UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN INGENIERÍA
MAESTRÍA EN INGENIERÍA ELÉCTRICA – TELECOMUNICACIONES

IMPLEMENTACIÓN DE LOS BLOQUES DE ENTRELAZADO, CODIFICACIÓN MODULADA Y MULTIPLEXACIÓN EN POTENCIA DE ATSC 3.0 EN GNU RADIO

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA

PRESENTA:
VIRIDIANA MARES RODRÍGUEZ

TUTOR
DR. JOSÉ MARÍA MATÍAS MARURI
FACULTAD DE INGENIERÍA

CDMX, DICIEMBRE 2018

JURADO ASIGNADO:

Presidente:	Dr. Martynyuk Oleksandr
Secretario:	Dr. Moctezuma Flores Miguel
1 er. Vocal:	Dr. Matías Maruri José María
2 d o. Vocal:	Dr. Ceballos Herrera Daniel Enrique
3 er. Vocal:	Dr. García Garduño Víctor

CDMX, Facultad de Ingeniería, UNAM.

TUTOR DE TESIS:

Dr. Matías Maruri José María

_____

FIRMA

Índice general

Índice de figuras **3**

Índice de cuadros **6**

Lista de Términos **9**

1. Introducción **10**
1.1. Justificación 11
1.2. Objetivos de la tesis 12
1.2.1. Objetivos generales 12
1.2.2. Objetivos particulares 12
1.3. Organización de la tesis 13

2. La capa física de ATSC 3.0 **14**
2.1. Características generales 14
2.1.1. Arquitectura 15
2.2. Formato de entrada 17
2.3. Entrelazado y codificación modulada 17
2.3.1. Etapa FEC 18
2.3.2. Entrelazado de bits 21
2.3.3. Modulación 24
2.4. Multiplexación en potencia 25
2.5. Entrelazado y framing 27
2.6. Generación de la forma de onda 29

3. Construcción de módulos en GNU Radio **32**
3.1. Módulos Out-Of-Tree de GNU Radio 32
3.2. Tipos de bloques en GNU Radio 33
3.3. Convención de nombres de los bloques en GNU Radio 34

3.4. La herramienta gr_modtool 34
3.5. Estructura de un módulo de GNU Radio 35
3.6. Módulo desarrollado en la tesis 37

4. Entrelazado y codificación modulada **39**
4.1. Codificador LDPC 40
4.1.1. Conceptos de códigos lineales de bloque 40
4.1.2. Representaciones de un código LDPC 42
4.1.3. Estructuras LDPC del estándar ATSC 3.0 44
4.1.4. Implementación 50
4.2. Entrelazado de bits 51
4.2.1. Implementación 52
4.3. Modulador 54
4.3.1. Implementación 54
4.3.2. Resultados 56

5. Multiplexación por nivel de potencia **61**
5.1. Implementación del bloque de multiplexación en potencia 66
5.1.1. Programación del bloque LDM 66
5.2. Resultados 68

6. Conclusiones **78**

A. Instalación de GNU Radio mediante Pybombs **80**

Índice de figuras

2.1. Diagrama completo del sistema transmisor de ATSC 3.0 [3]. 15
2.2. Diagrama simplificado del transmisor de ATSC 3.0 [3]. 16
2.3. Formato de los paquetes en banda base [3]. 18
2.4. Diagrama del bloque BICM. 18
2.5. Estructura de la trama FEC cuando un código externo es utilizado. . 19
2.6. Diagrama del intercalador de bits. 22
2.7. Palabra de código LDPC divida en grupos de 360 bits [3]. 23
2.8. Constelación 16-QAM para una tasa de 3/15 [3]. 25
2.9. Constelación 16-QAM para una tasa de 12/15 [3]. 25
2.10. Señal de dos capas obtenida después del módulo de LDM. 26
2.11. Bloque LDM para dos capas [3]. 27
2.12. Diagrama del bloque de intercalado y framing. 28
2.13. Diagrama del bloque generación de forma de onda. 29

3.1. Arquitectura de GNU Radio [20]. 36
3.2. Diagrama del módulo ATSC 3.0 programado en GNU Radio 38

4.1. Diagrama de la implementación del bloque de entrelazado y codificación modulada. 39
4.2. Matriz H para un código (8,4) [22]. 43
4.3. Gráfica de Tanner de un código (8,4) [22]. 44
4.4. Matriz circulante de un código cuasi cíclico [23]. 45
4.5. Matriz de chequeo de paridad de un código LDPC con estructura IRA [5]. 46
4.6. Matriz de paridad para una tasa de codificación de 13/15 y longitud de código igual a 16200 [3]. 48
4.7. Interfaz del codificador LDPC de ATSC 3.0. 51
4.8. Ejemplo de salida en consola de codificador LDPC. 51
4.9. Diagrama del intercalador de bits. 51

4.10. Diagrama de la implementación del intercalador de bits 53
4.11. Interfaz del intercalador de bits 53
4.12. Implementación del modulador de ATSC 3.0 55
4.13. Interfaz del modulador de ATSC 3.0 55
4.14. Bloques para la visualización de las constelaciones implementadas. . . 56
4.15. Constelación QPSK uniforme. 57
4.16. Constelación 16-QAM no uniforme para todas las tasas de codificación. 58
4.17. Constelación 64-QAM no uniforme para todas las tasas de codificación. 59
4.18. Constelación 256-QAM no uniforme para todas las tasas de codificación. 60

5.1. (a) TDM (b) FDM y (c) LDM para múltiples servicios de radiodifusión [7]. 62
5.2. Diagrama transmisor con arquitectura LDM [3]. 64
5.3. Bloque LDM para dos capas. 66
5.4. Interfaz con el usuario del Multiplexor en Potencia o LDM. 67
5.5. Propiedades modificables por el usuario del bloque LDM. 67
5.6. Implementación del multiplexor en potencia. 68
5.7. Ejemplo de salida en la consola de GNU Radio al ejecutarse el bloque LDM. 68
5.8. Programa en GNU Radio para la generación de 2 PLPs y para la obtención de las constelaciones después del bloque de multiplexación en potencia. 69
5.9. Superposición de una constelación QPSK (capa CL) y una constelación 16-QAM no uniforme (capa EL), con nivel de inyección de 13 dB. . . 70
5.10. Superposición de una constelación QPSK (capa CL) y una constelación 16-QAM no uniforme (capa EL), con nivel de inyección de 4 dB. . . . 71
5.11. Superposición de constelación QPSK (capa CL) y 64-QAM no uniforme (capa EL) con un nivel de inyección de 13 dB. 71
5.12. Superposición de constelación QPSK (capa CL) y 64-QAM no uniforme (capa EL) con un nivel de inyección de 5 dB. 72
5.13. Constelación 16-QAM en ambas capas con nivel de inyección de 13 dB. 72
5.14. Constelación 16-QAM en ambas capas con nivel de inyección de 20 dB. 73
5.15. Superposición de constelación BPSK (capa CL) y QPSK (capa EL) con un nivel de inyección de 5 dB. 74
5.16. Superposición de constelación BPSK (capa CL) y 16-QAM (capa EL) con un nivel de inyección de 5 dB. 74
5.17. Superposición de constelación 4-QAM (capa CL) y 16-QAM (capa EL) con un nivel de inyección de 9 dB y ruido agregado. 75

5.18. Superposición de constelación BPSK (capa CL) y 256-QAM no uniforme (capa EL) con un nivel de inyección de 3.5 dB. 76
5.19. Superposición de constelación BPSK (capa CL) y 256-QAM no uniforme (capa EL) con un nivel de inyección de 0 dB. 77

Índice de cuadros

2.1. Longitud de la carga útil k y longitud de bits paridad M para los 3 casos de codificación externa: uso de un codificador BCH, CRC o sin codificación externa [3]. 20
2.2. Longitud de la carga útil k y longitud de bits paridad M para los 3 casos de codificación externa: uso de un codificador BCH, CRC o sin codificación externa [3]. 21
2.3. Tipos de estructuras LDPC para cada longitud y tasa de codificación [3]. 22
2.4. Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 64800 bits. 24
2.5. Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 16200 bits. 24

3.1. Características de los bloques desarrollados en la tesis 38

4.1. Tipo de estructura LDPC dependiendo de la longitud del código en bits y de la tasa de codificación [3]. 45
4.2. Constante q_1 según el código a utilizar para la estructura tipo IRA. . 47
4.3. Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 64800 bits. 52
4.4. Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 16200 bits. 52

5.1. Factores de escala y normalización para el módulo LDM [3]. 65

Dedicatoria

A mi hijo Sebastian con mucho amor.
Con esfuerzo y dedicación puedes lograr todo lo que te propongas.

Agradecimientos

Gracias a mi gran amigo y amor Ricardo, quien ha vivido muy de cerca esta etapa en mi vida, me ha alentado a seguir adelante y ha sido un apoyo durante todas las dificultades.

Agradezco a mi familia por su gran apoyo en esta etapa de mi vida. A mis padres y abuelos, cuyas enseñanzas son la base del logro de esta meta. A mis hermanas, por su apoyo y sus palabras de aliento.

Gracias a mi hijo Sebastian, quien ha sido mi impulso y también me ha alentado con sus palabras.

Gracias a mi tutor el Dr. José María, por sus valiosas enseñanzas, su orientación académica y por su amistad. Me encuentro muy agradecida por todo el apoyo brindado durante la etapa de licenciatura y de maestría.

Gracias a mis compañeros y amigos, con quien compartí muchos conocimientos y momentos agradables.

Agradezco al Conacyt por el la beca económica otorgada, durante el apoyo, me fue asignado el CVU 780134.

Agradezco también, al proyecto Conacyt-CDTI número 189235, REFUTV, "Desarrollo de Redes en Frecuencia Única para Televisión Digital ATSC", por todos los aprendizajes adquiridos durante esta etapa, todos ellos a través de las personas que colaboran o colaboraron en el proyecto. También agradezco la amistad de todos aquellos con los que tuve el gusto de convivir dentro del proyecto REFUTV.

Lista de Términos

ALP: ATSC Link-Layer Protocol.
ATSC: Advanced Television Systems Committee.
BCH: Bose, Ray-Chaudhuri y Hocquenghem.
CP: Continual Pilots.
DVB-T2: Digital Video Broadcasting – Terrestrial 2.
FEC: Forward Error Correction.
FFT: Fast Fourier Transform.
IFFT: Inverse Fast Fourier Transform.
LDM: Layered Division Multiplexing.
LDPC: Low-Density Parity-Check.
MIMO: Multiple-Input Multiple-Output.
MISO: Multiple-Input Single-Output.
MPEG: Moving Picture Experts Group.
NUC: Non-Uniform Constellation.
OFDM: Orthogonal Frequency Division Multiplexing.
PLP: Physical Layer Pipe.
PRBS: Pseudo-Random Binary Sequence.
QAM: Quadrature Amplitude Modulation.
SDR: Software Defined Radio.
SISO: Single-Input Single-Output.
SNR: Signal to Noise Ratio.
SP: Scattered Pilots.
TS: Transport Stream.
USRP: Universal Software Radio Peripheral.

Capítulo 1

Introducción

Un problema existente en los sistemas de radiodifusión es la falta de flexibilidad para adaptarse a los requerimientos de la nueva generación de tecnologías. Igualmente se encuentra la incapacidad de incluir en el servicio a los receptores móviles y portátiles de forma eficiente [1].

ATSC (Advanced Television Systems Committee) es una organización internacional dedicada al desarrollo de estándares para televisión digital. El grupo ATSC dio a conocer su última versión denominada 3.0 a finales de 2017 [2]. La versión 3.0 es un estándar de TV digital de nueva generación, tiene como objetivo resolver los principales problemas que enfrentan los sistemas de radiodifusión actuales. Mejora la calidad de vídeo y audio, posee mayor robustez y flexibilidad en comparación con su estándar anterior (A/53), tanto para recepción fija como móvil y portátil [3]. Además el sistema está pensado para adaptarse a futuras tecnologías por lo que incluye señalización extensible.

Este estándar tiene diferencias significativas con respecto a su versión anterior, trabaja la modulación OFDM (Orthogonal Frequency Division Multiplexing), utiliza codificación LDPC (Low Density Parity Check) y una multiplexación en nivel de potencia llamada LDM (Layer-Division Multiplexing) [4].

LDM es una tecnología que resuelve el problema de transmitir de manera eficiente servicios para recepción fija y móvil simultáneamente. A diferencia de otras técnicas como FDM o TDM, LDM utiliza el 100 % del espectro durante el 100 % de tiempo disponible. Por otro lado, los códigos LDPC han demostrado tener un desempeño cercano al límite descrito por Shannon y son últimamente muy utilizados [5]. Los códigos LDPC se encuentran dentro de un módulo llamado BICM (Bit Interleaved and Coded Modulation) o módulo de entrelazado y codificación modulada, el cual se encarga de la codificación (uso de LDPC), entrelazado y modulación.

Otro aspecto a destacar del sistema ATSC 3.0 es que posee gran flexibilidad, tiene 12 posibles tasas de codificación, 6 tipos de constelaciones, 16 patrones diferentes de portadoras piloto, 12 diferentes intervalos de guarda y 2 tipos de entrelazado en tiempo [4].

En conjunto todas estas técnicas hacen del estándar ATSC 3.0 un sistema de radiodifusión muy robusto, mejorando el desempeño del sistema en general [6].

Por otra parte, la tecnología SDR (Software Defined Radio), que se basa en configurar la capa física de comunicaciones mediante software o firmware, ha tenido un gran crecimiento y se ha observado la tendencia de los sistemas hacia el software, debido a esto, es importante empezar a desarrollar módulos del sistema transmisor de ATSC 3.0 en esta tecnología. Esto además permitiría aumentar la flexibilidad inherente al sistema.

En esta tesis se desarrolla el módulo de entrelazado y codificación modulada (BICM), y el módulo de multiplexación en potencia (LDM). Estas implementaciones se hacen en GNU Radio debido a que es una herramienta de desarrollo que provee bloques de procesamiento de señal para implementar sistemas de radio definida por software. Cabe aclarar que la implementación se basa en los modos de operación típicos de ATSC 3.0.

1.1. Justificación

ATSC 3.0 posee una amplia flexibilidad, permite transmitir con robustez diferenciada y cuenta con una eficiencia espectral alta. Esto abre la posibilidad de dar servicio a receptores fijos y móviles de manera eficiente. Este estándar implementa varias mejoras respecto a su versión anterior y se intenta adaptar a las demandas de la tecnología actual y de posibles tecnologías futuras. De ahí la importancia sobre el estudio de este sistema y de las nuevas tecnologías que ha incorporado.

Con la finalidad de contar con una herramienta que permita el estudio de los diferentes modos de operación definidos en ATSC 3.0, se desea implementar los módulos BICM y LDM, los cuales incluyen las técnicas de entrelazado, codificación LDPC, modulación y multiplexación en nivel de potencia.

La implementación de todos los módulos se lleva a cabo en el software GNU Radio, éste permite la implementación de sistemas definidos por software. El desarrollo de los módulos en esta tecnología denominada SDR (Software Defined Radio) reduce el costo de implementación y hace posible una fácil reconfiguración o actualización de los parámetros de los bloques.

También es posible el estudio por separado de cada una de las técnicas descritas.

Sobre todo hablando de LDM y LDPC, las cuales son últimamente usadas debido a que han demostrado un buen desempeño [5] [7].

Por otro lado el desarrollo de los primeros bloques del transmisor de ATSC 3.0, como son el BICM y el LDM abre la posibilidad de la implementación de un sistema transmisor de ATSC 3.0 a futuro. Un transmisor de bajo costo definido por software, el cual permitiría el estudio de las principales características del sistema.

Para el caso del bloque de entrelazado y codificación modulada (BICM), es posible seleccionar entre 12 tasas de codificación (2/15 a 13/15), dos posibles longitudes de código (16200 o 64800) y 6 órdenes de constelación (QPSK y de 16-QAM a 4096-QAM). Además, la forma en que se intercalan los bits depende del codificador LDPC y del orden de la constelación. De este modo es posible seleccionar diferentes modos de operación, según las necesidades del radiodifusor.

Debido a que el bloque BICM de ATSC 3.0 cuenta con una gran flexibilidad en cuanto a la configuración de sus parámetros y bloques, el desarrollo del módulo se lleva a cabo con la finalidad de obtener los modos de operación más usuales.

1.2. Objetivos de la tesis

1.2.1. Objetivos generales

Esta tesis tiene como objetivo general el desarrollo del módulo de multiplexación en nivel de potencia LDM y del módulo de entrelazado y codificación modulada BICM, éste último es el módulo que alimenta al multiplexor de potencia.

Cabe resaltar que el módulo de entrelazado y codificación modulada está a su vez compuesto de bloques independientes entre sí: consiste en un codificador LDPC, un intercalador de bits y un modulador.

Todas las implementaciones serán realizadas en el software libre GNU Radio en una distribución con sistema operativo Linux.

Se pretende desarrollar solo algunos modos de operación definidos como obligatorios dentro del estándar ATSC 3.0, con la posibilidad de ampliar el número de modos de operación en un futuro.

1.2.2. Objetivos particulares

Como objetivos secundarios y derivados del módulo de entrelazado y codificación modulada (BICM), se tiene como objetivo desarrollar el bloque codificador LDPC, el bloque de entrelazado y el bloque de modulación.

En el caso del bloque LDM es necesario desarrollar previamente un controlador y un normalizador de nivel de potencia.

1.3. Organización de la tesis

Esta se compone de 6 capítulos:

Capítulo 1 Consiste en la introducción, los objetivos y la justificación de la tesis.

Capítulo 2 Contiene la descripción general de la capa física de ATSC 3.0.

Capítulo 3 Se describe de manera general la estructura de un módulo de GNU Radio, así como los conceptos fundamentales para la construcción de bloques en esta plataforma.

Capítulo 4 Se describen los fundamentos, la implementación y los resultados del bloque de entrelazado y codificación modulada (BICM), el cual consiste en un codificador LDPC, un intercalador de bits y un modulador.

Capítulo 5 Se describen los fundamentos, la implementación y resultados del bloque de multiplexado en nivel de potencia (LDM).

Capítulo 6 Conclusiones de la tesis y trabajo futuro.

Capítulo 2

La capa física de ATSC 3.0

2.1. Características generales

Una de las principales características del estándar ATSC 3.0 es la flexibilidad que es capaz de brindar al radiodifusor a través de los diferentes modos de operación que pueden ser configurados, los cuales dependerán de la robustez y la eficiencia deseada.

ATSC 3.0 utiliza la modulación OFDM y un codificador llamado LDPC (Low-Density Parity-Check), para el cual existen dos longitudes de código y 12 tasas de codificación. Utiliza 3 modos de multiplexación: en tiempo (TDM), en potencia (LDM, Layered Division Multiplexing) y en frecuencia (FDM), para los cuales existen 3 tipos de sub-tramas: SISO, MISO y MIMO. Para una amplia recepción a ecos de la señal, existen 12 posibilidades para la longitud del intervalo de guarda. Para la estimación de canal existen 16 patrones de portadoras piloto dispersas y continuas. Para contrarrestar el efecto Doppler existen tres tamaños de FFT: 8k, 16k, 32k [3].

En un canal de 6 MHz es posible transmitir menos de 1Mbps cuando se utiliza el modo de operación de menor capacidad y hasta 57 Mbps en el modo de operación de mayor capacidad. El estándar también soporta 7 u 8 MHz de ancho de banda [3].

Los datos son llevados en PLPs (Physical Layer Pipes), los cuales son canales lógicos a nivel de capa física, que pueden ser configurados con diferentes modos de operación, los modos de operación varían dependiendo de la robustez de la señal y de la capacidad del canal requerida [4].

Es posible utilizar hasta 4 PLPs simultáneos para transportar diferentes flujos de datos, con el objetivo de que sean ensamblados para formar un flujo de datos final. De acuerdo con lo anterior se pueden tener diferentes tipos de fuente de entrada como: audio, vídeo, vídeo mejorado y datos de la aplicación, cada uno de ellos enviado en un PLP individual a diferentes niveles de robustez. También en el caso de que diferentes

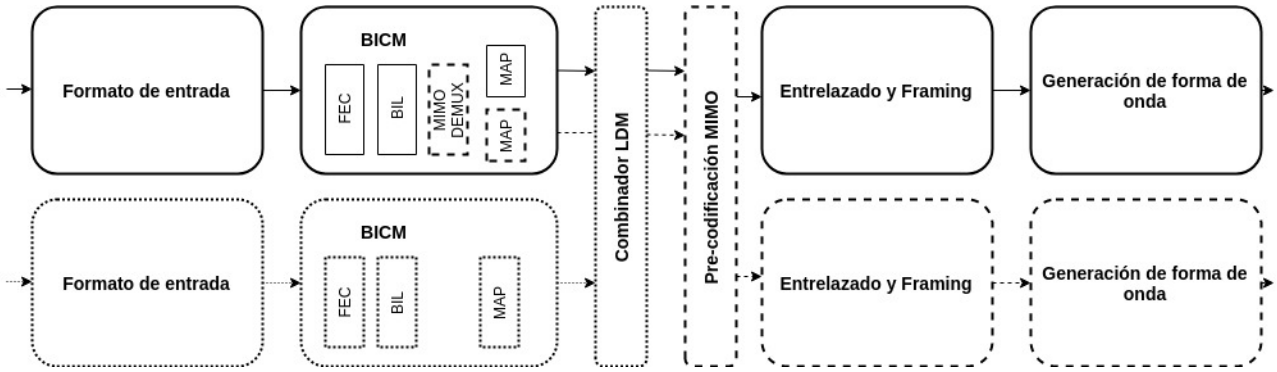

Figura 2.1: Diagrama completo del sistema transmisor de ATSC 3.0 [3].

productos a entregar requieran el mismo nivel de robustez de señal es posible que compartan PLPs. Además de todas las herramientas que ATSC 3.0 ofrece al radiodifusor para adaptar el sistema a sus necesidades, es posible implementar tecnologías nuevas sin afectar las existentes, ya que el sistema tiene señalización extensible [3] con la posibilidad de que sea adaptable a futuras tecnologías.

Por último cabe destacar que el estándar ATSC 3.0 no es compatible con su versión anterior utilizada: ATSC 1.0.

2.1.1. Arquitectura

En la figura 2.1 se muestra el diagrama completo del sistema transmisor, cabe aclarar que no todos los bloques se utilizan al mismo tiempo. Los bloques con línea punteada corresponden al bloque LDM mientras que los bloques con línea a rayas corresponden a bloques tipo MIMO. Cuando se utilizan los bloques tipo LDM (bloques con línea punteada) los bloques tipo MIMO no deben ser utilizados y viceversa. Los bloques con línea continua son comunes a todas las configuraciones [3].

La arquitectura del sistema transmisor se compone de cuatro partes principales: formato de entrada, entrelazado y modulación codificada (bloque denominado BICM), framing y entrelazado y por último generación de la forma de onda.

En esta tesis se estudian los bloques correspondientes a la multiplexación por división de capas o LDM, es decir los bloques dibujados con línea punteada de la figura 2.1. Esta técnica de multiplexación ha demostrado tener ciertas ventajas sobre otras técnicas como TDM o FDM, dichas ventajas se estudiarán en el apartado 2.4.

En la figura 2.2 se muestra el diagrama del sistema transmisor cuando se utiliza el bloque LDM. Se observa que se utilizan dos PLPs (Physical Layer Pipes). Un PLP es un flujo de datos que tiene una modulación y una codificación específica. Es decir, para cada PLP existe un bloque de formato de entrada y un bloque de BICM diferente

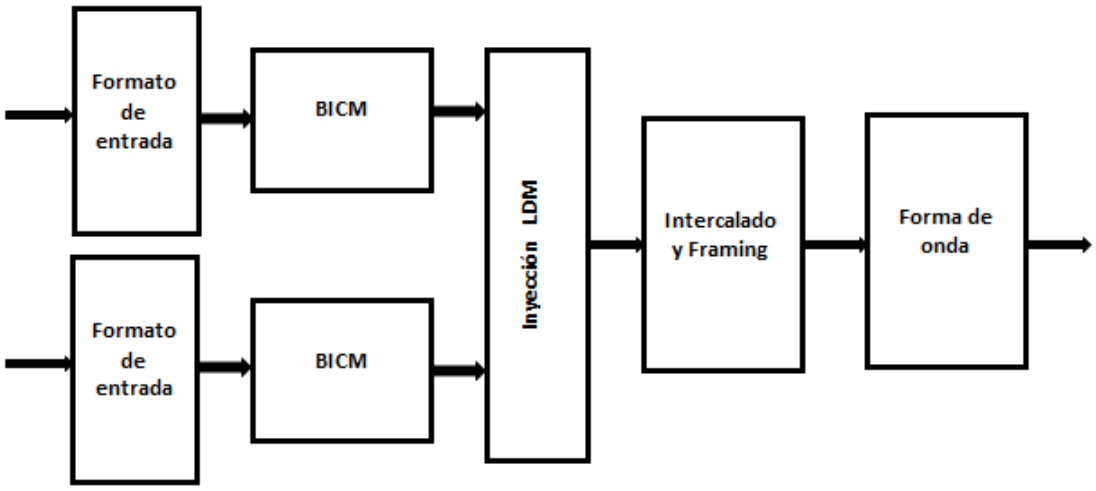

Figura 2.2: Diagrama simplificado del transmisor de ATSC 3.0 [3].

[3].

A continuación se da un panorama general de como la información es procesada bloque a bloque:

El flujo de datos entra al bloque llamado “Formato de entrada” el cuál de manera global se encarga de encapsular y comprimir la información.

Después los datos ingresan al bloque llamado “BICM”, encargado de la codificación, intercalado o entrelazado y modulación. Para la codificación se utiliza siempre un código LDPC (Low Density Parity Check) [3].

A la salida del BICM los datos ingresan al módulo de "LDM", donde los dos flujos de datos entrantes, que corresponden a los dos PLPs, se combinan con diferentes niveles de potencia.

El siguiente módulo es el de “Intercalado y framing”, consiste en tres partes: intercalado en tiempo, framing e intercalado en frecuencia. Después de que los datos se intercalan en tiempo y pasan a través del sub-bloque de framing, se obtienen símbolos OFDM, de esta forma el intercalador en frecuencia trabaja con símbolos OFDM [3].

Finalmente los datos llegan al último bloque llamado “Forma de onda”, el cual consiste de manera general en la inserción de portadoras piloto y en la inserción del intervalo de guarda [3]. Por último una señal llamada bootstrap se añade al comienzo de cada trama y con esto concluye la construcción de la señal de ATSC 3.0.

Con base en la idea general del transmisor de ATSC 3.0 mencionada anteriormente, se describirá de manera detallada cada bloque del sistema transmisor de la figura 2.2.

2.2. Formato de entrada

El primer bloque, denominado "formato de entrada", se compone a su vez de tres sub-bloques: encapsulación y compresión, controlador y framing en banda base.

1) Encapsulación y compresión

El protocolo de encapsulación de ATSC 3.0 es llamado ALP (ATSC Link-Layer Protocol). Los paquetes de datos de entrada pueden ser de varios tipos, por ejemplo, datos genéricos, MPEG-2 TS, datos del protocolo IP, etc [8]. El protocolo ALP hace posible que convivan los diferentes tipos de paquetes en uno solo (el formato ALP). La longitud de los paquetes ALP es variable, la máxima longitud es de 64 kB y la mínima es de 4 kB, incluyendo el encabezado [4].

Para la compresión de vídeo, se soportan diferentes tecnologías de codificación de vídeo, incluído el estándar H.265 [2], cuyo uso se específica en el documento A/341 citado en [9]. En cuanto a audio, se utiliza el AC-4 y el sistema MPEG-H [10] cuyo uso se especifica en [11] y en [12] respectivamente.

2) Controlador

Toma paquetes ALP y agrega información de configuración de flujo. Esta información permite controlar qué datos son enviados en que tiempo y con qué recursos, dependiendo de la calidad de servicio descrita en los metadatos. Es decir, le indica al siguiente sub-bloque (framming de banda base) cómo generar los paquetes banda base [4].

3) Creación de los framing en banda base

La construcción de los framing en banda base consiste en tres partes: construcción de paquetes, construcción de la cabecera y codificación de los paquetes. A la salida de este bloque se puede tener más de un PLP según sea necesario [4]. Cada paquete en banda base se compone de una cabecera y una carga útil. Como se muestra en la figura 2.3, la carga útil son los paquetes ALP. La longitud de los paquetes en banda base está determinada por la tasa de código y la longitud de código elegida en la etapa FEC.

Otra de las finalidades de este bloque es la de evitar secuencias repetitivas que podrían llevar a un mapeo incorrecto en las constelaciones. Todo paquete en banda base se codifica antes de la etapa FEC, incluyendo encabezado y carga útil [4].

2.3. Entrelazado y codificación modulada

El bloque de entrelazado y codificación modulada o también llamado BICM (Bit Interleaved and Coded Modulation) consiste en tres sub-bloques: la etapa FEC, el

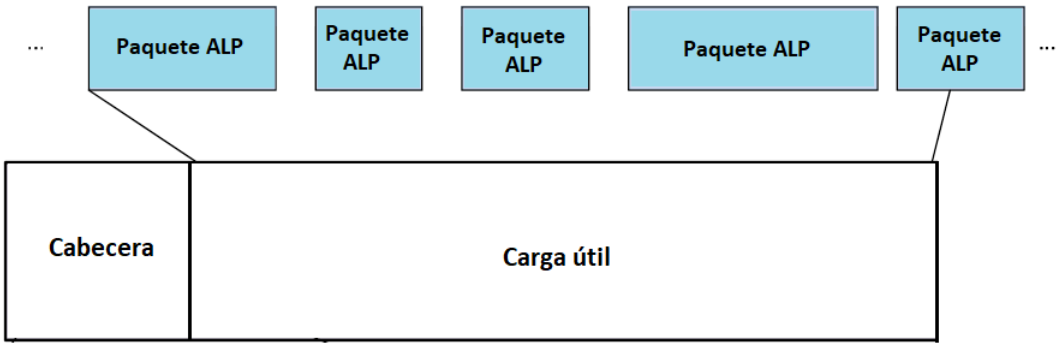

Figura 2.3: Formato de los paquetes en banda base [3].

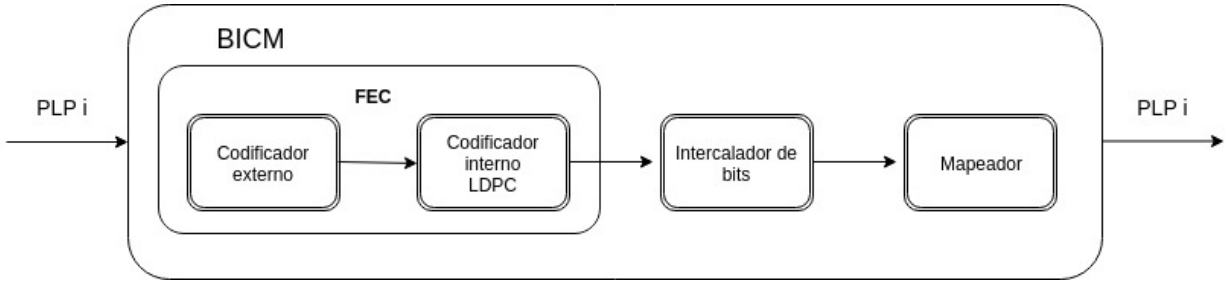

Figura 2.4: Diagrama del bloque BICM.

intercalado o entrelazado de bits y el mapeo, tal y como se muestra en la figura 2.4. A continuación se explican estos 3 bloques.

2.3.1. Etapa FEC

Consiste en dos códigos, un código interno y uno externo. El *código interno* es un LDPC y su uso es obligatorio.

Los códigos LDPC (Low Density Parity Check) son un tipo de codificación que ofrece una alta protección añadiendo un encabezado de longitud mínima [13]. Las longitudes de código LDPC deben ser de 16200 bits ó 64800 bits [4]. Los códigos LDPC de 16200 bits tienen menor latencia pero su rendimiento está por debajo de los códigos de longitud de 64800. Los códigos de longitud de 16200 bits pueden ser usados en ambientes donde la latencia es crítica o se prefiere una estructura de codificador y decodificador más sencilla. En general, se espera que los códigos LDPC de longitud 64800 bits sean la primera elección, debido a que presentan un rendimiento superior [3].

En cuanto a la tasa de codificación, se soportan doce tasas de codificación que van desde 2/15 hasta 13/15, ofreciendo desde una operación muy robusta hasta una

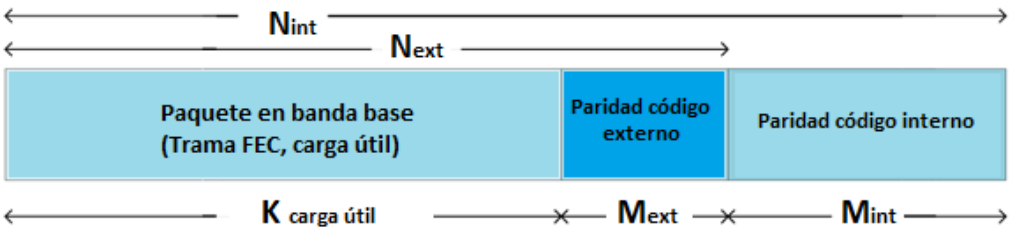

Figura 2.5: Estructura de la trama FEC cuando un código externo es utilizado.

operación de alta capacidad [4].

Respecto al *código externo*, éste tiene como función proveer una detección y/o corrección de errores adicional. Existen tres opciones: la primera opción es utilizar un código de Bose, Ray-Chaudhuri y Hocquenghem (BCH), la segunda opción es utilizar un código de Control de Redundancia Cíclica (CRC), y como tercera opción, se puede no utilizar un código externo. En este último caso no se proporciona ninguna corrección o detección de errores adicional.

Suponiendo que se utilice un codificador tipo BCH entonces se proporciona detección y corrección de errores adicional. Los bits que no pueden ser recuperados por el LDPC se reducen, corrigiendo hasta 12 bits erróneos [4]. El codificador CRC sólo proporciona detección de errores (sin corrección de errores adicional).

En la figura 2.5 se observa como está compuesta la trama de datos en la etapa FEC. La longitud total de la trama (N_{int}) es la longitud del código LDPC, denominado así porque el LDPC es el codificador interno. En la figura 2.5 también se observa que se denomina como N_{ext} a la longitud del codificador externo. La letra M es utilizada para denominar el número de bits de paridad añadidos, por tanto M_{int} y M_{ext} son los bits de paridad del codificador interno y del codificador externo respectivamente. Por último la letra k es utilizada para nombrar a la carga útil.

En la figura 2.5, el número de bits de paridad que añade el codificador externo (M_{ext}) depende de si se utiliza un código CRC o si se utiliza un BCH, cuando no se utiliza codificador externo M_{ext} es igual a cero. De igual forma la longitud de la carga útil (k) depende del codificador externo, cuando no hay codificador externo entonces $k = N_{ext}$, esto último mencionado se puede consultar en los cuadros 2.1 y 2.2, dónde se observa en ambas tablas que la columna 6 y 9 son iguales.

En los cuadros 2.1 y 2.2 también se puede apreciar que cuando se utiliza un código BCH, el número de bits que añade (M_{ext}) es de 192 si la longitud del LDPC (N_{int}) es de 64800 bits. Cuando la longitud del LDPC es 16200 bits se añaden 168 bits. En cambio, cuando se utiliza un código CRC siempre se añaden 32 bits sin importar la longitud (columna 5 en las tablas).

Longitud LDPC= 64800 bits

Tasa Cod.	K (BCH)	M ext (BCH)	K (CRC)	M ext (CRC)	K (s/ext)	M ext (s/ext)	M int	N ext
2/15	8448	192	8608	32	8640	0	56160	8640
3/15	12768	192	12928	32	12960	0	51840	12960
4/15	17088	192	17248	32	17280	0	47520	17280
5/15	21408	192	21568	32	21600	0	43200	21600
6/15	25728	192	25888	32	25920	0	38880	25920
7/15	30048	192	30208	32	30240	0	34560	30240
8/15	34368	192	34528	32	34560	0	30240	34560
9/15	38688	192	38848	32	38880	0	25920	38880
10/15	43008	192	43168	32	43200	0	21600	43200
11/15	47328	192	47488	32	47520	0	17280	47520
12/15	51648	192	51808	32	51840	0	12960	51840
13/15	55968	192	56128	32	56160	0	8640	56160

Cuadro 2.1: Longitud de la carga útil k y longitud de bits paridad M para los 3 casos de codificación externa: uso de un codificador BCH, CRC o sin codificación externa [3].

Es importante observar en las tablas mencionadas anteriormente que el número de bits de paridad del codificador LDPC denotado como M_{int} depende de la tasa de codificación. Por ejemplo, en el caso de una tasa de codificación de 13/15 el número de bits que añade el LDPC es de 8640 para una longitud de 64800 bits. Para el mismo ejemplo, si no se utiliza un codificador externo la longitud de la carga útil será de 56160 bits. Si se utiliza un codificador externo tipo BCH entonces la carga útil será de 55968 y cuando se usa un CRC la carga útil es de 56128 bits.

Tipos de estructuras LDPC en ATSC 3.0

Los códigos LDPC son la parte principal de la etapa FEC del sistema. ATSC 3.0 adoptó dos tipos de estructuras de códigos LDPC:

- Estructura tipo A o MET (Multi-Edge Type structure).
- Estructura tipo B o IRA (Irregular Repeat Accumulate structure).

Cabe destacar que el tipo A o MET presenta mejor desempeño a tasas bajas de codificación, mientras que el tipo B o IRA presenta mejor desempeño a tasas altas de codificación [5], [3]. Este desempeño fue descubierto a partir de parámetros como el FER (Frame Error Rate) y el BER (Bit Error Rate) en pruebas que se realizaron sobre

Longitud LDPC = 16200 bits

Tasa Cod.	K (BCH)	M ext (BCH)	K (CRC)	M ext (CRC)	K (s/ext)	M ext (s/ext)	M int	N ext
2/15	1992	168	2128	32	2160	0	14040	2160
3/15	3072	168	3208	32	3240	0	12960	3240
4/15	4152	168	4288	32	4320	0	11880	4320
5/15	5232	168	5368	32	5400	0	10800	5400
6/15	6312	168	6448	32	6480	0	9720	6480
7/15	7392	168	7528	32	7560	0	8640	7560
8/15	8472	168	8608	32	8640	0	7560	8640
9/15	9552	168	9688	32	9720	0	6480	9720
10/15	10632	168	10768	32	10800	0	5400	10800
11/15	11712	168	11848	32	11880	0	4320	11880
12/15	12792	168	12928	32	12960	0	3240	12960
13/15	13872	168	14008	32	14040	0	2160	14040

Cuadro 2.2: Longitud de la carga útil k y longitud de bits paridad M para los 3 casos de codificación externa: uso de un codificador BCH, CRC o sin codificación externa [3].

ambas estructuras utilizando las modulaciones BPSK y QPSK [5]. El funcionamiento de ambas estructuras se describe posteriormente, en el capítulo 4.

Para cada longitud de código existen las estructuras IRA y MET, dependiendo de la tasa, como se muestra en el cuadro 2.3.

2.3.2. Entrelazado de bits

La operación de entrelazado se lleva a cabo a través de 3 etapas: intercalador de paridad, el intercalador de grupos y el intercalador de bloques, como se muestra en la figura 2.6. Esta estructura permite una decodificación LDPC en paralelo, y optimiza el rendimiento del codificador LDPC a cualquier constelación [6] .

Al intercalador entran FEC frames y salen FEC frames intercalados. El tamaño de la trama FEC no cambia después del entrelazado o intercalado de bits.

El entrelazado o intercalado de bits depende de la tasa de codificación LPDC utilizada y del orden de constelación seleccionado [4]. Esto se hace con la finalidad de optimizar la eficiencia del canal para cada tasa de codificación y orden de constelación.

Intercalador de paridad

Esta etapa del bloque solo se debe utilizar para los códigos LDPC tipo B (estructura IRA). Para los códigos tipo A no se utiliza.

	Tipo de estructura LDPC	
Tasa de Codificación	N int = 64800	N int = 16200
2/15	A	A
3/15	A	A
4/15	A	A
5/15	A	A
6/15	B	B
7/15	A	B
8/15	B	B
9/15	B	B
10/15	B	B
11/15	B	B
12/15	B	B
13/15	B	B

Cuadro 2.3: Tipos de estructuras LDPC para cada longitud y tasa de codificación [3].

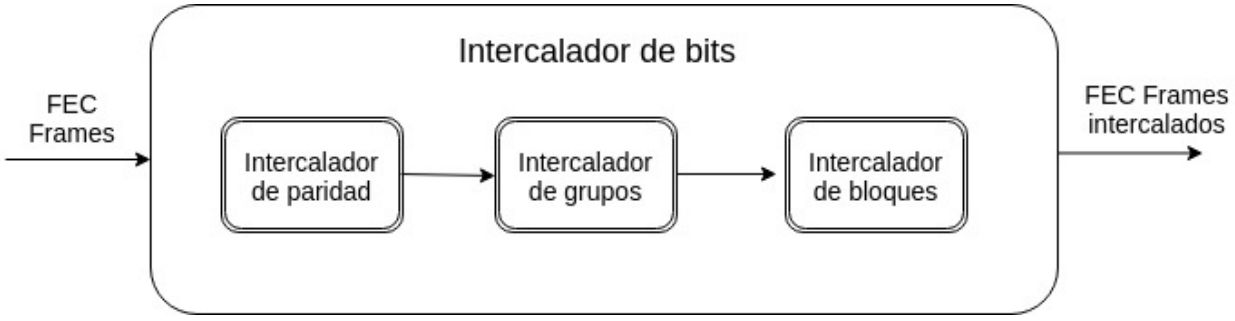

Figura 2.6: Diagrama del intercalador de bits.

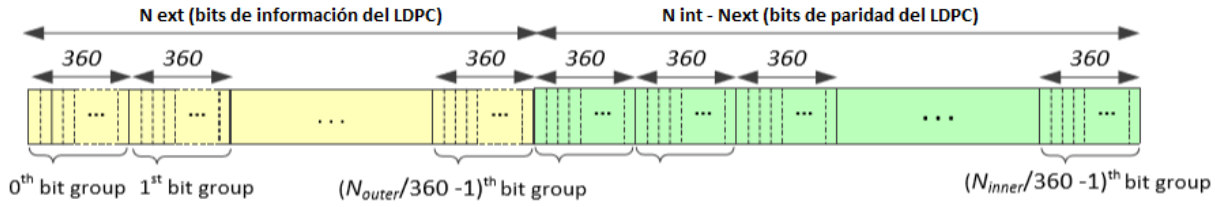

Figura 2.7: Palabra de código LDPC divida en grupos de 360 bits [3].

En el proceso de intercalado de paridad, como su nombre lo dice, solo los bits de paridad son entrelazados [3]. Esto permite una decodificación en paralelo.

Intercalador de grupos

La trama es divida en grupos de 360 bits, como se muestra en la figura 2.7. Posteriormente los grupos son reordenados en la trama según el número de grupo. La posición en la que los grupos son reorganizados depende de la tasa de codificación y del orden de la constelación. Esto permite la optimización del codificador LDPC y de la constelación utilizada [6].

Intercalador de bloques

El intercalador de bloques proporciona la localización final de bits para ser mapeados a símbolos dentro de una constelación [6]. Ya que el modulador toma grupos de bits de acuerdo con la modulación. Estos grupos de bits conforman el parámetro denominado *número de bits por símbolo*.

Existen dos tipos de intercalador de bloque: el tipo A y el tipo B, pero no se encuentran asociados a los tipos de estructura LDPC que también se le llama tipo A (MET) o tipo B (IRA). El uso de un intercalador de bloque tipo A o el uso del tipo B depende más bien de la tasa de codificación y del orden de la constelación a usarse.

El intercalador de bloque tipo A escribe los bits a la entrada en columnas y para la salida los lee en forma de filas. El tipo B trabaja sobre grupos de bits, escribe los bits a la entrada en forma de fila mientras que a la salida los lee en forma de columna.

En las tablas 2.4 y 2.5 se observa el tipo de intercalador de bloque que debe usarse, según el orden de la constelación y la tasa de codificación. En las tablas, los órdenes de constelación se denotan como 2, 4, 6, 8, 10 y 12, los cuales se asocian a las constelaciones QPSK, 16QAM, 64QAM, 256QAM, 1024QAM y 4096QAM respectivamente.

	2/15	3/15	4/15	5/15	6/15	7/15	8/15	9/15	10/15	11/15	12/15	13/15
2	A	A	A	A	A	A	A	A	A	A	A	A
4	A	A	A	B	A	A	B	B	A	A	A	A
6	A	A	A	A	A	B	A	B	B	A	A	B
8	A	A	A	B	B	B	B	A	B	B	A	B
10	A	A	A	B	A	B	A	B	B	B	A	A
12	A	A	A	A	A	B	A	A	A	A	A	A

Cuadro 2.4: Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 64800 bits.

	2/15	3/15	4/15	5/15	6/15	7/15	8/15	9/15	10/15	11/15	12/15	13/15
2	A	A	A	A	B	B	A	B	A	A	A	A
4	A	A	A	A	B	B	A	B	A	B	A	B
6	A	A	A	A	B	B	A	B	A	A	A	A
8	A	A	A	A	B	A	A	A	A	B	A	A

Cuadro 2.5: Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 16200 bits.

2.3.3. Modulación

En esta etapa los bits de entrada son mapeados a valores de tipo complejo, y situados en el plano IQ para formar las constelaciones QAM.

Se puede seleccionar de entre seis órdenes de modulación: QPSK uniforme y cinco tamaños de constelación no uniforme (NUC, Non-Uniform Constellation): 16-QAM, 64-QAM, 256-QAM, 1024-QAM y 4096-QAM [4]. El uso de constelaciones no uniformes reduce la SNR requerida, comparado con las constelaciones uniformes [6].

Cabe resaltar que para un mismo orden de constelación los valores complejos situados en el plano IQ varían según la tasa de codificación LDPC [3]. Esto se puede observar en la figura 2.8 y en la figura 2.9, donde se muestra la constelación 16-QAM para dos diferentes tasa de codificación. Lo anterior no sucede con la modulación QPSK ya que se utiliza la misma constelación para todas las tasas de codificación existentes.

Sin embargo, la constelación no varía con la longitud del código LDPC (es decir, se usa la misma constelación para los códigos LDPC de longitud 64800 y 16200 bits). A pesar de lo anterior mencionado y con el objetivo de reducir la complejidad en la implementación de este bloque no es obligatoria la implementación de todas las combinaciones de modulaciones y tasa de codificación [3].

Es importante hacer notar que el bloque BICM provee una eficiencia espectral muy alta. El modo más robusto es el QPSK con tasa de codificación LDPC de 2/15 y el modo de mayor capacidad es el 4096QAM con tasa de codificación de 13/15. La ganancia que se obtiene comparada con el sistema europeo DVB-T2 (Digital Video

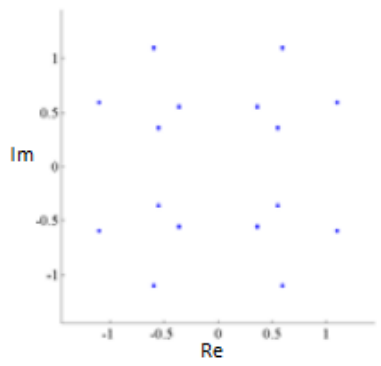

Figura 2.8: Constelación 16-QAM para una tasa de 3/15 [3].

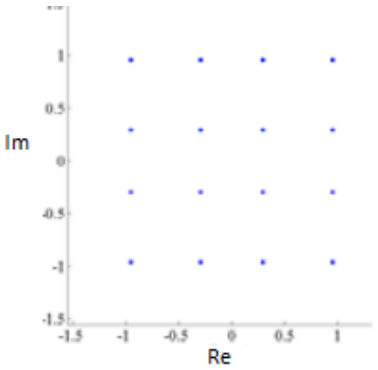

Figura 2.9: Constelación 16-QAM para una tasa de 12/15 [3].

Broadcasting – Terrestrial 2) es de 1 dB en algunos casos, en un canal de 6MHz [6].

2.4. Multiplexación en potencia

La multiplexación en nivel de potencia o LDM llamada así por sus siglas en inglés, Layered Division Multiplexing, es una tecnología que realiza la superposición de constelaciones con el fin de combinar dos flujos de datos (dos PLPs) con diferentes niveles de potencia, en un mismo canal de RF [4]. Por lo tanto los flujos de datos tienen modulación independiente y diferente codificación de canal.

Es decir, en la multiplexación LDM los diferentes contenidos de cada PLP que han pasado por diferentes procesos de entrelazado, codificación y modulación (BICM), son combinados con diferentes niveles de potencia.

LDM es una multiplexación no ortogonal basada en la superposición de espectro. A diferencia de otras técnicas como FDM o TDM, LDM utiliza el 100 % de espectro el 100 % de tiempo disponible para transmitir señales multicapa [7].

Al combinar los dos flujos de datos con diferentes niveles de potencia surge una señal que se compone de dos capas. A estas dos capas se les llama capa central o capa CL (Core Layer) y capa mejorada o capa EL (Enhanced Layer) [4].

Cuando es transmitida la señal multicapa, la capa central es decodificada directamente tratando la señal mejorada como ruido adicional. Para la decodificación de la señal EL, la capa CL tiene que ser cancelada con anterioridad [7].

Comparado con otras técnicas, LDM es mucho más eficiente a la hora de transmitir diferentes servicios, como fijo y móvil, en el mismo canal de RF. La superposición de espectro permite el uso flexible del canal de RF, permitiendo al radiodifusor transmitir simultáneamente diferentes servicios con robustez diferenciada [7].

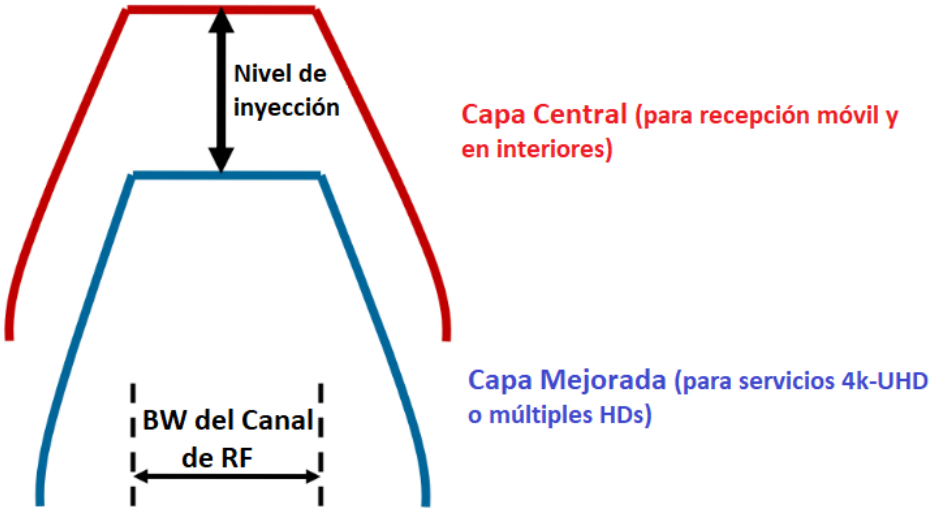

Figura 2.10: Señal de dos capas obtenida después del módulo de LDM.

La señal llamada CL es la que concentra la mayor potencia de la señal, aproximadamente el 72 % de la potencia en un caso típico, es una señal muy robusta pero de baja capacidad. La señal EL es usualmente configurada para entregar servicios de 4K-UHD o HDTV, los cuales requieren de una capacidad alta y de un SNR mayor, estos servicios van dirigidos a receptores fijos [7]. En la figura 2.10 se observa la señal de dos capas, obtenida después del bloque de LDM.

A continuación se muestran dos configuraciones posibles, una para cada capa:

- Señal CL: Modulación QPSK con una tasa de código de 4/15. Es la señal más robusta. Se obtienen 2.7 Mbps con un umbral de C/N de -2.9 dB.
- Señal EL: Modulación 64QAM NUC con tasa de codificación de 10/15. Es posible obtener 20.5 Mbps con un umbral de C/N de 12.9 dB [7].

Ambas señales comparten ciertos parámetros como tamaño de la FFT, patrones de portadoras piloto, longitud intervalo de guarda, entrelazado de tiempo y frecuencia, preámbulo o señal de bootsrap (señal para la sincronización con el receptor), como se observa en el diagrama de la figura 2.2. Después del bloque de LDM se procesa un solo flujo de datos.

En la figura 2.11 se observa el módulo o bloque de LDM. Un controlador de nivel es usado para reducir el nivel de potencia de la capa EL, después de combinar las señales, la potencia total es normalizada. Cabe resaltar que variando el control de nivel de inyección también se puede cambiar la robustez de la señal. El controlador

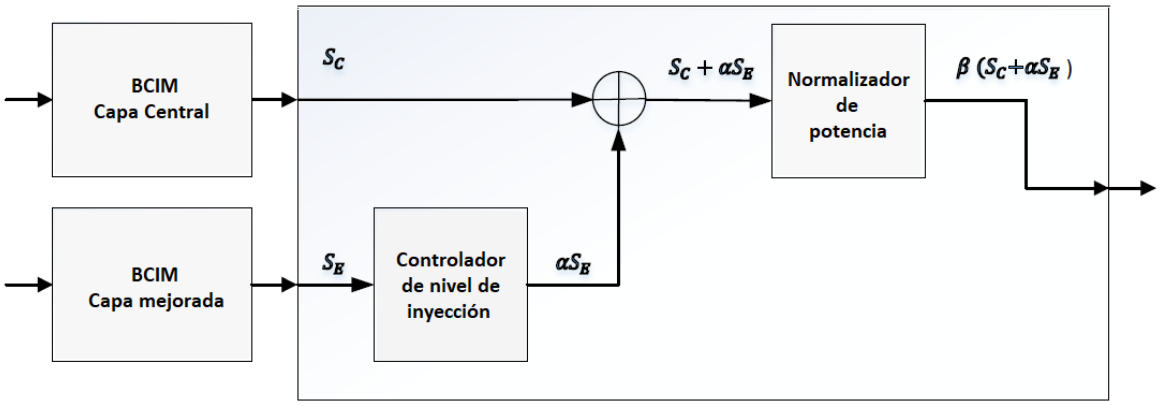

Figura 2.11: Bloque LDM para dos capas [3].

tiene niveles relativos respecto a la capa CL, seleccionables de entre 3 a 10 dB con incrementos de 0.5 dB [4].

2.5. Entrelazado y framing

En la figura 2.12 se muestran los sub-bloques que componen a este bloque. Como se observa, consta de tres partes: intercalado o entrelazado en tiempo, framing e intercalado o entrelazado en frecuencia.

Intercalado en tiempo

Los datos de entrada al bloque de intercalado en tiempo son las celdas provenientes después de la modulación dentro del BICM. Los datos a la salida de este bloque son igualmente celdas pero intercaladas en tiempo. Existen tres modos de intercalado en tiempo: sin intercalado en tiempo, intercalado convolucional o intercalado híbrido [3].

Un intercalador convolucional consiste en un conjunto de registros de desplazamiento, cada uno con un retraso fijo. El conjunto de registros sigue el principio FIFO (First In, First Out). El intercalado híbrido consiste en un intercalador de bloques y en un intercalador convolucional. En un intercalador de bloques los datos son guardados en forma de filas y después enviados por columna. Un intercalado de tipo convolucional presenta un mejor tiempo de retardo y requiere menor memoria que cualquier otro tipo de intercalado [14].

El modo de intercalado en tiempo que debe utilizarse depende del número de PLPs a transmitir. En caso de que se transmita un solo PLP se debe utilizar el modo

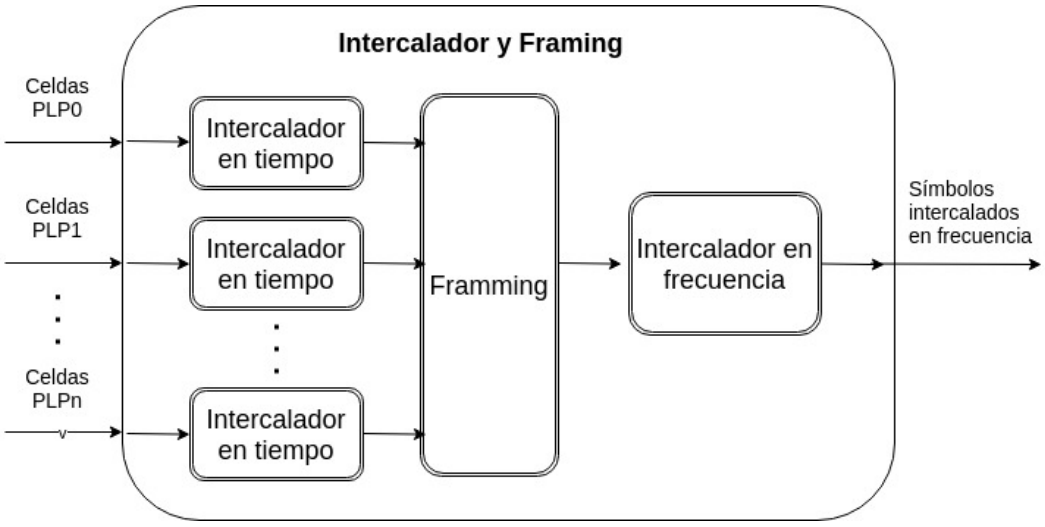

Figura 2.12: Diagrama del bloque de intercalado y framing.

convolucional. Cabe subrayar que cuando se utiliza el bloque LDM de dos capas (en cada capa un PLP) se está dentro del caso en el que se envía un solo PLP, ya que los dos flujos de datos han sido combinados para ser procesados como un único PLP [4]. Por lo tanto después del bloque LDM, el intercalado en tiempo será de tipo convolucional.

Framing

Un trama de ATSC 3.0 está compuesta por tres partes:

1. Una señal llamada bootstrap situada al principio de cada frame. Esta señal se explica en el apartado 2.6.

2. Una señal llamada preámbulo localizada inmediatamente después de la señal bootstrap.

3. Uno o más sub-frames situados después del preámbulo.

El preámbulo contiene las señales de control L1- Basic y L1-Detail, las cuales a su vez contienen la longitud de los datos de la carga útil. Cada sub-frame tiene parámetros configurados tales como: tamaño FFT, intervalo de guarda, patrón de portadoras piloto, el número de sub-portadoras piloto no utilizadas y el número de símbolos OFDM [4]. La duración máxima de los frames son de $5s$ y la mínima de $50ms$.

A la salida de este bloque se obtienen símbolos OFDM.

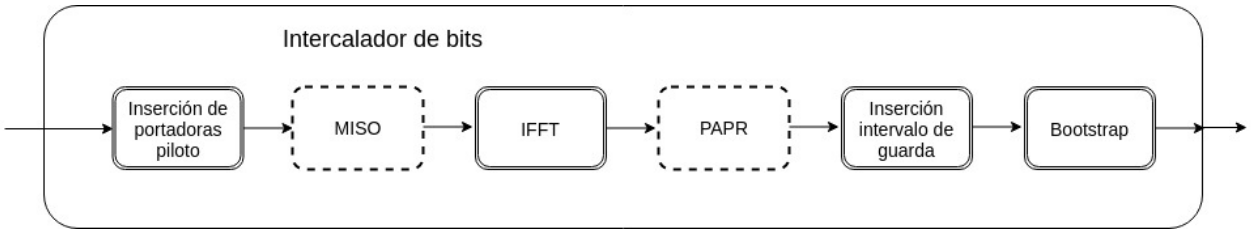

Figura 2.13: Diagrama del bloque generación de forma de onda.

Intercalado en frecuencia

El intercalado en frecuencia se lleva a cabo con el fin de contrarrestar los errores ráfaga que puedan presentarse en el dominio de la frecuencia.

Este bloque opera sobre símbolos OFDM y también puede aplicarse a los símbolos del preámbulo.

2.6. Generación de la forma de onda

En el diagrama mostrado en la figura 2.13 se observa la composición de esta etapa del sistema. Los bloques que se encuentran dibujados con línea punteada son bloques cuyo uso no es obligatorio.

Inserción de portadoras piloto

ATSC 3.0 emplea portadoras piloto como elementos capaces de ser usados para la sincronización en tiempo, frecuencia y de trama. También pueden ser usadas para llevar a cabo la estimación de canal, para identificar el modo de transmisión y para supervisar el ruido de fase. Existen 16 patrones de portadoras piloto dispersas SP (Scattered Pilots) [4].

En cuanto a las portadoras piloto continuas CP (Continual Pilots) se utilizan como mínimo 45, cuando el tamaño de la FFT es de 8k y se duplican conforme se duplica el tamaño de FFT [4]. Es decir para 16k el mínimo sería de 90 mientras que para 32k el mínimo de portadoras piloto continuas es de 180.

Multiple-Input Single Output (MISO)

OFDM puede contrarrestar el efecto del multi-trayecto de una señal con el uso del intervalo de guarda. Sin embargo para las situaciones en las que se reciben dos ecos con el mismo nivel de potencia fuera de la duración del intervalo de guarda se emplea este bloque.

ATSC 3.0 adoptó una técnica de pre-distorsión MISO (Multiple Input Single Output) conocida como TDCFS (Transmit Diversity Code Filter Set), que consiste en transmitir múltiples señales de diferentes transmisores de una red SFN, cada uno de los transmisores tiene un bloque TDCFS. El resultado es que se obtiene una decorrelación de las múltiples señales, todo esto con el fin de minimizar interferencias destructivas.

IFFT

Existen definidos tres tamaños de FFT: 8k, 16k y 32k y tres anchos de banda: 6, 7 y 8MHz.

Peak-to-Average Power Ratio (PAPR)

Este es un bloque opcional en el sistema, el cual se encarga de reducir el PAPR [3]. El PAPR es un parámetro que mide la relación entre la potencia máxima y la potencia promedio de una señal. En un sistema OFDM pueden existir valores pico altos después del bloque IFFT, es decir estos picos se encuentran en el dominio del tiempo.

Cuando una señal OFDM es transformada al dominio del tiempo, la señal resultante es la suma de todas las sub-portadoras y cuando todas las sub-portadoras se suman en fase, el resultado es un pico con un nivel de potencia mayor que la potencia promedio [15].

Tener un PAPR alto causa que la señal se degrade puesto que al llegar al amplificador analógico éste se ve obligado a trabajar en la región no lineal, causando distorsiones en la señal y consumiendo mayor potencia.

Inserción del intervalo de guarda

Existen 12 duraciones diferentes para el intervalo de guarda, las cuales van desde $27.78\mu s$ a los $703.70\mu s$ en un canal de 6MHz. Cabe aclarar, que el uso de los diferentes intervalos de guarda depende también del tamaño de la FFT. Expertos en campo de ATSC sugieren que el echo natural máximo es de $104\mu s$, fuera de una red SFN [4].

Bootstrap

Al comienzo de cada frame se añade una señal llamada bootstrap. Es una señal muy robusta puesto que tiene que ser captada por todos los receptores. Permite la sincronización con el receptor.

Puede ser recibida en canales muy ruidosos con un SNR menor a $-6dB$. En un canal AWGN el umbral es de $-9.5dB$ [4].

La señal de bootstrap consta de una serie de símbolos OFDM. El primer símbolo OFDM está destinado a la sincronización, los símbolos subsecuentes contienen información adicional.

ATSC 3.0 en su primer versión tiene un bootstrap que consta de 4 símbolos, como ya se mencionó antes, el primer símbolo es de sincronización, aunque también dentro del mismo es indicada la versión de ATSC 3.0 utilizada.

El segundo símbolo OFDM está destinado a los servicios de alerta de emergencia conocidos como EAS por sus siglas en inglés (Emergency Alert Service). El tercer símbolo contiene la información sobre la tasa de muestreo del frame. Por último, en el cuarto símbolo se indica la estructura del preámbulo, y los parámetros necesarios para iniciar la demodulación y decodificación del mismo [4].

Cabe destacar que inmediatamente después del bootstrap se encuentra el preámbulo. El preámbulo junto con el bootstrap conforman la capa 1 del modelo OSI, es decir contienen toda la información sobre la capa física del sistema [3]. Su función principal es la de permitir el acceso a la carga útil que lleva el PLP.

Capítulo 3

Construcción de módulos en GNU Radio

Como se mencionó en el capítulo 1, la implementación de los módulos se lleva acabo en el software GNU Radio, el cual permite crear sistemas de radio definidos por software. Debido a que es un software libre y se tiene acceso al código fuente es posible crear módulos con nuevas funcionalidades. En este capítulo se explicará la estructura de un módulo y el proceso de construcción de un nuevo módulo. Un módulo en GNU Radio es un conjunto de programas escritos en C++ o en Python. Un programa se denomina bloque, ya que GNU Radio trabaja con una interfaz de programación basada en bloques.

3.1. Módulos Out-Of-Tree de GNU Radio

Cuando se instala GNU Radio se cuenta con una serie de bloques que van desde analizadores de espectro hasta bloques para la construcción de sistemas de TV digital de algunos estándares como ATSC o DVB-T2. Es posible ampliar la variedad de bloques que existen en GNU Radio. Cada uno de los usuarios de este software puede hacerlo programando en C++ o en Python.

Los módulos que se añaden a GNU Radio de manera externa se denominan módulos OOT (Out-Of-Tree). Un módulo OOT en GNU Radio es, por tanto, un conjunto de bloques que se encuentran fuera del código fuente de GNU Radio. Es decir, un módulo OOT es creado con las funciones y necesidades propias, pero tiene funcionalidad con los módulos propios de GNU Radio [16].

La instalación de GNU Radio puede llevarse a cabo mediante Pybombs (Python Bundles Overlay Managed Build System) el cual está principalmente diseñado para

los usuarios de GNU Radio que gusten de trabajar con módulos OOT puesto que facilita las cosas para los desarrolladores de estos módulos [17]. El procedimiento de instalación mediante esta herramienta se encuentra en el apéndice A de esta tesis.

También es posible instalar GNU Radio mediante el siguiente comando, el cual a la hora de instalar los bloques se tienen que hacer pasos adicionales para poder usarlos, pero ambas instalaciones son correctas.

```
$ sudo apt install gnuradio
```

Con la finalidad de implementar módulos OOT en GNU Radio, es importante conocer algunas características de los módulos en general, así como la terminología usada y la clasificación de los mismos.

3.2. Tipos de bloques en GNU Radio

Es necesario mencionar que un *item* puede ser una muestra, un bit, un byte, un símbolo o un paquete de datos. Cada uno de los bloques procesa diferentes items. A continuación se enlistan los tipos de bloques que existen:

- Bloques síncronos (1:1): este tipo de bloque produce un item por cada item que entra.
- Bloques de decimación (N:1): el número de items a la entrada es un múltiplo entero del número de items a la salida.
- Bloques de interpolación (1:M): el número de items a la salida es un múltiplo entero del número de items a la entrada.
- Bloques de tipo general (N:M): Se trata de un bloque que a la entrada tiene N items y a la salida entrega M items. Dónde M y N son definidas por el desarrollador. [18].
- Bloques fuente/sumidero: estos bloques generan o consumen items.
- Bloques jerárquicos: son bloques construidos a partir de otros bloques [19].

3.3. Convención de nombres de los bloques en GNU Radio

El nombre del bloque debe especificar su función principal así como el tipo de dato a la entrada y a la salida [18]. El formato en general es el siguiente:

$< name > \_ < input >< ouput >$

Los tipos de datos que existen son:

- c: complejo (flotantes de 32 bits para ambas componentes I y Q).
- f: flotante (flotante de 32 bits, precisión simple con el estándar IEEE 754).
- i: entero (entero signado de 32 bits).
- s: entero corto (entero signado de 16 bits).
- b: byte (entero signado de 8 bits).

3.4. La herramienta gr_modtool

GNU Radio cuenta con una herramienta denominada *gr_modtool* cuya función es facilitar la creación de módulos, editando archivos automáticamente, entre otras cosas, con la finalidad de evitar repetir trabajo a la hora de crear un módulo. Esta herramienta se instala por defecto [16]. La creación de un nuevo módulo implica la repetición de código en cada módulo, esta herramienta evita que se tenga que escribir el mismo código una y otra vez en cada creación de un nuevo módulo. De esta forma solo es necesario programar la funcionalidad del módulo en sí mismo.

En caso de que la herramienta no este instalada por defecto, se puede instalar ejecutando el siguiente comando en la consola de Linux dentro del directorio donde se instaló GNU Radio:

```
$ git clone https://github.com/mbant/gr-modtool.git
```

La herramienta gr_modtool cuenta con los siguientes comandos:

disable	Deshabilita un bloque.
info	Proporciona información sobre un bloque.
remove	Elimina un bloque.
makexml	Construye automáticamente la interfaz visual de un bloque.
add	Añade un nuevo bloque a un módulo OOT.
newmod	Crea un nuevo módulo OOT.

3.5. Estructura de un módulo de GNU Radio

Un módulo en GNU Radio consta de los siguientes directorios: apps, cmake, docs, examples, grc, include, lib, python, swig y un archivo llamado CMakeLists.txt. Todo esto es creado automáticamente cuando se utiliza la herramienta *gr_modtool*.

La carpeta llamada *apps* incluye los ejecutables que haya creado el desarrollador del módulo. Los ejecutables pueden ser archivos con extensión grc propios de GNU Radio o archivos con extensión py propios de Python. Estos ejecutables se crean con la finalidad de tener ejemplos del uso del módulo o que aporten funcionalidad extra necesaria o útil para el módulo.

Con respecto a la carpeta *cmake* cabe destacar primero que *CMake* es una herramienta que se emplea para la construcción de proyectos multiplataforma. GNU Radio hace uso de esta herramienta para la construcción de los módulos. En cada carpeta dentro del módulo se encuentra un archivo llamado CMakeLists que básicamente contiene instrucciones para que CMake encuentre las librerías de GNU Radio y las librerías del módulo a crear [16].

En la carpeta *docs* se encuentran instrucciones sobre como extraer documentación sobre los archivos de C++ y de Python los cuales contienen la implementación de los bloques. GNU Radio utiliza *Doxygen* como estándar para la generación de su documentación [16]. Doxygen es una herramienta de programación para documentar software de manera fácil.

La carpeta *examples*, como su nombre lo dice, contiene ejemplos sobre el uso de los bloques que existen dentro del módulo. Es decir, esta carpeta puede contener archivos propios de GNU Radio para ejemplificar el uso de los bloques creados.

La carpeta *grc* contiene código XML, GNU Radio utiliza XML para crear la interfaz entre el código y el usuario, por lo tanto, para cada bloque debe existir una archivo XML.

La carpeta *include* se utiliza solo para bloques programados en C++. Contiene algunas librerías que se exportan desde los archivos de C++, también es posible generar nuestras propias librerías y ponerlas dentro de esta carpeta, con la finalidad de facilitar la implementación de los bloques.

La carpeta *lib* incluye los códigos de los bloques programados en C++. Mientras que en la carpeta *python* se guardan los bloques hechos en este lenguaje de programación.

Referente a la carpeta *swig*, cabe destacar que GNU Radio utiliza una herramienta llamada SWIG para conectar los programas escritos en C++ con Python. GNU Radio utiliza Python para generar el flujo de datos entre los bloques, el procesamiento de

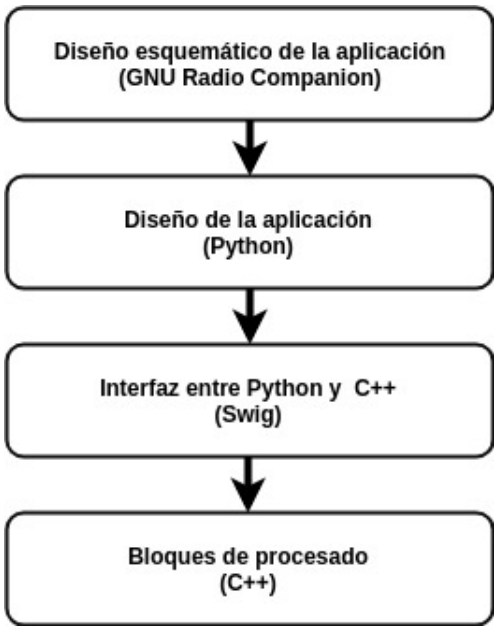

Figura 3.1: Arquitectura de GNU Radio [20].

datos dentro de cada bloque puede llevarse a cabo mediante C++ o mediante Python. Sí se requiere procesar grandes cantidades de datos, GNU Radio recomienda el uso de C++ para la implementación de los bloques. Es decir, la elección del lenguaje de programación para la implementación de los bloques depende del objetivo del bloque [19].

En la figura 3.1 se aprecia, en la parte superior, la primer capa de GNU Radio, la cual consiste en hacer el programa mediante bloques, una vez que se ejecuta el programa en GNU Radio, se genera un script de python, el cual contiene el flujo de datos entre los bloques, es decir todos los bloques utilizados se mandan a llamar desde python.

La siguiente capa es una interfaz para que python pueda llamar a bloques construidos en el lenguaje C++, la ultima capa de GNU Radio son los bloques desarrollados en C++, en esta capa se lleva a cabo el procesado de la información.

En caso de que el bloque haya sido desarrollado directamente en python no es necesaria la herramienta Swig. Cabe resaltar que la mayoría de bloques son programados en C++ debido a la eficiencia de este lenguaje de programación, ya que al ser un lenguaje compilado, es más rápida su ejecución, a diferencia de python que es un lenguaje interpretado.

3.6. Módulo desarrollado en la tesis

Como se explicó anteriormente, en GNU Radio un módulo es un grupo de bloques con un tema en común, por lo tanto existen módulos que van desde temas como operadores matemáticos hasta estándares de TV como ATSC 1.0 o DVB-T. Con el objetivo de agrupar los bloques que se crearon en esta tesis, se construyó un módulo en GNU Radio denominado ATSC 3.0.

Como se mencionó en la sección anterior, GNU Radio cuenta con una herramienta llamada *gr_modtool* que facilita la creación de nuevos módulos o módulos OOT, a continuación se muestran los comandos ingresados en la terminal de Linux [21] para la creación de un módulo.

```
$ gr_modtool newmod
Name of the new module: atsc3
```

Después del uso de este comando se obtuvo una carpeta llamada gr-atsc3, con la estructura y las carpetas descritas en la sección anterior (apps, cmake, grc, include, lib, python, swig, etc). El directorio llamado atsc3 es el que concentrará todos los bloques creados. En las carpetas *include* y *lib* se encuentran los bloques programados en C++, mientras que en la carpeta denominada *python* se encuentran los bloques programados en python.

Después de la creación de un nuevo módulo en GNU Radio, es necesario añadir bloques, esto se puede hacer como en el ejemplo siguiente:

```
$ gr_modtool add
GNU Radio module name identified: atsc3
Enter block type: general
Language (python/cpp): python
Language: Python
Enter name of block/code (without module name prefix): ldm_cc
Block/code identifier: ldm_cc
Enter valid argument list, including default arguments:
Add Python QA code? [Y/n] y
Adding file 'python/ldm_cc.py'...
Adding file 'python/qa_ldm_cc.py'...
Editing python/CMakeLists.txt...
Adding file 'grc/atsc32_ldm_cc.xml'...
Editing grc/CMakeLists.txt...
```

Nombre del bloque	Parámetros que recibe	Lenguaje	Tipo de dato a la entrada	Tipo de dato a la salida
Codificador LDPC	tasa de codificación, longitud de código	C++	byte	byte
Intercalador de bits	longitud de código	Python	byte	byte
Modulador	constelación, tasa de codificación	Python	byte	complejo
Multiplexor en potencia	Factor multiplicador	Python	complejo	complejo

Cuadro 3.1: Características de los bloques desarrollados en la tesis

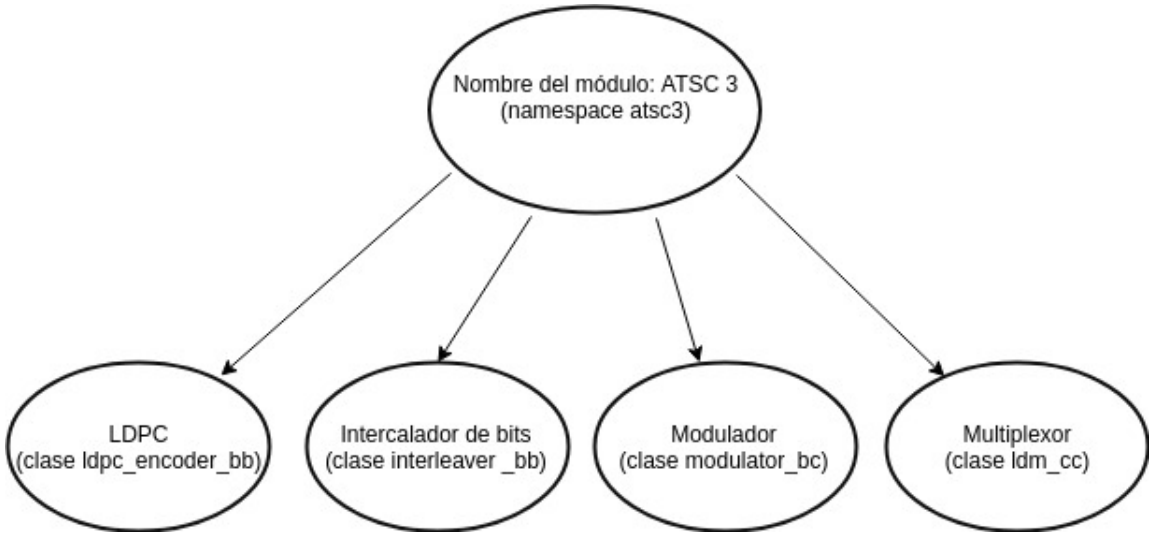

Figura 3.2: Diagrama del módulo ATSC 3.0 programado en GNU Radio

Cuando se añade un bloque, la herramienta pregunta por el tipo de bloque, el lenguaje de programación, el nombre del bloque, los argumentos y si se quiere añadir un archivo llamado *qa test* con el fin de verificar el correcto funcionamiento del bloque.

En total se añadieron 4 bloques al módulo ATSC 3.0 con las características del cuadro 3.1.

En el diagrama 3.2 se puede ver la estructura del módulo de ATSC 3.0. La programación en GNU Radio es orientada a objetos, en realidad el módulo denominado *atsc3* es un namespace, dentro de este namespace viven las clases asociadas a los bloques programados. El nombre de la clase se debe a la nomenclatura explicada en la sección 3.3.

Cabe aclarar que no todos los modos de operación fueron programados. En los siguientes capítulos se explica de manera detallada los modos programados para cada bloque.

Capítulo 4

Entrelazado y codificación modulada

Como se describió en el capitulo 2, sección 2.3, el bloque de entrelazado y codificación modulada o bloque BICM se compone a su vez de tres sub-bloques: un codificador LDPC, un intercalador y un modulador.

En la figura 4.1 se muestra el diagrama del bloque BICM implementado en esta tesis. La etapa FEC está compuesta solo por el codificador LDPC cuyo uso es obligatorio, no se implementa el codificador externo (que puede ser un CRC o un BCH) debido a que su uso no es obligatorio dentro del estándar.

A continuación, en las siguientes secciones del capítulo, se describen los fundamentos teóricos de cada bloque y su implementación.

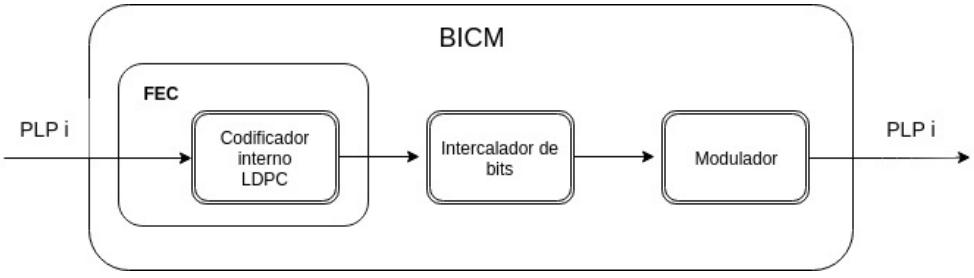

Figura 4.1: Diagrama de la implementación del bloque de entrelazado y codificación modulada.

4.1. Codificador LDPC

Este esquema de codificación debe su nombre a la característica que tiene su matriz de chequeo de paridad, ésta se compone de pocos unos en comparación con la cantidad de ceros dentro de la matriz [22], este tipo de matrices se denominan de *baja densidad*.

Los códigos LDPC (Low-density parity-check) fueron formalmente descritos en 1960 por Robert Gray Gallager y han sido últimamente utilizados debido al desarrollo de técnicas eficientes para su codificación y decodificación.

En muchos casos el desempeño de los códigos LDPC ha superado el desempeño de otros esquemas de codificación. Por esta razón los códigos LDPC se han adoptado en importantes estándares de telecomunicaciones como, el estándar ETSI DVB-S2, el IEEE 802.11n, el IEEE 802.16e y el estándar IEEE 802.20 [23]. Actualmente estos códigos son la técnica de codificación más prometedora, al acercarse a la capacidad del canal, es decir al límite propuesto por Shannon [24].

Los códigos LDPC son una clase de códigos lineales de bloque. Codificar y decodificar información es más sencillo cuando el código es lineal, de igual forma, por su estructura algebraica, es más sencillo describir un código lineal, en comparación cuando el código no es lineal [25]. Por esta razón los códigos lineales son los más estudiados.

4.1.1. Conceptos de códigos lineales de bloque

Definición de código lineal de bloque

El alfabeto de un código lineal es un campo finito denotado como F_q, donde q es el número de elementos del campo. Se define sobre el campo finito F_q un espacio vectorial formado por una secuencia ordenada de n elementos (n-tupla) denotado como F_q^n.

Se define también un un código llamado C, que es subconjunto del espacio vectorial F_q^n [25]. Dicho código llamado C de dimensión k es lineal si es un subespacio de F_q^n. De esta forma se puede decir que C es un código lineal (n, k) sobre el campo F_q. El código lineal C está formado por q^k palabras de código.

Es decir para cualquier combinación lineal de palabras de código, el resultado también es una palabra de código.

Un código lineal de bloque, binario de dimensión k y de longitud n denotado como $C(n, k)$ es una transformación como se describe en la ecuación 4.1:

$$C : F_2^k \Rightarrow F_2^n \tag{4.1}$$

Esta relación es unívoca y cada k-tupla (información) se asocia a una n-tupla (palabra de código).

Cuando la dimensión k de un código es grande y también lo es la longitud del código n, la codificación se vuelve muy compleja, sin embargo, la propiedad de linealidad reduce significativamente la complejidad del proceso de codificación.

Peso y distancia de Hamming

Un concepto importante es el peso de Hamming, el cual es el número de elementos diferentes de cero que contiene una palabra de código o vector. Otro concepto de importancia, es la distancia de Hamming, la cual se define como el número de bits diferentes entre dos vectores. Por lo tanto, la distancia mínima de un código es la distancia mínima de Hamming encontrada entre dos palabras cualesquiera del código [23].

Sí el código es lineal se cumple que la suma o resta de cualquiera de dos de sus palabras de código resulta en otra palabra de código. Por tanto también se debe cumplir que la distancia mínima del código y el peso mínimo del código sean iguales.

Códigos sistemáticos

Un código es sistemático sí las palabras de código contienen los vectores de información a los cuales se encuentran asociados.

Un caso de un código sistemático es aquel que añade al vector de información bits de redundancia o bits de paridad.

Matriz Generadora

Sí $C(n, k)$ es un código lineal de bloque, deben existir k palabras de código, linealmente independientes entre sí que formen una base del código: $\{g0, g1, ..., gk-1\}$. En consecuencia, cada palabra de código puede ser expresada como una combinación lineal de la base:

$$C = u_0 g_0 + u_1 g_1 + u_2 g_2 + ... + u_{k-1} g_{k-1} \tag{4.2}$$

Donde los coeficientes son el vector de información: $u = \{u0, u1, ..., uk - 1\}$. De forma matricial se tiene que:

$$C = u * G \tag{4.3}$$

Donde la matriz G se conoce como la *matriz generadora* del código. Es decir, la matriz generadora llamada G de un código $C(n, k)$ es cualquier matriz de tamaño k x n cuyas filas o renglones forman una base para C.

Las dos formas más comunes de representar a un código lineal es mediante una matriz generadora o mediante una matriz de paridad (concepto que se explica en el siguiente apartado) [23].

Matriz de paridad

Existe una matriz H que se relaciona con la matriz generadora G. Esta matriz es denominada *matriz de paridad* cuyo tamaño es de $(n - k)$ x n y debe satisfacer la ecuación 4.4 [25].

$$C * H = 0 \tag{4.4}$$

La relación entre la matriz generadora y la matriz de paridad de un código lineal de bloque es:

$$G * H^T = 0 \tag{4.5}$$

Si el código es sistemático, la matriz generadora puede ser representada como en la ecuación 4.6.

$$G = [I|P] \tag{4.6}$$

Dónde I representa una matriz identidad de dimensiones k x k y P es una matriz de tamaño k x r la cual representa el conjunto de ecuaciones de verificación de paridad [23].

Por lo tanto la matriz de paridad H puede ser expresada como en la ecuación 4.7.

$$H = [P^T|I] \tag{4.7}$$

4.1.2. Representaciones de un código LDPC

Es posible representar un código LDPC de dos formas: la primera es mediante una matriz y la segunda es mediante una gráfica [22].

Representación matricial

Para un código LDPC (8,4) se tiene una matriz de chequeo de paridad de 4 filas y 8 columnas como la que se muestra en la figura 4.2.

$$\mathbf{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Figura 4.2: Matriz H para un código (8,4) [22].

Para que una matriz de paridad H sea considerada como una *matriz de baja densidad* se debe cumplir que el peso de columna denotado como w_c sea mucho menor que la longitud del código n. Y que el peso de fila w_f sea mucho menor que la longitud del bloque de información k [22]. Es decir una matriz de paridad de baja densidad cumple con las condiciones siguientes:

$$w_c \ll n, w_f \ll k \tag{4.8}$$

Cabe aclarar que la matriz de la figura 4.2 no es estrictamente una matriz de baja densidad ya que no cumple con la condición anterior, puesto que en esta matriz se tienen los siguientes parámetros: w_c=2, n=8, w_f=4 y k=4. Las matrices de códigos LDPC suelen ser de mayor dimensión, por ejemplo, en el caso del estándar de ATSC 3.0 se tienen matrices de dimensiones de hasta 150 filas x 20 columnas. En la figura 4.6 se observa un ejemplo de una matriz de baja densidad dónde los bits no existentes, representan ceros en la matriz.

Representación gráfica

En 1981 Tanner introdujo una representación gráfica para los códigos LDPC. En la figura 4.3 se ve la gráfica de Tanner para el código LDPC descrito anteriormente (Figura 4.2).

La representación de Tanner consiste en la existencia de dos tipos de nodos, y solo es posible unir nodos de diferentes tipos. Los dos tipos de nodo existentes son los nodos de tipo variable y los nodos de tipo chequeo, denotados como v-nodos y c-nodos respectivamente [22].

La construcción de la gráfica se hace a partir del número de bits de paridad, para el código (8,4) son 4 y constituyen los nodos de chequeo (c-nodos). Los v-nodos son el número de bits de la palabra de código y en este caso son 8. Siguiendo la figura 4.3, un nodo f_i de tipo chequeo se conecta con un nodo c_j si en la matriz de paridad en la posición h_{ij} aparece un bit en 1. Siguiendo esto, a partir de la matriz H se genera

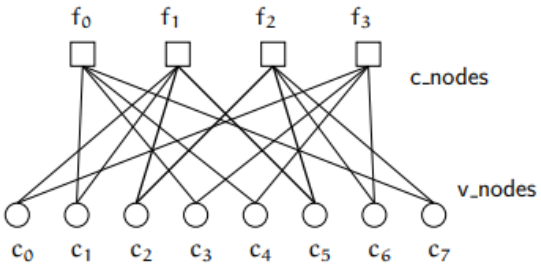

Figura 4.3: Gráfica de Tanner de un código (8,4) [22].

la representación gráfica de la figura 4.3.

4.1.3. Estructuras LDPC del estándar ATSC 3.0

Los códigos LDPC fueron adoptados por este estándar como mecanismo para la protección contra errores, estos códigos son la parte principal de la etapa FEC del sistema.

Los códigos adoptados se caracterizan por ser un tipo de códigos cuasi cíclicos y por lo tanto denominados QC-LDPC (Quasi Cyclic LDPC). Esta característica se debe a que la matriz H de los códigos LDPC de ATSC 3.0 puede ser transformada a su forma cuasi cíclica mediante la permutación de una fila y una columna adecuada [5].

Los códigos cuasi cíclicos son una clase importante de los códigos lineales de bloque debido a que pueden utilizar circuitos muy simples para llevar a cabo la codificación y decodificación [23]. También proporcionan otras ventajas como la decodificación en paralelo y la eficiencia en el uso de memoria. Por esto, otros estándares como DTMB (Digital Terrestrial Multimedia Broadcast) y DVB (Digital Video Broadcasting) han hecho uso de códigos QC-LDPC [5].

Un código cuasi cíclico tiene una matriz circulante como la que se muestra en la figura 4.4. En una matriz circulante cada fila de la matriz es igual a la fila anterior pero desplazada hacia la derecha una posición. Además se debe cumplir que la primera fila sea igual a la última fila, pero desplazada una posición a la izquierda [23].

Además de que los códigos LDPC elegidos para el estándar son de tipo cuasi cíclico se adoptaron dos diferentes estructuras o tipos, debido al algoritmo utilizado para obtener los bits de paridad:

- Tipo IRA (Irregular Repeat Accumulate structure).
- Tipo MET (Multi-Edge Type structure).

$$\begin{bmatrix} \mathbf{H_0} & \mathbf{H_1} \dots \mathbf{H_{p-1}} \\ \mathbf{H_{p-1}} & \mathbf{H_0} \dots \mathbf{H_{p-2}} \\ \vdots & \vdots \quad \ddots \vdots \\ \mathbf{H_1} & \mathbf{H_2} \dots \mathbf{H_0} \end{bmatrix}$$

Figura 4.4: Matriz circulante de un código cuasi cíclico [23].

	Tipo de Estructura	
Tasa de Codificación	N= 64800	N= 16200
2/15	MET	MET
3/15	MET	MET
4/15	MET	MET
5/15	MET	MET
6/15	IRA	IRA
7/15	MET	IRA
8/15	IRA	IRA
9/15	IRA	IRA
10/15	IRA	IRA
11/15	IRA	IRA
12/15	IRA	IRA
13/15	IRA	IRA

Cuadro 4.1: Tipo de estructura LDPC dependiendo de la longitud del código en bits y de la tasa de codificación [3].

Cabe destacar que el tipo MET presenta mejor desempeño a tasas bajas de codificación, mientras que el tipo IRA presenta mejor desempeño a tasas altas de codificación [5], [3]. Cada codificador implementa un algoritmo diferente y es debido a esto la diferencia de desempeño dependiendo la tasa de codificación.

Se adoptaron dos longitudes de código:

- Una longitud llamada corta de 16200 bits.
- Una longitud llamada larga de 64800 bits.

Para cada longitud de código existen las estructuras IRA y MET [3], dependiendo de la tasa, como se muestra a en el cuadro 4.1.

En esta tesis se realiza la implementación del codificador LDPC tipo IRA para ambas longitudes de código (64800 y 16200 bits). El codificador tipo IRA es el más utilizado, esto puede verse nuevamente en el cuadro 4.1. En la siguiente sección se describe a detalle el codificador LDPC de estructura IRA.

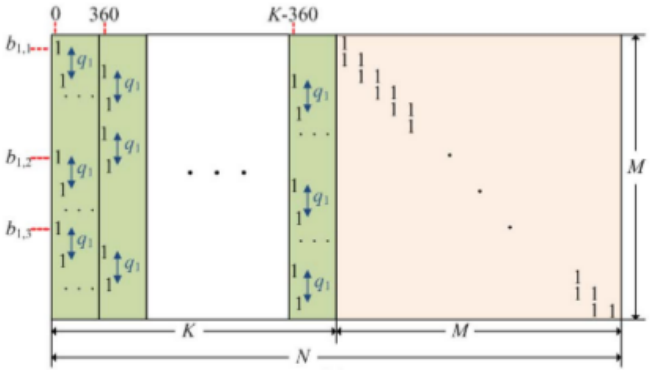

Figura 4.5: Matriz de chequeo de paridad de un código LDPC con estructura IRA [5].

Estructura tipo IRA

Los códigos IRA (Irregular Repeat-Accumulate) aparecieron en el año 2000 como una subclase de códigos LDPC. Los códigos LDPC tipo IRA tienen propiedades similares a los turbo códigos. Aunque de manera general los códigos LDPC presentan un mejor rendimiento, además de que reducen la complejidad de implementación de los turbo códigos[26].

Como su nombre lo dice, un LDPC tipo IRA presenta una matriz de chequeo de paridad irregular. Una matriz H irregular se caracteriza por tener un peso de fila (w_f) diferente al peso de columna (w_c), es decir $w_f \neq w_c$.

Cabe mencionar el concepto de matriz regular. Una matriz H es regular sí el peso de fila es igual al peso columna ($w_f = w_c$). Es decir, un código LDPC regular es aquel cuya matriz de paridad tiene todas las columnas con un grado igual, es decir, la misma cantidad de unos por columnas, lo mismo sucede con las filas [25].

La matriz de paridad H de un código LDPC tipo IRA puede ser dividida en dos partes: una parte de información y la otra parte de paridad como se muestra en la figura 4.5, la información resalta en verde y los bits de paridad resaltan en rosa. Se observa que la parte de paridad de la matriz H tiene una estructura triangular inferior de doble diagonal [27] [5].

Respecto a la parte de información, ésta tiene un desplazamiento denotado como q_1. Las columnas de información se obtienen del desplazamiento de su columna izquierda, exceptuando la columna número 360, 720, hasta la columna k-360. El parámetro q_1 es una constante del código, la cuál depende de el número de bits de paridad y es calculada como en la expresión 4.9.

$$q_1 = \frac{M}{L} \tag{4.9}$$

Tasa de codificación	q_1 (N=64800)	q_1 (N=16200)
6/15	108	27
7/15	-	24
8/15	84	21
9/15	72	18
10/15	60	15
11/15	48	12
12/15	36	9
13/15	24	6

Cuadro 4.2: Constante q_1 según el código a utilizar para la estructura tipo IRA.

En ATSC 3.0 el parámetro L es siempre igual a 360 y M es el número de bits de paridad. Es importante explicar que L es igual a 360 debido a que el algoritmo empleado realiza la codificación en grupos de 360 bits, proceso que será detallado más adelante [3].

En el cuadro 4.2 se tiene el parámetro q_1 para cada longitud de código y tasa de codificación [3], este parámetro será empleado más adelante para el cálculo de los bits de paridad. Estos valores fueron calculados de la tabla 2.1 y de la tabla 2.2 dónde se proporciona la longitud de los bits de paridad añadidos en cada caso de codificación. La estructura IRA se utiliza a partir de una tasa de codificación de 6/15, exceptuando la tasa 7/15 para una longitud de código de 64800.

El documento A/322 que describe la capa física, proporciona una serie de tablas para llevar a cabo el cálculo de los bits de paridad. Estas tablas son la descripción de la matriz de paridad H [28], como se explica a continuación.

Utilizando la representación de Tanner explicada anteriormente, cada fila de la tabla contiene la dirección de los nodos de tipo chequeo (bits de paridad), asociados a 360 nodos de tipo variable (solo bits de información) [27]. Es decir los bits de información son utilizados en grupos de 360 para el cálculo de los bits de paridad [27]. Cada fila dentro de la tabla son las direcciones de los bits de paridad asociados a ese grupo de 360 bits.

Por ejemplo, en la figura 4.6 se observa una matriz de paridad para un código de longitud corta (16200 bits) y una tasa de codificación de 13/15 [3]. De la tabla 2.2 se tiene que el número de bits de información para este caso es igual a 14,040, sí se agrupan los bits de información en grupos de 360 se tienen 39 grupos. Sí para cada grupo de 360 bits se tiene una fila con las direcciones de bits de paridad asociados a ese grupo, la tabla de direcciones debe tener 39 filas justo como se observa en la figura 4.6. Es decir todas las tablas de direcciones de bits de paridad (matrices H) tienen un número de filas igual a $K/360$ dónde K es la carga útil [28].

```
 1 71 334 645 779 786 1124 1131 1267 1379 1554 1766 1798 1939
 2 6 183 364 506 512 922 972 981 1039 1121 1537 1840 2111
 3 6 71 153 204 253 268 781 799 873 1118 1194 1661 2036
 4 6 247 353 581 921 940 1108 1146 1208 1268 1511 1527 1671
 5 6 37 466 548 747 1142 1203 1271 1512 1516 1837 1904 2125
 6 6 171 863 953 1025 1244 1378 1396 1723 1783 1816 1914 2121
 7 1268 1360 1647 1769
 8 6 458 1231 1414
 9 183 535 1244 1277
10 107 360 498 1456
11 6 2007 2059 2120
12 1480 1523 1670 1927
13 139 573 711 1790
14 6 1541 1889 2023
15 6 374 957 1174
16 287 423 872 1285
17 6 1809 1918
18 65 818 1396
19 590 766 2107
20 192 814 1843
21 775 1163 1256
22 42 735 1415
23 334 1008 2055
24 109 596 1785
25 406 534 1852
26 684 719 1543
27 401 465 1040
28 112 392 621
29 82 897 1950
30 887 1962 2125
31 793 1088 2159
32 723 919 1139
33 610 839 1302
34 218 1080 1816
35 627 1646 1749
36 496 1165 1741
37 916 1055 1662
38 182 722 945
39 5 595 1674
```

Figura 4.6: Matriz de paridad para una tasa de codificación de 13/15 y longitud de código igual a 16200 [3].

Dado que se trata de un código sistemático, los bits de información son parte de la palabra de código. Si los bits de información se denotan como: $s_0, s_1, s_2, ..., s_{K-1}$, el cálculo de los bits de paridad para la estructura tipo IRA se detalla paso por paso a continuación [3] [5]:

1. Se inicializan con ceros los bits de paridad [3].

   Sí N es la longitud del código y K es la longitud de los bits de información se tiene que:

   $$p_0 = p_1 = p_2 = p_3 = ... = p_{N-K-1} = 0 \tag{4.10}$$

2. Para $0 \leq k < K$ se realiza la acumulación del bit de información s_k al bit de paridad $p_{q(i,j,l)}$ para todos los j [3], es decir, para todos los elementos de la fila i, donde w es el número de elementos de la fila i:

   $$\begin{aligned} p_{q(i,0,l)} &= p_{q(i,0,l)} + s_k \\ p_{q(i,1,l)} &= p_{q(i,1,l)} + s_k \\ p_{q(i,2,l)} &= p_{q(i,2,l)} + s_k \\ . \quad & . \quad . \\ . \quad & . \quad . \\ p_{q(i,w-1,l)} &= p_{q(i,w-1,l)} + s_k \end{aligned}$$

   La expresión $q(i, j, 0)$ es la j entrada en la fila i de la tabla de direcciones de bits de paridad. Entonces $q(i, j, l) = q(i, j, 0) + Q_{ldpc} * l(modM)$ [3].

   Loa parámetro i y k son obtenidos de la siguiente forma: i= $\lfloor k/360 \rfloor$ y $l = k(mod360)$ [3].

3. Para obtener todos los bits de paridad se tiene la siguiente ecuación:

   $$p_k = p_k + p_{k-1}$$

   Después de esto todos los bits de paridad deben ser obtenidos [3].

Del análisis del algoritmo anterior se puede decir que la obtención de los bits de paridad se realiza en grupos de 360 bits. El parámetro i es el encargado de realizar los cambios de fila dentro de la matriz, esto se realiza cada 360 bits de información.

El parámetro l es igual a cero cuando k es igual a 360 o múltiplos de 360 ($l = k(mod360)$). Es decir cada que ocurre un cambio de fila se acumula un bit de información a los bits de paridad indicados por dicha fila. Posteriormente las acumulaciones se realizan a los bits de información obtenidos mediante el cálculo de la expresión $q(i, j, l) = q(i, j, 0) + Q_{ldpc} * l(modM)$.

4.1.4. Implementación

La implementación del codificador LDPC se realizó en el lenguaje C++ debido al cómputo necesario para el algoritmo tipo IRA explicado en la sección anterior. Para añadir un bloque a el módulo llamado ATSC 3.0 se usaron los siguientes comandos:

```
$ cp gr_modtool.py gr-atsc3
$ cd gr-atsc3
$ gr_modtool add
GNU Radio module name identified: atsc3
Enter block type: general
Language (python/cpp): cpp
Language: C++
Enter name of block/code (without module name prefix): ldpc_bb
Block/code identifier: ldpc_bb
Enter valid argument list, including default arguments:
Add Python QA code? [Y/n] y
```

En las líneas de comando se observa que el bloque se denominó *ldpc_bb*, el prefijo del bloque se debe a que éste recibe bytes a la entrada y genera bytes en la salida. El algoritmo fue implementado en la carpeta lib, en los archivos generados automáticamente por la herramienta llamada modtool.

Esta implementación se basó en el codificador LDPC utilizado en el estándar DVB-T2 cuya fuente de código puede encontrarse en [29] y [30]. El codificador de DVB-T2 tiene algunas similitudes con el codificador LDPC de ATSC 3.0, los dos cuentan con una estructura tipo IRA y longitudes de código iguales, pero manejan diferentes tasa de codificación. Cabe destacar que ATSC 3.0 maneja el doble de tasas de codificación que DVB-T2.

Las matrices utilizadas en la implementación de este código se encuentran en el anexo A del documento denominado A/322.

En la figura 4.7 se observa la interfaz gráfica del bloque LDPC, es posible configurar dos parámetros: la longitud de código y la tasa de codificación. Como lo dicta el estándar, la longitud de código puede ser de 16200 bits o de 64800 bits, la tasa de codificación va de 6/15 a 11/15.

Cuando se utiliza este bloque se puede ver en la consola de GNU Radio la configuración que se está ejecutando.

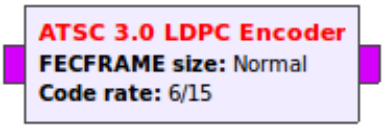

Figura 4.7: Interfaz del codificador LDPC de ATSC 3.0.

```
>>> Running a LDPC encoder
Code length: Short
Encoding at 6/15
>>> Running a LDPC encoder
Code length: Normal
Encoding at 6/15
```

Figura 4.8: Ejemplo de salida en consola de codificador LDPC.

4.2. Entrelazado de bits

El intercalado de bits se lleva a cabo en 3 sub-bloques: intercalador de paridad, intercalador de grupos y el intercalador de bloques, como se muestra en la figura 4.9.

El entrelazado o intercalado de bits depende de la tasa de codificación LPDC utilizada y del orden de constelación seleccionado [4]. Esto se hace con la finalidad de optimizar la eficiencia del canal para cada tasa de codificación y orden de constelación.

Intercalador de paridad

Este bloque solo se utiliza para los códigos LDPC con estructura IRA.

En el proceso de intercalado de paridad, como su nombre lo dice, solo los bits de paridad son entrelazados [3]. El objetivo de entrelazar solo los bits de paridad es el de obtener una estructura cuasi cíclica, similar a la estructura que tienen los bits de información. Como se vio en la sección anterior, una matriz H de tipo IRA en la parte

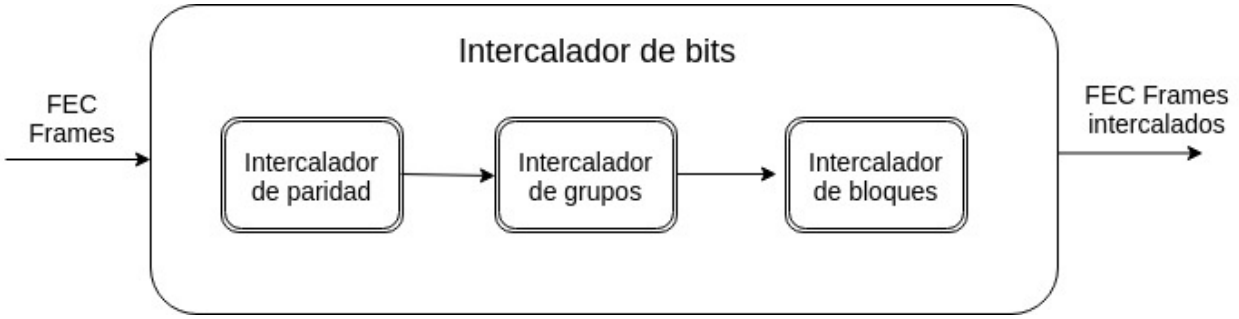

Figura 4.9: Diagrama del intercalador de bits.

	2/15	3/15	4/15	5/15	6/15	7/15	8/15	9/15	10/15	11/15	12/15	13/15
2	A	A	A	A	A	A	A	A	A	A	A	A
4	A	A	A	B	A	A	B	B	A	A	A	A
6	A	A	A	A	A	B	A	B	B	A	A	B
8	A	A	A	B	B	B	B	A	B	B	A	B
10	A	A	A	B	A	B	A	B	B	B	A	A
12	A	A	A	A	A	B	A	A	A	A	A	A

Cuadro 4.3: Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 64800 bits.

	2/15	3/15	4/15	5/15	6/15	7/15	8/15	9/15	10/15	11/15	12/15	13/15
2	A	A	A	A	B	B	A	B	A	A	A	A
4	A	A	A	A	B	B	A	B	A	B	A	B
6	A	A	A	A	B	B	A	B	A	A	A	A
8	A	A	A	A	B	A	A	A	A	B	A	A

Cuadro 4.4: Tipo de intercalador de bloque a utilizar para códigos LDPC de longitud de 16200 bits.

de paridad tiene una estructura de matriz de doble diagonal inferior.

Intercalador de grupos

La trama es divida en grupos de 360 bits. Posteriormente los grupos son reordenadas en la trama según el numero de grupo. La posición en la que los grupos son reorganizados depende de la tasa de codificación y del orden de la constelación.

Intercalador de bloques

Existen dos tipos de intercalador de bloque: el tipo A y el tipo B, su uso depende de la tasa de codificación y del orden de la constelación a usarse.

En las tablas 4.3 y 4.4 se observa el tipo de intercalador de bloque que debe usarse, según el orden de la constelación y la tasa de codificación. En las tablas, los órdenes de constelación se denotan como 2, 4, 6, 8, 10 y 12, los cuales se asocian a las constelaciones QPSK, 16QAM, 64QAM, 256QAM, 1024QAM y 4096QAM respectivamente.

En esta tesis solo se implementa el intercalador tipo A.

4.2.1. Implementación

La implementación del bloque de intercalado se realizó en Python debido a que se trata de un bloque de tipo jerárquico, es decir, está compuesto de otros bloques ya

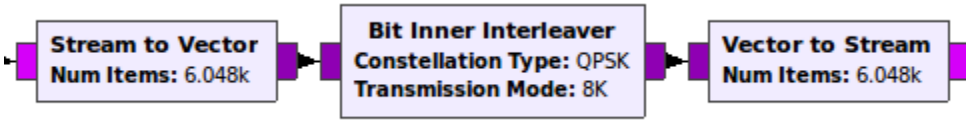

Figura 4.10: Diagrama de la implementación del intercalador de bits

Figura 4.11: Interfaz del intercalador de bits

existentes en GNU Radio.

Se realizó la adaptación del bloque de intercalado de DVB-T que ya existe en GNU Radio, esto es posible ya que el intercalador tipo A es igual al intercalador utilizado en DVB-T [31], el cual ya ha sido implementado en GNU Radio.

El diagrama de la implementación de este bloque se puede observar en la figura 4.10.

Se utilizaron dos bloques llamados *Stream to Vector* los cuales sirven para adaptar el número de bits a la entrada y salida del bloque intercalador perteneciente al estándar DVB-T al intercalador de ATSC 3.0. Estos bloques fueron programados para manejar las dos longitudes de código que existen en GNU Radio.

En la figura 4.11 se tiene la interfaz del bloque implementado, en el puede ser configurado solo un parámetro: la longitud de código (16200 o 64800 bits).

Cabe aclarar que debido a que solo se llevó a acabo la implementación del intercalador tipo A, éste se utilizó para todas las longitudes de código y para todas las constelaciones. Esto se realizó con el fin de obtener las tasas de codificación y las constelaciones que utilizan el codificador tipo B. Esta implementación impacta en la eficiencia del canal debido a que el objetivo de tener dos tipos de intercaladores es el de permitir la optimización de la eficiencia de canal. Por supuesto el receptor tiene que saber el tipo de intercalado utilizado.

Cuando se ejecuta este bloque se puede ver en consola la longitud de código que se está utilizando, ésta debe coincidir con la longitud de código del LDPC (bloque anterior).

4.3. Modulador

En este bloque los bits son mapeados a valores de tipo complejo para la obtención de las constelaciones. Existen 6 ordenes de constelaciones de las cuales solo un orden de constelación es uniforme, el resto es de tipo no uniforme. A continuación se enlistan las constelaciones en forma ascendente:

1. QPSK uniforme
2. 16-QAM no uniforme
3. 64-QAM no uniforme
4. 256-QAM no uniforme
5. 1024-QAM no uniforme
6. 4096-QAM no uniforme

Es importante mencionar que ATSC 3.0 ha optimizado cada orden de constelación para cada tasa de codificación LDPC, por lo tanto, cada orden de constelación varía ligeramente dependiendo de la tasa de codificación utilizada. Debido a que existen 12 tasas de codificación, hay 12 constelaciones para cada orden. Es decir, existen 12 constelaciones 16-QAM, una para cada tasa. Lo mismo sucede con 64, 256, 1024 y 4096-QAM. La excepción es la constelación QPSK, la cual es exactamente igual para todas las tasas de codificación [3].

Sin embargo, la misma constelación es utilizada para las dos longitudes de código que existen (16200 y 64800 bits), además entre ambas longitudes se presenta un rendimiento similar [5].

4.3.1. Implementación

En esta tesis se desarrollaron 4 ordenes de constelación:

1. QPSK uniforme
2. 16-QAM no uniforme
3. 64-QAM no uniforme
4. 256-QAM no uniforme

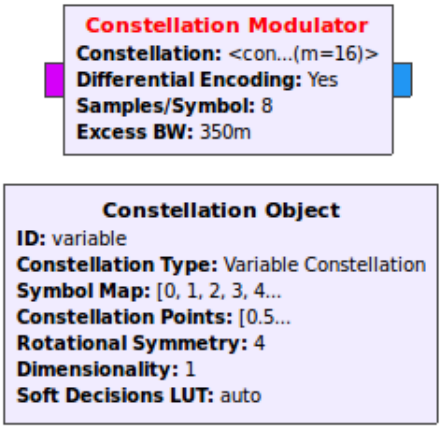

Figura 4.12: Implementación del modulador de ATSC 3.0

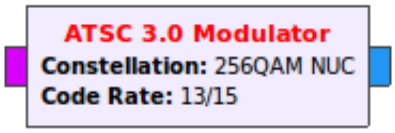

Figura 4.13: Interfaz del modulador de ATSC 3.0

Este modulador implementado en GNU Radios es también de tipo jerárquico ya que su implementación se realiza con ayuda de dos bloques de GNU Radio: *Constellation Object* y *Constellation Modulator* (figura 4.12).

Con el bloque *Constellation Object* es posible definir todas las constelaciones a utilizar mediante el mapeo de símbolos y la definición de puntos en el plano IQ, entre otras cosas. La definición de las constelaciones implementadas se pueden consultar en el apéndice B.

Mediante el bloque *Constellation Modulator* se realiza la modulación.

El modulador de ATSC 3.0 fue implementando en Python, en la figura 4.13 se observa la interfaz gráfica del modulador. Son necesarios dos parámetros para la generación de una constelación: el orden de la constelación y la tasa de codificación.

Todas las constelaciones excepto la QPSK son no uniformes, por lo que en el menú desplegable del bloque son identificadas con las siglas NUC (Non Uniform contellation). En cuanto a las tasas de codificación, se implementaron las 12 tasas existentes desde los 2/15 hasta los 13/15.

Otra parte de la implementación de este bloque consistió en la posibilidad de observar estas constelaciones en el plano IQ, con el objetivo de poder compararlas

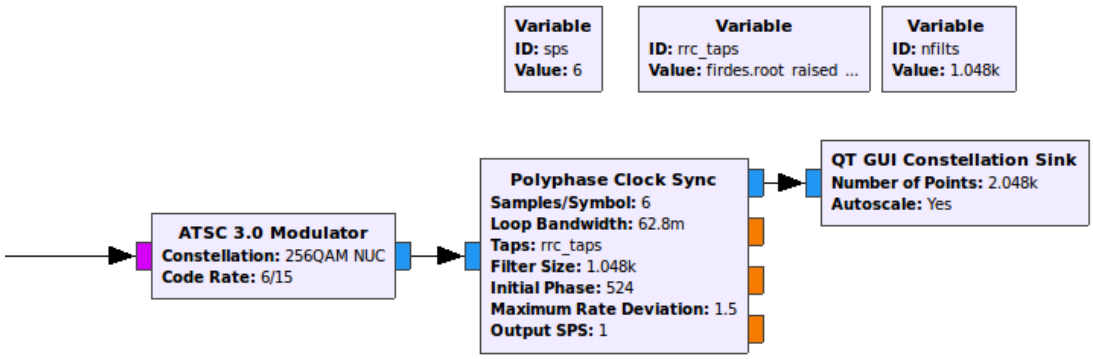

Figura 4.14: Bloques para la visualización de las constelaciones implementadas.

directamente con las proporcionadas en el estándar y verificar la funcionalidad del bloque. En la figura 4.14 se observan las bloques utilizados y su configuración para la visualización de las constelaciones en el plano IQ.

4.3.2. Resultados

Derivado del trabajo de este capítulo de la tesis, se desarrolló el módulo BICM de ATSC 3.0, y se demostró su funcionamiento. Los modos de operación desarrollados son los siguientes: codificador LDPC para tasas altas de codificación, intercalador de bits tipo A y modulador para los ordenes de constelación QPSK, 16-QAM, 64-QAM y 256-QAM. Además, se desarrolló un visualizador de constelaciones para comprobar que los modos de operación implementados funcionaran correctamente.

En este apartado se muestran las gráficas de las constelaciones programadas. En la figura 4.15 se tiene la constelación de menor capacidad dentro del estándar, sin importar la tasa de codificación seleccionada se genera un QPSK uniforme tal y como se define en el estándar. En la imagen también se observa que es posible añadir ruido a la constelación mediante una barra deslizable colocada en la parte superior, esto es posible para todas las constelaciones con la finalidad de observar las constelaciones con ruido añadido.

Las figuras 4.16, 4.17 y 4.18 agrupan en una sola imagen todas las constelaciones no uniformes según el orden de constelación implementado dentro de este bloque: 16-QAM, 64-QAM y 256-QAM. Cada imagen cuanta con 12 constelaciones las cuales corresponden a las 12 tasas de codificación. Los órdenes de constelación más altos (1024 y 4096-QAM) no fueron implementados debido a la capacidad de cómputo de los equipos.

En la figura 4.16 se observan las 12 constelaciones 16-QAM de ATSC 3.0. Para

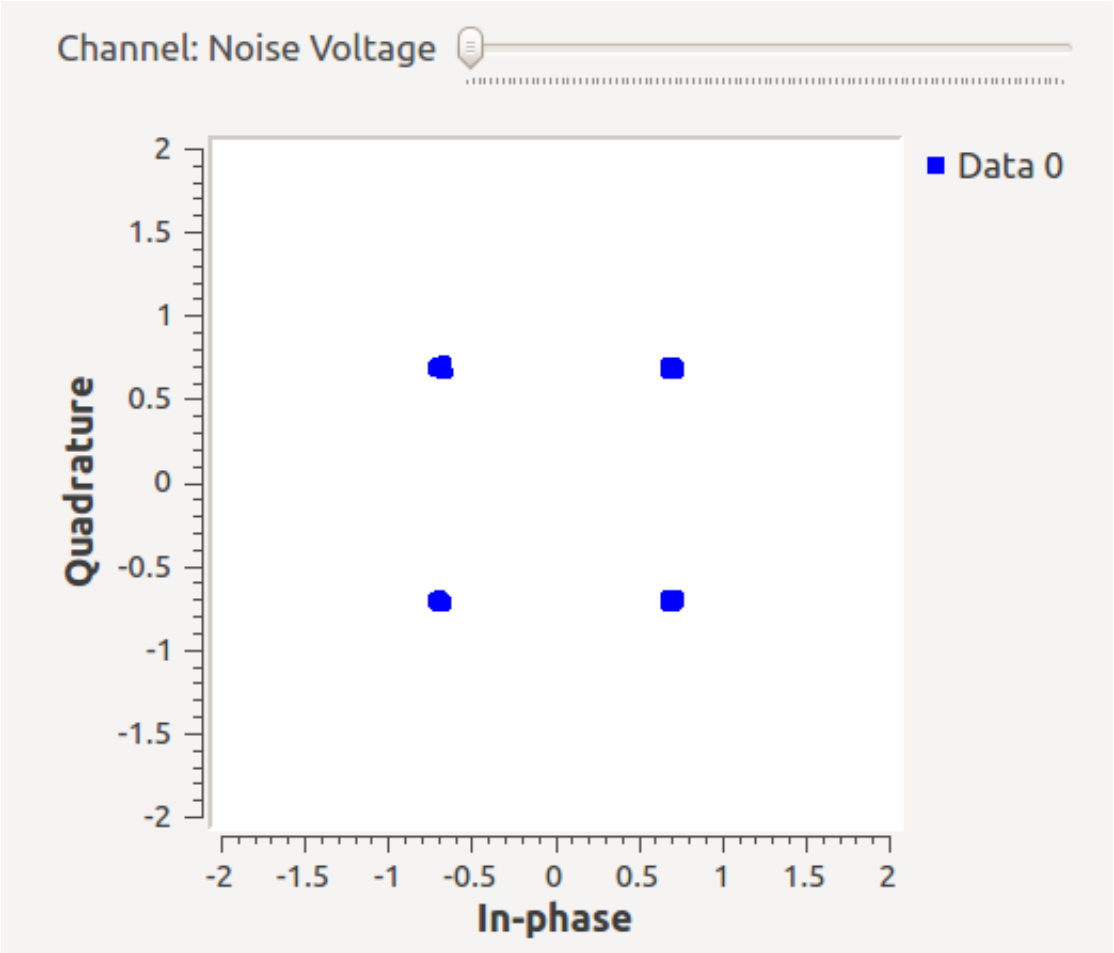

Figura 4.15: Constelación QPSK uniforme.

una tasa de 2/15, los puntos del cuadrante se encuentran tan cercanos que pareciera tratarse de un QPSK. Para tasas de codificación de 3/15 a 10/15 las constelaciones son parecidas entre sí mostrando un patrón circular, para tasas altas: 11/15, 12/15 y 13/15, el patrón de puntos es muy similar a lo que sería un 16-QAM uniforme, sin embargo no es uniforme.

Respecto a la figura 4.17 las 12 constelaciones de 64-QAM tienen un patrón similar al circular. Por último se obtuvo la figura 4.18 la cual muestra las constelaciones de orden 256, de igual manera todas las tasas de codificación muestran un patrón circular. En ambas figuras se observa que para la tasa de codificación más baja, los puntos en la constelación de cada cuadrante se encuentran muy cercanos entre sí.

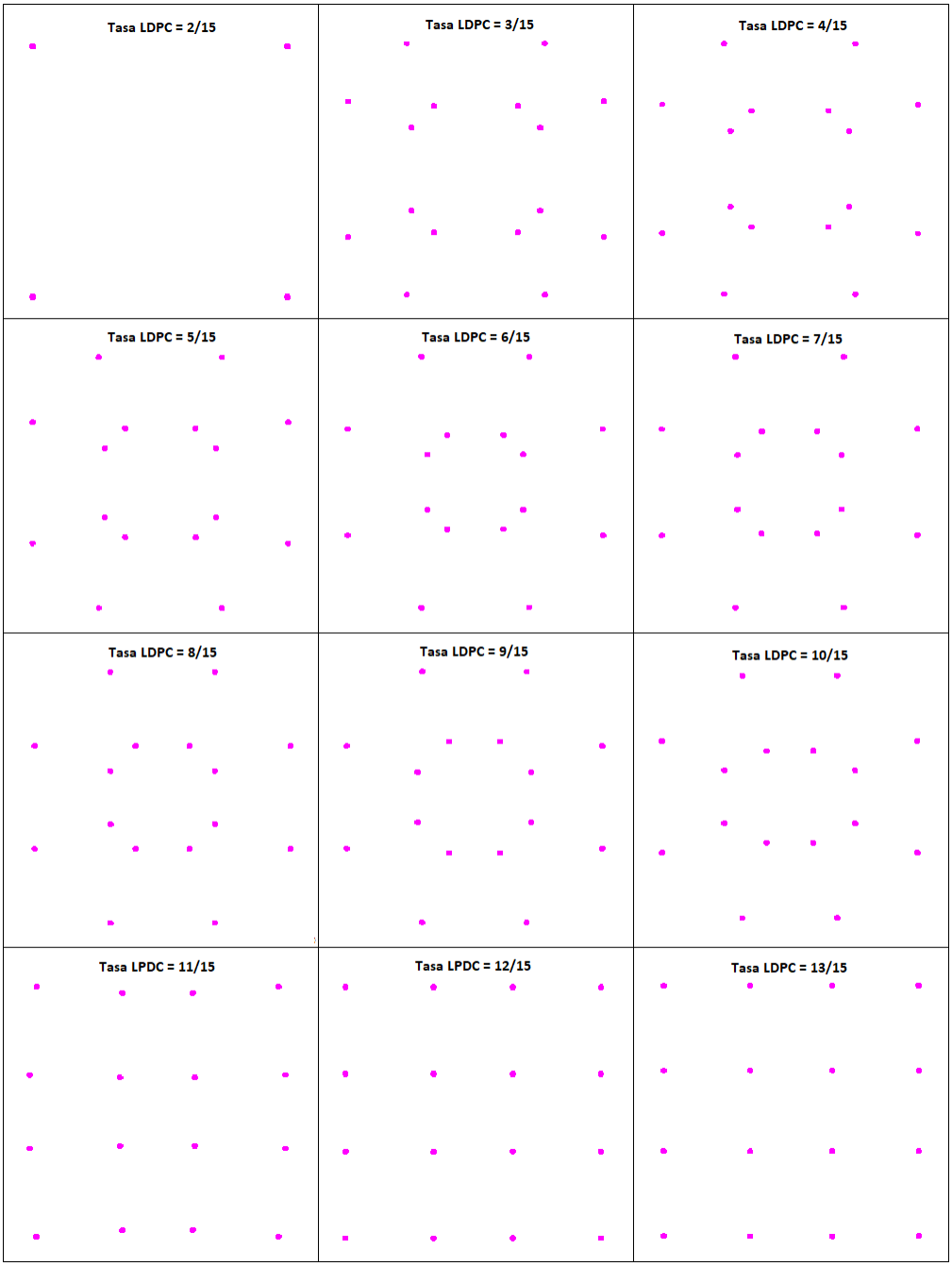

Figura 4.16: Constelación 16-QAM no uniforme para todas las tasas de codificación.

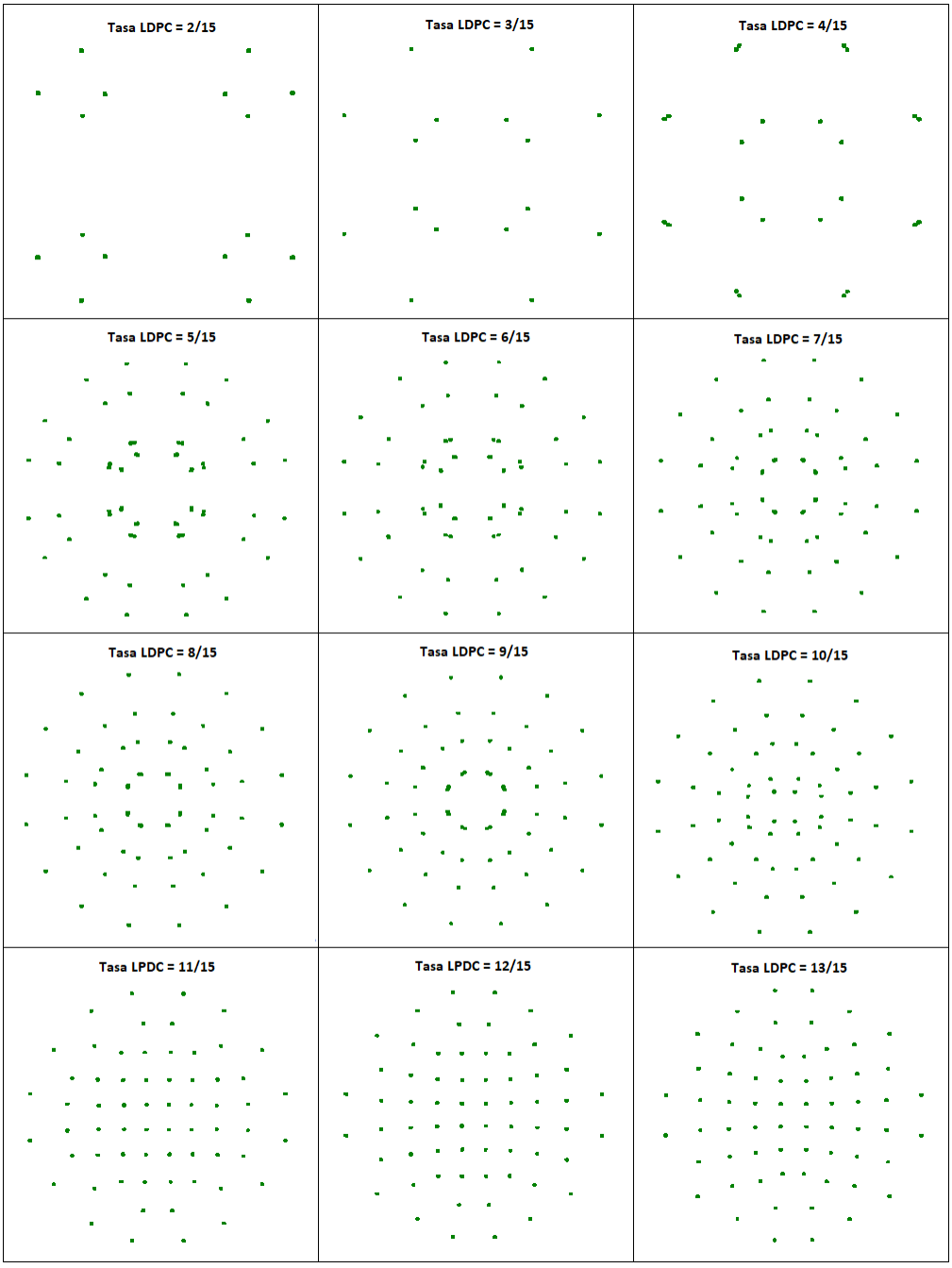

Figura 4.17: Constelación 64-QAM no uniforme para todas las tasas de codificación.

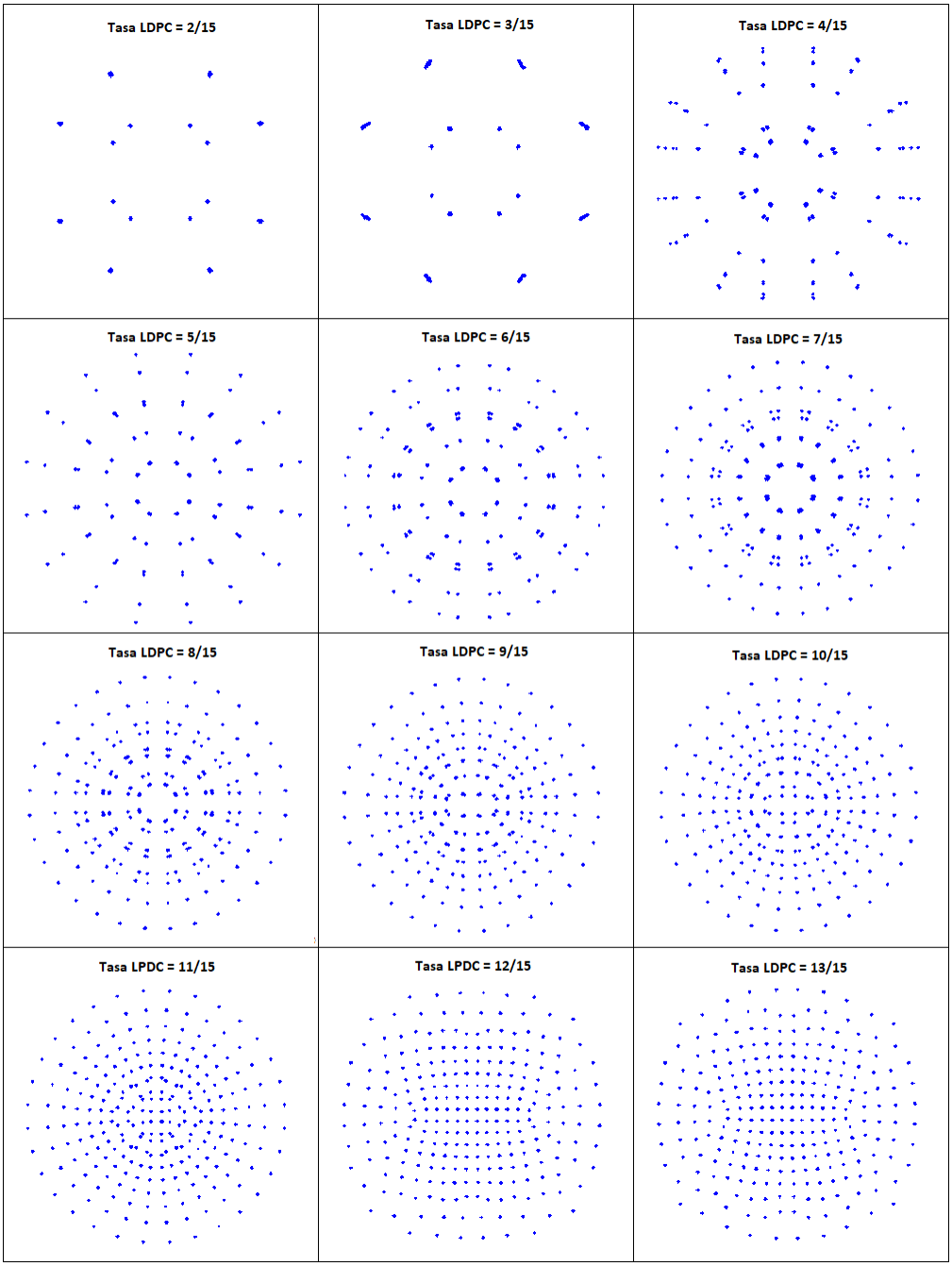

Figura 4.18: Constelación 256-QAM no uniforme para todas las tasas de codificación.

Capítulo 5

Multiplexación por nivel de potencia

La multiplexación por capas o por nivel de potencia, mejor conocida por sus siglas en inglés como LDM (Layered Division Multiplexing), es una multiplexación no ortogonal [7], utiliza el mismo rango de frecuencias para transmitir simultáneamente diferentes servicios de acuerdo a las necesidades del radiodifusor. Esta técnica consiste en una estructura de dos capas dentro del estándar ATSC 3.0. Esta estructura es capaz de transmitir dos señales simultáneamente a diferentes niveles de potencia y con robustez diferenciada. De esta forma es posible entregar servicios a receptores móviles y servicios de ultra alta definición (UHDTV) o de alta definición (HDTV) a receptores fijos simultáneamente.

Otros estándares han implementado diferentes soluciones para transmitir servicio a receptores móviles y fijos. La segunda generación de entandares DVB (Digital Video Broadcasting) transmite estos dos servicios haciendo uso de PLPs o mediante tramas que en un principio estaban pensadas para uso futuro [32], el inconveniente de estas dos soluciones es que se basan en la multiplexación por división de tiempo o TDM (Time Division Multiplexing). En el caso del estándar ISDB (Integrated Services Digital Broadcasting) se utiliza una multiplexación en frecuencia o también llamada por sus siglas en inglés FDM (Frequency Division Multiplexing) [32]. En la figura 5.1 se comparan las diferentes técnicas de multiplexación con respecto al aprovechamiento de recursos como tiempo y frecuencia, se aprecia que LDM aprovecha al 100 % ambos recursos para transmitir diferentes servicios.

La multiplexación en potencia trae consigo diferentes ventajas, transmite con una eficiencia espectral alta, mejorando la capacidad del canal en comparación con las técnicas de multiplexación TDM o FDM [33][32]. Esto sucede en los escenarios más

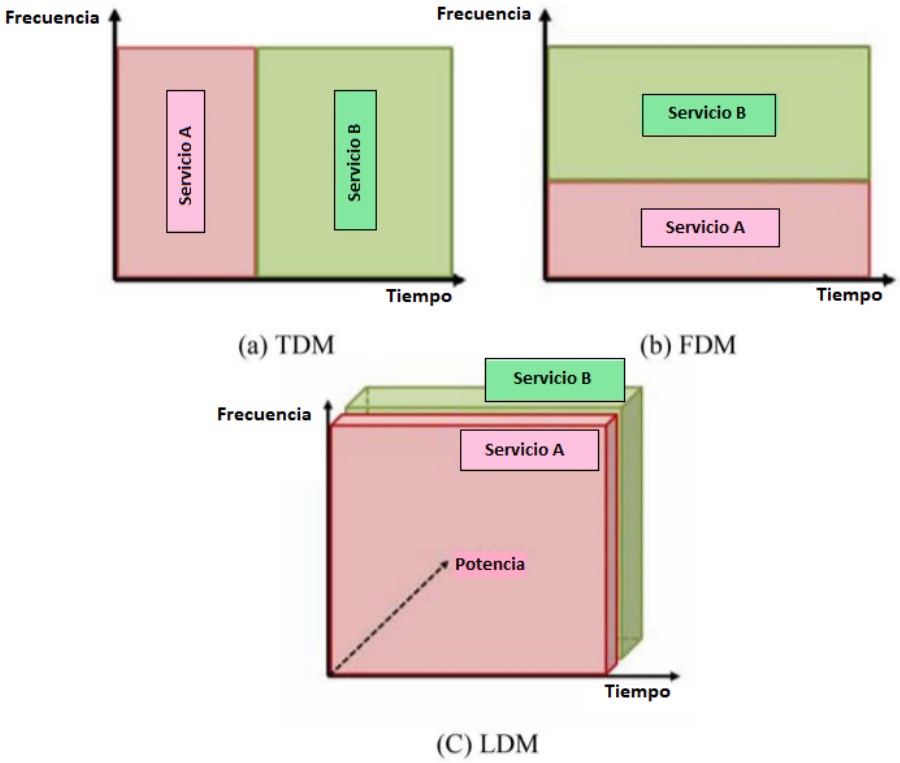

Figura 5.1: (a) TDM (b) FDM y (c) LDM para múltiples servicios de radiodifusión [7].

típicos de ATSC 3.0, es decir, cuando se trata de brindar servicio a receptores móviles con umbrales de SNR bajos (flujo de datos bajo, calidad de imagen baja) y para receptores fijos con umbrales de SNR altos (flujo de datos alto debido a calidad de imagen alta) [7] [33].

Se ha demostrado que LDM puede proveer una eficiencia espectral adicional de 2 b/s/Hz y hasta más de 2.5 b/s/Hz para servicios de recepción fija, mientras que para servicios de recepción móvil la velocidad de transmisión aumenta un 50 %, esto nuevamente en comparación con el uso de FDM/TDM [33]. Otra ventaja es que esta tecnología presenta mejor rendimiento cuando el objetivo es transmitir solo a receptores móviles, en este caso, ambas capas contienen servicios para receptores móviles, portátiles o para recepción en interiores, este estudio, encontrado en [34], se realizó para velocidades del receptor de 3 km/h y a 300 km/h y se comparó con un sistema TDM [34].

LDM es la suma de dos señales sincronizadas en tiempo y frecuencia [32]. Consiste en la superposición de espectros de dos o más PLPs, es decir, aunque el sistema LDM de ATSC 3.0 consiste de solo dos capas, cada capa puede contener uno o más PLPs.

La capa alta también conocida como *Core Layer* (CL) es la que concentra la mayor parte de la potencia y es utilizada para proporcionar servicio a receptores móviles y portátiles, por lo que es una señal muy robusta con un flujo de datos bajo. La capa baja, también conocida como *Enhanced Layer* (EL), es utilizada para dar servicio UHDTV o para transmitir múltiples servicios de HDTV a receptores fijos, por lo tanto, es usualmente una señal de alta capacidad pero baja robustez.

Típicamente, cuando se utiliza una estructura de 2 PLPs y 2 capas como en la figura 5.2, la longitud de código debe ser la misma (64800 ó 16200 bits) mientras que la tasa de codificación y la modulación pueden ser diferentes [3].

En la figura 5.2 se puede observar la arquitectura del sistema transmisor cuando se hace uso de una multiplexación en potencia.

En la figura 5.3 se muestra nuevamente el diagrama a bloques del módulo de multiplexación en potencia o LDM. Se observa que existe un controlador de nivel para regular el voltaje relativo que tiene la capa EL respecto a la capa CL. Este nivel es seleccionable de 0 dB a 25 dB como se muestra en la tabla 5.1 en la columna 1. Cabe resaltar que variando el controlador de nivel también varía la robustez de cada capa de la señal.

Para decodificar la capa alta o capa CL, la capa EL es tratada como interferencia adicional, por lo tanto, esta interferencia también es controlada mediante los diferentes niveles mencionados anteriormente. Es interesante resaltar que un decremento en el nivel de inyección de la capa EL, se traduce a un aumento en el umbral de SNR en

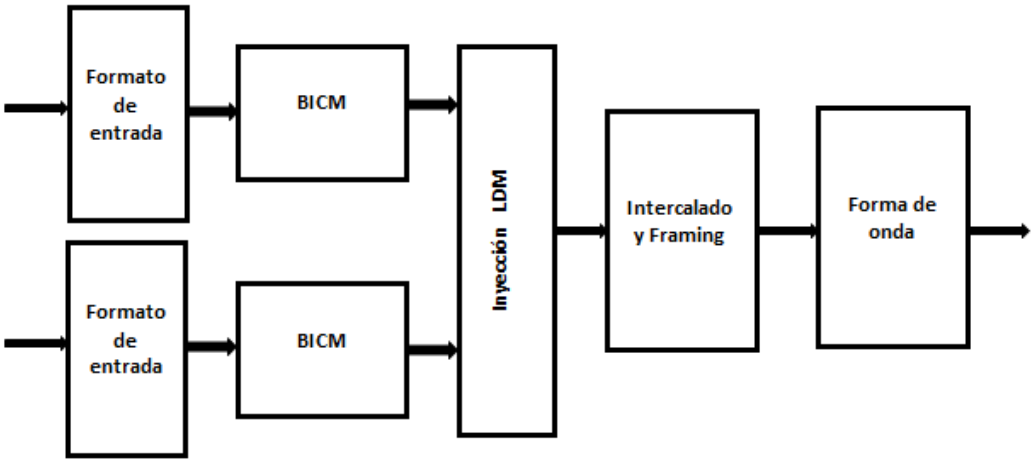

Figura 5.2: Diagrama transmisor con arquitectura LDM [3].

los receptores móviles y portátiles, es decir, a niveles de inyecciones bajos (pequeña diferencia de potencia entre capas) significa que una mayor parte de la potencia se comparte con la capa EL, por lo tanto el umbral necesario para recibir correctamente la capa CL aumenta [35], de la misma forma que el umbral necesario para recibir correctamente la capa EL disminuye.

Como dato adicional, para decodificar la capa EL, el receptor fijo necesita primero recibir y cancelar la capa CL [33]. Cabe resaltar que al adoptar esta tecnología la complejidad de implementación solo aumenta en un 15 % [35].

En el caso de receptores móviles o en interiores, es claro que éstos no requieren ninguna implementación adicional ya que solo necesitan decodificar la capa CL [7].

Después de la combinación de ambas capas (CL y EL), el voltaje total debe ser normalizado. El factor normalizador se denomina β y éste depende del factor denominado α que controla el nivel de potencia de la capa EL [3].

En la tabla 5.1 se muestran los factores α y β para los niveles de potencia relativa de la capa EL, estos niveles están en un rango dentro de los 0 dB a los 25 dB. En total se tiene 31 niveles de potencia relativos de la capa EL con respecto a la capa CL.

El porcentaje de potencia que concentra la capa CL puede ser calculada en función del nivel de inyección con la ecuación 5.1:

$$\%P_{CL} = \frac{1}{1 + 10^{-nivel/10}} \tag{5.1}$$

Por ejemplo para un nivel de inyección de 4 dB la potencia que concentra la capa CL es del 71.5 % [7].

Nivel de inyección [dB]	Factor de escalamiento α	Factor para normalizar β
0.0	1.0000000	0.7071068
+0.5	0.9440609	0.7271524
+1.0	0.8912509	0.7465331
+1.5	0.8413951	0.7651789
+2.0	0.7943282	0.7830305
+2.5	0.7498942	0.8000406
+3.0	0.7079458	0.8161736
+3.5	0.6683439	0.8314061
+4.0	0.6309573	0.8457262
+4.5	0.5956621	0.8591327
+5.0	0.5623413	0.8716346
+6.0	0.5011872	0.8940022
+7.0	0.4466836	0.9130512
+8.0	0.3981072	0.9290819
+9.0	0.3548134	0.9424353
+10.0	0.3162278	0.9534626
+11.0	0.2818383	0.9625032
+12.0	0.2511886	0.9698706
+13.0	0.2238721	0.9758449
+14.0	0.1995262	0.9806699
+15.0	0.1778279	0.9845540
+16.0	0.1584893	0.9876723
+17.0	0.1412538	0.9901705
+18.0	0.1258925	0.9921685
+19.0	0.1122018	0.9937642
+20.0	0.1000000	0.9950372
+21.0	0.0891251	0.9960519
+22.0	0.0794328	0.9968601
+23.0	0.0707946	0.9975034
+24.0	0.0630957	0.9980154
+25.0	0.0562341	0.9984226

Cuadro 5.1: Factores de escala y normalización para el módulo LDM [3].

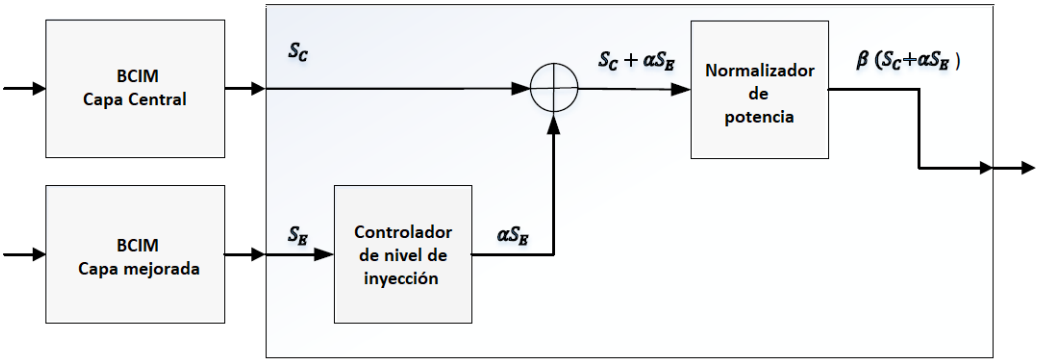

Figura 5.3: Bloque LDM para dos capas.

5.1. Implementación del bloque de multiplexación en potencia

Los datos a la entrada del bloque de multiplexación en potencia son de tipo complejo debido a que el bloque anterior es el de modulación. Los datos de salida son igualmente de tipo complejo debido a que este bloque realiza la superposición de espectros de dos señales con diferentes niveles de potencia.

Como se vio en el capítulo 3, sección 3.3, el nombre de bloque en GNU Radio debe especificar su función principal y el tipo de dato a la entrada y salida del bloque. Siguiendo el formato que propone GNU Radio, el bloque de multiplexación en potencia se denominó como sigue: *ldm_cc*.

En el caso de este bloque que es el multiplexor en potencia se decidió implementarlo en python debido a que no realiza grandes cantidades de cálculos sobre los datos. En cuanto al tipo de bloque, es de tipo jerárquico ya que se utilizan otros bloques para su construcción.

5.1.1. Programación del bloque LDM

Debido a que el módulo de multiplexación en potencia fue programado en python toda la lógica de este bloque se encuentra en un archivo llamado *ldm_cc.py*.

El bloque cuenta con dos puertos de entrada y uno de salida, esto se aprecia en la figura 5.4. Observando la figura, la entrada superior es la capa CL, mientras que la entrada o puerto inferior es la capa EL. Al posicionar el cursor sobre un puerto aparece el nombre del puerto como *CL* o *EL*.

Únicamente se recibe como parámetro el nivel de inyección de potencia deseado

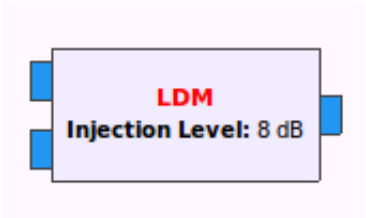

Figura 5.4: Interfaz con el usuario del Multiplexor en Potencia o LDM.

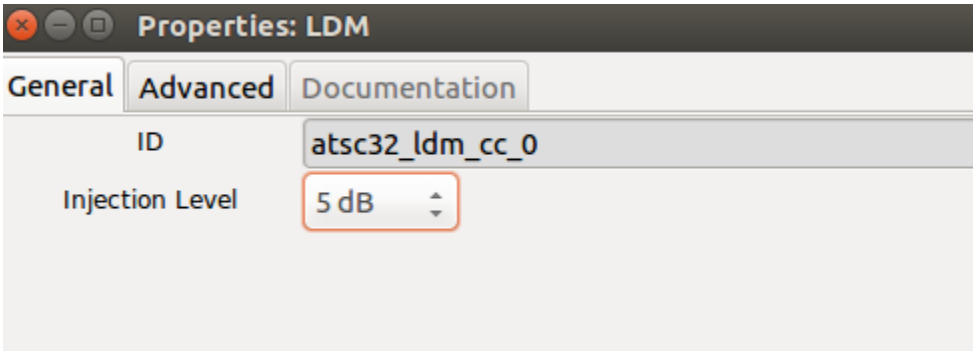

Figura 5.5: Propiedades modificables por el usuario del bloque LDM.

para la capa EL, el cual va de los 0 dB a los 25 dB (figura 5.5).

Los bloques que fueron utilizados para la creación de este bloque fueron los llamados *multiply_const_vcc* y *add_vcc*, pertenecientes a la fuente de código de GNU Radio. Se utilizaron dos bloques *multiply_const_vcc* y un bloque *add_vcc*, los cuales fueron suficientes para construir el controlador de nivel de potencia, el combinador de señales y el normalizador. En la figura 5.6 se observa el diagrama en GNU Radio de la implementación de este bloque, los parámetros configurados en los bloques denominados *Multiply Const* dependen del nivel de inyección, es decir estos son los factores α y β de la tabla 5.1, la señal que entra directamente al bloque llamado *Add* (combinador de señales) es la capa CL. La capa EL entra a un bloque llamado *Multiply Const* el cual es el controlador de nivel de potencia. El bloque *Multiply Const* de la derecha es el normalizador.

Se implementaron todos los modos de operación que establece el estándar ATSC 3.0 en el documento A/322 [3]. Todos estos parámetros se pueden consultar en la tabla 5.1.

Para la interfaz del bloque se trabajó sobre el archivo denominado *atsc3_ldm_cc.xml*. El usuario puede modificar el nivel de inyección de manera gráfica como se observa en la figura 5.5.

Cualquier usuario puede hacer uso de estos bloques descargando el módulo e instalándolo. Se integró al módulo un script para su instalación por lo cuál solo es

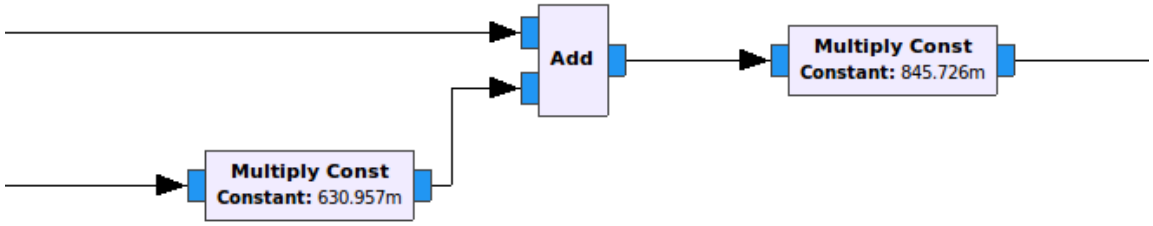

Figura 5.6: Implementación del multiplexor en potencia.

```
LDM:
Multiplier Factor Alfa = 0.9440609
Normalizing Factor Beta= 0.7271524
```

Figura 5.7: Ejemplo de salida en la consola de GNU Radio al ejecutarse el bloque LDM.

necesario ejecutar el script en una terminal con el siguiente comando:

```
$ . ./install_module_atsc3.sh
```

El srcipt utiliza la herramienta CMake y los comandos *make* y *makeinstall* para instalar el módulo en GNU Radio y éste pueda interactuar con todos los módulos existentes.

Por último cuando es ejecutado el bloque multiplexor, en la consola de GNU Radio se muestran los parámetros α y β que están siendo utilizados, se muestra un ejemplo en la figura 5.7, que corresponde a un nivel de inyección de 0.5 dB según la tabla 5.1.

En la figura 5.4 se muestra la interfaz del bloque con el usuario, mientras que en la figura 5.5 se muestran las propiedades del bloque.

5.2. Resultados

Para verificar el correcto funcionamiento del bloque de multiplexación en potencia, se desarrolló un programa en GNU Radio. Este programa genera 2 PLPs, los PLPs entran al bloque de multiplexación en potencia, y finalmente, tras el bloque de multiplexación, se muestra la constelación en el plano IQ con ayuda del graficador de constelaciones desarrollado en el capítulo 4. Cada PLP consiste en un bloque BICM con configuración diferente, el BICM se compone de un codificador LDPC,

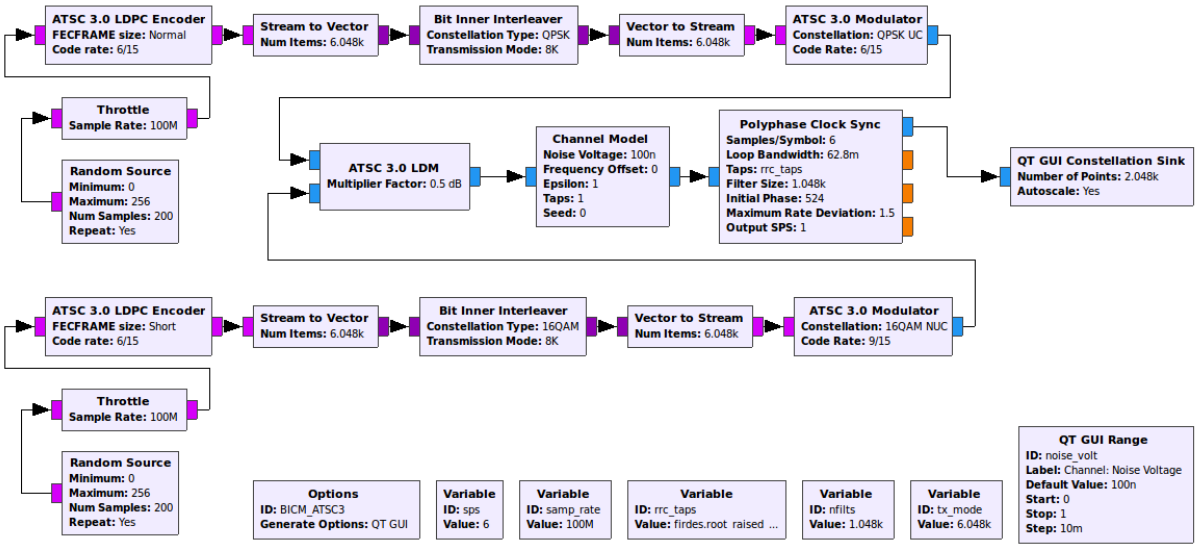

Figura 5.8: Programa en GNU Radio para la generación de 2 PLPs y para la obtención de las constelaciones después del bloque de multiplexación en potencia.

un intercalador y un modulador. Estos bloques son los desarrollados anteriormente y explicados en el capítulo 4.

El programa generado en GNU Radio para testear el bloque LDM se aprecia en la figura 5.8. Como fuente de información se utiliza un generador pseudoaleatorio. Los datos pasan a través del codificador LDPC y posteriormente por el intercalador de bits, el siguiente bloque es el modulador, hasta aquí se constituye un bloque BICM. El BICM superior fue configurado para actuar como la capa CL mientras que el inferior se configuró como una capa EL.

Los dos PLPs generados se combinan en el bloque LDM. Por último, los datos de salida del bloque LDM ingresan a dos bloques mediante los cuales se obtiene la constelación, y de esta forma, comprobar el correcto funcionamiento del bloque LDM. El resto de los bloques son utilizados como variables de configuración.

Con el programa de la figura 5.8 se generaron diferentes configuraciones para cada PLP. En general el PLP de la capa CL se configuró de manera más robusta, utilizando tasas bajas de codificación y constelaciones de baja capacidad. Por el contrario para el PLP de la capa EL se utilizaron constelaciones de alta capacidad y tasas altas de codificación. Todo esto de acuerdo a lo implementado dentro de esta tesis.

A continuación se muestran algunos ejemplos de constelaciones obtenidas después del bloque de LDM, es decir, estas constelaciones son el resultado final de todo el flujo implementado en esta tesis.

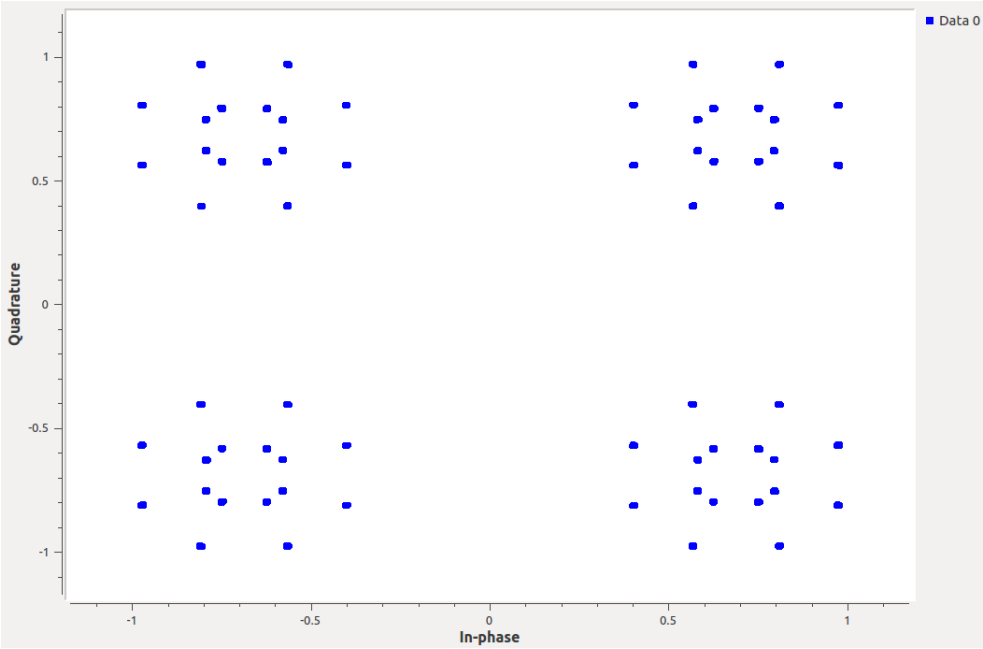

Figura 5.9: Superposición de una constelación QPSK (capa CL) y una constelación 16-QAM no uniforme (capa EL), con nivel de inyección de 13 dB.

Para la obtención de la figura 5.9 se configuró en la capa CL o capa alta un QPSK mientras que en la capa baja o capa EL se configuró un 16-QAM. Se pueden observar 4 constelaciones de orden 16, es decir el 16-QAM supeditado al QPSK. La disposición de los símbolos dentro de cada constelación 16-QAM se debe a la elección de la tasa de codificación, que fue de 6/15. Debido a que el nivel de inyección es de 13 dB, las 4 constelaciones debidas al QPSK se encuentran alejadas entre sí.

Un caso similar es la figura 5.10, donde se mantienen las constelaciones QPSK y 16 QAM para 6/15, y la diferencia es el nivel de inyección de 4 dB. Se puede observar que las 4 constelaciones del QPSK empiezan a mezclarse.

Otro caso son las figuras 5.11 y 5.12. Se trata de un QPSK en la capa CL con tasa de codificación de 6/15, mientras que en la capa EL se tiene un 64-QAM no uniforme para una tasa de codificación de 12/15. El nivel de inyección en la figura 5.11 es de 13 dB mientras que en la figura 5.12 es de 5 dB.

En las figuras 5.13 y 5.14, se muestra un caso dónde ambas capas se han configurado de manera idéntica, se utilizó un 16-QAM para una tasa de codificación de 12/15, el cuál es muy similar al 16-QAM uniforme. Se observa la superposición de dos constelaciones 16-QAM, una proveniente de la capa CL y la otra de la capa EL pero a diferentes niveles de potencia. En la figura 5.13 se utilizó un nivel de inyección de potencia de 13 dB mientras que en la figura 5.14 se utilizó un nivel de inyección de 20 dB.

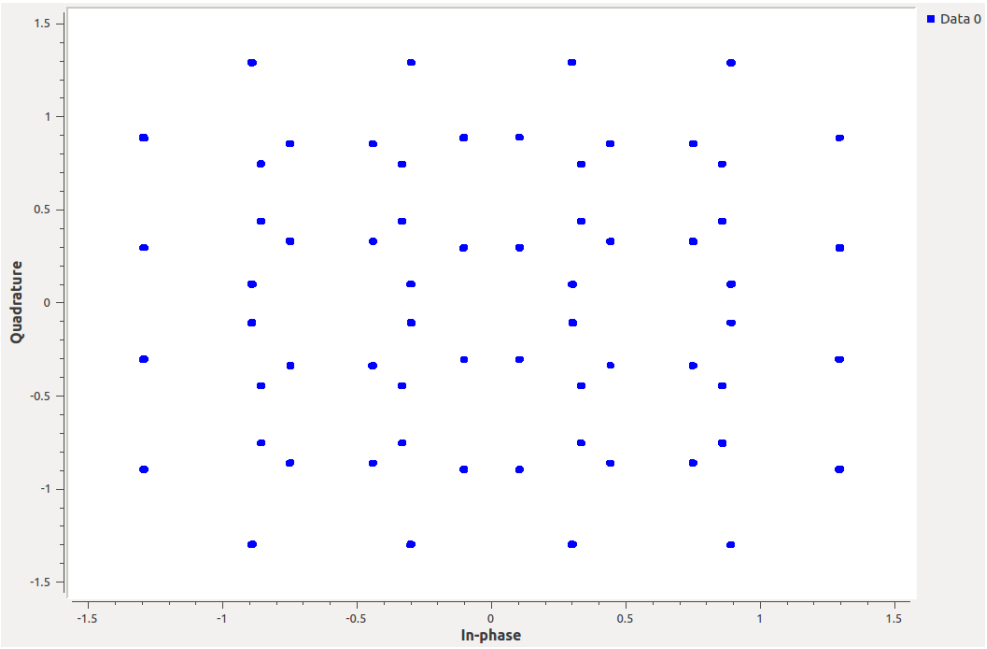

Figura 5.10: Superposición de una constelación QPSK (capa CL) y una constelación 16-QAM no uniforme (capa EL), con nivel de inyección de 4 dB.

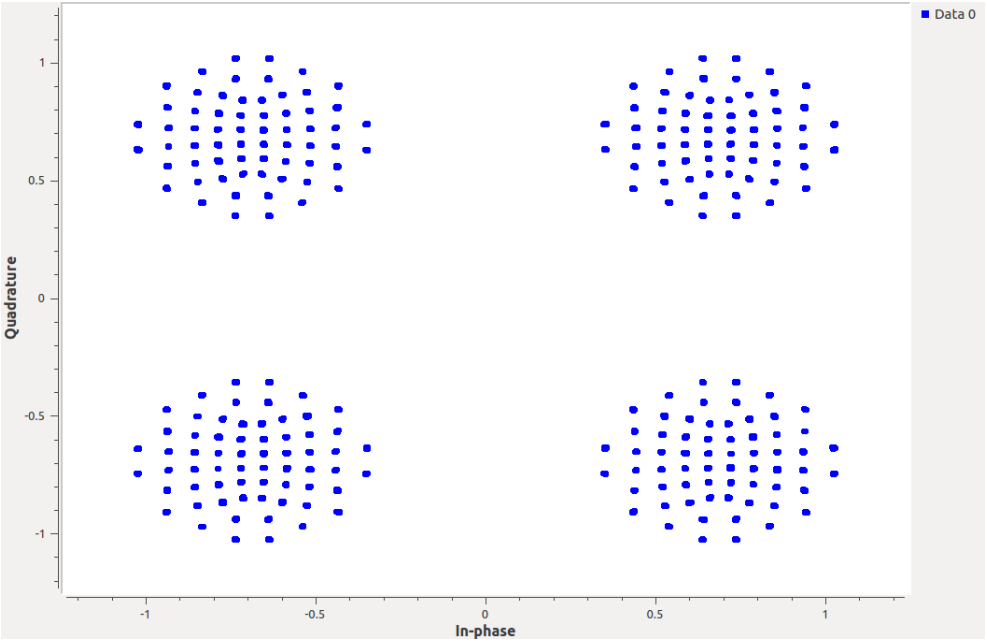

Figura 5.11: Superposición de constelación QPSK (capa CL) y 64-QAM no uniforme (capa EL) con un nivel de inyección de 13 dB.

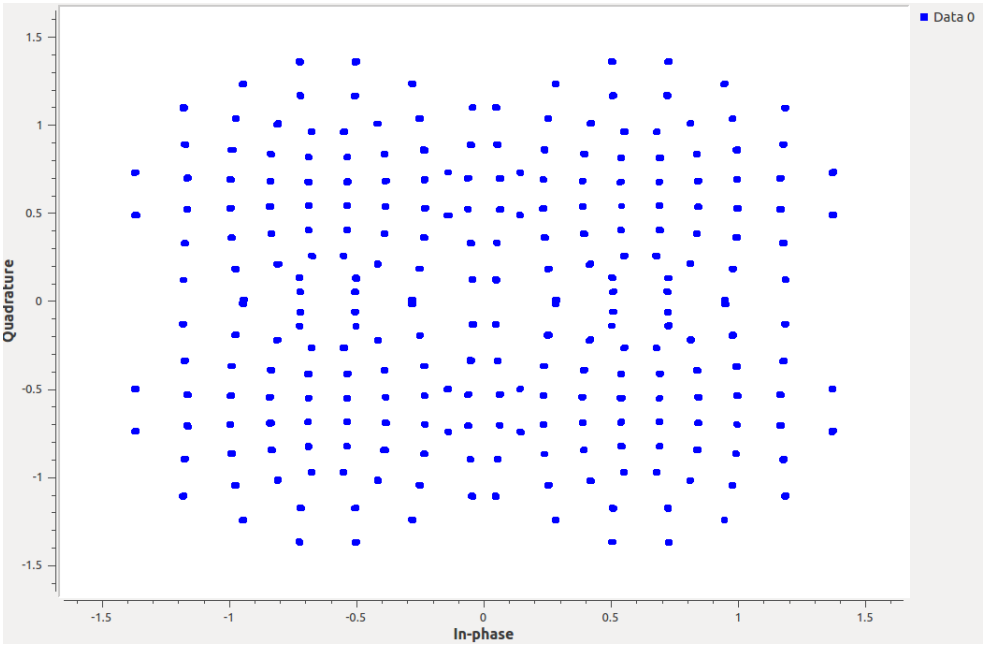

Figura 5.12: Superposición de constelación QPSK (capa CL) y 64-QAM no uniforme (capa EL) con un nivel de inyección de 5 dB.

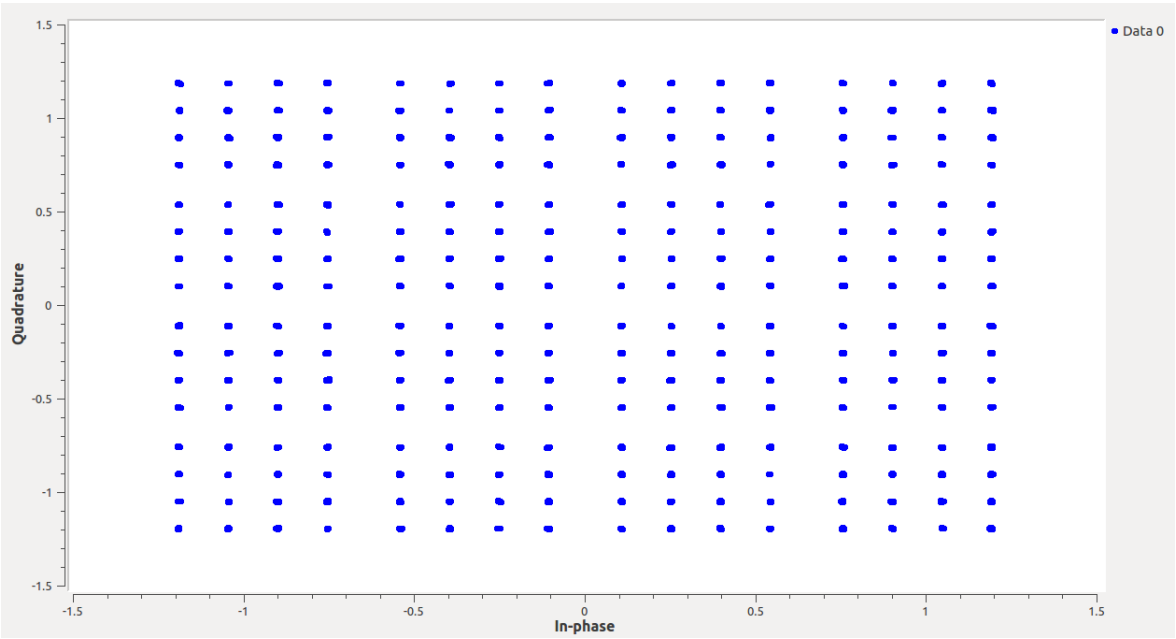

Figura 5.13: Constelación 16-QAM en ambas capas con nivel de inyección de 13 dB.

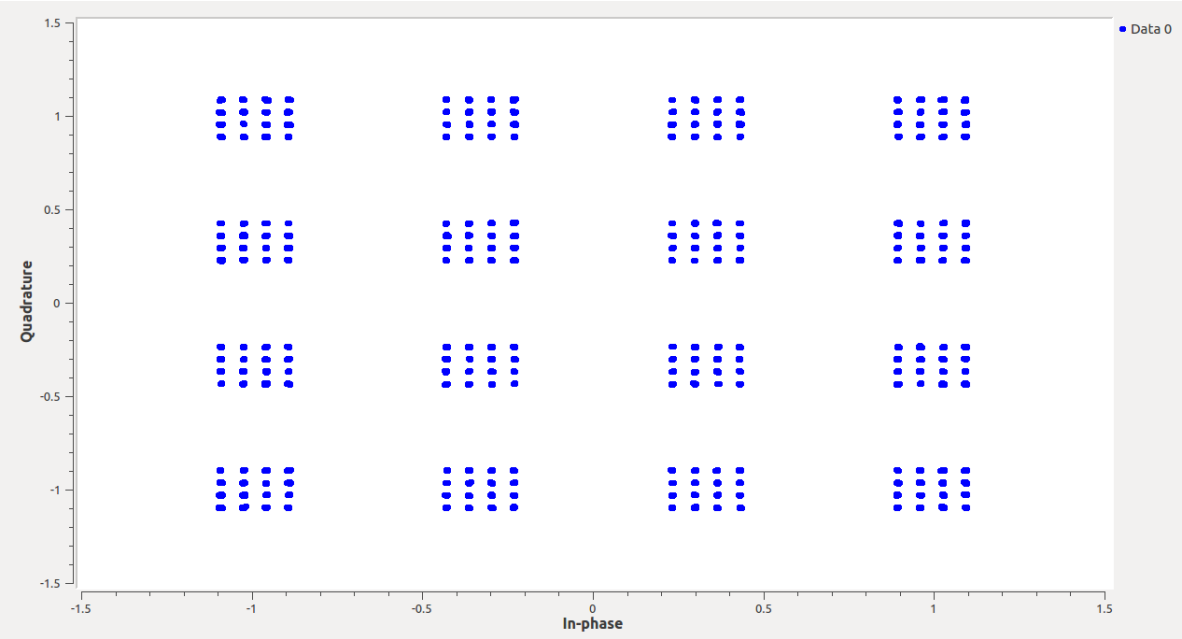

Figura 5.14: Constelación 16-QAM en ambas capas con nivel de inyección de 20 dB.

También se generaron otros escenarios que no existen dentro de ATSC 3.0, como es el caso de las figuras 5.15 y 5.16, puesto que no existe una constelación BPSK en el estándar. En la figura 5.15 se muestra una constelación obtenida con un nivel de inyección de 5 dB, donde en la capa CL se configuró un BPSK mientras que en la capa EL se configuró un QPSK.

También se aprecia en la figura 5.16 una constelación con nivel de potencia de 5 dB, en este caso se utiliza en la capa EL un 16-QAM y en la capa CL de nuevo un BPSK.

La superposición de constelaciones implica una interferencia adicional, es decir, la capa EL es interferencia adicional a la capa CL. En la figura 5.17, se tiene una superposición de un QPSK (CL) y un 16-QAM (EL) con ruido agregado, se puede observar que debido al ruido agregado la constelación 16-QAM se ha empezado a distorsionar.

Si el receptor es capaz de demodular y decodificar la capa CL, éste entonces reconstruye la señal CL y la resta a la señal recibida, si la SNR es suficientemente grande, el receptor será capaz de demodular y decodificar la capa EL.

Es importante notar que el nivel de inyección juega un papel importante en cuanto al nivel de interferencia entre capas. Con niveles de inyección bajos, los puntos de las constelaciones se acercan entre sí hasta traslaparse como se observa en la figuras 5.18 y 5.19. Ambas figuras tratan de un 256-QAM para una tasa de 8/15, En la primer figura las constelaciones empiezan a traslaparse con un nivel de inyección de 3.5 dB,

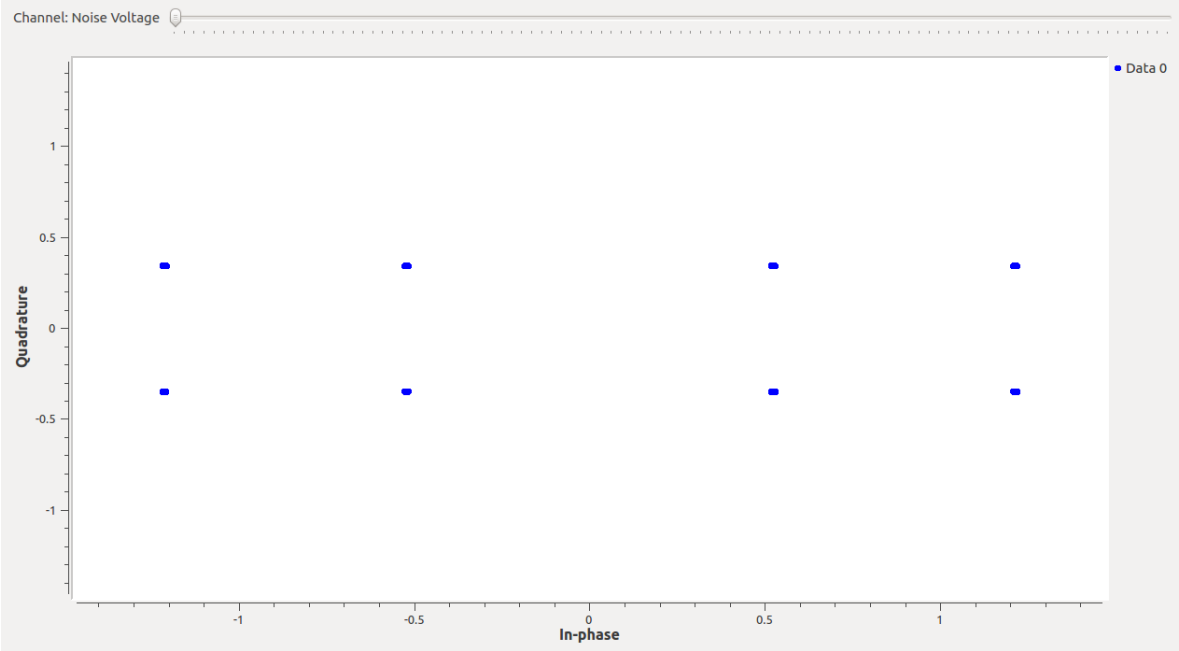

Figura 5.15: Superposición de constelación BPSK (capa CL) y QPSK (capa EL) con un nivel de inyección de 5 dB.

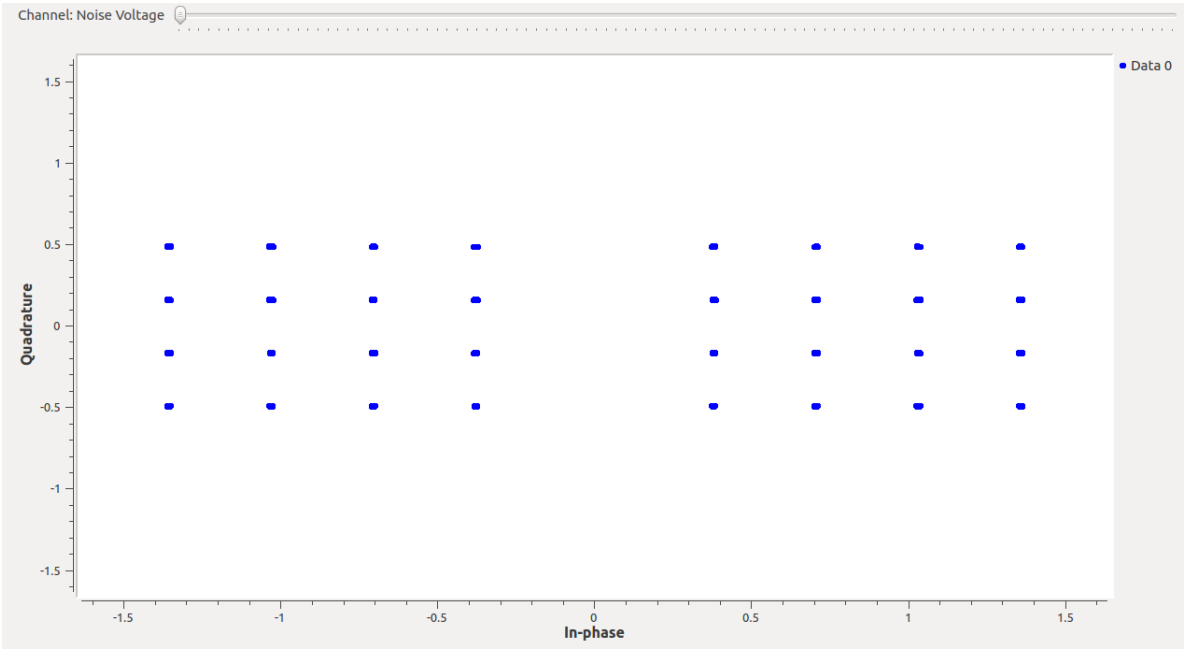

Figura 5.16: Superposición de constelación BPSK (capa CL) y 16-QAM (capa EL) con un nivel de inyección de 5 dB.

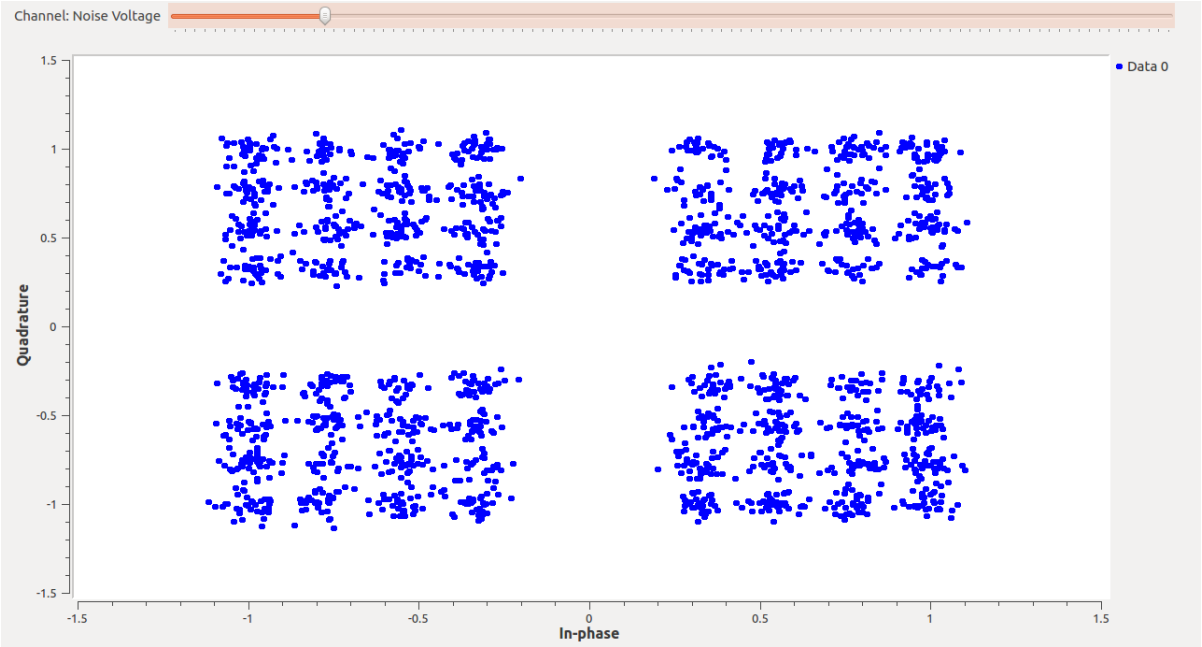

Figura 5.17: Superposición de constelación 4-QAM (capa CL) y 16-QAM (capa EL) con un nivel de inyección de 9 dB y ruido agregado.

en la figura 5.19 se tiene un nivel de inyección de 0 dB por lo que se han traslapado la mitad de la constelación.

Cabe destacar que un nivel típico de inyección es de 5 dB [1].

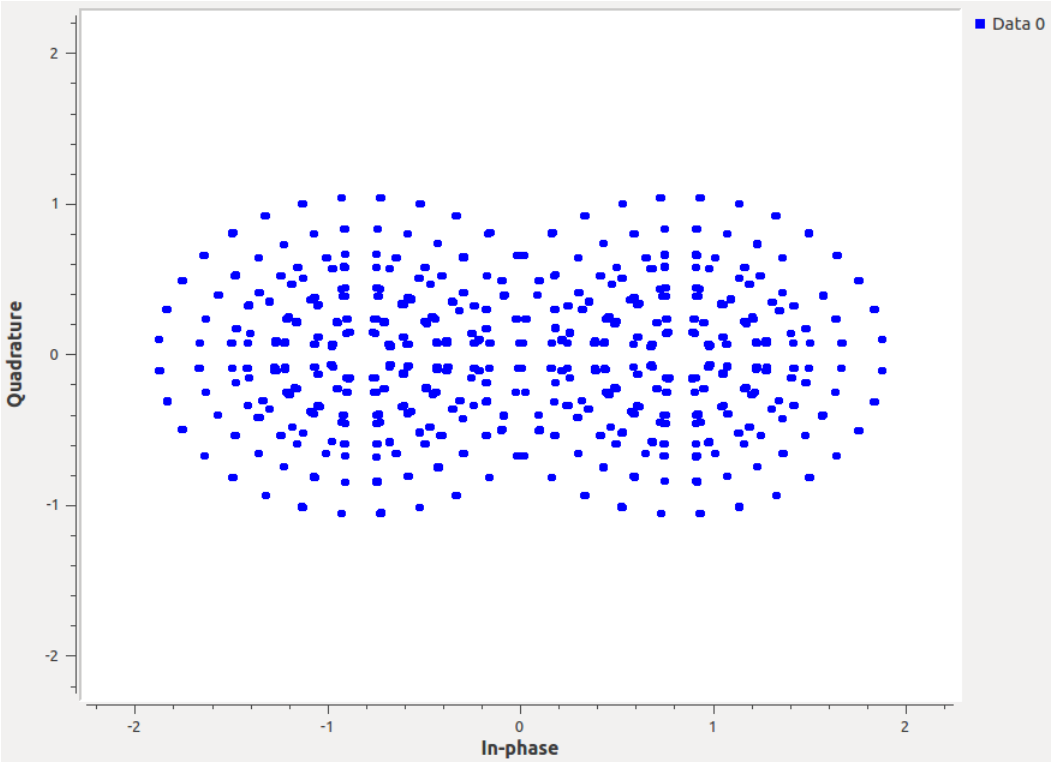

Figura 5.18: Superposición de constelación BPSK (capa CL) y 256-QAM no uniforme (capa EL) con un nivel de inyección de 3.5 dB.

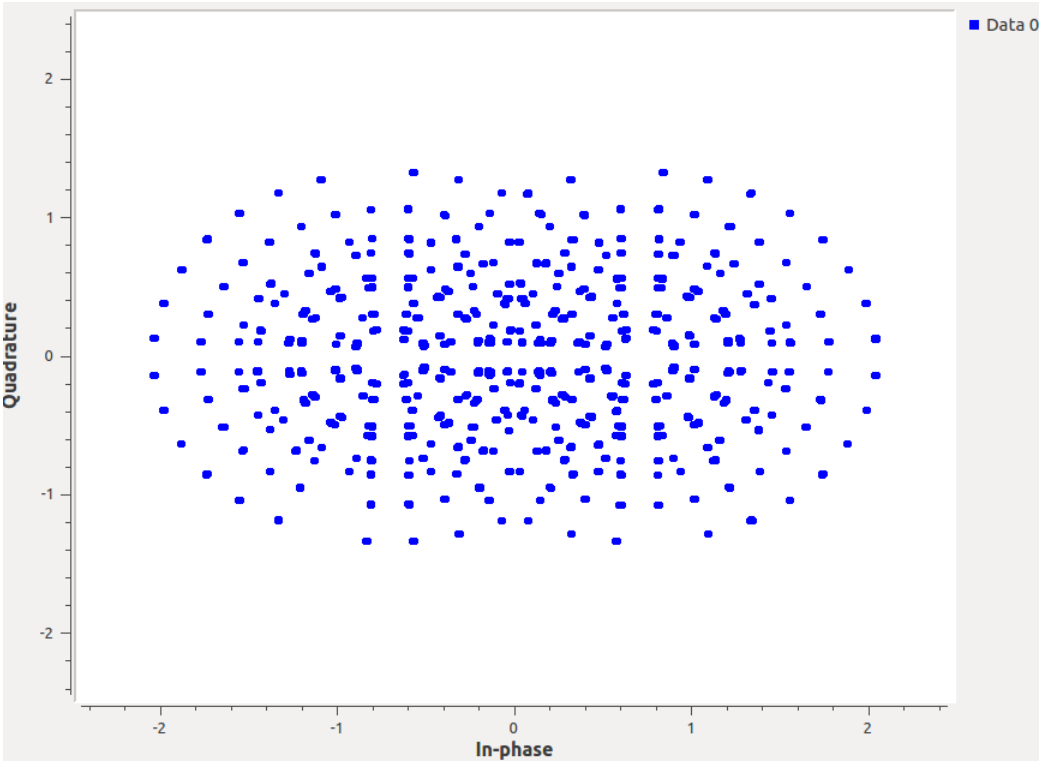

Figura 5.19: Superposición de constelación BPSK (capa CL) y 256-QAM no uniforme (capa EL) con un nivel de inyección de 0 dB.

Capítulo 6

Conclusiones

Esta tesis tuvo como objetivo principal el desarrollo en software de los módulos BICM y LDM de ATSC 3.0, con el fin de contar con una herramienta que permita el estudio de las nuevas técnicas empleadas por este estándar.

Para la implementación de los módulos, primero se realizó una investigación sobre la capa física del estándar, y debido a la amplia flexibilidad del estándar, se decidió implementar algunos modos de operación, dando prioridad a los modos catalogados como obligatorios.

A continuación se enlistan los 4 bloques desarrollados dentro de esta tesis y se describen los modos de operación implementados para cada bloque. Los primeros 3 bloques (Codificador LDPC, Intercalador de bits y Modulador) constituyen lo que se denomina bloque BICM, el bloque LDM es el multiplexor en potencia.

1. Codificador LDPC

    a) Tipo de estructura LDPC implementada: IRA.

    b) Tasas de codificación implementadas: 6/15, 7/15, 8/15, 9/15, 10/15, 11/15, 12/15 y 13/15.

    c) Longitudes de código implementadas: 64800 bits y 16200 bits.

2. Intercalador de bits

    a) Tipo de intercalador implementado: A.

3. Modulador

    a) Constelación QPSK uniforme. Se utiliza la misma constelación para todas las todas las tasas de codificación.

b) Constelación 16-QAM no uniforme: se implementó para todas las tasas de codificación desde 2/15 hasta 13/15. En total se tienen 12 constelaciones 16-QAM, una para cada tasa de codificación.

c) Constelación 64-QAM no uniforme: se implementó para todas las tasas de codificación desde 2/15 hasta 13/15. En total se tienen 12 constelaciones 64-QAM, una para cada tasa de codificación.

d) Constelación 256-QAM no uniforme: se implementó para todas las tasas de codificación desde 2/15 hasta 13/15. En total se tienen 12 constelaciones 256-QAM.

4. Multiplexor en nivel de potencia

a) Se implementaron todos lo niveles de inyección de 0 dB a 25 dB.

b) Se implementaron todos los factores de escalamiento y normalización de acuerdo al estándar.

El bloque LDM o multiplexor en potencia se implementó de manera completa de acuerdo al estándar.

La implementación de los bloques anteriores permitió la obtención de distintos PLPs con diferentes configuraciones, según todas las combinaciones que puedan obtenerse.

Apéndice A

Instalación de GNU Radio mediante Pybombs

Instalación recomendada cuando el objetivo principal es crear módulos OOT:

```
$ sudo apt-get update

$ sudo apt-get install python-pip

$ sudo apt-get install git

$ sudo pip install --upgrade git+https://github.com/gnuradio/pybombs.git

$ pybombs recipes add gr-recipes git+https://github.com/gnuradio/gr-recipes.git

$ pybombs recipes add gr-etcetera git+https://github.com/gnuradio/gr-etcetera.git

$ pybombs prefix init /path/to/prefix -a myprefix -R gnuradio-default

path to prefix puede ser /usr/local/ , /usr/local/.pybombs o /home/usuarioX/.pybombs

$ sudo ldconfig

$ cd /path/to/prefix

$ . ./setup_env.sh

$ sudo gnuradio-companion
```

Bibliografía

[1] J. Montalban, I. Angulo, C. Regueiro, Y. Wu, L. Zhang, S. Park, J. Lee, H. Kim, M. Velez y P. Angueira, «Performance Study of Layered Division Multiplexing Based on SDR Platform», *IEEE Transactions on Broadcasting*, vol. 61, n.º 3, 2015.

[2] Advanced Television Systems Committee, *ATSC Standard: ATSC 3.0 System (A/300)*. Washington, D.C., EUA: ATSC, 2017.

[3] ——, *ATSC Standard: Physical Layer Protocol (A/322)*. Washington, D.C., EUA: ATSC, 2016.

[4] L. Fay, L. Michael, D. Gómez-Barquero, N. Ammar y W. Caldwell, «An Overview of the ATSC 3.0 Physical Layer Specification», *IEEE Transactions on broadcasting*, vol. 62, n.º 1, págs. 159-171, 2016.

[5] K.-J. Kim, S. Myung, S.-I. Park, J.-Y. Lee, M. Kan, Y. Shinohara, J.-W. Shin y J. Kim, «Low-Density Parity-Check Codes for ATSC 3.0», *IEEE Transactions on broadcasting*, vol. 62, n.º 1, págs. 189-196, 2016.

[6] L. Michael y D. Gómez-Barquero, *Modulation and Coding for ATSC 3.0*. Gante, Bélgica: Broadband Multimedia Systems y Broadcasting (BMSB), IEEE International Symposium on, 2015.

[7] S. I. Park, J.-Y. Lee, S. Myoung, L. Zhang, Y. Wu, J. Montalbán, S. Kwon, B.-M. Lim, P. Angueira, H. M. Kim, N. Hur y J. Kim, «Low Complexity Layered Division Multiplexing for ATSC 3.0», *IEEE Transactions on Broadcasting*, vol. 62, n.º 1, págs. 233-243, 2016.

[8] Advanced Television Systems Committee, *ATSC Standard: Link-Layer Protocol (A/330)*. Washington, D.C., EUA: ATSC, 2016.

[9] ——, *ATSC Standard: Video HEVC With Amendments No. 1 and No. 2*. Washington, D.C., EUA: ATSC, 2018.

[10] V. M. Dionísio y C. Akamine, «Comparison of Terrestrial DTV Systems: ISDB-TB and ATSC 3.0», *SET International Journal of Broadcast Engineering*, vol. 3, n.º 1, págs. 8-14, 2017.

[11] Advanced Television Systems Committee, *ATSC Standard: A/342 Part 2, AC-4 System*. Washington, D.C., EUA: ATSC, 2017.

[12] ——, *ATSC Standard: A/342 Part 3, MPEG-H System*. Washington, D.C., EUA: ATSC, 2017.

[13] L. Fay, *ATSC 3.0 Physical Layer Overview*. Gante, Bélgica: Broadband Multimedia Systems y Broadcasting (BMSB), IEEE International Symposium on, 2015.

[14] X. Zhuo, G. Shengyong e Y. Fuqiang, «A NOVEL LOW-TIME-DELAY CONVOLUTIONAL INTERLEAVER AND ITS PERFORMANCE», *Information and Communications Technologies, IET International Conference on*, 2013.

[15] S. Bhada, P. Gulhaneb y A.S.Hiwalec, «PAPR Reduction scheme For OFDM», *Procedia Technology, SciVerse Science Direct,*, vol. 4, págs. 109-113, 2012.

[16] *Out Of Tree Modules*, 2018. dirección: https://wiki.gnuradio.org/index.php/OutOfTreeModules.

[17] *PyBOMBS*, 2018. dirección: https://pypi.org/project/PyBOMBS/.

[18] *Blocks Coding Guide*, 2018. dirección: https://wiki.gnuradio.org/index.php/OutOfTreeModules.

[19] *Guided Tutorial GNU Radio in C++*, 2018. dirección: https://wiki.gnuradio.org/index.php/Guided_Tutorial_GNU_Radio_in_C%5C%2B%5C%2B.

[20] *Codificación para el control de errores en Radios Definidos por Software*, 2018. dirección: http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/8823/Tesis.pdf?sequence=1.

[21] *Construcción de Módulos y Bloques en GNU Radio Companion*, 2018. dirección: http://eontool.github.io/blog/2013/08/20/construccion-de-modulos-y-bloques-en-gnu-radio-companion/.

[22] B. Leiner, *LDPC Codes a brief Tutorial*. Wien, Austria: Leiner, 2005. dirección: http://www.bernh.net/media/download/papers/ldpc.pdf.

[23] M. Baldi, *QC-LDPC Code-Based Cryptography*. Nueva York, EUA: Springer, 2014.

[24] Z. Li, L. Chen y L. Zeng, «Efficient encoding of quasi-cyclic low-density parity-check codes», *IEEE Transactions on Communications*, vol. 54, n.º 1, págs. 71-81, 2006.

[25] W. C. Huffman y V. Pless, *Fundamentals of Error-Correcting Codes*. Nueva York, EUA: Cambridge University Press, 2003.

[26] A. H. Khan y D. K. C. Roy, «Comparison of Turbo Codes and Low Density Parity Check Codes», *IOSR Journal of Electronics and Communication Engineering*, vol. 6, n.º 6, págs. 11-18, 2013.

[27] M. Eroz, F.-W. Sun y L.-N. Lee, «DVB-S2 low density parity check codes with near Shannon limit performance», *International Journal of Satellite Communications Network*, n.º 22, págs. 269-279, 2004.

[28] H. Ge, *Investigation of LDPC code in DVB-S2*. Sweden: Department of Electrical Engineering Linköpings Universitet, 2012.

[29] *A DVB-T2 transmitter for GNU Radio*, 2018. dirección: https://github.com/drmpeg/gr-dvbt2.

[30] *GNU Radio*, 2018. dirección: https://github.com/gnuradio/gnuradio/tree/master/gr-dtv/lib/dvb.

[31] V. M. Dionísio y C. Akamine, «ATSC 3.0 implementation in GNU Radio Companion», *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, 2017.

[32] J. Montalban, I. Angulo, C. Regueiro, Y. Wu, L. Zhang, S.-I. Park, J.-Y. Lee, H. M. Kim, M. Velez y P. Angueira, «Performance Study of Layered Division Multiplexing Based on SDR Platform», *IEEE Transactions on Broadcasting*, vol. 61, n.º 3, págs. 436-444, 2015.

[33] L. Zhang, W. Li, Y. Wu, X. Wang, S.-I. Park, H. M. Kim, J.-Y. Lee, P. Angueira y J. Montalban, «Layered-Division-Multiplexing: Theory and Practice», *IEEE Transactions on Broadcasting*, vol. 62, n.º 1, págs. 216-232, 2016.

[34] L. Zhang, W. Li, Y. Wu, K. Salehian, P. Angueira, J. Montalban, H. M. Kim, S.-I. Park, J.-Y. Lee y X. Wang, «Two-Layer Mobile Service Performance in LDM-based ATSC 3.0 System», *Broadband Multimedia Systems and Broadcasting (BMSB), IEEE International Symposium on*, 2016.

[35] C. Regueiro, J. Montalban, J. Barrueco, M. Velez, P. Angueira, Y. Wu, L. Zhang, S.-I. Park, J.-Y. Lee y H. M. Kim, «LDM Core Services Performance in ATSC 3.0», *IEEE Transactions on Broadcasting*, vol. 62, n.º 1, págs. 244-252, 2016.

DECLARACIÓN

La información presentada en este trabajo se obtuvo de diversas fuentes que se consideran fidedignas y se consignan puntualmente en las referencias. El uso dado a la información es de naturaleza estrictamente de investigación académica y de divulgación, sin fines de lucro o de otra índole. Se ha hecho también el mayor esfuerzo por acreditar debidamente datos, opiniones y contenidos presentados, por lo que cualquier error u omisión en ello, es del todo involuntario.

Ciudad de México, Octubre 2018

Viridiana Mares Rodríguez