*Radiofrecuencia*
RFoF	*Radio-Frequency-Over-Fiber*
RMS	*Root mean squared*
RoF	*Radio-over-Fiber*
SER	*Symbol Error Rate*
SER	*Symbol Error Rate*
SM	*Spectrum Management*
TF	*Time-Frequency*
TFC	*Time-Frequency Codes*
TFI	*Time-Frequency Interleaved*

TFI2	*Two-band TFI*
USB	*Universal Serial Bus*
UWB	*Ultra-Wide Band*
VOD	*Video on demand*
VPI	*Virtual Photonics Inc*
WiMAX	*Worldwide Interoperability for Microwave Access*
WPAN	*Wireless Personal Area Network*
ZPS	*Zero Padded Suffix*

Tabla de contenido

1 Introducción

Actualmente los sistemas de comunicaciones inalámbricas como pueden ser datos de Internet, canales de televisión, video a la carta (VOD, *video on demand*), comunicaciones combinadas de voz y datos, servicios multimedia, entre otros, necesitan mayor ancho de banda para poder transmitir más información.

Las señales de banda ultra-ancha (UWB, *Ultra-Wide Band*) han despertado gran interés por sus ventajas respeto a la alta velocidad de datos, el bajo consumo de energía y la inmunidad a interferencias en comunicaciones de corto alcance [1]

Para ello, en este proyecto se estudiará con profundidad la tecnología UWB basada en la definición dada por WiMedia Alliance [2] mediante sistemas basados en MB-OFDM (*Multi-Band Orthogonal-Frequency Division-Multiplexing*) en entornos de redes inalámbricas personales (WPAN, *Wireless Personal Area Network*) con la que se pueden transmitir datos a gran velocidad sobre un espectro amplio de frecuencias, a baja potencia y sin necesidad de visión directa entre transmisor y receptor para establecer la comunicación.

La señal UWB, por su limitada potencia de emisión, con una densidad de potencia interior a -41.3 dBm/MHz [2] sólo se puede transmitir hasta decenas de metros, para cubrir la necesidad de distribuir la señal de radio con mayor capacidad, se proponen sistemas flexibles y de bajo coste, por eso se utiliza la tecnología Radio-Sobre-Fibra (RoF, Radio-over-Fiber), con esta tecnología podremos distribuir la señal de radiofrecuencia (RF, Radiofrecuencia) mediante enlaces de fibra óptica hacia antenas o puntos de acceso inalámbricos para su recepción según la tecnología utilizada.

En el artículo [3] encontramos que al principio este sistema se utilizaba para transmitir GSM (proviene del francés *groupe spécial mobile, Global System for Mobile communications)* a la frecuencia de 1800MHz, pero hoy en día ya se pueden utilizar la totalidad de los sistemas inalámbricos existentes.

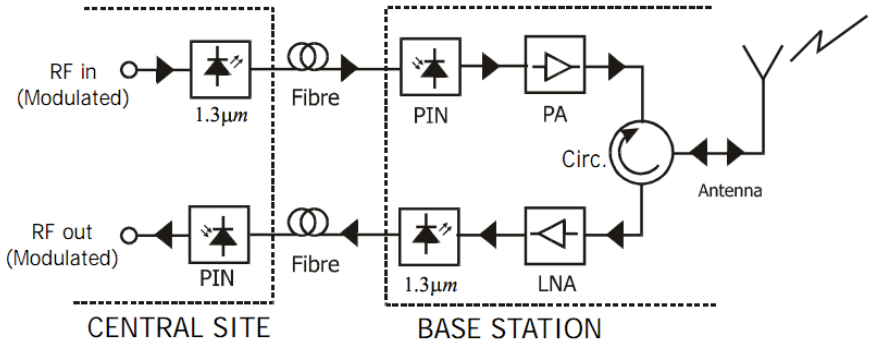

Figura 1. 1 Sistema Radio sobre Fibra (RoF)

La distribución de la señal mediante la tecnología RoF se basa en la conversión de la señal eléctrica en señal óptica en la estación central (CS, *Central Station*) y transmitida a través de fibra óptica a un terminal cliente (CPE, *Customer Premises Equipment*) donde se realiza otra conversión, en este caso de señal óptica a eléctrica, permitiendo el acceso al usuario.

La tecnología RoF se basa en distintos enlaces, ya que suelen requerir de una transmisión bidireccional por la red: en el enlace descendente la señal proviene de la estación central hasta el terminal cliente y el enlace ascendente procedente de un terminal cliente hasta la estación central.

En el enlace descendente, la señal procedente de la estación central es una señal de radiofrecuencia que se convierte en señal óptica mediante un modulador, en nuestro caso utilizaremos un modulador electro-óptico Mach-Zehnder (MZM), y viajará a través de fibra óptica hasta llegar al módulo de la antena dónde se convertirá nuevamente en una señal en radiofrecuencia.

En el enlace ascendente, la señal proviene el terminal cliente y se recibe mediante la antena, donde se transporta mediante cable coaxial hasta la estación base donde se realiza la conversión electro-óptica de la señal, viajando la señal por la fibra óptica hasta la estación central donde se convierte nuevamente a radiofrecuencia.

Las principales ventajas de utilizar sistemas basados en sistemas ópticos, es la baja atenuación frente a los 0,7 dB/m de un cable coaxial convencional, la fibra tiene unas pérdidas de 0,3 dB / km para 1550nm, y 0,5 dB / km para longitudes de onda de 1310nm, es decir, permite transportar señales a grandes distancias sin el uso de repetidores intermedios. Otra de las ventajas destacadas de los sistemas RoF es la inmunidad a interferencias electromagnéticas, ya que la fibra emplea luz emitida por un láser para la comunicación, con lo cual no se ve afectada por las radiaciones electromagnéticas del entorno, y finalmente el gran ancho de banda, el cual es mil veces superior al de un cable coaxial.

Pero como todas las tecnología también tiene sus desventajas, como la dispersión, es decir, el fenómeno por el cual un pulso se deforma a medida que se propaga a través de la fibra óptica, causando errores en la detección de los mismos en recepción además de añadir distorsión y la no linealidad, causando distorsiones en la señal debido a cualquier mecanismo del enlace óptico que no actúe de forma estrictamente lineal, como por ejemplo la propia fibra o la fuente óptica.

Se distinguen distintas técnicas para transportar la señal wireless mediante fibra óptica en base al tipo de señal que se transmite [4], en nuestro caso utilizaremos el método más sencillo, radio frecuencia sobre fibra, (RFoF, *Radio-Frequency-Over-Fiber*) en el que consiste en transportar directamente la señal de RF (Radiofrecuencia) modulando la portadora óptica con la propia señal sin necesidad de trasladar en frecuencia la estación base.

Para la modulación utilizaremos un modulador externo como el modulador Mach-Zehnder (MZM) mientras que en receptor sólo nos hará falta un receptor óptico como un fotodetector.

Los sistemas ROF utilizados a lo largo de este proyecto consisten en configuraciones bastante simples en cuanto al diseño. La estación base no necesita utilizar etapas de frecuencia intermedias. Aunque no estudiaremos sistemas complejos no tenemos que olvidar en que al tratarse de sistemas ópticos aparecerán pérdidas debidas a la dispersión cromática de la fibra sobre la señal.

2 La señal UWB

En este capítulo vamos a explicar brevemente el origen de las señales de banda ultra ancha (UWB o de su nombre en inglés *Utra Wide Band*) así como sus principales características y aplicaciones.

UWB consiste en una tecnología para la transferencia inalámbrica de datos. Hasta ahora las tecnologías más conocidas para esta aplicación son el Wi-Fi y el Bluetooth aunque estas tecnologías presentan importantes limitaciones cómo por ejemplo en la velocidad de trasmisión, la autonomía de funcionamiento y la seguridad.

La tecnología UWB se caracteriza por su gran ancho de banda espectral que proporciona una mayor tasa de datos. Se espera que la señal UWB pueda reemplazar las tecnologías Wi-Fi y Bluetooth en el ámbito de redes personales.

2.1 Competencias de la señal UWB

En este punto vamos a ver las competencias de la señal UWB y sus ventajas respecto otras tecnologías inalámbricas como puede ser el Bluetooth o el WiFi. La tecnología UWB es de gran interés en los entornos académicos e industriales debido a su versatilidad para un gran número de aplicaciones al mismo tiempo de su bajo coste y gran ancho de banda.

La tecnología UWB fue creada para ser utilizada en aplicaciones militares. Sus grandes aplicaciones, desde el principio de esta tecnología, han sido las comunicaciones y la vigilancia, es decir radar y localización, las cuales han sido extendidas al mercado en servicios de emergencia, vigilancia, seguridad o construcción. [5]

UWB trabaja transmitiendo la señal radio en una amplia franja de frecuencias (entre 3.1 y 10.6GHz.) en lugar de los sistemas basados en portadoras sobre una frecuencia fija.

La utilización de pulsos cortos, del orden de nanosegundos permite que su espectro se extienda a una gran banda de frecuencias haciendo que la densidad espectral de potencia sea muy baja (-41.3 dBm/MHz) obteniendo así un bajo consumo de potencia. Mientras que UWB necesita una potencia del orden de medio mW en el Bluetooth son de varias centenas de mW, esto se traduce en una mayor autonomía de los terminales que la incorporen.

Gracias a la utilización de los pulsos de corta duración para la transmisión se obtiene un posicionamiento preciso, siendo una gran candidata para aplicaciones de corto alcance (<10m) en las transmisiones de tipo radar o para el posicionamiento de vehículos.

El incremento del uso de la tecnología UWB viene gracias a las características inherentes de la señal como su inmunidad al desvanecimiento multicamino y su baja probabilidad de intercepción, su capacidad de establecer comunicaciones penetrando materiales.

Otra de sus ventajas es su alta velocidad de transmisión, en la tecnología UWB se pueden transmitir datos de hasta 1024 Mbps en lugar de los 54 Mbps del Wi-Fi o el Bluethooth con 1Mbps.

Además, UWB permite reutilización de espectro. Es decir, podemos tener una serie de dispositivos, comunicándose con nuestro ordenador a través de un canal, y a la vez, en otra zona, otra serie de dispositivos en el mismo canal, situación muy útil en la interconexión de dispositivos multimedia de un entorno residencial y de forma inalámbrica.

2.2 Implementaciones UWB

En este punto estudiaremos los entornos de aplicaciones UWB de corta distancia. La tecnología radio UWB se implementa de dos maneras, mediante portadoras libres o *carrieles* y en las que utilizan múltiples portadoras o *multi-carrier*. En la **[Figura 2. 1]** se muestra los distintos tipos de implementación de la tecnología radio UWB.

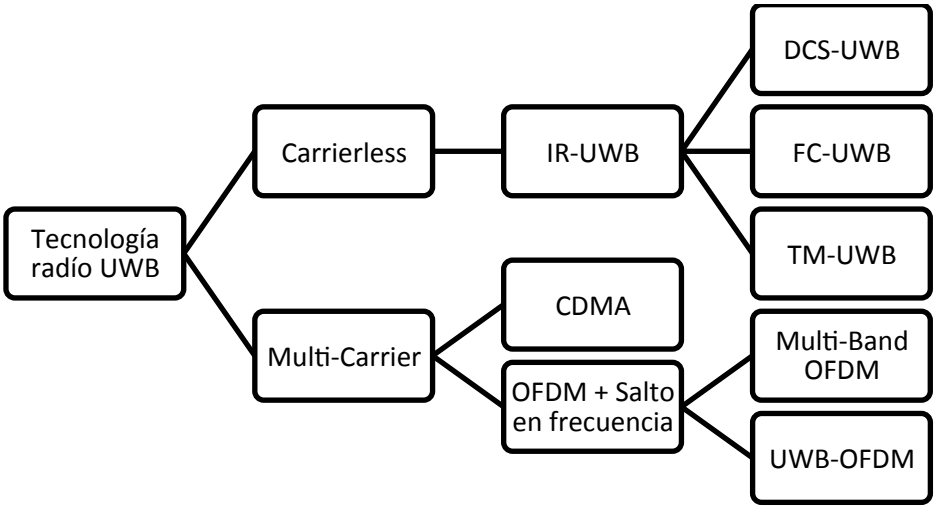

Figura 2. 1 Implementación de la tecnología UWB

2.2.1 Sistemas basados en impulse-radio (IR-UWB)

Los primeros sistemas de generación de señales IR-UWB (*Impulse Radio UWB*) se desarrollaron a principios de los años 70. La tecnología IR-UWB se basa en la transmisión de pulsos de duración muy corta. La transmisión de estos pulsos no requiere de una portadora, ya que debido a la duración del pulso se consigue adecuar su transmisión al ancho de banda deseado sin canalizaciones previas.

Habitualmente la generación de pulsos IR-UWB se basa en monociclos, aunque existen múltiples técnicas de generación de pulsos IR-UWB, como puede ser el tipo fotónico, moduladores electro-ópticos MZ funcionando en la zona no lineal para subir en frecuencia monociclos gaussianos o pulsos *doublet* generados mediante transmisores eléctricos comerciales entre otros.

Para su modulación en banda base se han propuestos diferentes técnicas basadas en tiempo o basadas en la forma del pulso. En las técnicas basadas en tiempo encontramos la modulación por posición del pulso (*Pulse Position Modulation* o PPM) y en función de la forma del pulso encontramos la modulación por amplitud del pulso (PAM*, Pulse Amplitud Modulation*), la modulación bifásica del pulso (BPM*, Biphase Pulse Modulation*), la modulación *On-Off keying* (OOK) y finalmente la modulación ortogonal del pulso (*Orthogonal Pulse Modulation*, OPM)

Sus aplicaciones más destacadas son los radares de reconocimiento de corto alcance y alta resolución, por ejemplo radares de alance de hasta 10m en 24GHz, sensores en automóviles para detectar y evitar obstáculos cercanos, captura de imágenes a través de estructuras y paredes entre otras.

2.2.2 Sistemas basados en Multi-Band OFDM

En este punto se va a realizar un breve resumen de los sistemas basados en Multi-Band OFDM, ya que en el punto 3 se va a desarrollar más determinadamente.

La propuesta de OFDM está liderada por la alianza multi-banda OFDM (MBOA*, Multiband OFDM Alliance*) de la que forman parte compañías como: Texas Instruments, Intel, Time Domain, entre otras.

En este tipo de transmisión digital, la información se divide en múltiplos flujos paralelos donde cada uno de ellos es transmitido en sub-bandas de 528MHz sobre portadoras ortogonales. La ortogonalidad permite que las portadoras adyacentes puedan solaparse disminuyendo así el ancho de banda necesario, siendo un sistema más eficiente en la utilización del espectro. Este esquema de transmisión digital es conocido como multiplexación por división ortogonal de frecuencia (OFDM*, Orthogonal Frecuency Division Multiplexing*). [6]

Una característica interesante para la transmisión digital utilizando OFDM y que se muestra como una clara ventaja frente a IR-UWB es su muy buena eficiencia en el manejo y control de efectos como: el desvanecimiento lento y rápido, y la dispersión temporal, generados por la propagación multicamino que se presenta en los canales móviles inalámbricos, igualmente OFDM ofrece un adecuado control del efecto doppler, generado este último por el movimiento del receptor y/o estación móvil y de los difusores del canal.

2.3 Reseña Histórica

El año 1973 se concedió la primera patente en comunicaciones UWB aunque este tipo de señal se llevaba trabajando desde tiempo atrás. Más adelante, a UWB le dieron varios nombres como BaseBand (Banda Base), Carrier-Free (Sin portadora) o de impulso en banda base.

A finales de los años 80 se empezó a utilizar el término UWB a partir de que se hicieron públicos los primeros desarrollos de la tecnología UWB de portadora libre por el departamento de defensa de los Estados Unidos que investigaban sus aplicaciones.

En el año 1994 se inventó el radar de impulsos de micro potencia (MIR) que propuso por primera vez un sistema radar UWB que operaba a muy baja potencia con la única alimentación de baterías. Este radar permitía implementar algoritmos de detección muy sensibles a la dirección de llegada.

El año 2000, la agencia de regulación de las comunicaciones (FCC*, Federal Communications Comission*) de los EEUU propuso regular el uso de la señal UWB para usos civiles, bajo la normativa [7]. En esta normativa se definen tres casos de uso de los dispositivos en el ámbito civil:

- Radiocomunicaciones y sistemas de medida
- Uso en comunicaciones vehiculares tipo radar
- Sistemas de reconocimiento o *imaging* como radares de penetración en tierra o muros, o sistemas de reconocimiento médico y de vigilancia

En febrero de 2002, la FCC publicó las primeras regularizaciones para el uso comercial sin licencia de sistemas UWB en Estados Unidos [8], delimitado el rango de frecuencias donde estos sistemas pueden operar entre 3.1 GHz y 10.6 GHz, con un límite de potencia de -41.3 dBm/MHz, debido a esta limitación de potencia el alcance de los sistemas UWB era bastante reducido.

En noviembre del 2005, la ITU-R (rama de radiocomunicaciones de la *International Telecommunication Union*) estableció sus recomendaciones SM (Spectrum Management) 1754, 1755, 1756 y 1757 como marco regulador para el desarrollo de regulaciones.

2.4 Regulación de la tecnología MB-OFDM UWB

La tecnología de multiportadora basada en OFDM está liderada por la alianza Multi-band OFDM (MBOA*, Multiband OFDM Alliance*) de la que forman parte compañías como Texas Instruments, Intel, Sony y Alereon.

Posteriormente, en el año 2005, este alianza se unió al grupo de interés WiMedia Alliance junto con Hewlett-Packard, Samsung y Texas Instruments con el objetivo de fomentar el uso y el desarrollo eficaz de esta tecnología UWB. Al mismo tiempo que MBOA comenzaba sus actividades en torno a MB-OFDM UWB, el grupo de interés UWB Forum, formado por Motorola y Freescale, se fundó para fomentar la tecnología UWB basada en DS-UWB (*Direct Sequence* UWB), como soporte radio para dispositivos de comunicaciones de elevado régimen binario y corto alcance.

Esta confrontación industrial en cuanto la tecnología ha tenido su reflejo en el proceso de estandarización, que ha llevado a cabo el grupo de trabajo del IEEE (IEEE*, Institute of Electrical and Electronics Engineers*). Este grupo de trabajo debía elegir la tecnología de base radio UWB para comunicaciones WPAN de elevado régimen binario (PHY, *Physical High Rate*) pero se disolvió sin decidir una estandarización concreta.

La estandarización a nivel físico dependía del grupo de acceso medio TG3b en el seno de IEEE 802.15. Tras acabar los trabajos de TG3a, el consorcio WiMedia-MBOA recogió las contribuciones UWB de nivel físico (PHY UWB) que se habían realizado y las presentó a la asociación internacional ECMA (*European Computer Manufacturers Association*). Ésta aceptó la propuesta y publicó a finales de 2005 un estándar para fabricación y posterior desarrollo de componentes basado en MB-OFDM UWB. Este estándar técnico es el ECMA-368 [9] que incorpora la capa física radio de la tecnología MB-OFDM UWB y también detalles sobre la capa de acceso al medio (MAC, Medium Access Layer).

Además, se lanzó conjuntamente la especificación ECMA-369 [10] en la cual se desarrolla la interfaz hardware MAC-PHY para MB-OFDM UWB.

En el año 2007, estas especificaciones fueron reconocidas como estándares ISO (ISO, Organización Internacional de Normalización). Esto ha supuesto la adopción definitiva de MB-OFDM UWB como tecnología para WPAN de elevado régimen binario.

En el año 2009, el grupo de interés especial WiMedia Alliance anunció sus planes de realizar la transferencia completa de su tecnología al grupo de fabricantes de USB (USB*, Universal Serial Bus*) para el desarrollo de Wireless USB y también al grupo de interés Bluetooth para el desarrollo del Bluetooth 3.0.

Como la base de este proyecto es la señal UWB Las especificaciones de la tecnología MB-OFDM UWB definidas en el estándar ECMA-368 [9] se van a desarrollar en el capítulo (3).

Una vez vista la regulación de la tecnología MB-OFDM UWB se van a explicar las especificaciones definidas en el estándar ECMA-368 [9].

3 ECMA 368

En este capítulo vamos a centrarnos en explicar cómo se especifica la capa física (PHY) de una señal inalámbrica de área personal (*personal area network* o PAN) para la utilización la máscara espectral de potencia MB-OFDM UWB situada en la banda de frecuencias entre 3.1 Ghz y 10.6 GHz y admitiendo velocidades de datos de 53.3 Mb/s, 80 Mb/s, 106.7 Mb/s, 160 Mb/s, 200 Mb/s, 320 Mb/s, 400 Mb/s, y 480 Mb/s siguiendo el estándar ECMA 368 [9].

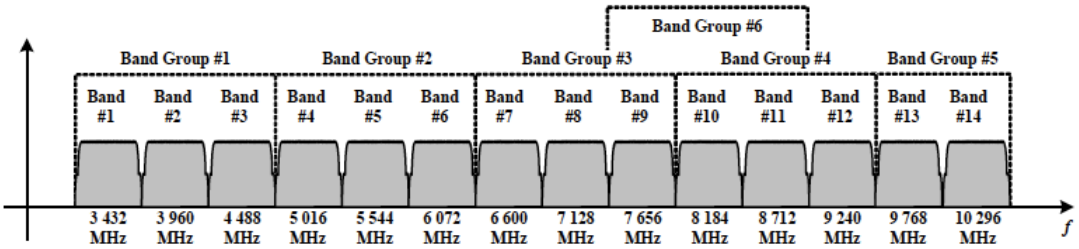

Figura 3. 1 Plan de frecuencias MB-OFDM [9]

La máscara espectral proporciona 14 canales o bandas de 528 MHz de ancho cada una, tal y como se define en [9]. Las 14 bandas se agrupan en 6 grupos de bandas (BG, *Band Group*), los primero cuatros grupos de banda corresponden con las primeras 12 bandas, repartiendo tres bandas para cada grupo de bandas, el BG5 está formado únicamente por dos bandas, siendo estas las bandas 13 y 14 y el BG6 comparte las bandas BG3 y BG4. En la **[Figura 3. 1]** se muestra como se asigna cada banda y grupo de banda mientras que en la **[Tabla 3. 1]** se muestra de forma detallada el plan de frecuencias para MB-OFDM UWB.

Grupo de Banda	BAND-ID (*nb*)	Frecuencia menor (MHz)	Frecuencia central (MHz)	Frecuencia superior (MHz)
1	1	3 168	3 432	3 696
	2	3 696	3 960	4 224
	3	4 224	4 488	4 752
2	4	4 752	5 016	5 280
	5	5 280	5 544	5 808
	6	5 808	6 072	6 336
3	7	6 336	6 600	6 864
	8	6 864	7 128	7 392
	9	7 392	7 656	7 920
0	10	7 920	8 184	8 448
	11	8 448	8 712	8 976
	12	8 976	9 240	9 504
5	13	9 504	9 768	10 032

	14	10 032	10 296	10 560
6	9	7 392	7 656	7 920
	10	7 920	8 184	8 448
	11	8 448	8 712	8 976

Tabla 3. 1 Plan detallado de frecuencias en MB-OFDM [9]

La canalización del sistema MB-OFDM UWB se basa en un conjunto de códigos de tiempo-frecuencia (TFC, *Time-Frequency Codes*) que codifican la frecuencia central, donde se transmite cada símbolo OFDM.

En el estándar se denomina TFI (TFI, *Time-Frequency Interleaved*) a aquellos códigos que saltan en frecuencia entre las tres bandas de un BG (Band Group). FFI (FFI, *Fixed-Frequency Interleaved*) son aquellos códigos que solo usan una banda y no tienen salto en frecuencia y por último TFI2 (TFI2, *Two-band TFI*) son aquellos códigos que realizan salto en frecuencia entre dos de las tres bandas de un BG.

Cada banda MB-OFDM UWB de ancho de banda de 528 MHz está dividida en 128 subportadoras ortogonales entre sí y separadas 4.125 MHz. La separación entre portadoras establece la duración temporal del símbolo OFDM, de valor 242.42ns. Al símbolo OFDM se le añade un prefijo cíclico nulo de 70.08ns para conformar una duración total de símbolo de 312.5ns. Los parámetros temporales del símbolo OFDM se recogen en la **[Tabla 3. 2].**

Parámetros	Descripción	Valor
f_s	Frecuencia de muestreo	528MHz
N_{FFT}	Número total de subportadoras (FFT)	128
N_D	Número de subportadoras de datos	100
N_P	Número de subportadoras pilotos	12
N_G	Número de subportadoras de guarda	10
N_T	Número total de subportadoras utilizadas	122($=N_D+N_P+N_G$)
D_f	Subportadoras de separación de frecuencia	4,125 MHz($=f_s/N_{FFT}$)
T_{FFT}	IFFT y FFT periodo	242.42 ns(Δ_f^{-1})
N_{ZPS}	Número de muestras en el rellenado de cero.	37
T_{ZPS}	Duración rellenado de cero en el tiempo	70,08 ns ($=N_{ZPS}/f_s$)
T_{SYM}	Intervalo de símbolo	312,5 ns($=T_{FFT}+T_{ZPS}$)
F_{SYM}	Velocidad de símbolo	3,2 MHz ($=T_{SYM}^{-1}$)
N_{SYM}	Total de numero de muestras por símbolo	165 ($=N_{FFT}+N_{ZPS}$)

Tabla 3. 2 Parámetros Temporales de la señal MF-OFDM [9]

En el conjunto de 128 subportadoras de un símbolo OFDM existen 6 portadoras nulas, una de ellas la central, tres en la parte derecha o superior y dos en la parte izquierda o baja de la banda. El resto de subportadoras se dividen en 12 subportadoras piloto, 10 subportadoras de guarda y 100 portadoras de datos como se puede observar en **[Figura 3. 18].**

El paquete MB-OFDM UWB está formado por una concatenación de símbolos OFDM con diversa funcionalidad, que va desde la sincronización, la estimación de canal, la cabecera del paquete, hasta la carga o *payload* de datos. El preámbulo de un paquete MB-OFDM está formado por aquellos símbolos dedicados a la sincronización y estimación de canal. Seguidamente se encuentra la cabecera del paquete, este subconjunto lo forman 42 símbolos OFDM como se muestra en la **[Tabla 3. 3]**. La longitud del conjunto de datos del paquete UWB varía en función del régimen binario escogido. Para el máximo régimen binario MB-OFDM UWB de 480 Mbit/s la longitud máxima de datos en un paquete son 1968 símbolos OFDM.

Parámetros	Descripción	Valor
N_{pf}	Número de símbolos en el paquete de la secuencia de sincronización	Estándar Preámbulo: 24 Burst Preámbulo: 12
T_{pf}	Duración del paquete de la secuencia de sincronización	Estándar Preámbulo: 7,5 µs Burst Preámbulo: 3,75 µs
N_{ce}	Número de símbolos en la secuencia de estimación del canal	6
T_{ce}	Duración del canal en la secuencia estimación.	1,875 µs
N_{sync}	Número de símbolos en el PLCP Preámbulo	Estándar Preámbulo: 30 Burst Preámbulo: 18
T_{sync}	Duración del PLCP Preámbulo	Estándar Preámbulo: 9,375 µs Burst Preámbulo: 5,625 µs
N_{hdr}	Número de símbolos en la cabecera PLCP	12
T_{hdr}	Duración de la cabecera PLCP	3,75 µs
N_{frame}	Número de símbolos en el PSDU	$6\times\left\lceil\frac{8\times LENGTH+38}{NIBP6S}\right\rceil$
T_{frame}	Duración del PSDU	$6\times\left\lceil\frac{8\times LENGTH+38}{NIBP6S}\right\rceil\times TSYM$
N_{packet}	Número total de símbolos en el paquete	$N_{sync}+N_{hdr}+N_{frame}$
T_{packet}	Duración del paquete	$(N_{sync}+N_{hdr}+N_{frame})\times T_{SYM}$

Tabla 3. 3 Características Funcionales de un paquete de la señal MB-OFDM UWB [9]

El estándar MB-OFDM UWB contempla dos tipos de modulaciones, QPSK (QPSK, *Quadrature-Phase Shift Keying*) y modulación de doble portadora DCM (DCM, *Dual Carrier Modulation*), para transmitir los datos sobre las 100 subportadoras disponibles. Al mismo tiempo, establece 10 regímenes binarios los cuales van desde 53.3 Mb/s hasta 480 Mb/s.

La modulación QPSK se usa para transmisiones de hasta 200 Mbit/s con diferentes tasas de codificación según sea el régimen binario. Para los regímenes binarios de 320, 400 y 480 Mbit/s se utiliza la modulación DCM. En esta modulación la señal se representa por grupos de 4 bits, y cada grupo de 4 bits se mapea en 2 constelaciones diferentes. Cada constelación posee una subportadora que está separada 50 subportadoras de la anterior, es decir, alrededor de unos 206 MHz, lo que aumenta la robustez de la modulación ante desvanecimientos simultáneos de ambas subportadoras.

En la **[Tabla 3. 4]** se puede observar la correspondencia entre la modulación, las tasas de codificación (*R*) y el régimen binario para comunicaciones MB-OFDM UWB.

Velocidad de datos (Mb/s)	Modulación	Tasa de codificación (R)	FDS	TDS	Coded Bits/ 6 OFDM Symbol (N_{CBP6S})	Info Bits / 6 OFDM Symbol (N_{IBP6S})
53.3	QPSK	1/3	SI	SI	300	100
80	QPSK	1/2	SI	SI	300	150
106.7	QPSK	1/3	NO	SI	600	200
160	QPSK	1/2	NO	SI	600	300
200	QPSK	5/8	NO	SI	600	375
320	DCM	1/2	NO	NO	1200	600
400	DCM	5/8	NO	NO	1200	750
480	DCM	3/4	NO	NO	1200	900

Tabla 3. 4 Parámetros de Modulación de la señal MB-OFDM UWB [9]

3.1 PPDU

En este punto vamos a convertir un PSDU (PSDU*, PHY Service Data Unit*) en un PPDU (PPDU *,PLCP Protocol Data Unit*). Durante la transmisión se añade previamente un preámbulo y una cabecera con el fin de crear los PPDU. En el receptor el preámbulo y la cabecera nos servirán como ayuda en la demodulación, decodificación de los PSDU.

La **[Figura 3. 2]** muestra el formato para el PPDU, está formado por 3 componentes, el PLCP (PLCP *, Physical Layer Convergence Protocol*) preámbulo, la cabecera PLCP y la PSDU.

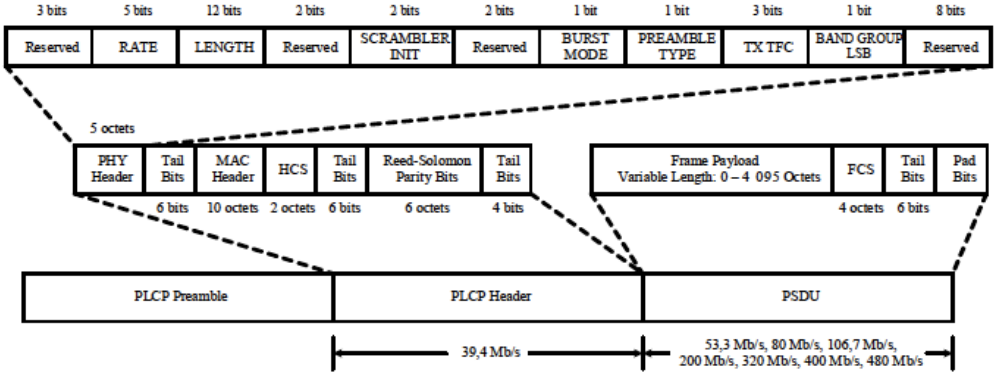

Figura 3. 2 Estructura PSDU [9]

El primer componente es el PLCP preámbulo, el objetivo principal es ayudar al receptor en el tiempo de sincronización mediante portadoras de compensación para la recuperación de la señal.

A continuación tenemos la cabecera, PLCP, con objetivo de transmitir la información necesaria tanto sobre la PHY y MAC, para ayudar la decodificación en el receptor del PSDU, está formado por la cabecera de PHY, la cabecera de MAC, la secuencia de verificación (HCS, *Header Check Sequence*), entre cabeceras se introducen unos bits de cola para devolver al codificador el estado cero. También se introduce un código Reed-Solomon para mejorar la robustez de la cabecera PLCP.

Y finalmente encontramos el PSDU, está compuesto por la secuencia a transmitir, juntamente con una secuencia de verificación (FCS, *Frame Check Sequence*), los bits de cola y finalmente los bits de relleno.

3.1.1 PLCP Preámbulo

El preámbulo PLCP se añade antes de la cabera PLCP para ayudar al receptor en el tiempo de sincronización, estimación del canal. El preámbulo consta de dos partes: una parte en el dominio del tiempo seguida de una porción en el dominio de la frecuencia que será la responsable de realizar la estimación del canal.

Se definen dos tipos de preámbulos, estándar PLCP preámbulo o el preámbulo de ráfagas (*Brust PLCP*). El preámbulo de ráfaga sólo se utiliza en modo de ráfaga, cuando una ráfaga de paquetes se transmite separada un tiempo mínimo, para velocidades de datos de 200 Mb/s o menores.

Sin embargo, para velocidades de datos superiores a 200Mb/s, en el primer paquete se utiliza el estándar PLCP mientras que el resto de paquetes puede utilizarse el estándar o el de ráfaga.

3.1.2 Standard PLCP preámbulo

En la **[Figura 3. 3]** se define la estructura del PLCP preámbulo estándar, tal y como se muestra se puede subdividir en dos partes distintas, una secuencia de sincronización de paquetes y una estimación del canal.

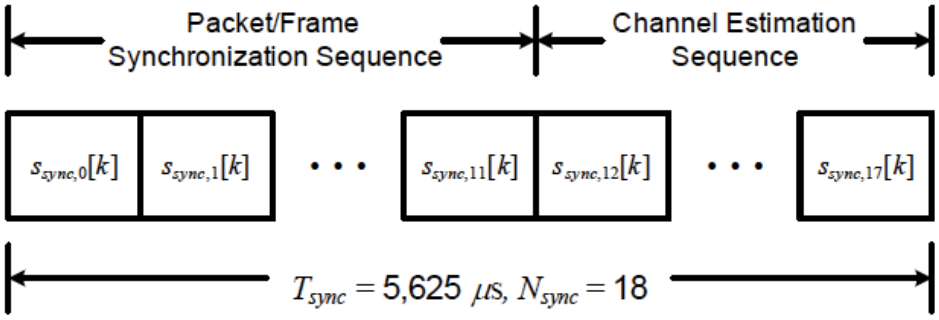

Figura 3. 3 Diagrama de bloques del estándar PLCP Preámbulo [9]

La secuencia de estimación de paquetes se debe formar tal y como se muestra en la **[Figura 3. 4].**

1: Para un determinado código tiempo-frecuencia, se selecciona la base, S_{base} [*l*], apropiada mediante las tablas que encontramos en el estándar [9] y una S_{cover} [l]

2: Se forma la secuencia S_{ext}[l] añadiendo NZPS a la secuencia S_{Base}[l] con longitud NFFT.

3: Finalmente, las K^{th} muestas de N^{th} en el estándar preámbulo $S_{sync,n}$[k] se forman de la siguiente manera:

$$S_{sync,n}[k] = S_{cover}[n] \times S_{ext}[k]$$

Donde $n \in [0, N_{PF} - 1]$, $k \in [0, N_{SYM} - 1]$, N_{PF} se define en la **[Tabla 3. 3]** y N_{SYM} se define en la **[Tabla 3. 2]**.

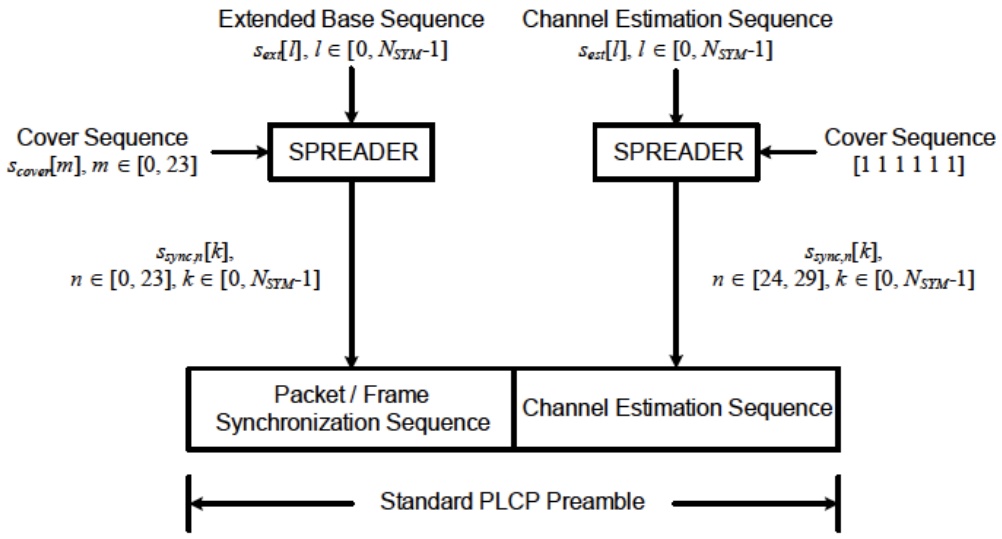

Figura 3. 4 Diagrama de bloques para la construcción del estándar PLCP Preámbulo [9]

La secuencia de estimación de canal se construye siguiendo la **[Figura 3. 4]** Una secuencia de estimación de canal $S_{est}[l]$ es creada mediante la transformada inversa discreta de Fourier (IDFT, *Inverse Discrete Fourier Transform*) de la secuencia del dominio de la frecuencia. A continuación se la añade un rellenado de ceros que consiste en NZPS para obtener la salida en el tiempo.

La secuencia de estimación del canal se crea añadiendo sucesivamente N_{ce} periodos de la secuencia base, se puede escribir como:

$$S_{sync,n}[k] = S_{ext}[k]$$

Donde n ∈ [N_{PF},N_{sync}- 1], k ∈ [0,N_{SYM}- 1], N_{PF} se define en la **[Tabla 3. 3]** y N_{SYM} se define en **[Tabla 3. 2]**.

3.1.3 Burst PLCP preámbulo

La estructura del preámbulo de tipo ráfaga es parecida al estándar, en la **[Figura 3. 5]** se define la estructura del PLCP preámbulo de ráfaga, tal y como se muestra se puede subdividir en dos partes distintas, una secuencia de sincronización de paquetes y una estimación del canal.

La secuencia de estimación de paquetes se debe formar tal y como se muestra en la **[Figura 3. 6].**

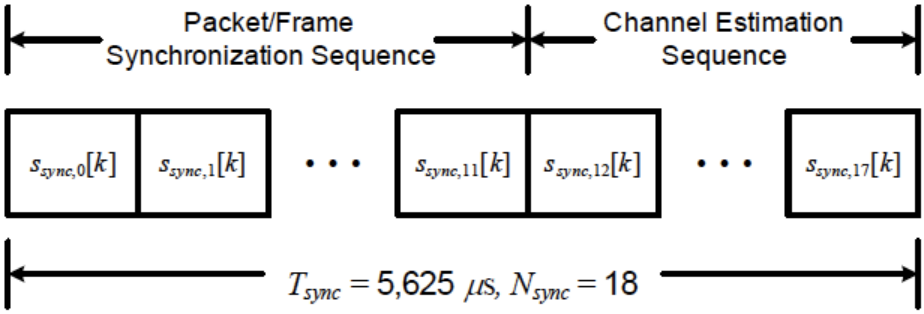

Figura 3. 5 Diagrama de Bloques del Brust PLCP Preámbulo [9]

1: Para un determinado código tiempo-frecuencia, se selecciona la base, S_{base} [*l*], apropiada mediante las tablas que se encuentran en el estándar [9] y una S_{cover} [l].

2: Se forma la secuencia S_{ext}[l] añadiendo NZPS a la secuencia S_{Base}[l] con longitud NFFT

3: Finalmente, las K^{th} muestas de N^{th} en el estándar preámbulo $S_{sync,n}$[k] se forman de la siguiente manera:

$$S_{sync,n}[k] = S_{cover}[n] \times S_{ext}[k]$$

Donde n ∈ [0, N_{PF} - 1], k ∈ [0, N_{SYM} - 1], N_{PF} se define en la **[Tabla 3. 3]** y N_{SYM} se define en **[Tabla 3. 2].**

La secuencia de estimación del canal se crea igual que en el caso del estándar preámbulo, añadiendo sucesivamente N_{ce} periodos de la secuencia base, matemáticamente se puede escribir como:

$$S_{sync,n}[k] = S_{ext}[k]$$

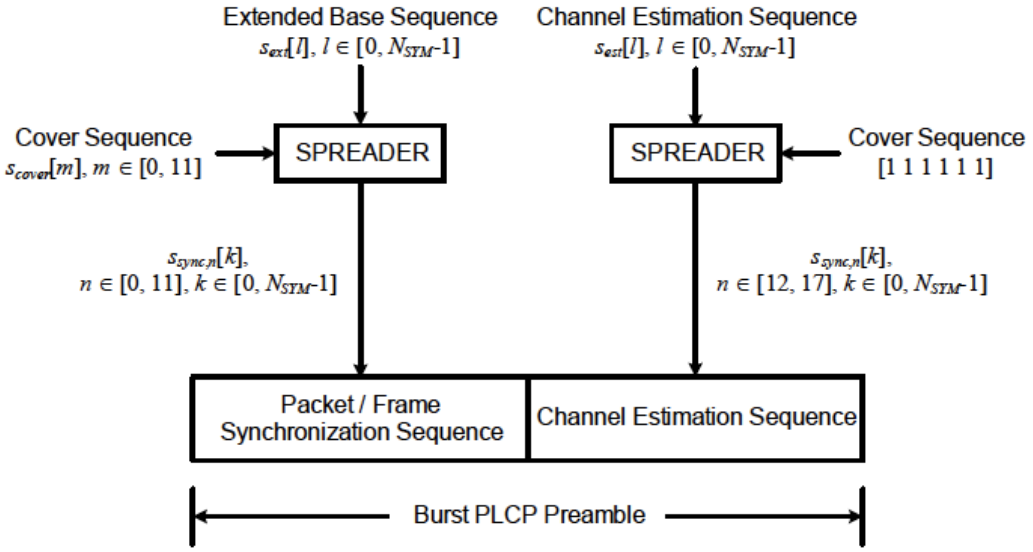

Figura 3. 6 Diagrama de Bloques para la construcción Brust PLCP Preambulo [9]

3.1.4 PLCP Header

La cabecera (PLCP Header), se añade después del preámbulo para transmitir la información sobre el PHY y MAC necesaria para que el receptor pueda descodificarla en el PSDU. En la **[Figura 3. 7]** se muestra cómo se forma el PLCP Header.

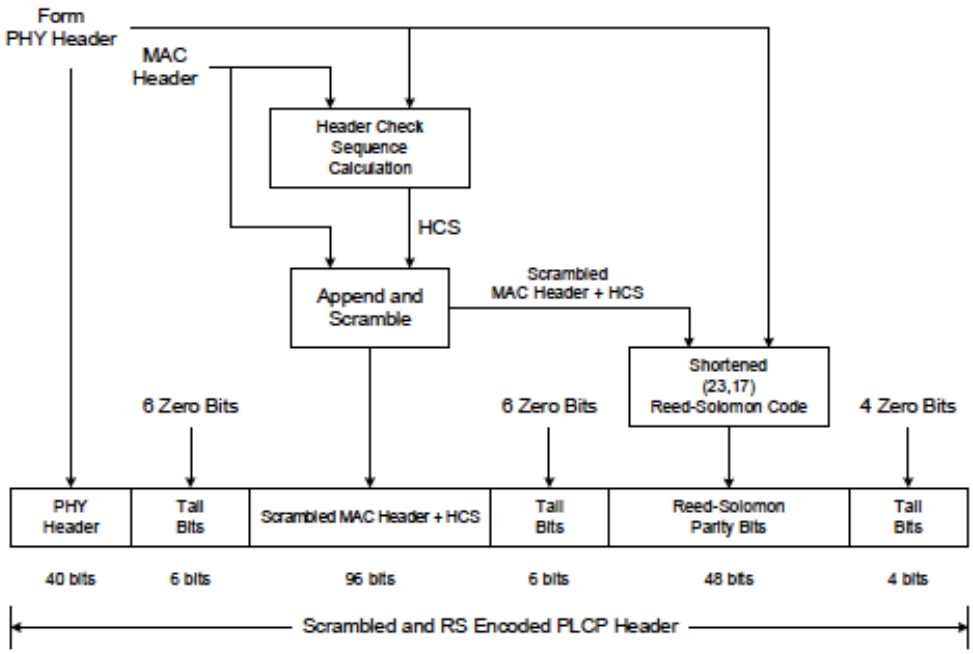

Figura 3. 7 Construcción de la cabecera PLCP [9]

Primero tenemos la información del PHY Header basada en la información proporcionada por el MAC a la que le tenemos que aplicar 6 bits de cola. A continuación, calculamos el valor de HCS (2 octetos) sobre el combinado de cabeceras PHY y MAC. A continuación, el valor resúltate HCS se añade a la cabecera del MAC y se le aplica 6 bits de cola. Finalmente, se aplica el código Reed-Solomon y se le añaden 4 bits de cola.

Una vez tenemos la secuencia de la cabecera, se codifica, se entrelaza, se mapea y finalmente se modula mediante el modulador OFDM tal y como se muestra en la siguiente **[Figura 3. 8]**.

Figura 3. 8 Codificación Cabecera [9]

3.1.5 PHY Header

El PHY header contiene información sobre la velocidad de datos de la MAC, el cuerpo, la longitud de la secuencia, información sobre el próximo paquete de datos. Está formado por una secuencia de 40 bits, numerados del 0 al 39 tal y como se muestra en la **[Figura 3. 9].** A continuación se irá explicando cómo se asigna cada bit.

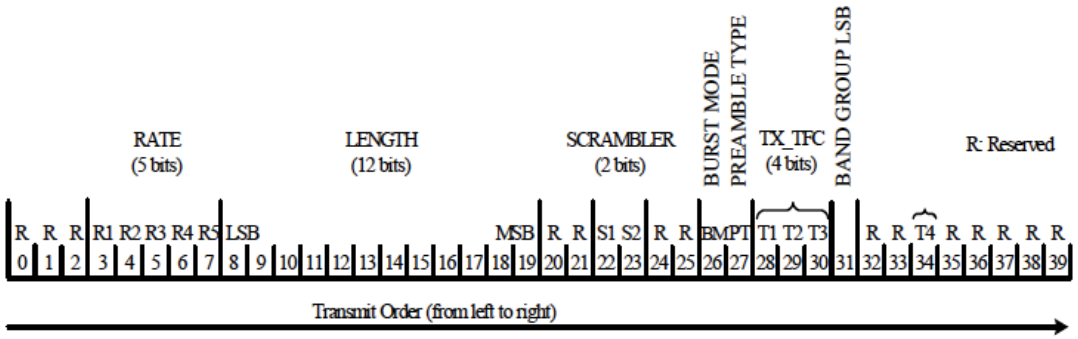

Figura 3. 9 Asignación Bits para PHY [9]

Los bits del 3 al 8, (*RATE*), corresponden con el tipo de modulación, la tasa de codificación. Es decir depende del de la velocidad de transmisión siguiendo la siguiente tabla **[Tabla 3. 5]**:

Velocidad de datos (Mb/s)	R1 - R5
53,3	00000
80	00001
106,7	00010
160	00011

200	00100
320	00101
400	00110
480	00111
Reservado	11111

Tabla 3. 5 Parámetros para la velocidad de datos [9]

Los bits del 8 al 19 (*LENGTH*), indican el número de octetos que hay en la secuencia, se representa mediante 12 bits. En el modo de preámbulo de ráfaga, el valor mínimo debe ser 1, en cambio, en el preámbulo tipo estándar el valor mínimo debe ser 0. El bit menos significativo se sitúa en primer lugar.

Los bits 22 y 23 (S1-S2) nos sirven para sincronizar el decodificador del receptor. El bit 26 (BM, *Burst Mode*) nos informa si el siguiente paquete es parte o no del que se está transmitiendo. Si el bit es "1" quiere decir que el próximo paquete no pertenece a la secuencia en cambio, si el bit es "0" significa que si pertenece. El bit 27 (PT, *Preamble Type*) nos indica si el siguiente preámbulo es de tipo ráfaga o estándar siguiendo la siguiente **[Tabla 3. 6]**.

Tipo De Preámbulo (PT), Bit utilizado en el siguiente paquete.	**Tipo De Preámbulo**
ZERO	Estándar Preámbulo
ONE	Burst Preámbulo

Tabla 3. 6 Tipo de Preámbulo [9]

Cuando se transmite con tasas de 200 Mb/s o superiores el bit se pone a cero, igualmente este bit solo tiene sentido cuando el preámbulo es de tipo de ráfaga, ya que cuando se transmite con un preámbulo estándar se debe poner a 0.

Los bits del 28 al 30, representar los 3 bits menos significativos de la TFC (*Time-Frequency Codes*), que representan el código de tiempo – frecuencia utilizada en el transmisor para el paquete que se está transmitiendo, definidos en la siguiente tabla **[Tabla 3. 7]**:

TF Code	**T1 - T4**
1	1000
2	0100
3	1100
4	0010
5	1010
6	0110
7	1110
8	0001

9	1001
10	0101
Reservados	Todos los otros valores

Tabla 3. 7 Codificación TX TFT [9]

El bit 31, nos sirve para indicar el bit menos significativo del grupo de banda utilizado en el transmisor, en función de la banda podrá un valor u otro siguiendo la tabla **[Tabla 3. 8]**:

Band Group	Band Group LSB (BG_LSB)
1, 3, 5	1
2, 4, 6	O

Tabla 3. 8 Codificación para BF LSB [9]

El bit 34 corresponde con el LBS del TFC que se utiliza en el transmisor. Todos los otros bits que no se han definido, se pondrán a cero, ya que son bits reservados para un futuro y el receptor no los tendrá en cuenta.

3.1.6 PSDU

El PSDU es el último componente principal del PPDU y se construye tal y como se muestra en la **[Figura 3. 10].**

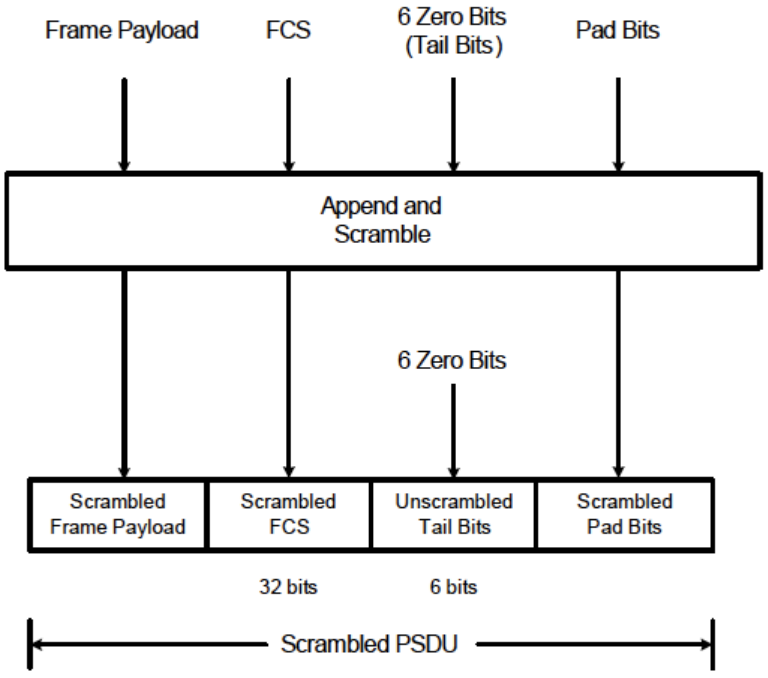

Figura 3. 10 Formato PSDU [9]

3.1.7 Pad Bits

Los bits de relleno se añaden después de los 6 bits de cola antes del mezclado y de la codificación. El rellenado con bits se realiza con el objetivo de asegurar que el PSDU cumpla las condiciones para poderlos entrelazar.

El número de bits de relleno (N_{pad}) viene fijada por la siguiente ecuación.

$$N_{pad} = N_{IBP6S} \times \left[\frac{8 \times LENGTH + 38}{N_{IBP6S}} \right] - \left(8 \times LENGTH + 38 \right)$$

3.2 Reed-Solomon

Para la cabecera del PLCP se utiliza un código Reed-Solomon R = 1 / 3, K = 7 para mejorar la robustez. El código Reed-Solomon está definido sobre un campo de Galois GF(2^8) con un polinomio primitivo $p(z) = z^8 + z^4 + z^3 + z^2 + 1$, donde α es la raíz del polinomio p(z). Por razones de brevedad, este campo de Galois se denota como F.

El elemento $M = b_7z^7 + b_6z^6 + b_5z^5 + b_4z^4 + b_3z^3 + b_2z^2 + b_1z + b^0$, donde $M \in F$, tiene la siguiente representación binaria $b_7b_6b_5b_4b_3b_2b_1b_0$, donde b_7 es el MSB (*Most-Significant Bit*) y b_0 es el LSB (*Least-Significant Bit*). El polinomio generador se obtiene de una reducción sistemática (255, 249) del código Reed-Salomón, que es especificado por el polinomio generador:

$$g(x) = \prod_{i=1}^{6} \left(x - \alpha^i \right) = x^6 + 126x^5 + 4x^4 + 158x^3 + 58x^2 + 49x + 117$$

Donde g(x) es el polinomio generador sobre F, $x \in F$ y los coeficientes son en notación decimal.

El mapeo de la información procedentes de los octetos $m = (m_{248}, m_{247}, \ldots, m_0)$ con los octetos codificados $c = (m_{248}, m_{247}, \ldots, r_5, r_4, .. r_0)$ se logra mediante el cálculo del resto del polinomio r(x)

$$r(x) = \sum_{i=0}^{5} r_i x^i = x^6 m(x) \bmod g(x)$$

Donde m(x) es el polinomio de la información

$$m(x) = \sum_{i=0}^{248} m_{ix^i}$$

Y r_i, i = 0,..., 5, y m_i, i = 0,..., 248, son elementos de F.

$$m_i = 0, i = 17, \ldots . . 248$$

Los 17 octetos del mensaje, están formador por la concatenación de los 5 octetos de la cabecera PHY y los 12 octetos de la cabecera MAC revueltos juntamente con el HCS.

El mensaje esta ordenado de la siguiente manera: m_{16} es el primer octeto del PHY Header, m_{15} es el segundo octeto del PHY, m_{12} es el último octeto del PHY, m_{11} es el primer octeto del MAC Header y HCS y m_0 es el último octeto de la cabecera de MAC y HCS. El primer bit de la cabecera PLCP es mapeado con el LSB del m_{16}, el 9^{th} bit de la cabecera PLCP es mapeado con el LSB m_{15} y así sucesivamente.

El orden de la paridad de los octetos es el siguiente: r_5 es el primer octeto, r_4 es el segundo octeto, y r_0 es el último octeto de la secuencia de paridad del código Reed-Solomon. Una vez más, la asignación en la sección de la paridad de Reed-Solomon de la cabecera PLCP es el bit LSB primero, de tal manera que el primer bit de la paridad de Reed-Solomon se asigna al LSB de r_5, el bit 9 de la paridad de Reed-Solomon se asigna al LSB de r_4,y así sucesivamente. En la **[Figura 3. 11]** se muestra una implementación del código sistemático Reed-Solomon.

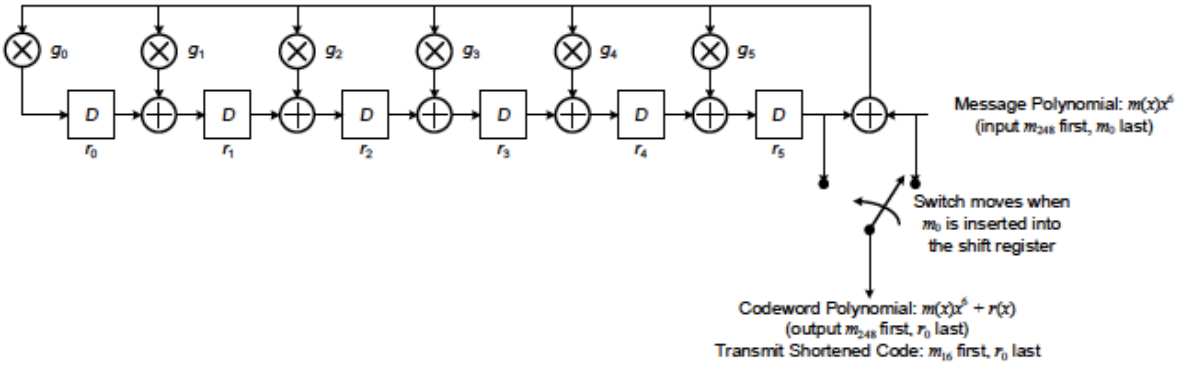

Figura 3. 11 Código sistemático Reed-Solomon [9]

3.3 Header check sequence

Debe haber una comprobación de la cabecera (HCS). Para ello la información de la cabecera PHY y MAC en la cabecera PLCP, deben estar protegidos con un octeto CCITT CRC-16, (CRC, *Cyclic Redundancy Check*). Los cálculos del HCS se realizarán antes de la codificación de datos.

3.4 Data Scrambler

Un mezclador se utiliza para la parte de la cabecera PLCP, es decir, para la cabecera de la MAC y HCS y para toda la PSDU. El codificador debe iniciarse por un valor de inicialización *seed value*, especificado por la MAC en el comienzo de la cabecera MAC y reiniciado al mismo valor al comienzo de la PSDU.

El polinomio generador g(D), procedente del generador de bit aleatorios PRBS (*Pseudo-Random Binary Sequence*), debe ser: $g(D)=1+D^{14}+D^{15,}$ donde D es un bit de retardo. Utilizando este generador, el PRBS correspondiente x[n] se genera como

$$x[n] = x[n - 14] \oplus x[n - 15], n = 0, 1, 2, \ldots \quad (12)$$

Donde "⊕" denota el módulo 2 en adición. La siguiente secuencia define el valor x_{int}, el cual es especificado por el valor *seed value*.

$$xinit = [xi[-1]xi[-2] \ \ldots \ xi[-14] \ xi[-15]]$$

Donde $x_i[-k]$ representa el valor binario inicial de la salida K^{th}. El mezclado de los bits de datos, v_m, se obtiene siguiendo la **[Figura 3. 12]**.

$$v[m] = s[m] \oplus x[m], m = 0, 1, 2, \ldots$$

Donde s[m] representa los bits de datos no codificados. El vector de inicialización se determina a partir del identificador *seed value* en la cabecera PLCP de la trama recibida.

El vector de inicialización de 15 bits correspondientes con el valor *seed value* se define en la **[Tabla 3. 9]**. En la MAC se establece el valor de *seed value* a 00 cuando el PHY se inicia y este valor se incrementa en un contador de renovación de 2 bits por cada trama enviada por el PHY.

Seed Identifier (S1, S2)	**Seed Value** ***xinit* = *xi*[-1] *xi*[-2]… *xi*[-15]**	**PRBS Salida primeros 16 16 bits** ***x*[0] *x*[1] … *x*[15]**
00	0011 1111 1111 111	0000 0000 0000 1000
01	0111 1111 1111 111	0000 0000 0000 0100
10	1011 1111 1111 111	0000 0000 0000 1110
11	1111 1111 1111 111	0000 0000 0000 0010

Tabla 3. 9 Scrambler Seed Selection [9]

Todos los paquetes consecutivos, incluidas las retransmisiones, deberán ser enviados con diferentes *seed value*.

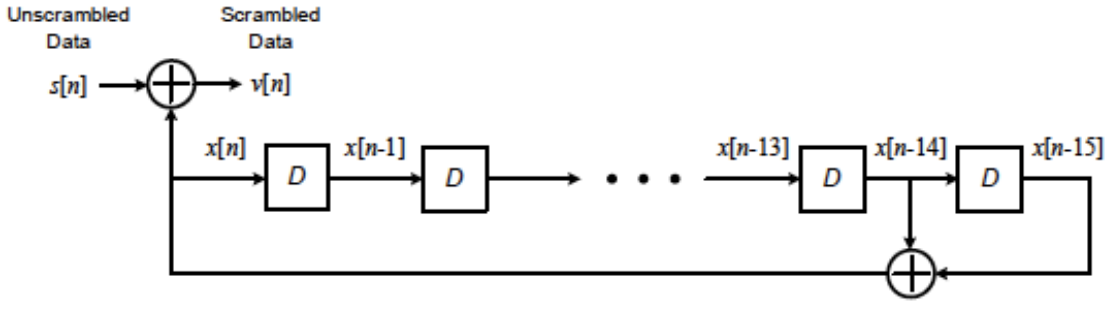

Figura 3. 12 Diagrama de bloques del Scrambler [9]

3.5 Tail Bits

Son necesarios para devolver al codificador convolucional en estado *zero*, ayudando así a evitar posibles errores en él, se sitúan en la cabecera PHY y MAC, HCS y Reed-Solomon…

Después del PHY header y HSC introduciremos seis ceros y después del Reed-Solomon introduciremos cuatro ceros.

3.6 Convolutional Encoder

El codificador convolucional utilizará R=1/3, con un generador de polinomios tal que $g_0=133_8$, $g_1 = 165_8$, y $g_2 = 171_8$, tal como se define en la **[Figura 3. 13].**

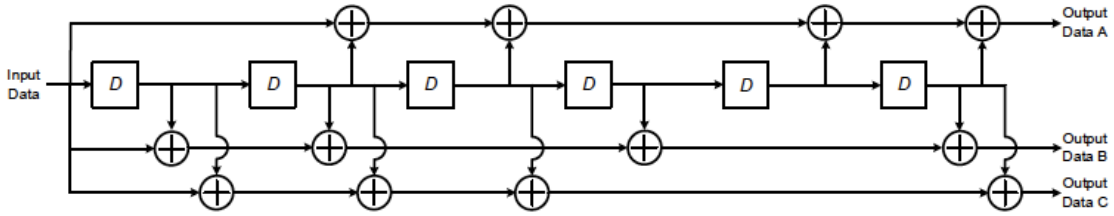

Figura 3. 13 Diagrama De Bloques Del Convolucional Encoder. [9]

El bit denominado como "A" será el primer bit generado por el codificador, seguido por el "B" y finalmente por el denominado "C".

La cabecera PHY, la cabecera MAC, HCS y los bits de cola deben codificarse mediante un codificador convolucional de tasa R=1/3, en cambio, el cuerpo de la trama MAC y los bits de cola se codifican con una tasa R=1/3, 1/2, 5/8 o 3/4, dependiendo de la velocidad de transmisión que se desee. Se recomienda un decodificador con algoritmo de Viterbi.

3.7 Bit interleaving

Antes de la modulación, la secuencia de bits la tenemos que entrelazar para así proporcionales robustez frente posible errores. Este proceso se realiza mediante tres pasos, tal y como se muestra en la **[Figura 3. 14]**.

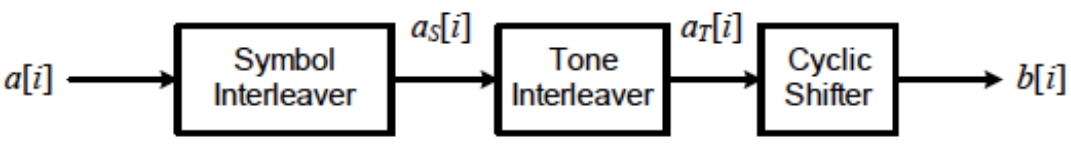

Figura 3. 14 Proceso Bit Interleaving [9]

En primer lugar encontramos, el entrelazado de símbolo, que nos agrupa los bits a través de 6 símbolos consecutivos de la OFDM. La salida del entrelazado de símbolo viene dada por la siguiente relación:

$$a_s[i] = \left\lfloor \left| \frac{i}{N_{CBPS}} \right| + \left(\frac{6}{N_{TDS}} \right) \times \mathrm{mod}(i, N_{CBPS}) \right\rfloor$$

Después, tenemos *Intra-Symbol tone Interteaver*, el cual nos entrelaza bits a través de las portadoras de datos dentro de un símbolo de OFDM, ofreciendo robustez frente interferencias de banda estrecha.

La salida del entrelazado de símbolos, que esta agrupada en bloques de *NCBPS* bits, se intercambia usando un bloque de tamaño $N_{Tint} \times 10$.

$$a_T[i] = a_s\left[\left| \frac{i}{N_{T\,\mathrm{int}}} \right| + 10 \times \mathrm{mod}(i, N_{T\,\mathrm{int}}) \right]$$

Finalmente realizados el *Intra symbol ciyclic shifter* que cíclicamente desplaza los bits en los sucesivos símbolos OFDM

$$a\,[i] = a_T\left[m(i) \times N_{CBPS} + \mathrm{mod}(i + m(i) \times N_{CYC}, N_{CBPS}) \right]$$

3.8 Constellation mapping

En este punto vamos a describir la manera de mapear la señal binaria para obtener una señal de números complejos. Dependiendo de la tasa binaria que tengamos utilizaremos una técnica u otra, para velocidades de datos de 200 Mb/s o inferiores, se realizará mediante una constelación QPSK, y si la velocidad de datos a transmitir es de 320 Mb/s o superior se utilizará una constelación *dual–carrier modulation* (DCM).

3.8.1 QPSK

El código y los datos de entrada binaria se intercalan en serie, b[i] donde i = 0, 1, 2,..., para posteriormente dividirlos en grupos de dos bits y convertirlo en un número complejo. Este número complejo representa una de los cuatro puntos de la constelación QPSK. La conversión se realiza de acuerdo con el código de asignación de la constelación, que se define en la **[Figura 3. 15]**.

Con el bit de entrada, b [2k], donde k = 0, 1, 2,..., los valores de salida, d [k], se forman al multiplicar (2x b [2k] -1)+ j (2x b [2k+1] -1) por un valor de factor de normalización de $K_{mod} = 1/\sqrt{2}$ como se describe en la siguiente ecuación:

$$d[k] = K_{MOD} \times [(2 \times b[2k] - 1) + j(2 \times b[2k+1] - 1)],$$

Para QPSK, b [2k] determina el valor de I, y b [2k +1] determina el valor de Q, tal como se define en la **[Tabla 3. 10]**.

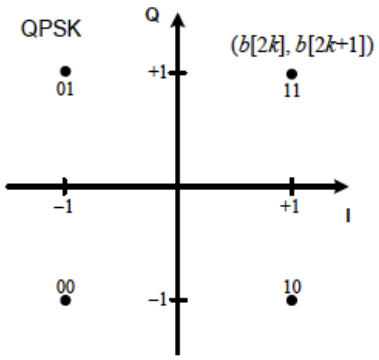

Figura 3. 15 QPSK Constelación

Input Bit (*b*[2*k*], *b*[2*k*+1])	***I*-out**	***Q*-out**
00	-1	**-1**
01	-1	1
10	1	-1
11	1	1

Tabla 3. 10 Encoding QPSK [9]

3.8.2 Dual Carrier Modulation (DCM)

El código y los datos de entrada binaria se intercalan en serie, b [i], se divide en grupos de 200 bits y se convierten en 100 números complejos usando una técnica llamada de doble portadora de modulación (DCM). La conversión se realiza de la siguiente manera:

1. El código de 200 bits se agrupan en 50 grupos de 4 bits. Cada grupo está representado como (b [g (k)], b [g (k) +1], b [g (k) + 50], b [g (k) + 51]), donde k ∈ [0,49].

$$g(k) = \begin{cases} 2k & k \in [0,24] \\ 2k + 50 & k \in [25,49] \end{cases}$$

2. Cada grupo de 4 bits (b [g (k)], b [g (k) +1], b [g (k) + 50], b [g (k)+ 51]), representa un punto de la constelación, tal como se define en la **[Figura 3. 16]** y en la **[Figura 3. 17].** Para posteriormente convertirlo en dos números complejos (d [k], d [k + 50]) en función de la siguiente **[Tabla 3. 11].**

Input Bit (*b*[*g*(*k*)], *b*[*g*(*k*)+1], *b*[*g*(*k*) + 50)], *b*[*g*(*k*) + 51])	***d*[*k*] *I*-out**	***d*[*k*] *Q*-out**	***d*[*k* + 50] *I*-out**	***d*[*k* + 50] *Q*-out**

0000	-3	-3	1	1
0001	-3	-1	1	-3
0010	-3	1	1	3
0011	-3	3	1	-1
0100	-1	-3	-3	1
0101	-1	-1	-3	-3
0110	-1	1	-3	3
0111	-1	3	-3	-1
1000	1	-3	3	1
1001	1	-1	3	-3
1010	1	1	3	3
1011	1	3	3	-1
1100	1	-3	-1	1
1101	3	-1	-1	-3
1110	3	1	-1	3
1111	3	3	-1	-1

Tabla 3. 11 DMC Encoding [9]

3. Finalmente los números complejos se normalizan mediante el factor $K_{mod}=1/\sqrt{10}$.

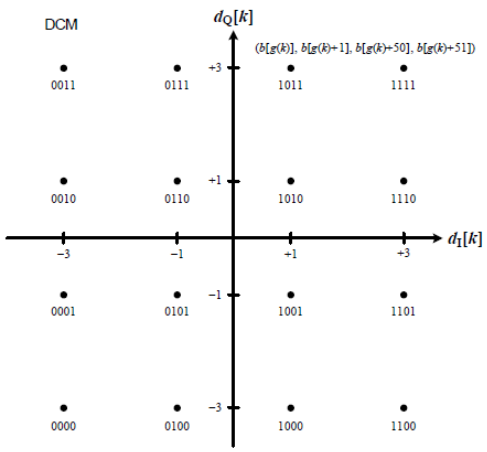

Figura 3. 16 DCM Encoding, Mapeo para d[k], [9]

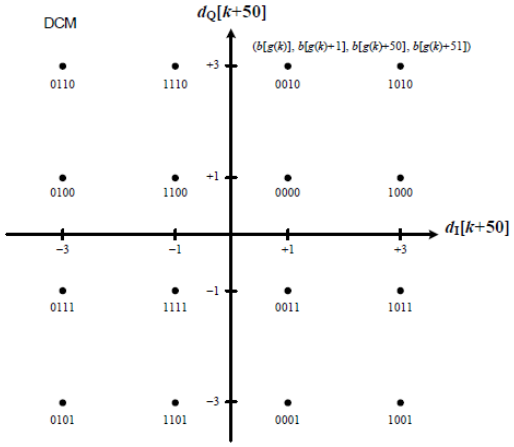

Figura 3. 17 DCM Encoding, Mapeo para d[k+50], [9]

3.9 Modulación OFDM

Según el estándar ECMA 368, la señal es modulada a través de un modulador tipo OFDM. Para ello, se utiliza 110 subportadoras, de las cuales 100 de son de datos y 10 son de guarda por banda de transmisión. Además se utiliza 12 portadoras piloto para la detección coherente.

La señal en el tiempo discreto, $s_n[k]$ se obtiene mediante la aplicación de la IDFT a la secuencia de valores complejos tal y como se muestra a continuación:

$$S_{hd,n}[k] = \frac{1}{\sqrt{N_{FFT}}} \left[\begin{array}{l} \sum_{l=0}^{N_D} C_{D,n}[l] \exp(2j\pi M_D[l]k / N_{FFT}) + \\ \sum_{l=0}^{N_G} C_{G,n}[l] \exp(2j\pi M_G[l]k / N_{FFT}) + \\ \sum_{l=0}^{N_P} C_{P,n}[l] \exp(2j\pi M_P[l]k / N_{FFT}) \end{array} \right]$$

Donde k ∈ [0, NFFT - 1], n ∈ [Nsync, Npacket - 1], N_D es el número de subportadoras de datos, N_G es el número de subportadoras de guarda, N_P es el número de subportadoras piloto, NFFT es el número total de subportadoras, y C_D, n [l], G_C, n [l], C_P, n [l] son los números complejos puesto l^{th}, de las subportadoras de datos, guarda, y pilotos del n^{th} símbolo OFDM, respectivamente.

La relación entre el C_D, n[l] ,G_C, n [l], C_P, n[l] se definen en apartados posteriores Las funciones de M_D [l], M_G [l], y M_P [l] definen el mapeo de los índices [0, N_D−1], [0, N_G−1], y [0, N_P−1] de las subportadoras en frecuencia [−N_T / 2, N_T / 2] excluyendo el 0, se definen tal y como se muestra en las siguientes figuras:

$$M_{G,n}[l] = \begin{cases} -61 + l & l \in \left[0, \frac{N_G}{2} - 1\right] \\ 52 + l & l \in \left[\frac{N_G}{2}, N_G - 1\right] \end{cases}$$

$$M_P[l] = -55 + 10L \quad l \in [0, N_P - 1]$$

$$M_{D,n}[l] = \begin{cases} l-56 & l = 0 \\ l-55 & 1 \le l \le 9 \\ l-54 & 10 \le l \le 18 \\ l-53 & 19 \le l \le 27 \\ l-52 & 28 \le l \le 36 \\ l-51 & 37 \le l \le 45 \\ l-50 & 46 \le l \le 49 \\ l-49 & 50 \le l \le 53 \\ l-48 & 54 \le l \le 62 \\ l-47 & 63 \le l \le 71 \\ l-46 & 72 \le l \le 80 \\ l-45 & 81 \le l \le 89 \\ l-44 & 90 \le l \le 98 \\ l-43 & l = 99 \end{cases}$$

El mapeo de las subportadoras de datos, piloto y guarda dentro de un símbolo OFDM se define en la **[Figura 3. 18].**

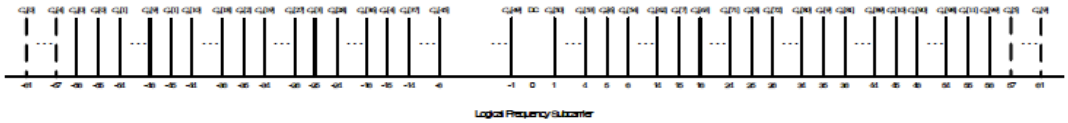

Figura 3. 18 Mapeo en frecuencia lógica de las subportadoras de datos, guarda y piloto [9]

Finalmente, las señales de tiempo discreto para la cabecera PLCP, $s_{hdr,n}[k]$, y la PSDU, $s_{frame,n}[k]$, se creará de la siguiente manera, añadiendo un sufijo con rellenado de ceros (ZPS, Zero Padded Suffix) para cada salida IDFT:

$$S_{hrd,n}[k] = \begin{cases} S_n[k] & k \in [0, N_{FFT} - 1] \\ 0 & k \in [N_{FFT} - 1, N_{SYM} - 1] \end{cases}$$

Para n ∈ [N_{sync}, N_{sync} + N_{hdr} − 1], and

$$S_{frame,n}[k] = \begin{cases} S_n[k] & k \in [0, N_{FFT} - 1] \\ 0 & k \in [N_{FFT} - 1, N_{SYM} - 1] \end{cases}$$

Para n ∈ [N_{sync} + N_{hdr}, N_{packet} − 1]. El rellenado con ceros se suele utilizar para reducir los efectos del *multi-path* para proporcionar un intervalo de guarda para que el transmisor y el receptor puedan cambiar las frecuencias centrales.

3.9.1 Consideraciones para la implementación

Una forma común de implementar una trasformada de Fourier inversa discreta es haciendo uso del algoritmo de la relación inversa de la transformada rápida de Fourier (IFFT). En este caso, la frecuencia de la subportadoras [$-N_T/2$, $N_T/2$] se asignan de acuerdo a la **[Figura 3. 19].**

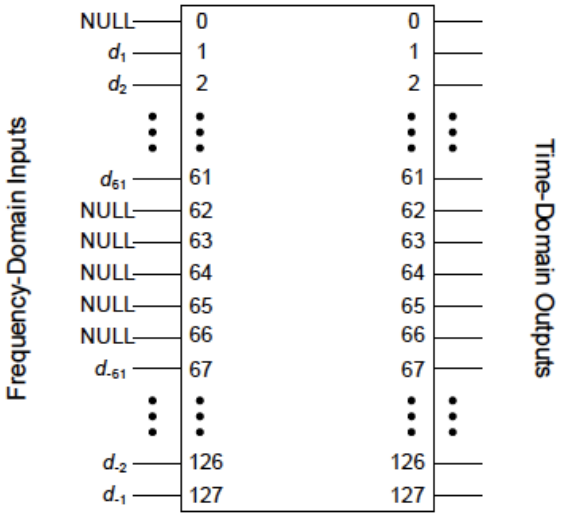

Figura 3. 19 Relación entre entrada y salida para la IFFT [9]

Las subportadoras de la 1 a la 61 se asignan a las mismas entradas de la IFFT mientras que las subportadoras del -61 a -1 se asignan a las entradas 67 hasta la 127 de la IFFT. El resto de entradas, 62 a 66 y la entrada 0, se ponen a cero.

3.9.2 Subportadoras de Datos

El mapeo de la secuencia de números complejos y las subportadoras de datos depende de la parte del PPDU, preámbulo, cabecera o PSDU y de la velocidad de datos.

3.9.3 Mapeo para la cabecera:

Tanto en el dominio temporal como en la frecuencia se utiliza esta técnica. Para el caso de la cabecera, $d_{hdr}[k]$, donde k= 0,1,2,3…. será la secuencia d[*k*] definida en (3.8.1), $d_{hdr}[k]$ se agrupa en conjuntos de $N_D/2=50$ números complejos. Este grupo de números complejos deben ser mapeados en l^{th} subportadoras de datos de los n^{th} OFDM símbolos, $C_{D,n,}[l]$ tal y como se muestra a continuación

$$C_{D\,2n}[l] = d_{hdr}\left[\frac{N_D}{4}\times\left(2n - N_{sync}\right)+1\right]$$

$$C_{D\,2n}\left[l+\frac{N_D}{2}\right] = d_{hdr}{}^{*}\left[\frac{N_D}{4}\times\left(2n - N_{sync}\right)+\left(\frac{N_D}{2}-1-l\right)\right]$$

$$C_{D\,2n+1}[l] = p_{spread}[n]\times d_{hdr}\left[\frac{N_D}{4}\times\left(2n - N_{sync}\right)+l\right]$$

$$C_{D\,2n+1}\left[l+\frac{N_D}{2}\right] = p_{spread}[n]\times d_{hdr}{}^{*}\left[\frac{N_D}{4}\times\left(2n - N_{sync}\right)+\left(\frac{N_D}{2}-1-l\right)\right]$$

Dónde:

$$p_{spread}[n] = p\left[\mathrm{mod}\left(n-\frac{N_{sync}}{2}\right)+6, N_{FFT}-1\right]$$

P[n] tiene una longitud de 127 secuencias pseudo-aleatorias, $l\in\left[0,\frac{N_D}{2}-1\right], n\in\left[\frac{N_{sync}}{2},\frac{N_{sync}+N_{hdr}}{2}-1\right]$, N_D es el número de portadoras de datos, N_{sync} es el número de símbolos en el Preámbulo y N_{hdr} es el número de símbolos en la cabecera.

3.9.4 Mapeo para velocidad de datos 53,3 Mb/s y 80 Mb/s

Este caso es para el dominio temporal tanto como el frecuencial y velocidades de datos de 53,3 Mb/s y 80 Mb/s en el PSDU, la secuencia de números complejos, $d_{frame}[k]$, donde k= 0,1,2,3…. está definida en punto (3.8.1).

La secuencia $d_{frame}[k]$ se agrupa en conjuntos de $N_D/2=50$ números complejos que se proyectarán sobre l^{th} subportadoras de datos de los n^{th} OFDM símbolos, $C_{D,n,}[l]$ tal y como se muestra a continuación.

$$C_{D,2n}[l] = d_{fame}\left[\frac{N_D}{4}\times\left(2n - N_{Sya} - N_{hdr}\right)+l\right]$$

$$C_{D,2n}\left[l,\frac{N_D}{2}\right] = d_{fame}{}^{*}\left[\frac{N_D}{4}\times\left(2n - N_{Sya} - N_{hdr}\right)+\left(\frac{N_D}{2}-1-l\right)\right]$$

$$C_{D,2n+1}[l] = p_{spead}[n]\times d_{fame}\left[\frac{N_D}{4}\times\left(2n - N_{Sya} - N_{hdr}\right)+l\right]$$

$$C_{D,2n+1}\left[l,\frac{N_D}{2}\right] = p_{spead}[n]\times d_{fame}{}^{*}\left[\frac{N_D}{4}\times\left(2n - N_{Sya} - N_{hdr}\right)+\left(\frac{N_D}{2}-1-l\right)\right.$$

Donde $P_{spread}[n]$ está definido en el mapeo para la cabecera (3.9.3**¡Error! No se ncuentra el origen de la referencia.**) y P[n] tiene una longitud de 127 pseudo-aleatorias secuencias, cuyos valores se definen en la tabla 33 del estándar [9], $l \in \left[0, \frac{N_D}{2} - 1\right], n \in \left[\frac{N_{sync} + N_{hdr}}{2}, \frac{N_{packet}}{2} - 1\right]$, N_D es el número de portadoras de datos, N_{sync} es el número de símbolos en el Preámbulo y N_{hdr} es el número de símbolos en la cabecera.

3.9.5 Mapeo para velocidad de datos 106,7 Mb/s, 160 Mb/s, y 200 Mb/s

Para el caso de velocidades de datos de 106,7 Mb/s, 160 Mb/s, y 200 Mb/s sólo se utiliza en el domino temporal. La secuencia de números complejos, $d_{frame}[k]$, donde k= 0,1,2,3…. está definida en el punto (3.8) de la PSDU.

La secuencia $d_{frame}[k]$ se agrupa en conjuntos de N_D=100 números complejos que se proyectarán sobre l^{th} subportadoras de datos de los n^{th} OFDM símbolos, $C_{D,n,}[l]$ tal y como se muestra a continuación.

$$C_{D,2n}[l] = d_{frame}\left[\frac{N_D}{2} \times (2n - N_{sync} - N_{hdr}) + l\right]$$

$$C_{D,2n+1}[l] = p_{spread}[n] \times \left\{ \begin{array}{l} imag\left(d_{frame}\left[\frac{N_D}{2} \times (2n - N_{sync} - N_{hdr}) + (N_D - 1 - l)\right]\right) \\ + j \quad real\left(d_{frame}\left[\frac{N_D}{2} \times (2n - N_{sync} - N_{hdr}) + (N_D - 1 - l)\right]\right) \end{array} \right\}$$

Donde $P_{spread}[n]$ está definido en el mapeo para la cabecera (3.9.3) y P[n] tiene una longitud de 127 pseudo-aleatorias secuencias, cuyos valores se definen en la tabla 33 del estándar [9], $l \in [0, N_D - 1], n \in \left[\frac{N_{sync} + N_{hdr}}{2}, \frac{N_{paket}}{2} - 1,\right]$, N_D es el número de portadoras de datos, N_{sync} es el número de símbolos en el Preámbulo y N_{hdr} es el número de símbolos en la cabecera

3.9.6 Mapeo para velocidad de datos 320 Mb/s, 400 Mb/s, y 480 Mb/s

Para el caso de velocidades de datos de 320 Mb/s, 400 Mb/s, y 480 Mb/s. La secuencia de números complejos, $d_{frame}[k]$, donde k= 0,1,2,3…. está definida en el punto (3.9.3) de la PSDU.

La secuencia $d_{frame}[k]$ se agrupa en conjuntos de N_D=100 números complejos que se proyectarán sobre l^{th} subportadoras de datos de los n^{th} OFDM símbolos, $C_{D,n,}[l]$ tal y como se muestra a continuación

$$C_{D,n}[l] = d_{frame}\left[N_D \times (n - N_{sync} - N_{hdr}) + l\right]$$

Donde $p_{spread}[n]$ está definido en (3.9.3) y P[n] tiene una longitud de 127 pseudo-aleatorias secuencias, cuyos valores se definen en la tabla 33 del estándar [9], $l \in [0, N_D - 1], n \in [N_{sync} + N_{hdr}, N_{paket} - 1]$, N_D es el número de portadoras de datos, N_{sync} es el número de símbolos en el Preámbulo y N_{hdr} es el número de símbolos en la cabecera

3.9.7 Subportadoras de Guarda

Para cada símbolo OFDM, a partir de la secuencia de estimación de canal del PLCP preámbulo, habrá diez subportadoras de guarda, 5 en cada extremo de la banda de frecuencias ocupada.

La relación entre los niveles de potencia de las subportadoras de guarda y la de las subportadoras de datos depende de la implementación. Esta relación se mantiene constante dentro de un paquete, es decir, desde el inicio del canal estimación de la secuencia hasta el final del paquete. Además, los niveles de potencia de las subportadoras de guarda serán elegidos de tal manera que se garantice que la señal transmitida reúna los requisitos de ancho de banda ocupado.

Las 10 subportadoras de guardia se encuentran en los bordes del símbolo OFDM, en las frecuencias lógicas de subportadoras -61, -60,..., -57, y 57, 58,..., 61. Los datos sobre estas subportadoras deben ser creados tal y como se define a continuación:

$$C_{G,n}[l] = \begin{cases} C_{D,n}[l] & l \in \left[0, \frac{N_G}{2} - 1\right] \\ C_{D,n}[l] + 90 & l \in \left[\frac{N_G}{2}, N_G - 1\right] \end{cases}$$

Donde $C_{G,n}[l]$ es el l^{th} subportadora de guarda del n^{th} ODFM símbolo $n \in [N_{sync}, N_{packet} - 1]$, N_{sync}, es el número de símbolos en el preámbulo PLCP y N_{packet} es el número totales de los símbolos en el paquete.

3.9.8 Subportadoras Pilot

Se introducirán 12 portadoras pilotos para cada símbolo OFDM y detrás del PLCP preámbulo, con el objetivo de permitir la detección coherente y proporcionar robustez frente a las compensaciones de frecuencia y ruido de fase.

Estas subportadoras se colocarán en las frecuencias lógicas -55,-45, -35, -25, -15, -5, 5, 15, 25, 35, 45 y 55. El valor de dichas portadoras depende de la parte del PPDU y la velocidad de datos.

3.9.9 Mapeo para la cabecera:

Para la parte de la cabecera PLCP del PPDU, la información de l[th] subportadoras pilotos para n[th] OFDM símbolos se define como

$$C_{p,2n}[l] = p\left[\mathrm{mod}\left(n - \frac{N_{sync}}{2}, N_{FFT} - 1\right)\right] \times d_{pilot,cs}[l]$$

$$C_{p,2n+1}[l] = p\left[\mathrm{mod}\left(n - \frac{N_{sync}}{2}, N_{FFT} - 1\right)\right] \times p_{spread}[n] \times d_{pilot,cs}[l]$$

Donde

$$d_{pilot,cs}[l] = \begin{cases} \frac{1-j}{\sqrt{2}} & l = 0,3 \\ \frac{-1+j}{\sqrt{2}} & l = 1,2,4,5 \\ \frac{1+j}{\sqrt{2}} & l = 8,11 \\ \frac{-1-j}{\sqrt{2}} & l = 6,7,9,10 \end{cases}$$

Donde p[n] está definido en la tabla 33 del estándar [9], $p_{spread}[n]$ está definido en el mapeo para la cabecera (3.9.3), $n \in \left[\frac{N_{sync}}{2}, \frac{N_{sync} + N_{hdr}}{2} - 1\right]$ donde N_{sync} es el número de símbolos en el preámbulo y N_{hdr} es el número de símbolos en la cabecera.

3.9.10 Mapeo para velocidad de datos 53,3 Mb/s y 80 Mb/s

Para el PPDU y velocidades de datos de 53,3 Mb/s y 80 Mb/s, la información de l[th] subportadoras pilotos para n[th] OFDM símbolos se define como

$$C_{p,2n}[l] = p\left[\mathrm{mod}\left(n - \frac{N_{sync}}{2}, N_{FFT} - 1\right)\right] \times d_{pilot,cs}[l]$$

$$C_{p,2n+1}[l] = p\left[\mathrm{mod}\left(n - \frac{N_{sync}}{2}, N_{FFT} - 1\right)\right] \times p_{spread}[n] \times d_{pilot,cs}[l]$$

Donde $d_{pilot}[n]$ está definido en el mapeo para la cabecera (3.9.3), p[n] está definido en la tabla 33 del estándar [9] $p_{spread}[n]$ está definido en el mapeo para la cabecera (3.9.3), $n \in \left[\frac{N_{sync}+N_{hdr}}{2}, \frac{N_{packet}}{2}-1\right]$ donde N_{sync} es el número de símbolos en el preámbulo, N_{hdr} es el número de símbolos en la cabecera y N_{packet} es el número de símbolos en el paquete.

3.9.11 Mapeo para velocidad de datos 106,7 Mb/s, 160 Mb/s, y 200 Mb/s

Para el PPDU y velocidades de datos de 106,7 Mb/s, 160 Mb/s, y 200 Mb/s, la información de l^{th} subportadoras pilotos para n^{th} OFDM símbolos se define como

$$C_{p,2n}[l] = p\left[\mathrm{mod}\left(n-\frac{N_{sync}}{2}, N_{FFT}-1\right)\right] \times d_{pilot,ncs}[l]$$

$$C_{p,2n+1}[l] = p\left[\mathrm{mod}\left(n-\frac{N_{sync}}{2}, N_{FFT}-1\right)\right] \times p_{spread}[n] \times d_{pilot,ncs}[l]$$

Dónde:

$$d_{pilot,ncd}[l] = \begin{cases} \frac{1+j}{\sqrt{2}} & l = 0,3,8,11 \\ \frac{-1-j}{\sqrt{2}} & l = 1,2,4,5,6,7,9,10 \end{cases}$$

Donde p[n] está definido en tabla 33 del estándar [9], $p_{spread}[n]$ está definido en el mapeo para la cabecera (3.9.3), $n \in \left[\frac{N_{sync}+N_{hdr}}{2}, \frac{N_{packet}}{2}-1\right]$ donde N_{sync} es el número de símbolos en el preámbulo, N_{hdr} es el número de símbolos en la cabecera y N_{packet} es el número de símbolos en el paquete.

3.9.12 Mapeo para velocidad de datos 320 Mb/s, 400 Mb/s, y 480 Mb/s

Para el PPDU y velocidades de datos de 320 Mb/s, 400 Mb/s, y 480 Mb/s, la información de l^{th} subportadoras pilotos para n^{th} OFDM símbolos se define como

$$C_{p,n}[l] = p\left[\mathrm{mod}\left(n-N_{sync}-\frac{N_{hdr}}{2}, N_{FFT}-1\right)\right] \times d_{pilot,ncs}[l]$$

Donde $d_{pilot}[n]$ está definido en (3.9.3), p[n] está definido en la tabla 33 del estándar [9], $n \in \left[N_{sync} + N_{hdr} + N_{packet} - 1\right]$ donde N_{sync} es el número de símbolos en el preámbulo, N_{hdr} es el número de símbolos en la cabecera y N_{packet} es el número de símbolos en el paquete.

4 Generación ECMA 368 mediante Matlab:

En este punto vamos a simular mediante Matlab una señal UWB con una velocidad de datos de 480Mb/s, para ello seguiremos el estándar ECMA 368 [9]. En esta simulación sólo elegiremos una banda. Como se ha explicado en el punto (3) la señal UWB con velocidades de datos superiores a 200Mb/s se mapea con una constelación tipo DCM. A lo largo de este capítulo se van a nombrar funciones que se encuentran en el anexo.

Tal y cómo se ha comentado, simularemos una señal con una velocidad de datos de 480Mb/s, por eso primero almacenaremos en la variable Rb, referente a la velocidad de datos de dicho valor.

```
Rb=480e6;          % Bit Rate de la señal
```

Siguiendo el estándar, seguiremos definiendo los parámetros que nos caracterizan una señal con una velocidad de datos de 480Mb/s y siguiendo las tablas relacionadas con la frecuencia **[**¡Error! No se encuentra el origen de la referencia.**]** y el tiempo **[Tabla 3. 2].**

```
Rate=[0 0 1 1 1];
R=3/4;
FDS=0;
TDS=0;
Ncbp6s=1200;
Nibp6s=900;
```

```
fs=528e6;          % Sampling frequency
NFFT= 128;         % Number of carriers of the OFDM system(FFT size)
Nd=100;            % Number of data subcarriers
Np=12;             % Number of pilot subcarriers
Ng=10;             % Number of guard subcarriers
Nt=122;            % Total number of subcarriers used
 Df=4.125e6;       % Subcarrier frequency spacing
Tfft=242.42e-9;    % IFFT and FFT period (information length)
Nzps=37;           % Number of samples in zero-padded suffix
Tzps= 70.08e-9;    % Zero-padded suffix duration in time
Tsym=312.5e-9;     % Symbol interval
Fsym=3.2e6;        % Symbol rate
Nsym=165;          % Total number of samples per symbol
```

Una vez ya tenemos los parámetros que nos caracterizan la señal en el tiempo y en la frecuencia seguiremos definiendo parámetros, ahora los relacionados con el PHY siguiendo la **[Tabla 3. 3].**

```
if 0<Rb<200e6 % Brust Preamble
   Npf=12;
   Tpf=3.75e-6;
```

```
   Nsync=18;
   Tsync=5.625e-6;
end

if Rb>=200e6; % Standard Preamble
   Npf=24;
   Tpf=7.5e-6;
   Nsync=30;
   Tsync=9.375e-6;
end
Nce =6;
Tce=1.875e-6;
Nhdr=12;          %Number of symbols in the PLCP header
Thdr=3.75e-6;      %Duration of the PLCP header
Nframe=6.*ceil(((8*4800+38)/Nibp6s));
Tframe=6*ceil(((8*4800+38)/Nibp6s))*Tsym;
Npacket=Nsync+Nhdr+Nframe;
Tpacket=(Nsync+Nhdr+Nframe)*Tsym;
```

Tal y cómo se ha comentado el capítulo anterior, el PSDU está compuesto de tres bloques: el preámbulo, la cabecera y el PPDU. Para crear la señal iremos definiendo cada uno de estos bloques

4.1 PLCP Preámbulo

Cómo hemos definido la tasa binaria en 480Mbps, y tal como el estándar nos indica, el primer paquete debe ser de tipo estándar y los siguientes, pueden ser tanto de tipo estándar cómo de ráfaga.

Los valores necesarios para realizar el preámbulo son: el número de símbolos en el paquete de sincronización, la duración del paquete de sincronización, el número de símbolos del PLCP preámbulo y la duración para una tasa binaria de 480Mbps, obtenidos previamente cuando hemos definido los parámetros de entrada.

Por otro lado, necesitaremos las variables S_{base} y S_{cover}. Primero obtendremos S_{base}, la cual depende del valor TF y siguiendo las tablas del estándar ECMA 368 [9] se obtiene:

```
Sbase(1,:)=[0.6564,-1.3671,-0.9958,-1.3981,0.8481,1.0892,-0.8621,1.1512,0.9602,-1.3581,-0.8354,-
1.3249,1.0964,1.3334,  -0.7378,1.3565,0.9361,-0.8212,-0.2662,-0.6866,0.8437,1.1237,  0.3265,  1.0511
,0.7927,-0.3363,-0.1342,-0.1546,0.6955, 1.0608,-0.1600,0.9442,-0.0844,1.1974,1.2261,1.4401,-0.5988,-
0.4675,0.8520,-0.8922,-0.5603,1.1886,1.1128,  1.0833,-0.9073,-1.6227,1.0013,-1.6067,0.3360,-1.3136,-
1.4447,-1.7238,1.0287,0.6100,-0.9237,1.2618,0.5974,-1.0976,-0.9776,-0.9982,0.8967,1.7640,    1.0211,
```

```
1.6913, -0.2095, 1.1640, 1.2334, 1.5338, -0.8844, -0.3857, 0.7730,-0.9754,-0.2315, 0.5579, 0.4035
,0.4248, -0.3359, -0.9914,0.5975,-0.8408,0.3587,-0.9604,-1.0002,-1.1636,0.9590,0.7137, 0.6776,
0.9824, -0.5454, 1.1022, 1.6485, 1.3307, -1.2852, -1.2659, 0.9435, -1.6809, 0.4232, -1.2684, -1.8151,-
1.4829,1.0302,0.9419,-1.1472, 1.4858, -0.6794, 0.9573, 1.0807, 1.1445, -1.2312, -0.6643, 0.3836,-
1.1482, -0.0353 -0.6747, -1.1653, -0.8896, 0.2414, 0.1160, -0.6987, 0.4781, 0.1821, -1.0672 ,-0.9676,-
1.2321, 0.5003, 0.7419, -0.8934, 0.8391];
```

El valor S_{cover} también lo encontramos en el estándar ECMA 368 [9] que depende del tipo de preámbulo que se utiliza y del TF (*Time-Frequency*). Como en nuestro caso utilizaremos un preámbulo tipo estándar, el valor S_{cover} es el siguiente:

```
Scover=[1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,-1,-1,-1];
```

Con todos estos parámetros ya podemos aplicar la función *PLCP_Preamble*, en la que se realiza la parte del estándar preámbulo y con la que obtenemos la secuencia previa a la cabecera.

```
[Ssync,Preamble_PLCP]=PLCP_Preamble(Npf,Nsym,Sbase,Scover);
```

Para el diseño del preámbulo primero se realiza el paquete relacionado con la sincronización y a continuación, la estimación del canal.

```
%Packet/Frame Synchronization Sequence
L1=length(Sbase);
Sext(1,1:L1)=[Sbase];
Sext(L1+1:Nsym)=[0];

for n=1:Npf
   Ssync(((n-1)*Nsym+1):(n*Nsym))=Sext.*Scover(n);
end

%Channel Estimation Sequence
channel_est_r_fren=[0,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,-1,1,1,1,-1,-1,1,1,1,1,...
-1,-1,1,-1,1,1,1,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,1,-1,1,1,-1,-1,-1,-1,0,0,0,0,0,-1,...
-1,-1,-1,-1,1,1,-1,1,1,1,1,1,-1,1,-1,-1,1,1,-1,-1,1,1,1,-1,1,-1,-1,1,1,1,1,-1,-1,1,1,1,1,...
-1,1,1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,-1,-1,1,-1,1,1];

channel_est_i_fren = [0,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,-1,1,1,1,-1,-1,1,1,1,1,...
-1,-1,1,-1,1,1,1,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,1,1,-1,1,1,-1,-1,-1,-1,0,0,0,0,0,1,...
1,1,1,1,-1,-1,1,-1,-1,-1,-1,-1,1,1,-1,1,1,1,-1,-1,1,1,-1,-1,-1,1,-1,1,1,1,-1,-1,-1,-1,1,1,1,...
-1,-1,-1,1,-1,-1,-1,1,-1,1,-1,1,1,1,-1,-1,1,-1,1,-1,1,1,1,-1,1,-1,-1];

channel_est_fren=channel_est_r_fren+1j*channel_est_i_fren;
channel_est=(ifft(channel_est_fren))/sqrt(2);

L=length(channel_est);
Schannel_est_165(1:L)=[channel_est];
Schannel_est_165(L+1:Nsym)=[0];

for ni=Npf+1:30
```

```
   Ssync(((n-1)*Nsym+1):(n*Nsym))=Schannel_est_165;
end
end
```

4.2 PLCP Header

Después del preámbulo, encontramos la cabecera. Para ello se ha creado la función *PLCP_Header*, en la que primero buscaremos el valor de las variables: *RATE, LENGTH, SCRAMBLER, BRUST MODE, PREAMBLE TYPE, TX_TFC* y *BAND GROUPS LSB* con las que podremos crear la secuencia de 40 bits correspondiente al PHY tal y como se muestra en la **[Figura 3. 9].**

```
[S_RS_PLCP_Header,Seed_Value]=PLCP_Header
```

```
PHY(4:8)=[Rate];
PHY(9:20)=[length_bit];
PHY(23)=[S1];
PHY(24)=[S2];
PHY(27)=[BM];
PHY(28)=[PT];
PHY(29:31)=[T1_T4(1:3)];
PHY(32)=[BG_LSB];
PHY(35)=[T1_T4(4)];
```

El siguiente paso es crear la secuencia relacionada con el revuelto de la cabecera MAC juntamente con la de verificación de cabecera. Para ello primero definiremos la MAC, que consistirá en 80 bits aleatorios los que se juntarán con la secuencia creada referente al PHY. Una vez tenemos esta secuencia, aplicaremos la verificación de cabecera mediante la función CRC_16.

```
MAC=randi([0:1],1,80);
PHY_MAC=[PHY,MAC];
LPHY_MAC=length(PHY_MAC);

[HCS]=CRC_16(PHY_MAC,LPHY_MAC);
```

Ahora ya podemos realizar el mezclado de la secuencia de verificación junto la cabecera. Para ello utilizaremos la función *Scrambed_MAC_Header_HCS* con la que obtendremos los 96 bits siguiendo el diagrama de la **[Figura 3. 7]** y aplicar el código Reed-Solomon.

Finalmente, solo nos queda realizar los bits de cola que necesitaremos para crear la secuencia final y que irán situados al final del PHY, de la secuencia MAC+HCS y del código Reed Solomon. Los bits de cola se representan mediante ceros vectores de longitud 4 o 6 ceros.

```
Tail_bits_1=zeros(1,6);
Tail_bits_2=zeros(1,6);
Tail_bits_3=zeros(1,4);
```

Una vez realizados todos los pasos previos, podemos formar la secuencia de cabecera. Para ello ordenamos las secuencias que hemos obtenido: primero la PHY, introducimos los 6 bits de cola, luego la mezcla de la MAC con la secuencia de verificación seguido de unos bits de cola para finalizar con el código Reed-Solomon y los últimos bits de cola, obteniendo así la secuencia lista para convolucionar, mapear y modular.

```
PLCP_Header=[PHY,Tail_bits_1,MAC_Header_HCS,Tail_bits_2,RS_Bits,Tail_bits_3]
```

Para la codificación de la cabecera, se van a crear distintas funciones. Primero aplicaremos el codificador, el cual está definido en el estándar. Para este caso la tasa de codificación a utilizar es ¾.

```
[CONV_Header]=CONV_ENC_HEADER(S_RS_PLCP_Header);
```

Una vez tenemos la secuencia codificada, podemos aplicar la función *Bit_Inter_Header,* la cual nos entrelaza la secuencia para obtener más robustez frente a posibles errores. En este caso, aunque nuestro objetivo es transmitir una señal con una velocidad de datos de 480Mb/s, la cabecera siempre se entrelaza con una velocidad de 39.4Mb/s.

```
[BITINT_Header]=BIT_INTER_HEADER(CONV_Header,Ncbp6s,Ncbps,Ntds,Ntint,Ncyc);
```

Una vez ya tenemos la secuencia codificada y entrelazada, podemos aplicar el mapeo con el que obtendremos una secuencia de números complejos. La parte de la cabecera del PPDU se mapea utilizando la constelación tipo QPSK .En la **[Figura 4. 1]** se obtiene la constelación resultante de aplicar el mapeo tipo QPSK en la secuencia de la cabecera.

```
[MAP_HEADER]=MAPPED_HEADER(BITINT_Header);
```

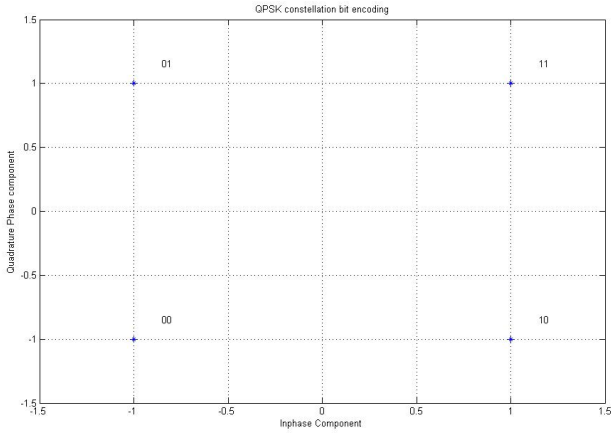

Figura 4. 1 Constelación QPSK

Finalmente se aplica el modulador OFDM. Para ello primero se mapea la señal, a continuación se aplican las portadoras de guarda siguiendo con las de datos y pilotos, para crear la secuencia con todas las portadoras y aplicar la IFFT. Para la realización de la modulación OFDM se han creado varias funciones.

```
[sym]=MAPPING_HEADER(MAP_HEADER);
[Guard]=GUARD_HEADER(sym);
[Pilot]=PILOT_HEADER(sym);
[Carrier]=CARRIER_HEADER(sym,Guard,Pilot);
IFFT_CARRIER_HEADER=reshape(Carrier,128,12);
IFFT_HEADER=ifft(IFFT_CARRIER_HEADER);
```

4.3 PSDU

Una vez tenemos la secuencia que define al preámbulo y la de cabecera, procedemos a diseñar la última parte del PPDU, el PSDU.

En primer lugar se ha creado una función llamada *PSDU* en la que se realiza el mezclador del frame, la comprobación de redundancia cíclica (*FSC*), se junta con 6 bits ceros y finalmente se le aplica el Pad Bits. Para ello primero se busca el valor N_{pad} y luego se realiza el mezclado siguiendo el formato de la **[Figura 3. 12].**

```
[Scramer_PSDU]=PSDU(Seed_Value);
```

Una vez tenemos la secuencia mezclada, podemos codificarla mediante la función *CONV_ENC_PSDU* y entrelazarla con la función *BIT_INTER_PSDU.*

```
[CONV_PSDU]=CONV_ENC_PSDU(Scramer_PSDU);
[BITINT_PSDU]=BIT_INTER_PSDU(CONV_PSDU,Ncbp6s,Ncbps,Ntds,Ntint,Ncyc);
```

Cuando tenemos la secuencia entrelazada tenemos que aplicar el mapeo de la constelación. En este caso, como estamos trabajando con una señal con una velocidad de datos de 480Mb/s, el mapeo sea tipo DCM, por eso se ha creado la función *MAP_PSDU.*

```
[DCM,DCM50,TX_DCM_DCM50]=MAP_PSDU(BITINT_PSDU);
```

En dicha función primero cogeremos la secuencia de bits, procedente del entrelazado y la dividiremos en bloques de 200 bits, tal y como se muestra a continuación:

```
for i=1:(length(PRBS)/200)
   for k=1:25
      out((i-1)*50+k,1)=PRBS((i-1)*200+2*(k-1)+1);
      out((i-1)*50+k,2)=PRBS((i-1)*200+2*(k-1)+2);
      out((i-1)*50+k,3)=PRBS((i-1)*200+2*(k-1)+51);
      out((i-1)*50+k,4)=PRBS((i-1)*200+2*(k-1)+52);
   end
   for k=26:50
      out((i-1)*50+k,1)=PRBS((i-1)*200+2*(k-1)+51);
      out((i-1)*50+k,2)=PRBS((i-1)*200+2*(k-1)+52);
      out((i-1)*50+k,3)=PRBS((i-1)*200+2*(k-1)+101);
```

```
    out((i-1)*50+k,4)=PRBS((i-1)*200+2*(k-1)+102);
  end
end
```

Una vez ya tenemos realizado la división en bloques de 200 bits procedemos a mapear dicha secuencia siguiendo la **[Tabla 3. 11],** obteniendo así la secuencia ($b[g(k)]$, $b[g(k)+1]$, $b[g(k) + 50)]$, $b[g(k) + 51]$) formando la secuencia de números complejos que darán lugar a las dos constelaciones mostradas en la **[Figura 4. 2]**.

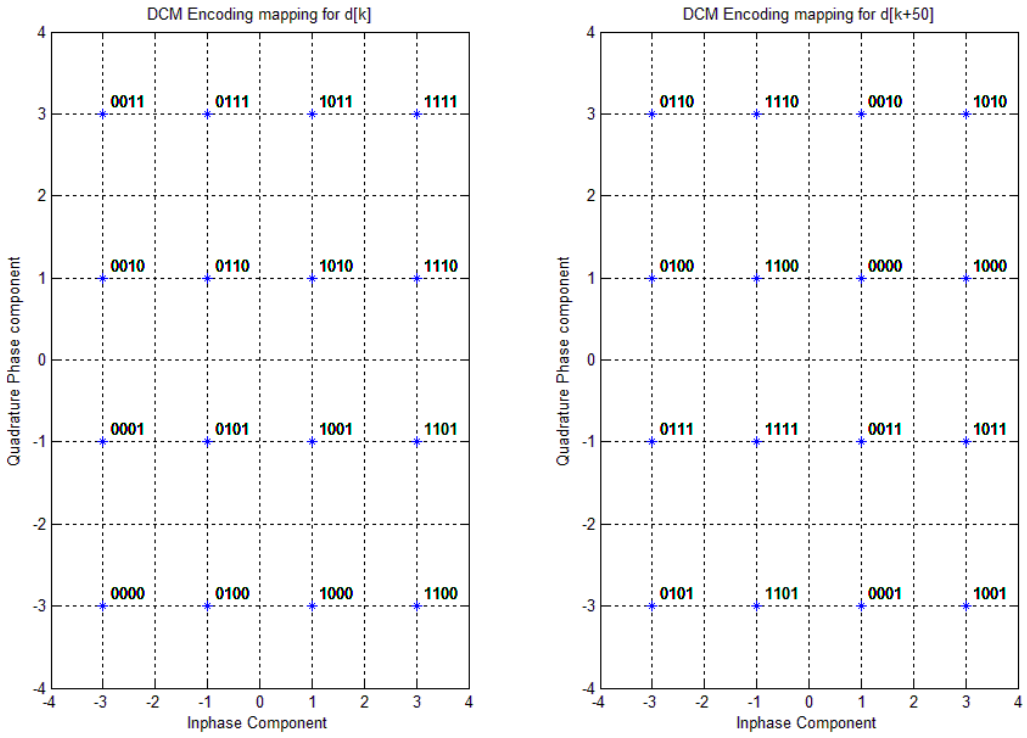

Figura 4. 2 Constelaciones DCM[k] y DCM[k+50]

Para finalizar el mapeo procederemos a multiplicar la secuencia obtenida, ($b[g(k)]$, $b[g(k)+1]$, $b[g(k) + 50)]$, $b[g(k) + 51]$), por el factor K_{mod} para poderla modular con el modulador OFDM.

El siguiente paso es aplicar el modulador OFDM. Para ello, se han ido creando paso a paso las portadoras de datos, pilotos y guarda con las que realizaremos el mapeo de todas ellas finalizando con la aplicación de la IFFT.

Empezaremos creando las portadoras relacionadas con los datos, para ello se ha creado la función *TX_Data_PSDU* en la que para cada portadora de datos se le ha introducido el valor de la señal obteniendo **[Figura 4. 3]**, obteniendo así las 100 portadoras de datos.

```
[TData,Data]=TX_Data_PSDU(TX_DCM_DCM50,Nd);
```

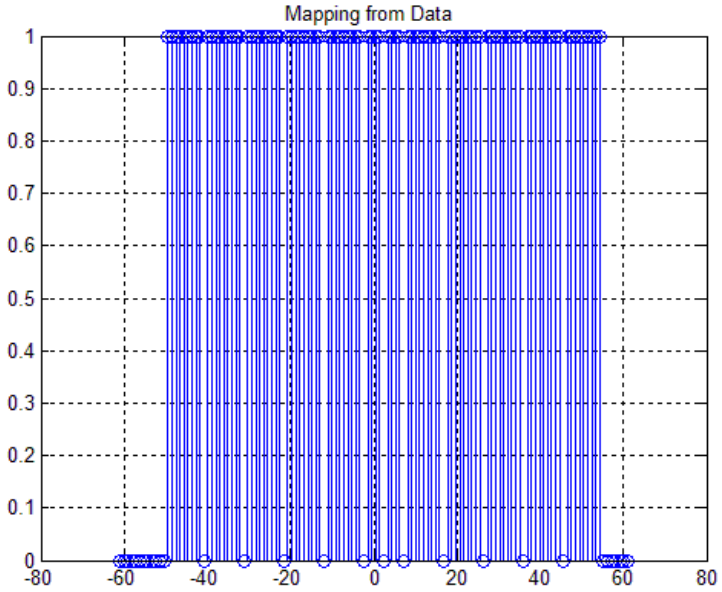

Figura 4. 3 Portadoras de datos

A continuación se aplicarán las portadoras de guarda, tal y como se describe en el punto anterior (3.9.7) mediante la función, *TX_Guard*, en las portadoras de guarda no existen distinciones entre la cabecera o el PSDU, por eso se utilizará la misma función utilizada para la cabecera.

```
[TGuard,Guard]=TX_Guard(TX_DCM_DCM50,Ng);
```

Las portadoras de guarda corresponden con las 10 subportadoras relacionas con las frecuencias lógicas -61, -60,…, -57, y 57, 58,…, 61.

```
for n=1:(length(TX_DCM_DCM50))/100
    for i=1:((Ng/2)+1)
      TGuard((n-1)*128+i)=0;
   end

   for i=(Ng/2)+1:(Ng)
      TGuard((n-1)*128+i)=TX_DCM_DCM50(((n-1)*10+i)-5);
   end

   for i=124:128
      TGuard((n-1)*128+i)=TX_DCM_DCM50(((n-1)*10+i)-118);
   end
end
```

Para terminar con las subportadoras, introduciremos las portadoras pilotos, definidas en el (3.9.8) en que se explica cómo distribuir dichas portadoras para una velocidad de datos de 480Mb/s en las frecuencias lógicas -55,-45, -35, -25, -15, -5, 5, 15, 25, 35, 45 y 55, obteniendo la **[Figura 4. 4]:**

```
[TPilot,Pilot]=TX_Pilot_PSDU(TX_DCM_DCM50,Np);
```

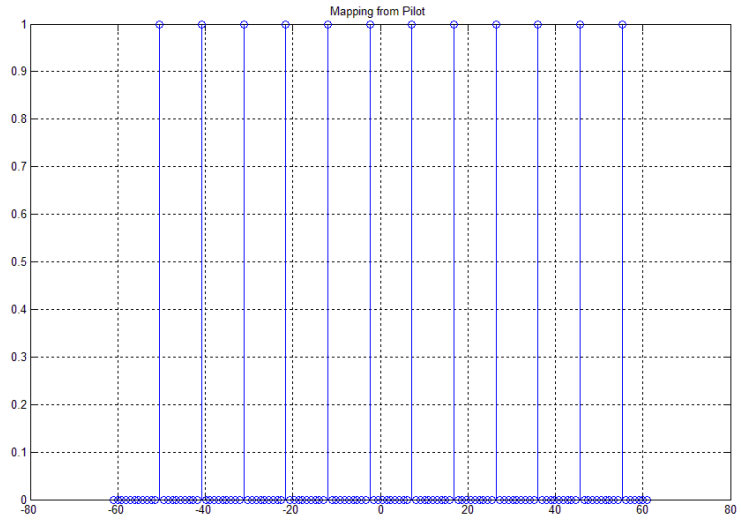

Figura 4. 4 Portadoras Piloto

Una vez tenemos las subportadoras de datos, guarda y piloto se procede a crear la secuencia total, mediante la función *CARRIER_PSDU.* En ella reorganizaremos la secuencia de tal manera que las subportadoras de la 62 a la 66 serán ceros, de la 1 a la 61 se asignen a las entradas 1-61 de la IFFT y las subportadoras -61 hasta -1 se asignen a las entradas 67 hasta 127 de la IFFT siguiendo la **[Figura 3. 19].**

```
[Carrier]=CARRIER_PSDU(TX_DCM_DCM50,TData,TGuard,TPilot);
```

```
b=(length((TX_DCM_DCM50))/100)*128;

Data=[TData,zeros(1,b-(length(TData)))];
Guard=[TGuard,zeros(1,b-(length(TGuard)))];
Pilot=[TPilot,zeros(1,b-(length(TPilot)))];

S=Data+Pilot+Guard;

for i=1:length(TX_DCM_DCM50)/100
   for k=1:62
      Carrier((i-1)*128+k)=S((i-1)*128+k+66);
   end
   for k=63:128
```

```
    Carrier((i-1)*128+k)=S((i-1)*128+k-62);
  end
end
```

Una vez obtenida la secuencia con las subportadoras, es momento de reordenarla y aplicar la IFFT para finalmente introducirle el pad-zero:

```
reshp_carrier=reshape(Carrier,128,length(Carrier)/128);
TX_ifft_carrier=ifft(reshp_carrier,128);
pads=zeros((165-128),length(Carrier)/128);
PPDU=[TX_ifft_carrier;pads] ;
```

Ahora ya tenemos creadas las tres partes del PSDU, finalmente nos queda unirlas, para ello cogeremos las tres secuencias previamente diseñadas, PLCP preámbulo, PLCP cabecera y PPDU y se creará la función s[n] tal y como se muestra en el código.

```
for n=1:6
  for i=1:Nsync
    s((n-1)*300+i)=Preamble_PLCP(((n-1)*6+i));
  end

  for i=(Nsync+1):(Nsync+Nhdr)
    s((n-1)*300+i)=HEADER(((n-1)*6+i-Nsync));
  end

  for i=(Nsync+Nhdr+1):(Npacket)-1
    s((n-1)*300+i)=PPDU(((n-1)*6+i-Nsync-Nhdr));
  end
end
```

5 Simulaciones

Para la realización de las simulaciones es imprescindible un software capaz de simular la señal creada previamente con Matlab utilizando el programa de simulación "Virtual Photonics Inc." (VPI).

Figura 5. 1 Logo VPI Photonics

VPI nos permitirá representar la señal en el dominio óptico y eléctrico para poder verificar y evaluar sistemas que incorpora el programa, mediante demostraciones, o crear de nuevos. Para ello, es imprescindible entender el funcionamiento interno del programa.

5.1 Estructura interna de VPI:

El programa VPI está organizado de una forma muy jerárquica basada en estrellas, galaxias y universo. Permitiendo así, crear subsistemas dentro de un sistema global.

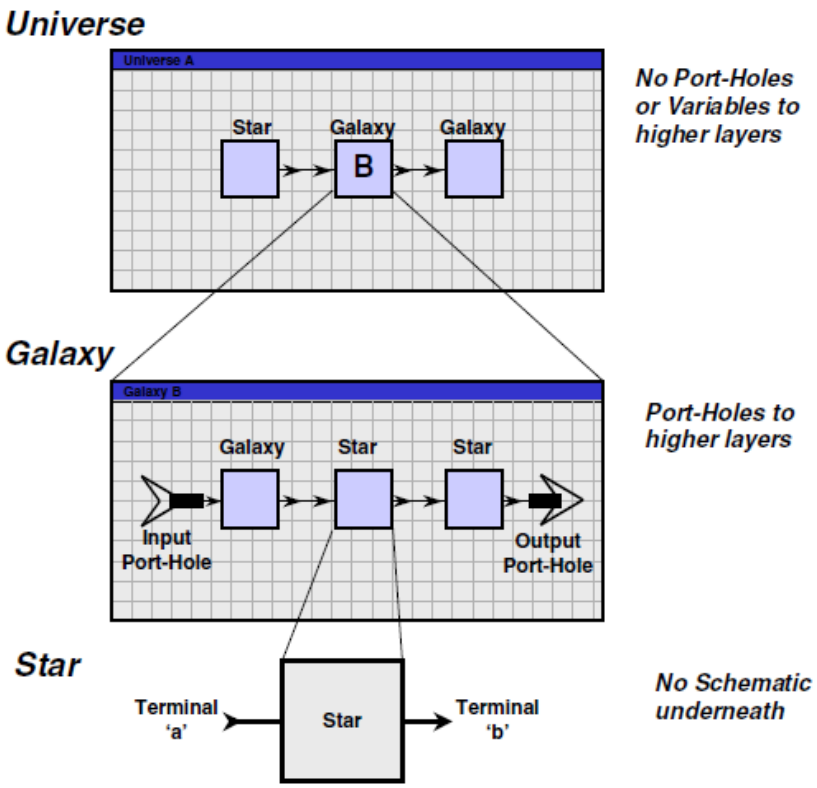

Figura 5. 2 Estructura Interna VPI [11]

El universo, tal y como su nombre indica no hay nada fuera de él, es decir, engloba las galaxias y estrellas, es el nivel más alto de la jerarquía. En él no hay conexiones externas y es a través el cual se va a realizar las simulaciones.

Las estrellas es el nivel más bajo, en ellas se representan los componentes básicos o módulos. Las estrellas no podrán funcionar si no se activa el universo en el que pertenecen.

Y en el punto intermedio tenemos las galaxias, que son un conjunto de estrellas conectadas entre ellas, con un puerto de entrada y uno de salida, pudiendo crear una galaxia dentro de otra galaxia. Las galaxias solo van a poder ser simuladas dentro de un universo.

5.2 Parámetros del VPI:

Todos los módulos de la simulación tienen cierto parámetros en común, denominadas parámetros de fondo, mediante los cuales el programa realizará cálculos a lo largo de la simulación, por eso es importante entenderlos para poder obtener un buen resultado.

- *Time Window* (ventana temporal): Es el intervalo temporal que usará el programa para realizar las simulaciones. El VPI analizará el sistema durante el tiempo que marque el parámetro *Time Window*, al mismo tiempo también fija la resolución espectral de la simulación.
- *Sample Rate Default*: Especifica la frecuencia de muestreo y se define como el número de muestras por segundo. También determina la resolución temporal.
- *Sample Mode Center Frequency*: Generalmente las señales ópticas tiene una frecuencia mucho mayor a las señales eléctricas. Para evitar un correcto muestreo, se necesita una frecuencia muy alta. Para evitar problemas, VPI trata las señales haciendo un equivalente paso bajo utilizando esta variable.
- *Sample Mode Bandwidth*: Define una resolución temporal y ancho de banda de la simulación.
- *Bit Rate Default*: Define la tasa de transmisión de bit en emisor.
- *In Band Noise Bins*: Este parámetro tiene dos estados (ON u OFF). El estado ON, define ruido dentro de la banda. Y el estado OFF, añade ruido aleatorio en la banda.
- *Boundary Conditions*: Permite especificar las condiciones de la simulación, si es periódica o aperiódica.

- *Logical Information*: Esta es una herramienta que utiliza VPI para enviar información entre diferentes módulos de una misma simulación.

Aparte de todas estas variables, existe la posibilidad de crear nuestras propias variables globales. Esto es bastante útil ya que podemos almacenar variables que se vayan a utilizar en varios bloques y así evitarnos tener que cambiarlas dentro de cada bloque.

5.3 Restricciones de los parámetros globales:

El software VPI utiliza dos vectores, uno temporal y el otro en el dominio de la frecuencia:

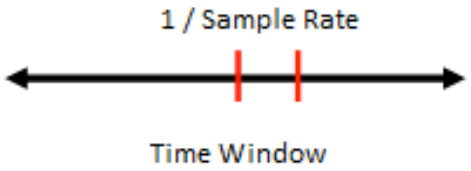

Figura 5. 3 Vector en el dominio del tiempo

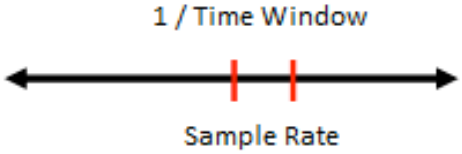

Figura 5. 4 Vector en el dominio de la frecuencia

Teniendo en cuenta esto dos vectores podemos explicar otras limitaciones de dicho software. Por un lado tenemos el *Time Window* el cual debe ser un número entero de bit, cumpliendo la siguiente ecuación:

$$TW = \frac{N}{BR}$$

Donde N es un número entero que representa el número total de bits en la simulación y BR es el *Bit Rate*. Por el otro lado, VPI trabaja con el algoritmo FFT cuando trabaja con señales periódicas cumpliendo la siguiente condición:

$$Samples = TW \times SR = 2^m$$

Por lo tanto, el número de muestras tiene que ser potencia de dos. Cumpliendo estas dos condiciones, el valor de la frecuencia de muestreo tiene que cumplir que:

$$SR = \frac{2^m}{N \times BR}$$

5.4 Módulos Principales

En este punto vamos a describir cuales son los principales módulos que se van a utilizar a lo largo de las simulaciones, sus funciones, el modo de operación, las variables que los definen…

5.4.1 Cosimulation Interface

Un bloque muy útil del VPI es el *CoSimInterface* con el que podremos utilizar una aplicación externa del VPI, como por ejemplo Matlab o Phyton, para el procesamiento de datos mediante programación. Este bloque ha sido muy útil en la realización del proyecto, ya que el software VPI no contiene el mapeo tipo DCM, y mediante este módulo se ha podido simular el estándar ECMA 368 [9].

Figura 5. 5 Módulo Cosimulation Interface [11]

Para el funcionamiento de este bloque, se deben configurar los parámetros siguiendo **[Figura 5. 6]**, en el *InterfaceType* elegiremos el software con el que previamente hemos realizado el programa, del que extraeremos la información siendo Matlab o Python. En la variable *RunCommand* utilizaremos la secuencia que se tiene que ejecutar, donde X es la salida e IN es la entrada, *Function* es la función, con el mismo nombre que la utilizada en Matlab y que se encontrará en la carpeta Input.

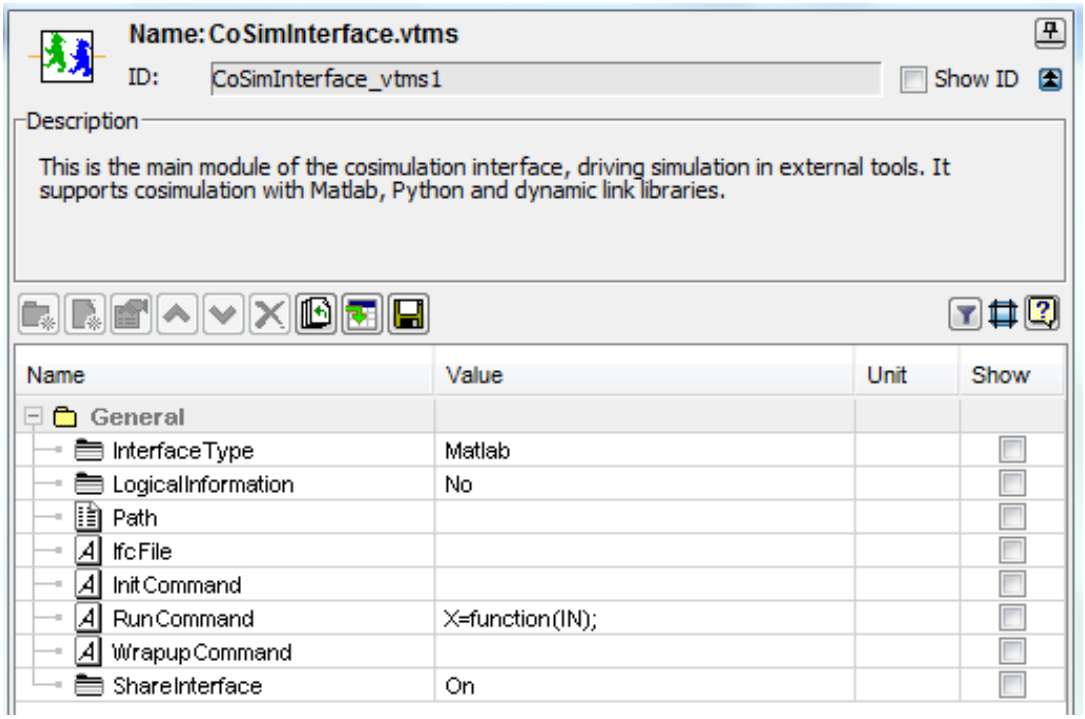

Figura 5. 6 Configuración Módulo Cosiminterface

VPI ofrece otros módulos, los *CosInput* que actúan como entradas y los *CosOutput* que actúan como salidas del módulo principal, *CosInterface*. Se utilizan para indicar el tipo de datos que van a ser insertados y extraídos a la módulo *CosInterface*, permitiendo el uso de señales eléctricas, ópticas, secuencias de números complejos…

En la **[Figura 5. 7]** se ha realizado un ejemplo en el que entra una matriz de números complejos al módulo *Cosinterface* y tras aplicar el programa se extrae otra matriz de números complejos.

Figura 5. 7 Interconexión módulo CosInterface con matrices de números complejos

El módulo *Cosinterface* será un dispositivo clave en las simulaciones de este proyecto ya que con él se podrá interactuar entre las funciones creadas previamente con Matlab y aplicarlas al software VPI.

5.4.2 PRBS

El módulo PRBS o pseudo generador de números aleatorios, genera distintos tipos de secuencias pseudo-aleatorias de datos, por ejemplo PRBS o secuencias de *De Bruijn* de orden N. El módulo PRBS produce una secuencia de bit de longitud M, siendo M la relación:

$$M = TimeWindow \times BitRateDefault$$

Figura 5. 8 Módulo PRBS [11]

5.4.3 Codificador OFDM

Este módulo genera la señal eléctrica correspondiente a la parte real e imaginaria de la señal OFDM. La entrada es una secuencia de bit, generalmente procedente del módulo PRBS. En este módulo podemos elegir qué tipo de OFDM queremos, si la clásica OFDM o la multi-tono DMT. Para la codificación podemos elegir entre M-QAM o M-PSK. También se le introduce un prefijo cíclico, y podemos elegir portadoras pilotos.

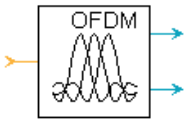

Figura 5. 9 Módulo Codificador OFDM [11]

5.4.4 Decodificador OFDM

Este módulo decodifica la señal OFDM o DMT, generada por el codificador OFDM. Realiza la demodulación, extrae el prefijo cíclico y las portadoras pilotos y realiza una ecualización. La salida se entrega en dos partes, en relación a la parte real y la imaginaria de la señal.

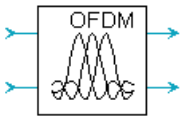

Figura 5. 10 Decodificador OFDM [11]

5.4.5 Pulse Raised Cos QAM

Este módulo genera la respuesta Nyquist de una señal eléctrica. Se utiliza antes de un modulador o cómo parte de un receptor.

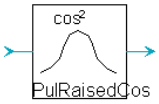

Figura 5. 11 Pulse Raised Cos QAM, [11]

5.4.6 El Amplificador Eléctrico

La entrada es la señal eléctrica y la salida es la misma señal amplificada mediante ruido blanco tipo Gaussiano.

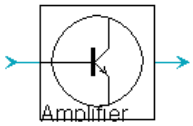

Figura 5. 12 El amplificador eléctrico, [11]

5.4.7 Logic Add Channel

Este módulo nos permite asignar canales lógicos de la señal. Permitiendo almacenar información de la señal para posteriores cálculos.

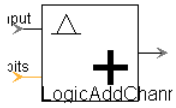

Figura 5. 13 Lodig Add Channel, [11]

5.4.8 Symbol Error Rate Estimation

Existen varios módulos con los que estudiar la señal, con este podemos ver el SER (*Symbol Error Rate*) y el EVM (*Error Vector Magnitude*) de una señal eléctrica. Para realizar dichos cálculos utiliza las componentes en fase y cuadratura de la señal de entrada.

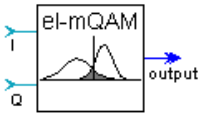

Figura 5. 14 Symbol Error Rate Estimation, [11]

5.4.9 Láser CW

El módulo *laserCW* genera una señal óptica mediante el láser el que produce una onda continua (CW, *Continuous Wave*), VPI dispone de distintos láseres más sofisticados como por ejemplo el *LaserCW_DSM*, pero a lo largo de nuestras simulaciones utilizaremos el láser de onda continua.

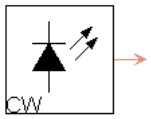

Figura 5. 15 Módulo Láser CW, [11]

5.4.10 Modulador Difencial MZ

El modulador Mach-Zehnder (MZ) se basa en el fenómeno electro-óptico producido por algunos cristales en él se realiza una transformación de fase en una modulación en amplitud.

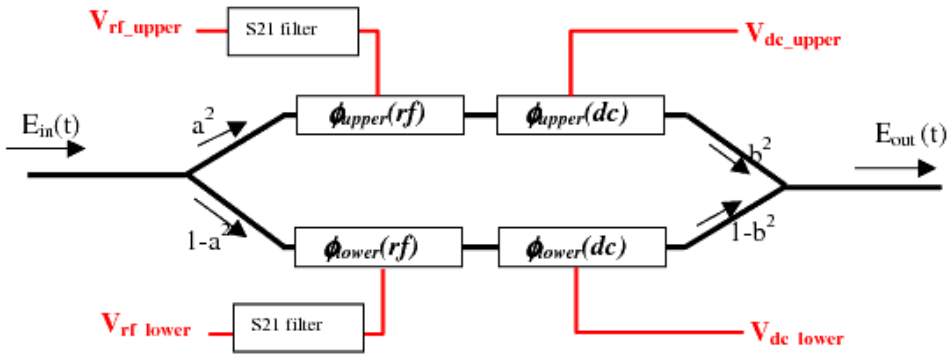

Figura 5. 16 Esquema modulador MZ, [11]

En la **[Figura 5. 16]** se representa el modulador Mach-Zehnder correspondiente al manual del VPI, Se necesita alimentar el modulador con una tensión de Bias ya que se envían señales con ciclos positivos y negativos, esta tensión de Bias hace que el modulador trabaje en zona lineal.

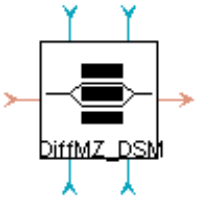

Figura 5. 17 Modulador MZ, [11]

5.4.11 Fotodiodo

El fotodiodo nos servirá para detectar la señal óptica y convertirla en señal eléctrica, podemos seleccionar si queremos el modelo PIN o el APD, podemos seleccionar el valor de corriente, el ruido entre otros.

Figura 5. 18 Fotodiodo, [11]

5.4.12 Signal Analyzer

Este módulo trabaja como una interface con VPI Photonics Analyser, utilizada para visualizar y analizar señales tanto eléctricas como ópticas.

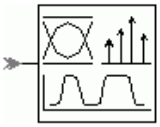

Figura 5. 19 Signal Analyzer, [11]

5.4.13 PackBlockEl/ UnpackBlockEl

Este par de módulos sirven para transformar los datos de entrada en bloques eléctricos, o la inversa. La tasa de muestreo viene determinada por el parámetro global TimeWindow.

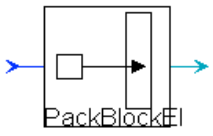

Figura 5. 20 Pack Block El, [11]

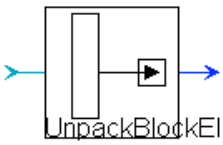

Figura 5. 21 UnPack Block El, [11]

5.5 Generation and Detection of OFDM Signals

En este punto vamos a coger un primer contacto con el software, explicando el funcionamiento de la simulación *Generation and Detection of OFDM Signals* que incorpora el programa.

5.5.1 Esquema General

Empezaremos con la demostración que el software incluye en ella podremos ver cómo se transmite y se recibe la señal mediante los coders/decoders OFDM siguiendo el esquema de la **[Figura 5. 22]**

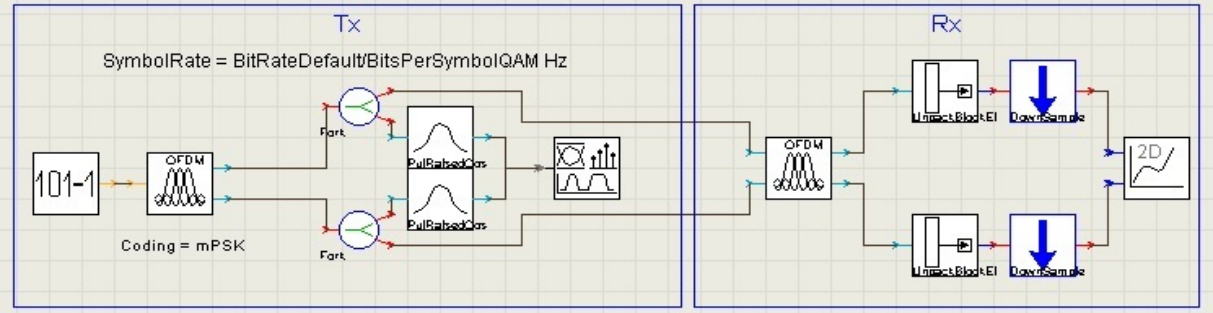

Figura 5. 22 Generación y Detección de señales OFDM

En esta simulación una señal banda base OFDM de genera mediante el codificador OFDM que está conectado al decodificador OFDM mediante cables, es decir, la señal es recibida directamente sin pasar por ningún canal entre transmisor y receptor, en otras palabras, es un canal de transmisión ideal.

Para iniciar la transmisión se necesita un generador de números aleatorios de secuencia de bits, PRBS, la salida de este módulo es una secuencia de números aleatorios enteros, formando un vector de tamaño *TimeWindow*BitRate,* que va conectada directamente al modulador OFDM.

Mediante los parámetros del codificador OFDM podemos configurar nuestro esquema, omitir el relleno con ceros, introducir o no prefijo cíclico, definir el factor M de la modulación QAM mediante el parámetro *BitsPerSymbolQAM*.

Una vez convertidos los bits en símbolos QAM *(Quadrature Amplitude Modulation),* el codificador realiza una operación IFFT para cada símbolo N que forman un símbolo de OFDM, obteniendo así una secuencia de tamaño *TimeWindow*BitRate/BitsPerSymbolQAM*. A continuación se produce un muestreo para obtener la longitud de la secuencia original *(TimeWindow*BitRate)* y que coincida con la frecuencia de muestreo requerida por VPI, obteniendo así la salida de las muestras eléctricas, que serán representadas mediante dos componentes, una para la parte real y otra para la imaginaria.

La salida está conectada directamente al receptor OFDM, y al mismo tiempo esta conectada a unos filtros *PulseRaisedCosQAM* con el objetivo de dar forma al pulso, mediante la respuesta Nyquist, para su visualización.

En la recepción, el módulo decodificador es el encargado de realizar las operaciones inversas que se han llevado a cabo en el módulo codificador.

Para la visualización de la señal, en este caso utilizaremos un analizador de la señal, capaz de representar la constelación recibida. Para ello, cómo las salidas del decodificador está en el dominio eléctrico, tiene que ser transformado mediante el módulo *UnPackBlockEl* así obtendremos las muestra numéricas.

Según las necesidades del VPI, el número de muestras de la secuencia original debe ser al menos igual al número de muestras de la señal eléctricas que se van a representar, por eso, como la secuencia procedente del codificador se organiza en símbolos, se supone que esta secuencia es internamente muestreada para poder ser procesada correctamente por el software, por lo tanto se introducen dos módulos que nos proporcionaran la cantidad exacta de los símbolos de la información que va a ser representada por el siguiente modulo. El factor de reducción va indicada en el editor de parámetros de estos bloques y se establece en:

Factor= {SampleRateDefault}/({BitRateDefault}/{BitsPerSymbolQAM})

5.5.2 Parámetros codificador y decodificador:

En la **[Figura 5. 23]** se muestra los parámetros del codificador OFDM que se pueden modificar, el decodificador utiliza los mismos parámetros, ya que realiza las mismas funciones que el codificador pero en orden inverso.

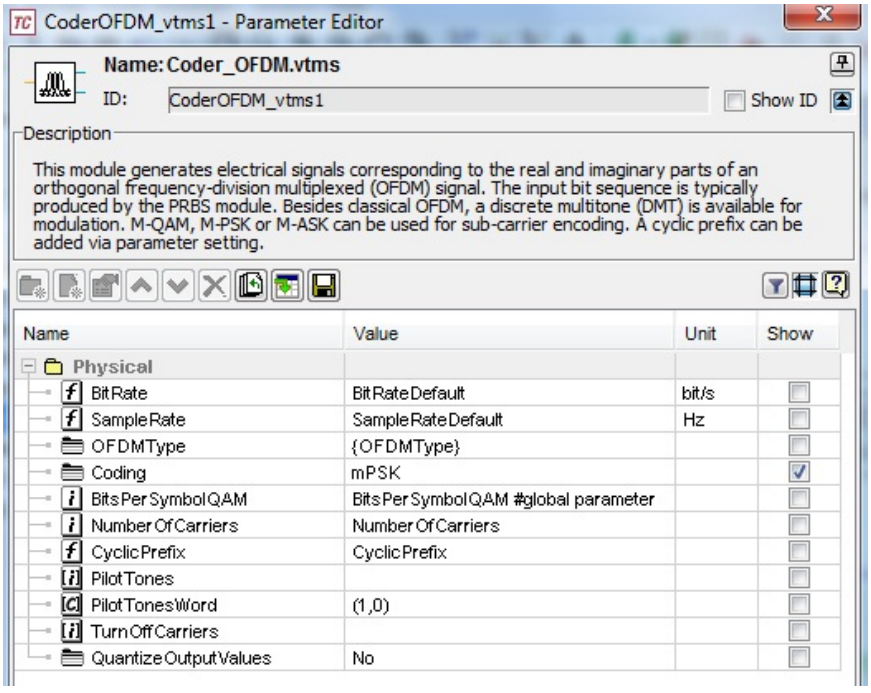

Figura 5. 23 Parámetros Codificador OFDM

Los parámetros *SampleRate, BitRate* y *BitPerSymbolQAM* son compartidos con el esquemático (universo), por lo que cualquier cambio en sus valores no tendrá ningún efecto en la simulación, a menos que se cambien en los parámetros del universo, el resto de parámetros son

- *OFDMType:* Nos permite elegir entre dos tipos de modulación: OFDM, sin rellenado con ceros, DMT (Multi Tono Discreto), donde la mitad de las entradas de la IFFT son el complejo conjurado de la otra mitad y relleno de ceros.
- *Coding:* En el que seleccionamos el tipo de codificación, mQAM o MPSK (PSK, *Phase Shift Keying*).
- *NumberOfCarriers:* Número de subportadoras de la modulación.
- *CyclicPrefix:* La longitud del prefijo cíclico en relación con la longitud del símbolo, con valores entre 0.0 a 1.0.
- *QuantizeOutputValues:* Si este parámetro se establece en *Sí*, los valores de salida del codificador se cuantifican. El número de niveles de cuantificación y el nivel de cuantificación más alto deben ser indicados después.

5.6 OFDM for Long-Haul Transmission Demo

Este esquemático es otra demostración que incorpora el software VPI, en ella se analizará en detalle la modulación OFDM en largas distancias. Consiste en una modulación en amplitud y una conversión ascendente, la cual pasa por una transmisión a través de un enlace de fibra óptica de 1.000 Km, una vez la señal a recorrido el sistema óptico se recibe mediante una conversión descendente para finalmente enviarla al receptor.

5.6.1 Esquema General

Como se observa en la **[Figura 5. 24],** el universo esta formador por una galaxia que es el transmisor OFDM, y otra que es el receptor OFDM, en ella se realiza la conversión de la señal y se le aplicará el codificador y decodificador de la señal.

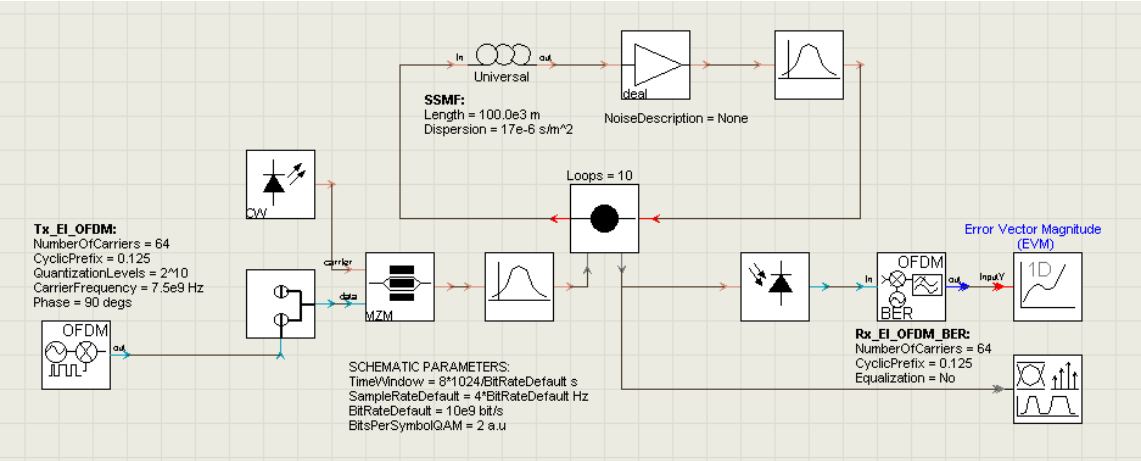

Figura 5. 24 OFDM Longh-Haul

Otro punto que se observa es el canal óptico y el enlace con la fibra. Al mismo tiempo, el esquemático muestra tres tipos distintos de resultados para la salida, la constelación recibida, el espectro de la entrada en el receptor y la magnitud del vector de error (EVM), el cual nos permite deducir el comportamiento del receptor, a la salida del receptor.

Los parámetros globales más importantes de este esquemáticos son el *TimeWindow,* la frecuencia de muestreo, la tasa de bits y la variable *BitsPerSymbolQAM* que en este caso está definido con valor 4 ya que nuestro sistema consta de una modulación en amplitud tipo 4-QAM.

Para una tasa de bits de 10 Gbps con una modulación 4-QAM tipo OFDM de un ancho de banda de 5 GHz se necesita una portadora de radiofrecuencia de 7.5 GHz, por lo tanto el parámetro *SampleRateDefault* deberá estar definido en $2^3 \times \frac{BitRateDefault}{BitPerSymbolQAM}$ con el objetivo de tener una ventana de simulación de 4 veces mayor al ancho de banda de señal óptica.

5.6.2 El Transmisor OFDM

En la **[Figura 5. 25]** se muestra los bloques que forman la galaxia del transmisor OFDM, en ella se pueden ver 3 bloques principales el codificador OFDM, la conformación de pulsos y la conversión al domino frecuencia. Existe un último bloque que nos permite guardar la información para cálculos.

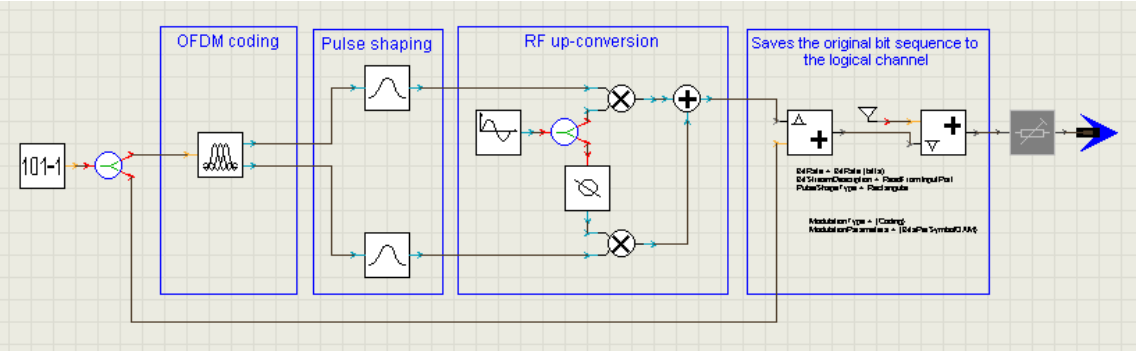

Figura 5. 25 El Transmisor OFDM

La transmisión de la señal empieza con el módulo PRBS, a la salida de este módulo obtenemos una secuencia aleatoria de números enteros de tamaño $BitRate \times TimeWindow$ que se introducirá al codificador OFDM.

El codificador no nos permite la opción de visualizar cómo esta formado, pero sí que podemos cambiar sus parámetros generales como por ejemplo, el prefijo cíclico que es la duración de prefijo de cada símbolo, y está definido por defecto a 0.125, es decir el 12.5%. Lo podemos modificar pero siempre teniendo en cuenta que se trata del porcentaje total del tamaño de la FFT (*Fast Fourier Transform*).

En el módulo del codificador, también se realiza una operación de muestreo, en la que cada símbolo que se transmite se repite un número determinado de veces. El factor de sobremuestreo se obtiene cómo la relación de la frecuencia de muestreo y la velocidad de símbolos, tal y como se muestra en la siguiente ecuación.

$$factor = SampleRateDefault \times \frac{BitRateDefault}{BitPerSymbolQAM}$$

A continuación encontramos dos módulos que consisten en unos filtros tipo coseno alzado con un factor de roll-off de 0.2 para realizar la conformación de los pulsos.

Seguimos el esquema con la conversión en RF, el cual consiste en una señal tipo sinusoidal a una frecuencia de 7.5 GHz y un desfasador.

Para terminar el esquema de transmisión tenemos el bloque *LogicAddChannel* el que se utiliza para enviar información entre módulos dentro de la misma simulación para posibles cálculos de resultados.

5.6.3 El Receptor OFDM

Una vez visto el transmisor OFDM vamos a ver como se recibe la señal, para ello nos fijamos en la galaxia correspondiente al receptor, *RX_OFDM* del esquema general, en ella podemos ver el esquema mostrado en la siguiente **[Figura 5. 26].**

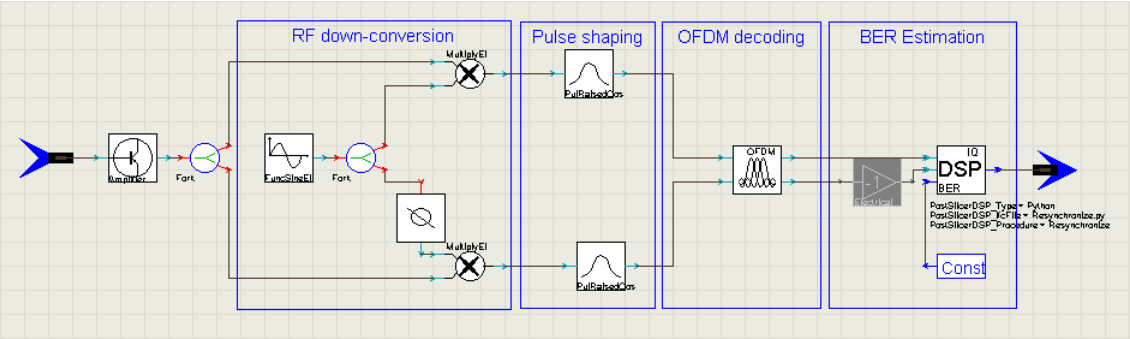

Figura 5. 26 El Receptor OFDM

Tal y como teníamos en el transmisor, el receptor está formado de 3 etapas: la primera la conversión RF, seguido de la conformación de los pulsos y finalmente el decodificador de la señal OFDM. Hay un bloque extra que nos permite realizar los cálculos de la BER (*Bit Error Rate*).

El primer módulo que encontramos es el conversor, con una frecuencia centrada en 7.5Ghz tal y cómo habíamos definido en el transmisor. A continuación tenemos los filtros tipo coseno alzado con el mismo roll off y finalmente encontramos la galaxia del decodificador OFDM, que realiza los mismos pasos que el transmisor OFDM pero en sentido inverso. Un detalle del decodificador es que en lugar de convertir los símbolos en una secuencia de bits, los convierte en dos salidas una real y otra imaginaria.

En el último bloque del receptor encontramos el módulo *BER_mQAM_DSP* el cual nos calcula la tasa de error de símbolo (SER, *Symbol Error Rate*) o la magnitud del vector de error (EVM) de una señal *mQAM, MPSK* o *DmPSK* teniendo en cuenta las entradas eléctricas I y Q procedentes del módulo de decodificador OFDM.

5.6.4 El Canal Óptico

El canal óptico esta formador por distintas etapas. Se empieza con el modulador óptico para posteriormente pasar la señal por la fibra y finalmente detectarla mediante un fotodiodo de detección directa.

En la **[Figura 5. 27]** se muestra la etapa de modulación óptica, donde el modulador óptico MZN tiene dos entradas, una procedente de la señal eléctrica creada por el transmisor OFDM y la otra óptica procedente de un láser de onda continua.

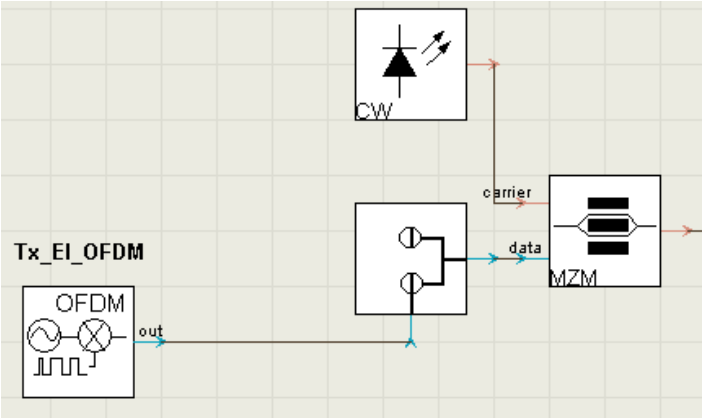

Figura 5. 27 El Modulador Óptico

Los parámetros más destacados del láser CW son la frecuencia de emisión, fijada a 193.12THz y la potencia media, establecida a 5mW. Un filtro óptico se introduce después del modulador MZN con el objetivo de eliminar la banda lateral inferior resultante de la modulación óptica.

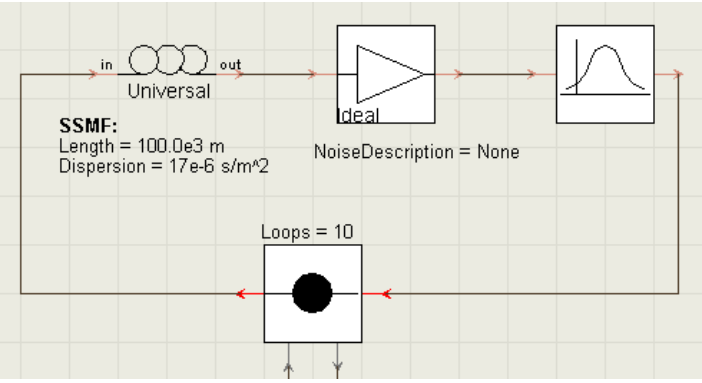

Figura 5. 28 Enlace Fibra Óptica

El enlace de fibra es la siguiente etapa del canal óptico, mostrado en la **[Figura 5. 28].** Para que el sistema se aproxime lo máximo a la realidad se ha creado un circuito donde se realizan 10 vueltas de 100 Km cada una. Se ha introducido un amplificador óptico con el objetivo de compensar las pérdidas y un filtrado óptico para eliminar el ruido resultante de la amplificación.

Para terminar con el sistema óptico, solo nos hace falta un receptor, para ello utilizaremos un fotodiodo, en este caso utilizaremos uno de tipo PIN.

5.7 Simulación FIVER

En este punto vamos a simular un escenario procedente del proyecto FIVER (*Full-Converged Quintuple-Play Integrated Optical-Wirless Access Architectures*). Aunque en el mismo se estudiaba el funcionamiento y alcance de una combinación de distintas señales, LTE (*Long Term Evolution*), WiMAX (*Worldwide Interoperability for Microwave Access*) y UWB, transmitidas por un mismo canal óptico, únicamente nos centraremos en el estudio de UWB para poderla comparar posteriormente con otras simulaciones.

5.7.1 Esquema General

En la **[Figura 5. 29]** se representa el esquema general de la simulación a llevar a cabo, consta de tres etapas bien diferenciadas, el primer bloque es el transmisor seguido de un par de amplificadores eléctricos, la señal eléctrica es modulada ópticamente mediante un MZ conectado a un láser. Una vez tenemos la señal en el dominio óptico pasa por la fibra para ser detectado por el diodo y convertirlo nuevamente en una señal eléctrica, para ser amplificado nuevamente y recibido por el receptor.

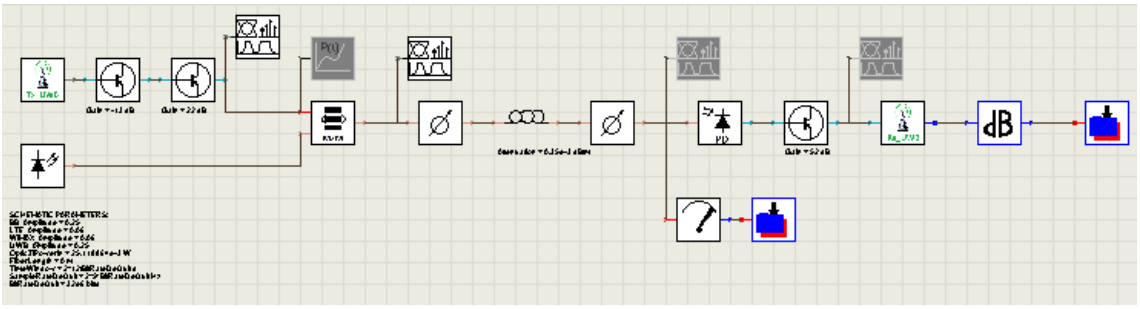

Figura 5. 29 Esquema general FIVER

5.7.2 El Transmisor OFDM-UWB

El esquema de la señal transmitida es el de la **[Figura 5. 30].** En el proyecto original este bloque consistía en la generación de las tres señales (LTE + WiMAX +UWB) las cuales se juntaban en un mismo canal, a distintas frecuencias y se amplificaban todas ellas para convertirlas al dominio óptico.

Para nuestro caso, en el que solo se transmite la señal UWB, empieza con el bloque PRBS con el que nos generará la secuencia de bits aleatorios que entrará en el codificador OFDM y al mismo tiempo al módulo *LogAddChannel* para sus posteriores cálculos. En este caso el codificador OFDM está configurado para una señal UWB con una velocidad de datos de 512Mbps y 128 portadoras, un prefijo cíclico de 0,25 y un ancho de banda de 528 MHz. Una vez configurado el codificador OFDM, se realiza la conversación en pulsos mediante el módulo *PulseRaisedCosQAM* y finalmente lo subimos a una frecuencia central de 3.96 GHz.

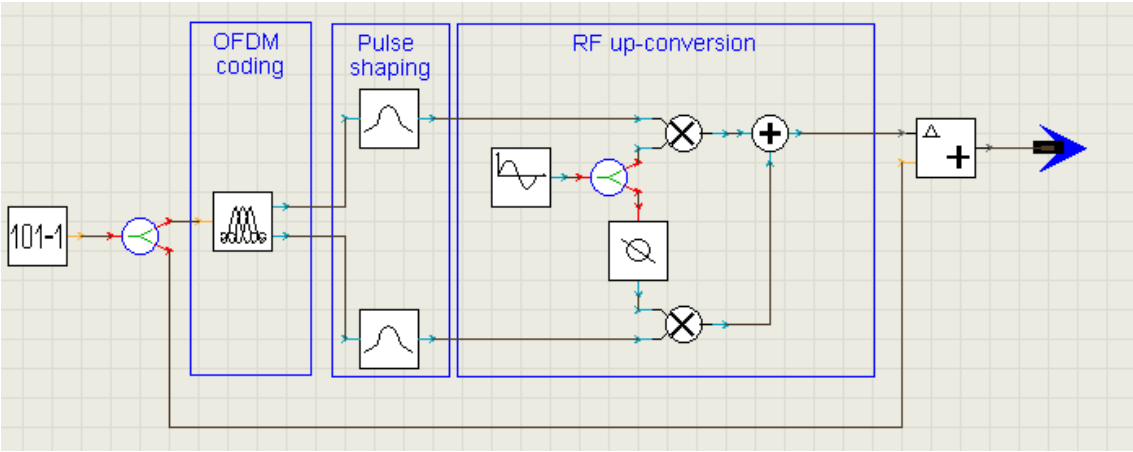

Figura 5. 30 El Transmisor OFDM-UWB

La generación de las otras señales estudiadas en dicho proyecto se realizaba del mismo modo, y finalmente se juntaban todas para ser amplificadas y transmitidas por el mismo canal.

La **[Figura 5. 31]** representa el espectro eléctrico de la señal UWB amplificada justo antes de entrar en el modulador OFDM. Tal y cómo se ha definido es un pulso centrado en 3.96Ghz, con un ancho de banda de 528Mhz. La potencia de dicha señal es -34dBm.

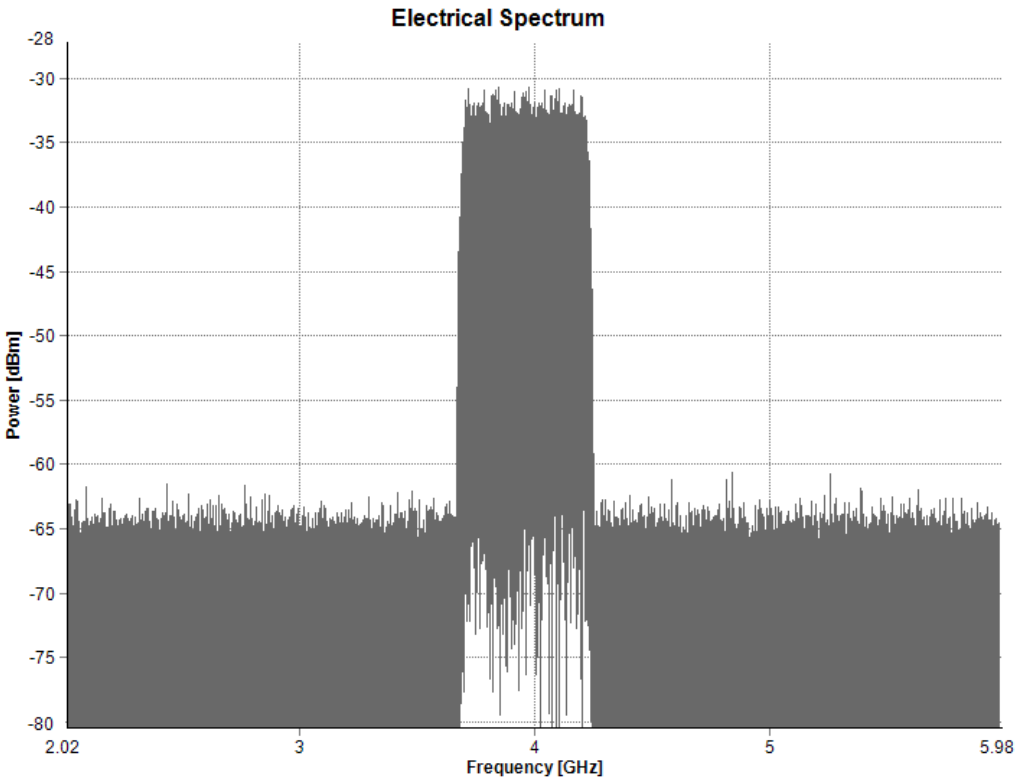

Figura 5. 31 Espectro Eléctrico Señal UWB FIVER

5.7.3 El Sistema Óptico

Una vez tenemos la señal en el dominio eléctrico lista para transmitir, se codifica mediante la modulación OFDM y se sube en frecuencia para posteriormente conectarla al modulador óptico MZ juntamente con la señal procedente de un láser. Así obtenemos la señal en el dominio óptico lista para pasarla por la fibra.

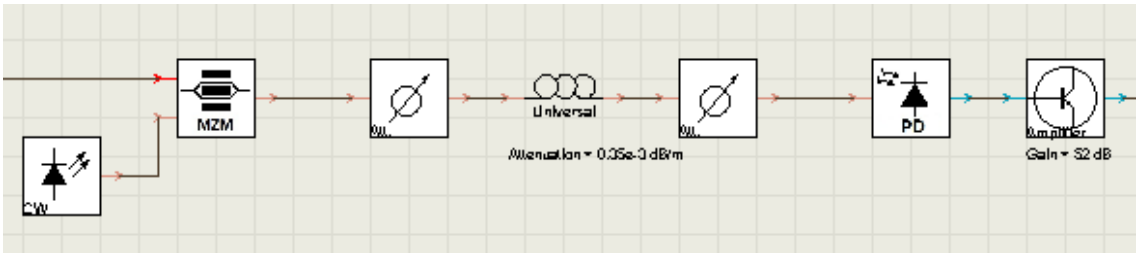

Figura 5. 32 Sistema Óptico FIVER

Se van a realizar distintas simulaciones cambiando la longitud de la fibra óptica. Para ello en los parámetros generales se ha creado la variable *Fiber_Length* para facilitar el cambio de dichos parámetros.

Una vez la señal pasa por la fibra se detecta mediante el fotodiodo tipo PIN y se pasa por el módulo de recepción, del mismo modo que el transmisor está formado por distintos módulos creando una galaxia.

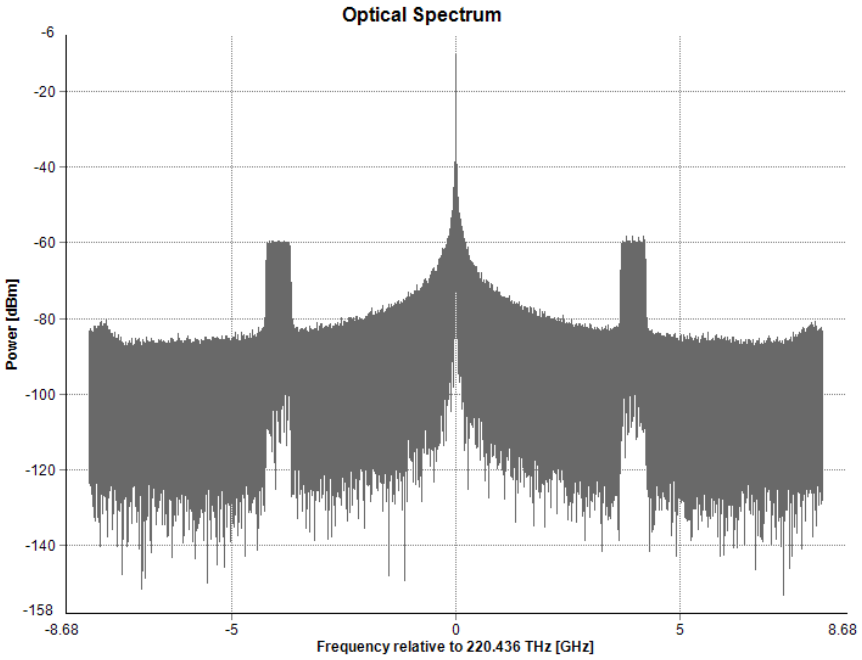

Figura 5. 33 Espectro en el dominio Óptico de la señal UWB FIVER

El espectro de la señal en el dominio óptico es el correspondiente a la **[Figura 5. 33].** En este caso se ha representado la señal sin fibra. Como se observa, el pulso está centrado en la frecuencia central definida previamente con valor 3.96GHz. La **[Figura 5. 34]** representa la misma señal UWB pero en este caso se le ha incorporado una distancia de fibra óptica de 15Km. Realizando una comparación de ambos casos se obtiene para el caso sin fibra una potencia óptica de -62dBm en cambio, para el caso de 15Km de fibra óptica la potencia es de -68dBm.

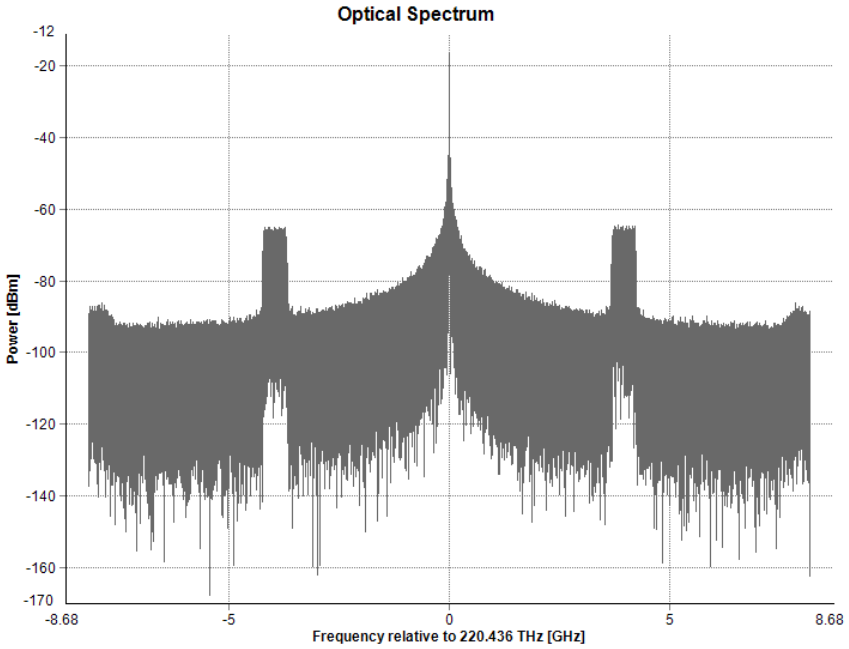

Figura 5. 34 Espectro en el dominio Óptico con longitud de fibra de 15Km de la señal UWB FIVER

5.7.4 El Receptor OFDM-UWB

El esquema de recepción de la señal es el representado en **[Figura 5. 35]**. En recepción tenemos el mismo esquema utilizado en transmisión pero a la inversa.

Para la recepción, primero amplificamos la señal, una vez amplificada bajamos en frecuencia, multiplicando con la misma frecuencia central utilizada en la transmisión y el mismo ancho de banda. Después realizamos la conformación de pulsos para pasarla por el decodificador OFDM con los mismos parámetros que en el transmisor. Finalmente enviamos la señal al módulo *BER_El_mQAM* con el que podremos realizar el estudio del EVM y al mismo tiempo la pasamos en el dominio frecuencial para dibujar su constelación.

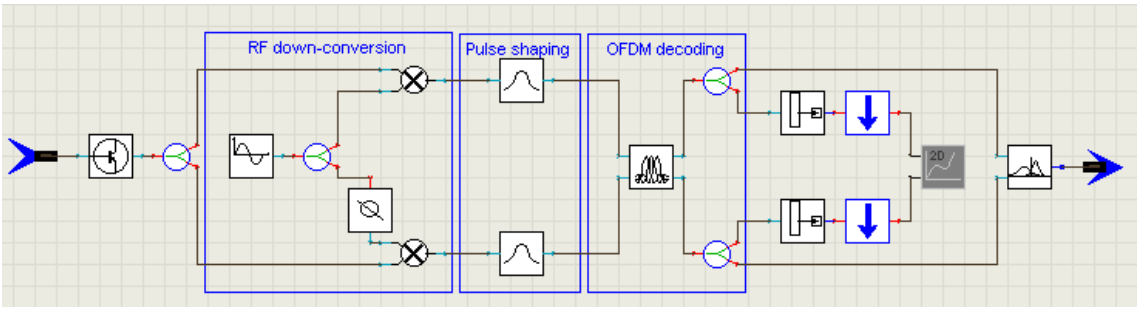

Figura 5. 35 Receptor OFDM-UWB

5.7.5 Resultados

Finalmente solo nos queda descubrir cómo varia la señal en función de la potencia óptica de entrada y de la longitud de la fibra óptica, para ello se han realizado varios barridos. En la **[Figura 5. 36]** se muestra cómo va cambiando el valor EVM en función de la potencia óptica de entrada proporcionada por el láser.

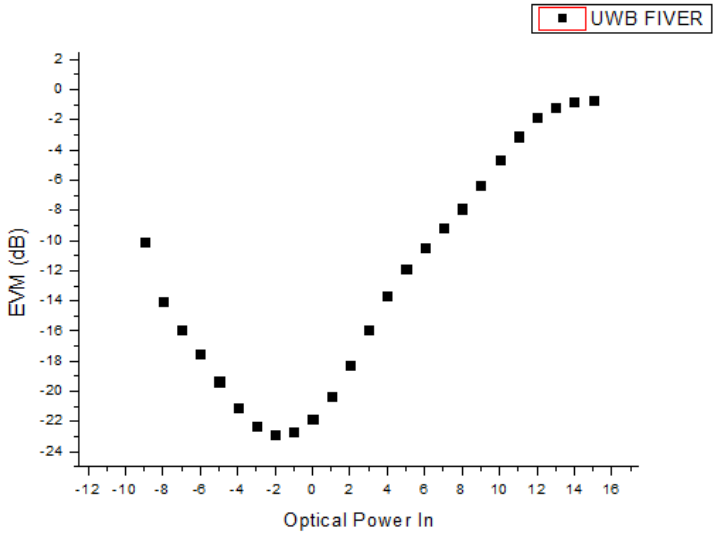

Figura 5. 36 EVM En Función De La Potencia Óptica De Entrada

Otro estudio que se llevó a cabo fue el estudio del comportamiento de la señal para una potencia óptica fija para distintas longitudes de fibra óptica. En la **[Figura 5. 37]** se muestra cómo va variando el valor EVM para distintas longitudes para una potencia fija de 14dBm. En este caso, el EVM está dentro de los límites del estándar [9], es decir se transmite y recibe correctamente la señal.

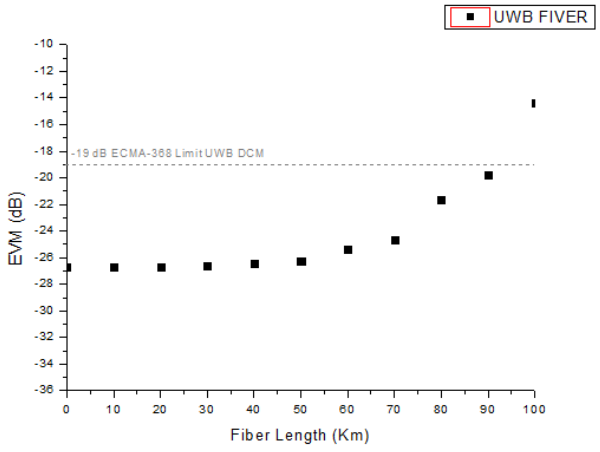

Figura 5. 37 EVM En Función De La Longitud De Fibra Para 14 dBm

Pero si cambiamos la potencia de entrada, en este caso a 10 dBm, obtenemos los resultados **[Figura 5. 38]** en este caso, la distancia máxima que podemos transmitir la señal, cumpliendo los limites se ha reducido a 80Km.

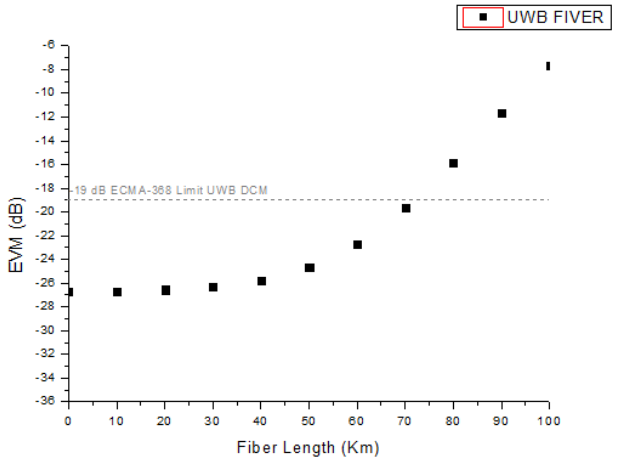

Figura 5. 38 EVM En Función De La Longitud De Fibra Para 10 dBm

5.8 Simulación OFDM DCM UWB

En este punto vamos a simular la señal UWB para una velocidad de datos de 480 Mbps siguiendo el estándar ECMA 368 [9]. En este escenario, la señal UWB se realiza mediante el mapeo tipo DCM. El software VPI no incorpora ningún módulo para poder realizar el DCM por eso haremos uso del módulo *CosInterface* con el que mediante Matlab se le introducirá el mapeo tipo DCM.

En esta simulación evaluaremos cómo varía el $RMS_{error,}$(RMS, *Root mean squared*), en función de la distancia y de la potencia óptica enviada. En ella, se ha variado la potencia óptica de entrada desde -14dB a 14dB y la longitud de fibra llegando hasta los 100Km. Para el cálculo del RMS_{error} se utiliza la función que nos proporciona el estándar [9].

$$RMS_{error} = \frac{1}{N_f}\sum_{i=1}^{N_f}\sqrt{\sum_{N=1+N_{sync}+N_{hdr}}^{N_{packet}}\left[\frac{\sum_{k=1}^{N_D}\left|R_{D,n}[k]-C_{D,n}[k]\right|^2+\sum_{k=1}^{N_P}\left|R_{p,n}[k]-C_{P,n}[k]\right|^2}{(N_D+N_P)N_{frame}P_o}\right]}$$

La función RMS_{error} depende de las portadoras de datos, y las pilotos, tanto las transmitidas como las recibidas. Los valores de RMS_{error} vienen fijados por la siguiente tabla donde se muestra el valor RMS_{error} para constelaciones sin atenuación o bien para con constelaciones de 2 a 12 dB. Obteniendo un valor de error RMS de entre -19.5dB hasta -17dB.

	Relative Constellation RMS Error		
Velocidad de datos	**Relative Constellation RMS Error**	**TX Attenuation of 2, 4, 6 dB (All TFCs)**	**TX Attenuation of 8, 10, 12 dB (All TFCs)**
320 Mb/s, 400 Mb/s, 480 Mb/s	−19,5 dB	-18,0 dB	-17,0 dB

Figura 5. 39 Error relativo a la constelación OFDM-DCM UWB

5.8.1 Escenario OFDM-DCM-UWB

El esquema del escenario a simular es el de **[Figura 5. 39].** Como en sistemas anteriores, el esquema se basa en tres grandes bloques, el transmisor donde se genera la señal, el sistema óptico con el modulador MZ, la fibra y el fotodetector y el receptor. Para el estudio de este escenario se han creado unas galaxias para representar las constelaciones, tanto las transmitidas como las recibirlas.

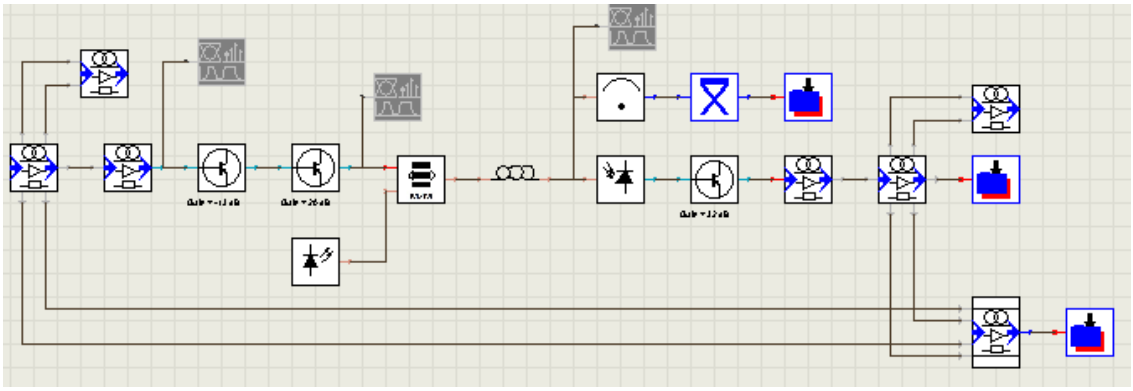

Figura 5. 40 Escenario OFDM-DCM-UWB

5.8.2 El Transmisor OFDM-DCM-UWB

Iremos explicando paso a paso cómo se ha creado el sistema, para ello primero miraremos el formato del transmisor representado en **[Figura 5. 41]**. Empieza con una secuencia de bits aleatorios generada por el módulo PRBS, esta secuencia será entregada al *CosInterface*, en el que se le ha introducido el código de Matlab.

En este punto se aplican todas las funciones relacionadas con el transmisor, es decir la creación de la señal UWB juntamente con el mapeo de la señal mediante DCM y la modulación OFDM, obteniendo cinco salidas. Las dos primeras salidas corresponden a las secuencias DCM[*k*] y DCM [*k+50*] con las que representaremos las constelaciones de entrada, la siguiente salida es la señal modulada mediante OFDM lista para la transmisión y finalmente tenemos dos salidas correspondientes a las portadoras pilotos y datos que nos servirán para calcular el RMS_{error} de la señal.

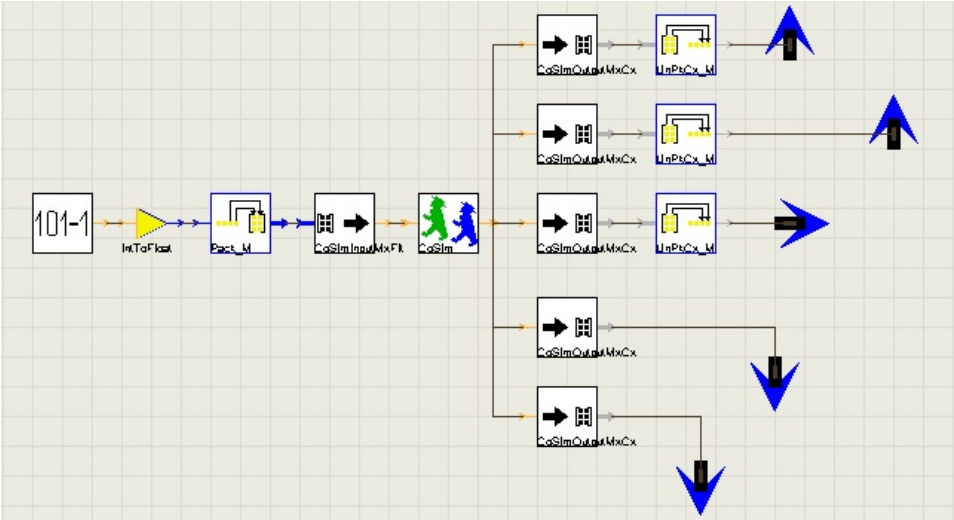

Figura 5. 41 Transmisor OFDM-DCM UWB

5.8.1 Conversión RF-UP

Una vez tenemos la señal lista para su transmisión, la tenemos que subir en frecuencia, para ello se utiliza un sistema parecido al utilizado en transmisor OFDM del punto (5.6.2). La señal de números complejos se convierte en señal eléctrica y se sube en frecuencia tal y como se muestra en la **[Figura 5. 42]**

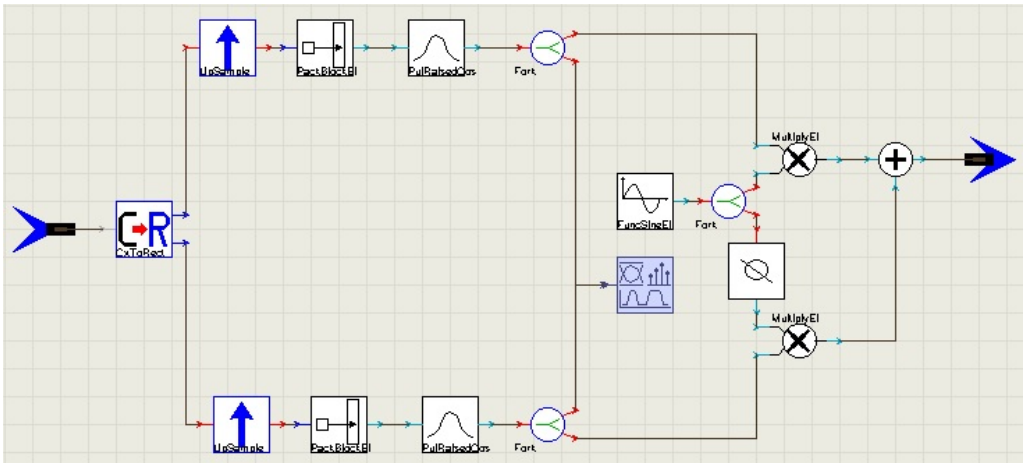

Figura 5. 42 RF Up DCM-OFDM UWB

Como se observa, primero encontramos un muestreo, seguido de la conversión eléctrica juntamente con la conformación de pulsos para finalmente subirlo en frecuencia mediante los módulos *FuncSineEl* y *PhaseShiftEl.* El resultado de esta conversión se muestra en la **[Figura 5. 43]**, donde se representa la señal en el dominio eléctrico. En este caso se ha representado la primera banda de la señal, como se observa es un pulso de ancho de banda de 528 Mhz centrado en 3.432MHz y con una potencia de -41dBm.

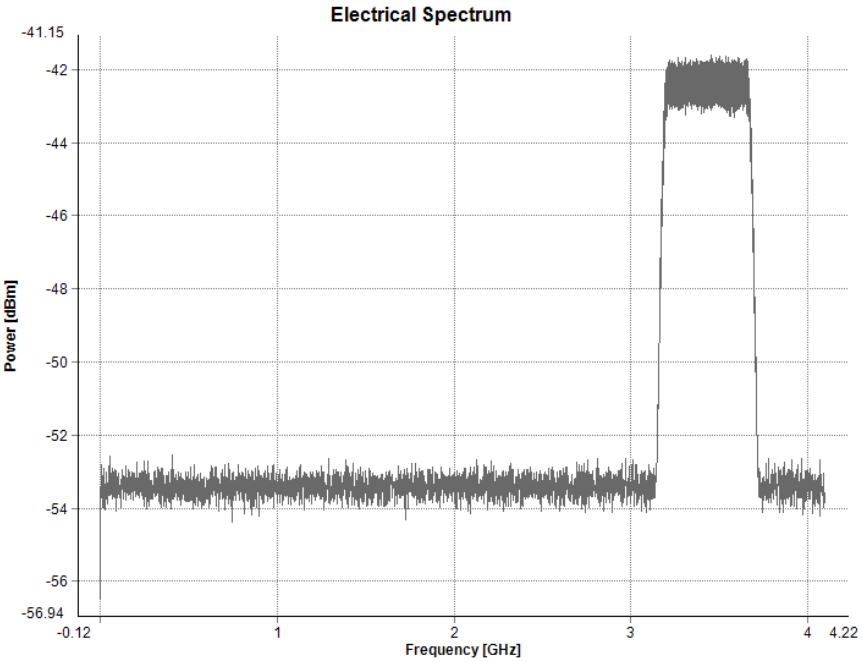

Figura 5. 43 Espectro eléctrico OFDM-DCM UWB

5.8.2 El Sistema Óptico

Una vez ya tenemos la señal eléctrica, el siguiente paso es pasarla en el dómino óptico. Para convertirla, utilizaremos un láser CW conectado al modulador Mach-Zehnder (MZ) juntamente con la señal eléctrica procedente del transmisor OFDM-DCM UWB. Una vez la señal ya está en el dominio óptico se pasa a través de la fibra y se detecta mediante un fotodiodo convirtiéndola nuevamente en una señal eléctrica.

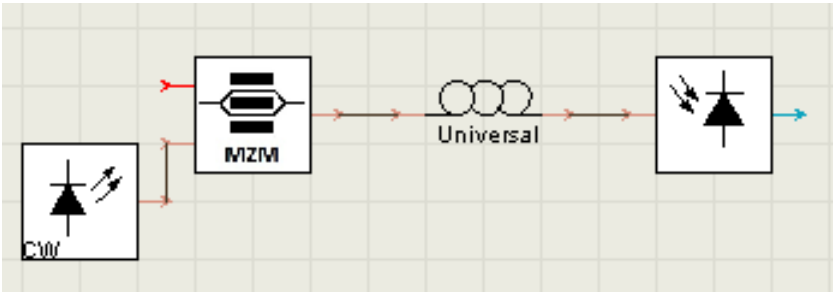

Figura 5. 44 Sistema Óptico OFDM-DCM UWB

En la **[Figura 5. 45**] se ha representado el espectro óptico de la señal después de la fibra, con una potencia óptica de entrada de 15dBm y una distancia de fibra óptica de 10Km.

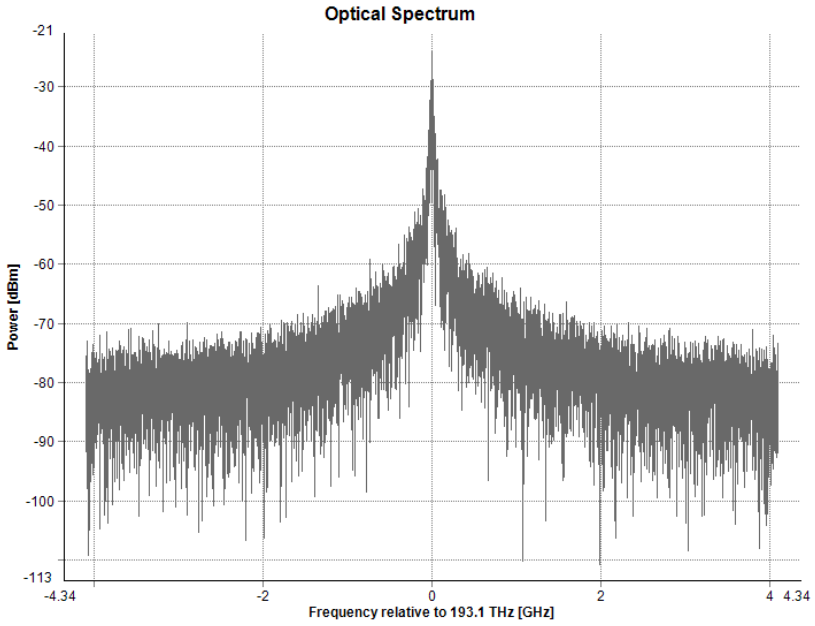

Figura 5. 45 Espectro Óptico señal UWB, Distancia 10Km y 15dBm de Potencia De Entrada

5.8.3 Conversión RF-Down

Tal y cómo se ha realizado en el transmisor, se debe realizar otra conversión, en este caso es del dominio óptico al dominio eléctrico. Para ello la señal se tiene que pasar por el sistema de conversión RF-Down. La conversión a señal eléctrica se realiza mediante el esquema de **[Figura 5. 46].**

Mediante los módulos *FuncSineEl* y el *PhaseShisftEl* se localiza la señal en el la frecuencia, para pasarla a través de los filtros para convertirla en números enteros y finalmente en números complejos.

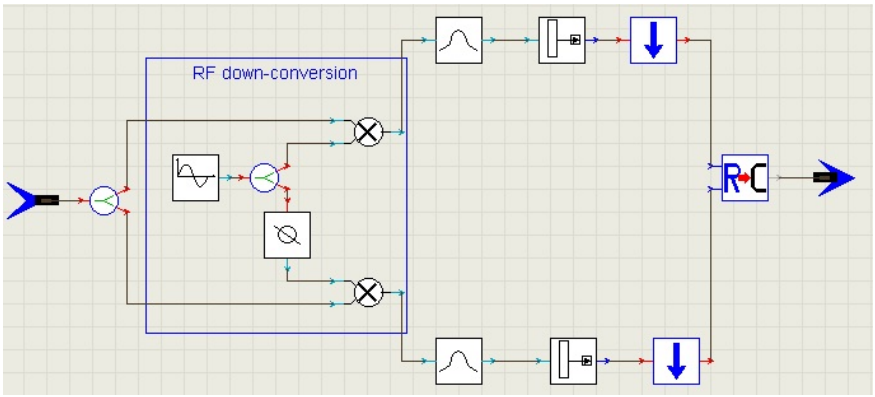

Figura 5. 46 RF-Down

5.8.4 El Receptor OFDM-DCM-UWB

El receptor OFDM-DCM tiene la misma forma que el transmisor. La señal se introduce al módulo *CosInterface* donde se le aplican las funciones relacionadas con la decodificación y se obtienen cinco salidas: las relacionadas con las secuencias DCM[*k*] y la DCM [*k+50*] para poder realizar las constelaciones, la señal recibida y las secuencias correspondientes a las portadoras pilotos y de datos con las que se realizará el cálculo del error RMS.

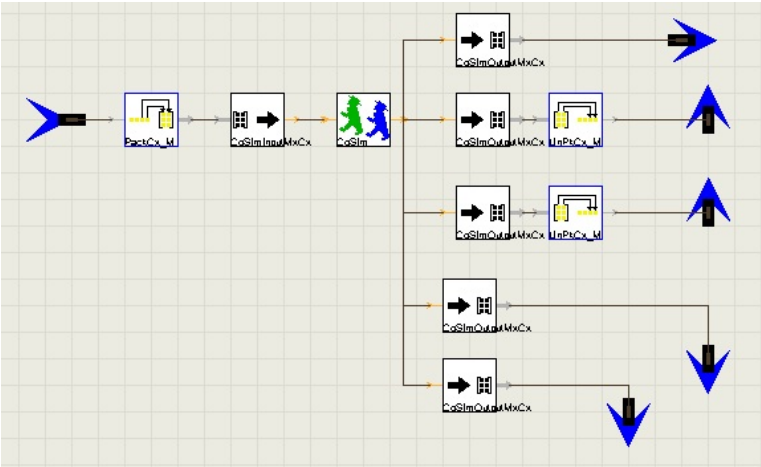

Figura 5. 47 Receptor OFDM-DCM UWB

5.8.5 Resultados OFDM DCM UWB

Para terminar con la simulación, vamos a estudiar cómo varia la señal en función de la potencia de entrada al sistema y de la longitud de fibra. Empezaremos realizando un barrido a la misma potencia óptica, como por ejemplo 14dB, y variando la longitud de fibra de 0km hasta 100Km, obteniendo los resultados de la **[Figura 5. 48]**.

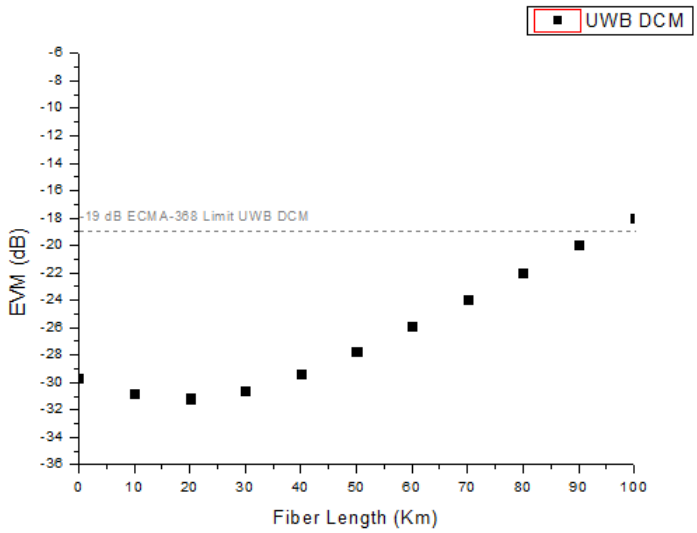

Figura 5. 48 EVM en función de la longitud de fibra, para una potencia de entrada de 14dBm

Tal y como muestra la figura, y siguiendo las especificaciones del estándar, el que nos marca el valor del error RMS máximo de 19dBm se observa que para este caso se puede logar una distancia de hasta 90Km.

En cambio, en la **[Figura 5. 49]** se representa el mismo caso, es decir, una variación de longitud de fibra óptica, pero en este caso hemos cambiado la potencia de entrada a 10dB. Al reducir la potencia de entrada se reduce la distancia de fibra permitida para la transmisión, es decir el alcance. Cómo se observa, en lugar de los 90Km que podíamos introducir de fibra óptica en el caso anterior, cuando la potencia de entrada es de 10 dBm se reduce a 75Km.

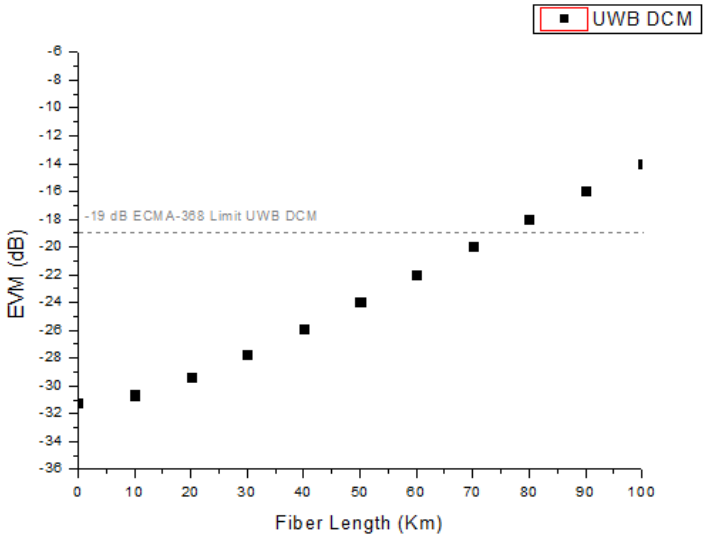

Figura 5. 49 EVM en función de la longitud de fibra, para una potencia de entrada de 10dBm

6 Conclusiones

En este proyecto se ha realizado el estudio de distintas arquitecturas radio sobre fibra para la transmisión de señales radio en coexistencia empleando la modulación OFDM.

En los capítulos 1 y 2 se han estudiado los sistemas RoF y ha descrito brevemente el origen de las señales de banda ultra ancha, sus principales características, sus competencias y sus aplicaciones.

En el siguiente capítulo nos hemos centrado en el estándar ECMA 368, hemos definido como se especifica la capa física de una señal inalámbrica. En él se ha estudiado como se forma la secuencia a transmitir, cómo se mapea, se convoluciona y se modula en función de la velocidad de datos transmitidos. Es importante entender este capítulo ya que es la base de nuestro proyecto.

En el capítulo 4 se ha generado la señal UWB, con una velocidad de datos de 480Mb/s mediante el software Matlab. Para el diseño de dicha señal, se ha hecho uso del estándar ECMA 368. Se han ido creando las señales paso a paso, tomando especial atención al tipo de mapeo, ya que dichas funciones nos serán muy útiles en el futuro.

Finalmente, en el capítulo 5 se han estudiado distintas arquitecturas radio sobre fibra, para ello primero hemos tomado un primer contacto con el software de simulación, VPI. Empezamos viendo cómo se organiza el programa, para poder generar los escenarios a simular. A continuación se definieron los módulos y los parámetros más característicos y se tomó un primer contacto mediante dos demostraciones que incorpora el software.

Visto ya cómo se organiza, y cuáles son los parámetros generales de las simulaciones, empezamos a crear los distintos escenarios a estudiar. En todos los casos se estudiará cómo se comporta la señal UWB, modulada mediante OFDM en función de la fibra óptica introducida.

El primero escenario a simular es el correspondiente al proyecto FIVER, en él se la señal esta mapeada mediante el mapeo QPSK y el segundo escenario a simular es el correspondiente a una señal UWB con un mapeo de tipo DCM, para este caso, se ha hecho uso de las funciones creadas mediante Matlab debido a que el software de simulación VPI no incorpora ningún módulo que realice dicho mapeo.

Una vez realizado las simulaciones correspondientes se puede hacer una comparación de ambos casos. Para esto, se utilizará una potencia óptica fija de entrada y se irá variando la longitud de la fibra óptica. Las potencias de entrada a estudiar son 14dBm y 10 dBm.

El símbolo rojo representa el EVM correspondiente a la modulación tipo DCM y el símbolo negro representa el EVM del proyecto FIVER. Dependiendo de la potencia de entrada se obtienen distancias más elevadas.

Comparando dichos casos, cuando se transmite 14 dB se observa que para ambas simulaciones la señal puede tener un alcance de hasta 100Km en cambio, en el segundo caso, con una potencia de entrada de 10dBm la distancia máxima se ha reducido a 75Km.

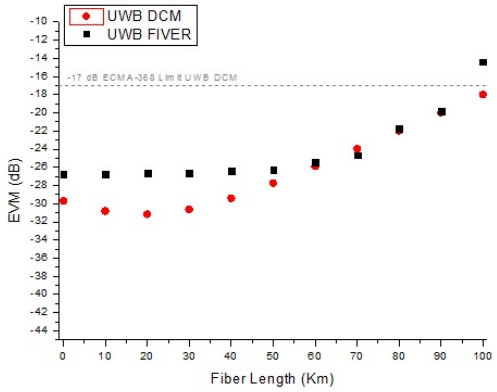

Figura 6. 1 Longitud fibra vs EVM 14 dBm

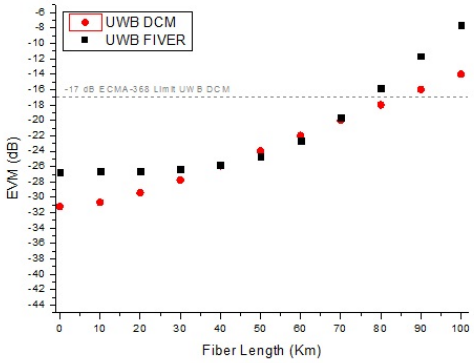

Figura 6. 2 Longitud fibra vs EVM 10 dBm

7 Bibliografía

[1] D. Porcine, P. Research, and W. Hirt, *"Ultra-wideband radio technology: potential and challenges ahead",* IEEE Commun. Mag. 41, 66–74 (2003).

[2] Federal Communications Commission, *"Revision of part 15 of the commissions rules regarding ultra-wideband transmission systems",* (2002).

[3] *"Revision of Part 15 of the Commission's Rules Regarding Ultra-Wideband Transmission Systems," First Note and Order, Federal Communications Commission, ETDocket 98- 153 (2002).*

[4] C. Lim, A. Nirmalathas, M. Bakaul, P. Gamage, K.-.L Lee, Y. Yang, D.Novak and R. Waterhouse, *"Fiber-Wireless Networks and Subsystem Technologies",* vol. 28, no. 4, pp. 390-405, Feb. 15, 2010., in IEEE/OSA J. Lightwave Technol.

[5] W. Zhuang, X.S, Shen, Q. Bi , *Ultra-wideband wireless communications,,* Wirel. Commun. Mob. Comput. 2003. 3:663–685 (DOI: 10.1002/wcm.149).

[6] A.Lowery and J.Armstrong, *"Orthogonal-frequency-division multiplexing for dispersion compensation of long-haul optical systems",* Optics Express 2079, Vol. 14, No. 6, March 2006.

[7] Federal Communications Commissionl, *"New public and broadband internet access among uses authorization of ultra-wideband technology",* http://www.fcc.gov/Bureaus/Engineering_Technology/News_Releases/2002/nret0203.htm, press released 14 2002.

[8] Federal Communications Commission, *Revision of Part 15 of the commission's rules regarding ultra-wideband transmission systems,,* first report and order," ET-Docket 98-153, FCC, Washington, DC, Feb. 2002.

[9] ECMA-368, *High rate ultra wideband PHY and MAC standard, 3rd.*

[10] Standard ECMA-369, *MAC-PHY Interface for ECMA-368,* 3rd edition (December 2008).

[11] VPI, User's Manual.

Anexo: Funciones Matlab

function [Ssync,Preamble_PLCP]=PLCP_Preamble(Npf,Nsym,Sbase,Scover)

```
%% Standard PLCP preamble__Packet/Frame Synchronization Sequence
L1=length(Sbase);
Sext(1,1:L1)=[Sbase];
Sext(L1+1:Nsym)=[0];

for n=1:Npf
   Ssync(((n-1)*Nsym+1):(n*Nsym))=Sext.*Scover(n);
end

%Channel Estimation Sequence
channel_est_r_fren=[0,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,-1,1,1,1,-1,-1,1,1,1,1, -1,-
1,1,-1,1,1,1,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,1,1,-1,1,1,-1,-1,-1,-1,-1,0,0,0,0,0,-1, -1,-1,-1,-1,1,1,-1,1,1,1,1,1,-
1,1,-1,-1,1,1,-1,-1,1,1,1,-1,1,-1,-1,1,1,1,1,-1,-1,1,1,1, -1,1,1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,-1,-
1,1,-1,1,1];

channel_est_i_fren = [0,1,1,-1,1,-1,-1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,1,-1,1,1,1,-1,1,1,1,-1,-1,1,1,1,1, -
1,-1,1,-1,1,1,1,-1,-1,1,1,-1,-1,1,-1,1,1,1,1,1,-1,1,1,-1,-1,-1,-1,-1,0,0,0,0,0,1, 1,1,1,1,-1,-1,1,-1,-1,-1,-
1,-1,1,-1,1,1,-1,-1,1,1,1,-1,-1,-1,1,-1,1,1,1,-1,-1,-1,1,1, -1,-1,-1,1,-1,-1,-1,1,-1,1,-1,-1,1,1,1,-1,-1,1,-
1,1,-1,1,1,-1,1,-1,-1];

channel_est_fren=channel_est_r_fren+1j*channel_est_i_fren;
channel_est=(ifft(channel_est_fren))/sqrt(2);

L=length(channel_est);
Schannel_est_165(1:L)=[channel_est];
Schannel_est_165(L+1:Nsym)=[0];

for ni=Npf+1:30
   Ssync(((n-1)*Nsym+1):(n*Nsym))=Schannel_est_165;
end

Preamble_PLCP=reshape(Ssync,165,(length(Ssync)/165));
```

end

---

function [S_RS_PLCP_Header,Seed_Value]=PLCP_Header

```
%% PHY_Header
PHY=zeros(1,40);
Rate=[0 0 1 1 1];  % RATE para tasa binaria 480Mbps
```

```
PT=0;           % Standard Preamble
T1_T4=[1 0 0 0];   % Para TF=1
BG_LSB=1;       % Para BAND GROUP (BG=1)
BM=1;           %El siguiente paquete no pertenece a la secuencia.

length_bit=[0 0 0 0 0 0 0 0 0 0 rand(1,2)>0.5];

   % SCRAMBLER INIT
   seed_identifier=rand(1,2)>0.5;
      S1=seed_identifier(1);
      S2=seed_identifier(2);
         if [S1,S2]==[0 0]
            Seed_Value=[0 0 1 1 1 1 1 1 1 1 1 1 1 1];
            PRBS=[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0];
         end

         if [S1,S2]==[0 1]
            Seed_Value=[0 1 1 1 1 1 1 1 1 1 1 1 1 1];
            PRBS=[0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0];
         end

         if [S1,S2]==[1 0]
            Seed_Value=[1 0 1 1 1 1 1 1 1 1 1 1 1 1];
            PRBS=[0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0];
         end

         if [S1,S2]==[1 1]
            Seed_Value=[1 1 1 1 1 1 1 1 1 1 1 1 1 1];
            PRBS=[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0];
         End

PHY(4:8)=[Rate];
PHY(9:20)=[length_bit];
PHY(23)=[S1];
PHY(24)=[S2];
PHY(27)=[BM];
PHY(28)=[PT];
PHY(29:31)=[T1_T4(1:3)];
PHY(32)=[BG_LSB];
PHY(35)=[T1_T4(4)];

%% Tail bits
Tail_bits_1=zeros(1,6);
Tail_bits_2=zeros(1,6);
Tail_bits_3=zeros(1,4);
```

```
%% Scambred MAC Header + HCS
MAC=randi([0:1],1,80);
PHY_MAC=[PHY,MAC];
LPHY_MAC=length(PHY_MAC);

%% Header check sequence
data_serial=uint8(PHY_MAC);
Ldata_Input=LPHY_MAC;
Preset_Register=ones(1,16);

for i=1:Ldata_Input
   out=bitxor(Preset_Register(16),data_serial(i));
   Preset_Register(16)=Preset_Register(15);
   Preset_Register(15)=Preset_Register(14);
   Preset_Register(14)=Preset_Register(13);
   Preset_Register(13)=bitxor(Preset_Register(12),out);
   Preset_Register(12)=Preset_Register(11);
   Preset_Register(11)=Preset_Register(10);
   Preset_Register(10)=Preset_Register(9);
   Preset_Register(9)=Preset_Register(8);
   Preset_Register(8)=Preset_Register(7);
   Preset_Register(7)=Preset_Register(6);
   Preset_Register(6)=bitxor(Preset_Register(5),out);
   Preset_Register(5)=Preset_Register(4);
   Preset_Register(4)=Preset_Register(3);
   Preset_Register(3)=Preset_Register(2);
   Preset_Register(2)=Preset_Register(1);
   Preset_Register(1)=out;
end

for i=1:16
   HCS(17-i)=1-Preset_Register(i);
end

%% Scambred MAC Header + HCS
data_Scrambled=[MAC,HCS];
Seed_Value;
L_data_Scrambled=length(MAC)+16;

for i=1:L_data_Scrambled
   out=bitxor(Seed_Value(14),Seed_Value(15));
   Seed_Value(15)=Seed_Value(14);
   Seed_Value(14)=Seed_Value(13);
   Seed_Value(13)=Seed_Value(12);
```

```
    Seed_Value(12)=Seed_Value(11);
    Seed_Value(11)=Seed_Value(10);
    Seed_Value(10)=Seed_Value(9);
    Seed_Value(9)=Seed_Value(8);
    Seed_Value(8)=Seed_Value(7);
    Seed_Value(7)=Seed_Value(6);
    Seed_Value(6)=Seed_Value(5);
    Seed_Value(5)=Seed_Value(4);
    Seed_Value(4)=Seed_Value(3);
    Seed_Value(3)=Seed_Value(2);
    Seed_Value(2)=Seed_Value(1);
    Seed_Value(1)=out;
    final_out(i)=bitxor(data_Scrambled(i),out);
end

MAC_Header_HCS=double(final_out);

%% Reed-Solomon outer code for the PLCP header
datain=[PHY,MAC_Header_HCS];
data_appends=[zeros(1,232*8),datain];
data_temp=zeros(1,249);
for p=1:249
    for q=1:8
        data_temp(p)=(data_appends((p-1)*8+q))*(2^(8-q))+data_temp(p);
    end
end
data_uint=uint16(data_temp);
for i=1:6
    reg(i)=uint16(0);
end
g5=126;
g4=4;
g3=158;
g2=58;
g1=49;
g0=117;
for t=1:249
    temp=bitxor(reg(6), data_uint(t));
    reg(6)=bitxor(GF8_multiply(g5,temp),reg(5));
    reg(5)=bitxor(GF8_multiply(g4,temp),reg(4));
    reg(4)=bitxor(GF8_multiply(g3,temp),reg(3));
    reg(3)=bitxor(GF8_multiply(g2,temp),reg(2));
    reg(2)=bitxor(GF8_multiply(g1,temp),reg(1));
    reg(1)=GF8_multiply(g0,temp);
end
```

```
reg_bin1=dec2bin(reg(1));
reg_bin2=dec2bin(reg(2));
reg_bin3=dec2bin(reg(3));
reg_bin4=dec2bin(reg(4));
reg_bin5=dec2bin(reg(5));
reg_bin6=dec2bin(reg(6));

len_reg1=max(size(reg_bin1));
len_reg2=max(size(reg_bin2));
len_reg3=max(size(reg_bin3));
len_reg4=max(size(reg_bin4));
len_reg5=max(size(reg_bin5));
len_reg6=max(size(reg_bin6));

reg_dou1=double(reg_bin1)-48;
reg_dou2=double(reg_bin2)-48;
reg_dou3=double(reg_bin3)-48;
reg_dou4=double(reg_bin4)-48;
reg_dou5=double(reg_bin5)-48;
reg_dou6=double(reg_bin6)-48;

RS_Bits=[zeros(1,8-len_reg6), reg_dou6,...
    zeros(1,8-len_reg5), reg_dou5,...
    zeros(1,8-len_reg4), reg_dou4,...
    zeros(1,8-len_reg3), reg_dou3,...
    zeros(1,8-len_reg2), reg_dou2,...
    zeros(1,8-len_reg1), reg_dou1,...
    ]  ;
 % Scrambled and RS Encoded PLCP Header
S_RS_PLCP_Header=[PHY,Tail_bits_1,MAC_Header_HCS,Tail_bits_2,RS_Bits,Tail_bits_3];
```

end

---

function [CONV_Header]=CONV_ENC_HEADER(S_RS_PLCP_Header)

```
LPLCP_Header=length(S_RS_PLCP_Header);
data_u=uint16(S_RS_PLCP_Header);

for i=1:6
   dr(i)=uint16(0);
end

for j=1:LPLCP_Header
   a=bitxor(data_u(j),dr(2));
```

```
   a=bitxor(a,dr(3));
   a=bitxor(a,dr(5));
   a=bitxor(a,dr(6));
   b=bitxor(data_u(j),dr(1));
   b=bitxor(b,dr(2));
   b=bitxor(b,dr(4));
   b=bitxor(b,dr(6));
   c=bitxor(data_u(j),dr(1));
   c=bitxor(c,dr(2));
   c=bitxor(c,dr(3));
   c=bitxor(c,dr(6));
   dr(6)=dr(5);
   dr(5)=dr(4);
   dr(4)=dr(3);
   dr(3)=dr(2);
   dr(2)=dr(1);
   dr(1)=data_u(j);
   out_temp(3*j-2)=a;
   out_temp(3*j-1)=b;
   out_temp(3*j-0)=c
end
CONV_Header=double(out_temp);

end
```

```
function [BITINT_Header]=BIT_INTER_HEADER(CONV_Header,Ncbp6s,Ncbps,Ntds,Ntint,Ncyc)

%% SYMBOL INTERLEAVER
for i=1:(length(CONV_Header)/Ncbp6s)
   for j=1:Ncbp6s
      as((i-1)*Ncbp6s+j)=CONV_Header((i-1)*Ncbp6s+floor((j-1)/Ncbps)+(6/Ntds)*mod((j-
1),Ncbps)+1);
   end
end

%% INTRA - SYMBOL TONE INTERLEAVING
for i=1:(length(CONV_Header)/Ncbps)
   for j=1:Ncbps
      at((i-1)*Ncbps+j)=as((i-1)*Ncbps+floor((j-1)/Ntint)+10*mod((j-1),Ntint)+1);
   end
end

%% INTRA - SYMBOL CYCLIC SHIFTS
for i=1:(length(CONV_Header)/Ncbp6s)
```

```
   for j=1:Ncbp6s
      BITINT_Header ((i-1)*Ncbp6s+j)=at((i-1)*Ncbp6s+(floor((j-1)/Ncbps))*Ncbps+mod(((j-
1)+(floor((j-1)/Ncbps))*Ncyc),Ncbps)+1);
   end
end

end
```

```
function  [MAP_HEADER]=MAPPED_HEADER(BITINT_Header)

for i=1:(length(BITINT_Header)/2)
   MAP_HEADER(i)=((2*BITINT_Header((i-1)*2+1)-1)+j*(2*BITINT_Header((i-1)*2+2)-1));
end

% Constelación QPSK - DCM
figure(1)
plot(real(MAP_HEADER),imag(MAP_HEADER),'*');
xlabel('Inphase Component');
ylabel('Quadrature Phase component');
title('MAP_HEADER');
grid
axis([-2,2,-2,2]);

MAP_HEADER=(1/sqrt(2)).*MAP_HEADER;
end
```

```
function [sym]=MAPPING_HEADER(MAP_HEADER)

data=double(MAP_HEADER);
p(7)=-1;
p(8)=1;
p(9)=-1;
p(10)=-1;
p(11)=-1;
p(12)=-1;
for i=1:(length(MAP_HEADER)/50)
   for j=1:50
      sym((i-1)*200+j)=data((i-1)*50+j);
      sym((i-1)*200+101-j)=conj(data((i-1)*50+j));
      sym((i-1)*200+100+j)=p(mod(i+6,127))*data((i-1)*50+j);
      sym((i-1)*200+100+101-j)=p(mod(i+6,127))*conj(data((i-1)*50+j));
   end
end
```

```
out=sym;

end
```

```
function [Guard]=GUARD_HEADER(sym)
for i=1:(length(sym)/100)
      Guard ((i-1)*10+1)=sym((i-1)*100+1);
      Guard ((i-1)*10+2)=sym((i-1)*100+2);
      Guard i-1)*10+3)=sym((i-1)*100+3);
      Guard ((i-1)*10+4)=sym((i-1)*100+4);
      Guard ((i-1)*10+5)=sym((i-1)*100+5);
Guard ((i-1)*10+6)=sym((i-1)*100+96);
Guard ((i-1)*10+7)=sym((i-1)*100+97);
Guard ((i-1)*10+8)=sym((i-1)*100+98);
Guard ((i-1)*10+9)=sym((i-1)*100+99);
Guard ((i-1)*10+10)=sym((i-1)*100+100);
end
end
```

```
function [Pilot]=PILOT_HEADER(sym)
dpi(1)=(1-j)/sqrt(2);
dpi(4)=(1-j)/sqrt(2);
dpi(2)=(-1+j)/sqrt(2);
dpi(3)=(-1+j)/sqrt(2);
dpi(5)=(-1+j)/sqrt(2);
dpi(6)=(-1+j)/sqrt(2);
dpi(9)=(1+j)/sqrt(2);
dpi(12)=(1+j)/sqrt(2);
dpi(7)=(-1-j)/sqrt(2);
dpi(8)=(-1-j)/sqrt(2);
dpi(10)=(-1-j)/sqrt(2);
dpi(11)=(-1-j)/sqrt(2);

p(1)=1;
p(2)=1;
p(3)=1;
p(4)=1;
p(5)=-1;
p(6)=-1;

sp(1)=-1;
sp(2)=1;
sp(3)=-1;
```

```
sp(4)=-1;
sp(5)=-1;
sp(6)=-1;

for i=1:(length(sym)/200)
   for k=1:12
      Pilot((i-1)*24+k)=dpi(k)*p(i);
      Pilot((i-1)*24+12+k)=dpi(k)*p(i)*sp(i);
   end
end

end
```

```
function [Carrier]=CARRIER_HEADER(sym,Guard,Pilot)

for i=1:12
   temp((i-1)*128+1)=0;
   temp((i-1)*128+2)=0;
   temp((i-1)*128+3)=0;
   temp((i-1)*128+4)=0;
   temp((i-1)*128+5)=0;
   temp((i-1)*128+67)=0;

   temp((i-1)*128+6)=Guard((i-1)*10+1);
   temp((i-1)*128+7)=Guard((i-1)*10+2);
   temp((i-1)*128+8)=Guard((i-1)*10+3);
   temp((i-1)*128+9)=Guard((i-1)*10+4);
   temp((i-1)*128+10)=Guard((i-1)*10+5);
   temp((i-1)*128+124)=Guard((i-1)*10+6);
   temp((i-1)*128+125)=Guard((i-1)*10+7);
   temp((i-1)*128+126)=Guard((i-1)*10+8);
   temp((i-1)*128+127)=Guard((i-1)*10+9);
   temp((i-1)*128+128)=Guard((i-1)*10+10);

   for k=1:12
      temp((i-1)*128+12+(k-1)*10)=Pilot((i-1)*12+k);
   end
      temp((i-1)*128+11)=sym((i-1)*100+1);
   temp((i-1)*128+123)=sym((i-1)*100+100);
      for k=1:9
      temp((i-1)*128+12+k)=sym((i-1)*100+k+1);
   end
      for k=10:18
      temp((i-1)*128+13+k)=sym((i-1)*100+k+1);
```

```
    end
      for k=19:27
      temp((i-1)*128+14+k)=sym((i-1)*100+k+1);
    end
     for k=28:36
      temp((i-1)*128+15+k)=sym((i-1)*100+k+1);
    end
    for k=37:45
      temp((i-1)*128+16+k)=sym((i-1)*100+k+1);
    end
    for k=46:49
      temp((i-1)*128+17+k)=sym((i-1)*100+k+1);
    end
    for k=50:53
      temp((i-1)*128+18+k)=sym((i-1)*100+k+1);
    end
    for k=54:62
      temp((i-1)*128+19+k)=sym((i-1)*100+k+1);
    end
      for k=63:71
      temp((i-1)*128+20+k)=sym((i-1)*100+k+1);
    end
    for k=72:80
      temp((i-1)*128+21+k)=sym((i-1)*100+k+1);
    end
    for k=81:89
      temp((i-1)*128+22+k)=sym((i-1)*100+k+1);
    end
    for k=90:98
      temp((i-1)*128+23+k)=sym((i-1)*100+k+1);
    end
end

for i=1:12
    for k=1:62
      switch_data((i-1)*128+k)=temp((i-1)*128+k+66);
    end
    for k=63:128
      switch_data((i-1)*128+k)=temp((i-1)*128+k-62);
    end
end
Carrier=switch_data;
end
```

function [Scramer_PSDU]=PSDU(Seed_Value)

```
length_bit=[0 0 0 0 0 0 0 0 0 0 0 1];
matrix=length_bit.*[2^0,2^1,2^2,2^3,2^4,2^5,2^6,2^7,2^8,2^9,2^10,2^11];
l_playload_bit=sum(matrix)*8;
playload_serial=rand(1,l_playload_bit)>0.5;
data=uint8(playload_serial);

%% Comprobacion De redundancia ciclica CR32, Scramble_FCS
for i=1:32
register(i)=uint8(1);
end
```

```
for i=1:l_playload_bit
   temp=bitxor(register(32),data(i));
   register(32)=register(31);
   register(31)=register(30);
   register(30)=register(29);
   register(29)=register(28);
   register(28)=register(27);
   register(27)=bitxor(register(26),temp);
   register(26)=register(25);
   register(25)=register(24);
   register(24)=bitxor(register(23),temp);
   register(23)=bitxor(register(22),temp);
   register(22)=register(21);
   register(21)=register(20);
   register(20)=register(19);
   register(19)=register(18);
   register(18)=register(17);
   register(17)=bitxor(register(16),temp);
   register(16)=register(15);
   register(15)=register(14);
   register(14)=register(13);
   register(13)=bitxor(register(12),temp);
   register(12)=bitxor(register(11),temp);
   register(11)=bitxor(register(10),temp);
   register(10)=register(9);
   register(9)=bitxor(register(8),temp);
   register(8)=bitxor(register(7),temp);
   register(7)=register(6);
   register(6)=bitxor(register(5),temp);
   register(5)=bitxor(register(4),temp);
   register(4)=register(3);
   register(3)=bitxor(register(2),temp);
   register(2)=bitxor(register(1),temp);
   register(1)=temp;
 end
 for k=32:-1:1
   Scramble_FCS(33-k)=1-register(k);
end

%% Unscrambled Tail Bits
playload_6zero_pad=[double(playload_serial),Scramble_FCS,zeros(1,6)];

%% Scrambled Pad Bits
Nibp6s=900;
```

```
Playload_appended=Nibp6s.*ceil((l_playload_bit+38)/Nibp6s)-(l_playload_bit+38);
Npad=Playload_appended;
Playload_appened_data=[playload_6zero_pad,zeros(1,Playload_appended)];
seed_identifier=[1 1];

if isequal(seed_identifier,[0 0])
  for i=1:15
     reg=[0 0 1 1 1 1 1 1 1 1 1 1 1 1 1];
  end
elseif isequal(seed_identifier,[0 1])
   for i=1:15
     reg=[0 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
  end
elseif isequal(seed_identifier,[1 0])
   for i=1:15
     reg=[1 0 1 1 1 1 1 1 1 1 1 1 1 1 1];
   end
elseif isequal(seed_identifier,[1 1])
   for i=1:15
     reg=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];
   end
end

data=uint8(Playload_appened_data);

for j=1:length(Playload_appened_data)
temp=bitxor(reg(14),reg(15));
reg(15)=reg(14);
reg(14)=reg(13);
reg(13)=reg(12);
reg(12)=reg(11);
reg(11)=reg(10);
reg(10)=reg(9);
reg(9)=reg(8);
reg(8)=reg(7);
reg(7)=reg(6);
reg(6)=reg(5);
reg(5)=reg(4);
reg(4)=reg(3);
reg(3)=reg(2);
reg(2)=reg(1);
reg(1)=temp;
out_temp(j)=bitxor(data(j),temp);
end
```

```
Scambred_playload=double(out_temp);
Scramer_PSDU=[Scambred_playload(1:length(l_playload_bit+32)),zeros(1,6),Scambred_playload((l
_playload_bit+39):length(Scambred_playload))];

end
```

```
function [CONV_PSDU]=CONV_ENC_PSDU(Scramer_PSDU)

LScramer_PSDU=length(Scramer_PSDU);

data_u=uint16(Scramer_PSDU);
for i=1:6
   dr(i)=uint16(0);
end

for j=1:LScramer_PSDU
   a=bitxor(data_u(j),dr(2));
   a=bitxor(a,dr(3));
   a=bitxor(a,dr(5));
   a=bitxor(a,dr(6));
   b=bitxor(data_u(j),dr(1));
   b=bitxor(b,dr(2));
   b=bitxor(b,dr(4));
   b=bitxor(b,dr(6));
   c=bitxor(data_u(j),dr(1));
   c=bitxor(c,dr(2));
   c=bitxor(c,dr(3));
   c=bitxor(c,dr(6));
   dr(6)=dr(5);
   dr(5)=dr(4);
   dr(4)=dr(3);
   dr(3)=dr(2);
   dr(2)=dr(1);
   dr(1)=data_u(j);
   out_temp(3*j-2)=a;
   out_temp(3*j-1)=b;
   out_temp(3*j-0)=c;
end

CONV_PSDU=double(out_temp);
end
```

---

function [BITINT_PSDU]=BIT_INTER_PSDU(CONV_PSDU,Ncbp6s,Ncbps,Ntds,Ntint,Ncyc);

```
for i=1:(length(CONV_PSDU)/9)
      CONV_PSDU((i-1)*4+1)=CONV_PSDU((i-1)*9+1);
      CONV_PSDU((i-1)*4+2)=CONV_PSDU((i-1)*9+2);
      CONV_PSDU((i-1)*4+3)=CONV_PSDU((i-1)*9+6);
      CONV_PSDU((i-1)*4+4)=CONV_PSDU((i-1)*9+9);
end

%% SYMBOL INTERLEAVER
for i=1:(length(CONV_PSDU)/Ncbp6s)
   for j=1:Ncbp6s
      as((i-1)*Ncbp6s+j)=CONV_PSDU((i-1)*Ncbp6s+floor((j-1)/Ncbps)+(6/Ntds)*mod((j-
1),Ncbps)+1);
   end
end

%% INTRA - SYMBOL TONE INTERLEAVING
for i=1:(length(CONV_PSDU)/Ncbp6s)
   for j=1:Ncbp6s
      at((i-1)*Ncbp6s+j)=as((i-1)*Ncbps+floor((j-1)/Ntint)+10*mod((j-1),Ntint)+1);
   end
end

%% INTRA - SYMBOL CYCLIC SHIFTS
for i=1:(length(CONV_PSDU)/Ncbp6s)
   for j=1:Ncbp6s
      b((i-1)*Ncbp6s+j)=at((i-1)*Ncbp6s+(floor((j-1)/Ncbps))*Ncbps+mod(((j-1)+(floor((j-
1)/Ncbps))*Ncyc),Ncbps)+1);
   end
end
BITINT_PSDU=b;
```

end

---

function [DCM,DCM50,TX_DCM_DCM50]=MAP_PSDU(PRBS)

```
for i=1:(length(PRBS)/200)
   for k=1:25
      out((i-1)*50+k,1)=PRBS((i-1)*200+2*(k-1)+1);
      out((i-1)*50+k,2)=PRBS((i-1)*200+2*(k-1)+2);
      out((i-1)*50+k,3)=PRBS((i-1)*200+2*(k-1)+51);
```

```
      out((i-1)*50+k,4)=PRBS((i-1)*200+2*(k-1)+52);
    end
    for k=26:50
      out((i-1)*50+k,1)=PRBS((i-1)*200+2*(k-1)+51);
      out((i-1)*50+k,2)=PRBS((i-1)*200+2*(k-1)+52);
      out((i-1)*50+k,3)=PRBS((i-1)*200+2*(k-1)+101);
      out((i-1)*50+k,4)=PRBS((i-1)*200+2*(k-1)+102);
    end
end

for i=1:(length(PRBS)/200)
    for k=1:50
      if out((i-1)+k,1:4)== [0 0 0 0]
        a((i-1)*50+k)=0;
        d((i-1)*50+k)=(-3-1j*3);
        d50((i-1)*50+k)=(1+1j);
      end

      if out((i-1)+k,1:4)==[0 0 0 1]
        a((i-1)*50+k)=1;
        d((i-1)*50+k)=(-3 -1j);
        d50((i-1)*50+k)=(1 -1j*3);
      end

      if out((i-1)+k,1:4)==[0 0 1 0]
        a((i-1)*50+k)=2;
        d((i-1)*50+k)=(-3+1j);
        d50((i-1)*50+k)=(1 +1j*3);
      end

      if out((i-1)+k,1:4)==[0 0 1 1]
        a((i-1)*50+k)=3;
        d((i-1)*50+k)=(-3+1j*3);
        d50((i-1)*50+k)=(1-1j);
      end

      if out((i-1)+k,1:4)==[0 1 0 0]
        a((i-1)*50+k)=4;
        d((i-1)*50+k)=(-1 -1j*3);
        d50((i-1)*50+k)=(-3+1j);
      end
      if out((i-1)+k,1:4)==[0 1 0 1]
        a((i-1)*50+k)=5;
        d((i-1)*50+k)=(-1 -1j);
```

```
   d50((i-1)*50+k)=(-3 -1j*3);
end
if out((i-1)+k,1:4)==[0 1 1 0]
   a((i-1)*50+k)=6;
   d((i-1)*50+k)=(-1+1j);
   d50((i-1)*50+k)=(-3+1j*3);
end

if out((i-1)+k,1:4)==[0 1 1 1]
   a((i-1)*50+k)=7;
   d((i-1)*50+k)=(-1+1j*3);
   d50((i-1)*50+k)=(-3 -1j);
end

if out((i-1)+k,1:4)==[1 0 0 0]
   a((i-1)*50+k)=8;
   d((i-1)*50+k)=(1 -1j*3);
   d50((i-1)*50+k)=(3+1j);
end

if out((i-1)+k,1:4)==[1 0 0 1]
   a((i-1)*50+k)=9;
   d((i-1)*50+k)=(1 -1j);
   d50((i-1)*50+k)=(3 -1j*3);
end

if out((i-1)+k,1:4)==[1 0 1 0]
   a((i-1)*50+k)=10;
   d((i-1)*50+k)=(1+1j);
   d50((i-1)*50+k)=(3+1j*3);
end

if out((i-1)+k,1:4)==[1 0 1 1]
   a((i-1)*50+k)=11;
   d((i-1)*50+k)=(1+1j*3);
   d50((i-1)*50+k)=(3 -1j);
end

if out((i-1)+k,1:4)==[1 1 0 0]
   a((i-1)*50+k)=12;
   d((i-1)*50+k)=(3 -1j*3);
   d50((i-1)*50+k)=(-1+1j);
end
```

```
        if out((i-1)+k,1:4)==[1 1 0 1]
           a((i-1)*50+k)=13;
           d((i-1)*50+k)=(3-1j);
           d50((i-1)*50+k)=(-1-1j*3);
        end

        if out((i-1)+k,1:4)==[1 1 1 0]
           a((i-1)*50+k)=14;
           d((i-1)*50+k)=(3+1j);
           d50((i-1)*50+k)=(-1+1j*3);
        end

        if out((i-1)+k,1:4)==[1 1 1 1]
           a((i-1)*50+k)=15;
           d((i-1)*50+k)=(3+1j*3);
           d50((i-1)*50+k)=(-1 -1j);

        end
     end
end

DCM=double(d);
DCM50=double(d50);

for n=1:(length(PRBS)/200)
   for k=1:50
      TX_DCM_DCM50((n-1)*100+k)=(1/sqrt(10))*DCM((n-1)*50+k);
      TX_DCM_DCM50((n-1)*100+k+50)=(1/sqrt(10))*DCM50((n-1)*50+k);
   end
end

end
```

```
function [TData,Data]=TX_Data_PSDU(TX_DCM_DCM50,Nd)

for i=1:length(TX_DCM_DCM50)/100

   MD((i-1)*128+11)=((i-1)*100+1);
   TData((i-1)*128+11)=TX_DCM_DCM50((i-1)*100+1);

   MD((i-1)*128+123)=((i-1)*100+100);
   TData((i-1)*128+123)=TX_DCM_DCM50((i-1)*100+100);
```

```
for k=1:9
   MD(k)=(k-1)-56;
   TData((i-1)*128+12+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+12+k)=((i-1)*100+k+1);
end

for k=10:18
   MD(k)=(k-1)-56;
   TData((i-1)*128+13+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+13+k)=((i-1)*100+k+1);
end

for k=19:27
   MD(k)=(k-1)-56;
   TData((i-1)*128+14+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+14+k)=((i-1)*100+k+1);
end

for k=28:36
   MD(k)=(k-1)-56;
   TData((i-1)*128+15+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+15+k)=((i-1)*100+k+1);
end

for k=37:45
   MD(k)=(k-1)-56;
   TData((i-1)*128+16+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+16+k)=((i-1)*100+k+1);
end

for k=46:49
   MD(k)=(k-1)-56;
   TData((i-1)*128+17+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+17+k)=((i-1)*100+k+1);
end

for k=50:53
   MD(k)=(k-1)-56;
   TData((i-1)*128+18+k)=TX_DCM_DCM50((i-1)*100+k+1);
   TTX_Data((i-1)*128+18+k)=((i-1)*100+k+1);
end

for k=54:62
   MD(k)=(k-1)-56;
```

```
    TData((i-1)*128+19+k)=TX_DCM_DCM50((i-1)*100+k+1);
    TTX_Data((i-1)*128+19+k)=((i-1)*100+k+1);
  end

  for k=63:71
    MD(k)=(k-1)-56;
    TData((i-1)*128+20+k)=TX_DCM_DCM50((i-1)*100+k+1);
    TTX_Data((i-1)*128+20+k)=((i-1)*100+k+1);
  end

  for k=72:80
    MD(k)=(k-1)-56;
    TData((i-1)*128+21+k)=TX_DCM_DCM50((i-1)*100+k+1);
    TTX_Data((i-1)*128+21+k)=((i-1)*100+k+1);
  end

  for k=81:89
    MD(k)=(k-1)-56;
    TData((i-1)*128+22+k)=TX_DCM_DCM50((i-1)*100+k+1);
    TTX_Data((i-1)*128+22+k)=((i-1)*100+k+1);
  end

  for k=90:98
    MD(k)=(k-1)-56;
    TData((i-1)*128+23+k)=TX_DCM_DCM50((i-1)*100+k+1);
    TTX_Data((i-1)*128+23+k)=((i-1)*100+k+1);
  end

end

Data=TTX_Data;

end
```

```
function [TGuard,Guard]=TX_Guard(TX_DCM_DCM50,Ng)

for n=1:(length(TX_DCM_DCM50))/100

   for i=1:((Ng/2)+1)
    MG(i)=(-62+(i));
    TGuard((n-1)*128+i)=0;
    TTX_Guard((n-1)*128+i)=0;
  end
```

```
   for i=(Ng/2)+1:(Ng)
      MG(i)=52+(i)-1;
      TGuard((n-1)*128+i)=TX_DCM_DCM50(((n-1)*10+i)-5);
      TTX_Guard((n-1)*128+i)=(((n-1)*10+i)-5);
   end

   for i=124:128
      TGuard((n-1)*128+i)=TX_DCM_DCM50(((n-1)*10+i)-118);
      TTX_Guard((n-1)*128+i)=(((n-1)*10+i)-118);
   end

end

Guard=TTX_Guard;

end
```

---

function [TPilot,Pilot]=TX_Pilot_PSDU(TX_DCM_DCM50,Np)

```
p=[1,1,1,1,-1,-1,-1,1,-1,-1,-1,-1,1,1,-1,1,-1,-1,1,1,-1,1,1,-1,1,1,-1,1,1,1,1,1,1,-1,1, 1,1,-1,1,1,-1,-1,1,1,1,-
1,1,-1,-1,-1,1,-1,1,-1,-1,1,-1,-1,1,1,1,1,1,1,-1,-1,1,1, -1,-1,1,-1,1,-1,1,1,-1,-1,-1,1,1,-1,-1,-1,1,1,-1,-1,1,-
1,1,1,1,1,-1,1,-1,1,-1,1,-1,-1,-1,-1,1,-1,1,1,-1,1,-1,1,1,1,-1,-1,1,-1,-1,-1,1,1,1,-1,-1,-1,-1,-1,-1];
 dp(1)=(1+1j)/sqrt(2);
dp(4)=(1+1j)/sqrt(2);
dp(9)=(1+1j)/sqrt(2);
dp(12)=(1+1j)/sqrt(2);
dp(2)=(-1-1j)/sqrt(2);
dp(3)=(-1-1j)/sqrt(2);
dp(5)=(-1-1j)/sqrt(2);
dp(6)=(-1-1j)/sqrt(2);
dp(7)=(-1-1j)/sqrt(2);
dp(8)=(-1-1j)/sqrt(2);
dp(10)=(-1-1j)/sqrt(2);
dp(11)=(-1-1j)/sqrt(2);

for i=1:(length(TX_DCM_DCM50)/100)
   for k=1:Np
      MP(k)=-55+10*(k-1);
      Mpilot((i-1)*12+k)=dp(k)*p(mod(i+15,127)+1);
      TPilot((i-1)*128+12+(k-1)*10)=Mpilot((i-1)*12+k);
      TTX_Pilot((i-1)*128+12+(k-1)*10)=((i-1)*12+k);
   end
```

```
end
Pilot=TTX_Pilot;
end
function [Carrier]=CARRIER_PSDU(TX_DCM_DCM50,TData,TGuard,TPilot)

b=(length((TX_DCM_DCM50))/100)*128;
Data=[TData,zeros(1,b-(length(TData)))];
Guard=[TGuard,zeros(1,b-(length(TGuard)))];
Pilot=[TPilot,zeros(1,b-(length(TPilot)))];

size(Data);
size(Guard);
size(Pilot);
S=Data+Pilot+Guard;
T=[Data',Pilot',Guard',S'];

for i=1:length(TX_DCM_DCM50)/100
   for k=1:62
      Carrier((i-1)*128+k)=S((i-1)*128+k+66);
   end
   for k=63:128
      Carrier((i-1)*128+k)=S((i-1)*128+k-62);
   end
end
end
```