PFC. Proyecto final de carrera

SIMULADOR DE LA CAPA FÍSICA DE DVB-T PARA LAVICAD

Autor: Carlos S. Marcos Fernández

Directores: Margarita Cabrera Bean y Francesc Rey Micolau

Departamento de Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería de Telecomunicaciones de Barcelona
Universidad Politécnica de Cataluña

Barcelona, Julio 2008

Agradecimientos

Son muchas las personas que me han ayudado y dado soporte en la realización de este Proyecto.

En primer lugar, dar las gracias a Margarita Cabrera Bean y Francesc Rey Micolau, directores del proyecto, por la ayuda, motivación y seguimiento que me han proporcionado a lo largo de todo este tiempo, estando siempre presentes en el momento que les necesitaba.

Mi más sincero agradecimiento a Pedro Espinosa, por su inestimable ayuda a lo largo de la implementación Java del proyecto, y por hacer más amenas las tardes de trabajo.

Gracias a los hermanos Carlos y Francisco Vargas, por la ayuda prestada en el empleo del aplicativo Contenedor, parte principal de su proyecto LaViCAD.

En el plano personal, a Sonia le agradezco estar ahí porque las largas horas de trabajo se hacen menos pesadas sabiendo que al final del día se encuentra ella.

No puedo dejar de agradecer a mis padres, Alberto e Isabel, todo lo que han hecho para que pueda llegar hasta este punto, cuantas veces se lo agradezca tantas se quedarán cortas. Mi hermano Alberto, también han aportado su granito de arena haciéndome reír cuando más lo necesitaba.

A mis compañeros de universidad, tanto por los momentos difíciles como los divertidos compartidos en el Campus Nord y fuera de él.

Y a mis amigos, que han soportado mi mal humor en momentos de estrés, y han sabido divertirse conmigo cuando lo necesitaba.

A todos, muchas gracias.

ÍNDICE GENERAL

Agradecimientos ... 3
1 Introducción ... 9
1.1 Motivación ... 11
1.2 Objetivos ... 12
1.3 Estructura de la memoria ... 14
2 Sistema DVB-T ... 15
2.1 DVB Project ... 15
2.2 Estándares DVB ... 17
2.2.1 ETSI ... 17
2.2.2 Estándares de transmisión ... 19
2.2.3 Otros estándares ... 22
2.3 DVB-T ... 23
2.3.1 Introducción al sistema DVB-T ... 23
2.3.2 Características del DVB-T ... 24
2.3.2.1 Especificaciones ... 24
2.3.2.1.1 Estándar ETSI EN 300 468 - Codificación de fuente ... 24
2.3.2.1.2 Estándar ETSI EN 300 744 ... 26
2.3.2.2 Implementación ... 27
2.3.2.2.1 Aspectos de transmisión del sistema DVB-T ... 28
2.3.2.2.2 Implementación del MPEG-2 ... 29
2.3.3 Despliegue del DVB-T y alternativas a nivel mundial ... 30
2.3.3.1 Sistemas de televisión digital terrestre ... 32
2.3.3.2 Comparativa DVB-T, ISDB-T y ATSC ... 34
2.3.4 Situación del DVB-T a nivel nacional ... 35
2.3.4.1 Especificaciones del sistema DVB-T ... 36
2.3.4.2 Frecuencias de empleo ... 37
2.3.4.3 Situación en Cataluña ... 40
2.3.4.3.1 Frecuencias de empleo ... 41
2.3.5 Futuro del DVB-T ... 42
2.3.5.1 DVB-T2 ... 42
2.3.5.2 Alta definición ... 43
3 Proyecto COM@WEB ... 45
3.1 Introducción ... 45
3.2 Formación mediante las TICs ... 46
3.2.1 Revolución docente ... 47
3.2.2 Necesidad de las TICs ... 48
3.2.2.1 ¿Por qué tenemos que integrar las TIC's? ... 48
3.2.2.2 Momentos clave de aplicación práctica de las TIC's. ... 48
3.2.3 Las TIC's en el sistema universitario español ... 48
3.2.3.1 Servicios TIC en la UPC ... 49
3.3 LaViCAD ... 49
3.3.1 Applets, la nueva docencia ... 50
3.3.2 Descripción funcional ... 51
3.3.3 Ejemplos de uso ... 53
3.3.4 Aplicativo Contenedor ... 53
4 Descripción del proyecto ... 57
4.1 OFDM ... 58
4.2 Especificaciones del sistema implementado ... 60

4.3 Simulador DVB-T 64
4.3.1 Etapa 1: Generación de la señal 66
4.3.1.1 Parámetros de entrada 66
4.3.1.2 Procesado de la señal 66
4.3.1.3 Resultados 69
4.3.1.4 Simulación 16 paquetes MPEG-2 TS 70
4.3.1.5 Retos de programación 71
4.3.1.6 Análisis de la etapa 71
4.3.2 Etapa 2: Codificación externa (Reed-Solomon) y entrelazado externo (Forney) 72
4.3.2.1 Parámetros de entrada 72
4.3.2.2 Procesado de la señal 72
4.3.2.3 Resultados 75
4.3.2.4 Simulación 16 paquetes transporte MPEG-2 77
4.3.2.5 Retos de programación 77
4.3.2.5.1 Retos en la programación MATLAB 77
4.3.2.5.2 Retos en la programación Java 79
4.3.2.6 Análisis de la etapa 80
4.3.3 Etapa 3: Codificador interno (convolucional) + Entrelazado interno + Mapping de símbolos 81
4.3.3.1 Parámetros de entrada 81
4.3.3.2 Procesado de la señal 82
4.3.3.3 Resultados 88
4.3.3.4 Simulación 16 paquetes transporte MPEG-2 90
4.3.3.5 Retos de programación 91
4.3.3.5.1 Retos en la programación MATLAB 91
4.3.3.5.2 Retos en la programación Java 94
4.3.3.6 Análisis de la etapa 94
4.3.4 Etapa 4: OFDM 96
4.3.4.1 Parámetros de entrada 96
4.3.4.2 Procesado de la señal 97
4.3.4.3 Resultados 103
4.3.4.4 Simulación 16 paquetes transporte MPEG-2 104
4.3.4.5 Retos de programación 105
4.3.4.5.1 Retos en la programación MATLAB 106
4.3.4.5.2 Retos en la programación Java 106
4.3.4.6 Análisis de la etapa 106
4.3.5 Etapa 5: Canal 107
4.3.5.1 Parámetros de entrada 107
4.3.5.2 Procesado de la señal 108
4.3.5.3 Resultados 111
4.3.5.4 Simulación 16 paquetes transporte MPEG-2 113
4.3.5.5 Retos de programación 114
4.3.5.5.1 Retos en la programación MATLAB 114
4.3.5.5.2 Retos en la programación Java 115
4.3.5.6 Análisis de la etapa 115
4.3.6 Etapa 6: Demodulación OFDM 117
4.3.6.1 Parámetros de entrada 117
4.3.6.2 Procesado de la señal 118
4.3.6.3 Resultados 120

4.3.6.4 Simulación 16 paquetes transporte MPEG-2 122
4.3.6.5 Retos de programación 123
4.3.6.6 Análisis de la etapa 124
4.3.7 Etapa 7: Demapping + Desentrelazador interno + Decodificador de Viterbi 125
4.3.7.1 Parámetros de entrada 125
4.3.7.2 Procesado de la señal 125
4.3.7.3 Resultados 129
4.3.7.4 Simulación 16 paquetes transporte MPEG-2 131
4.3.7.5 Retos de programación 132
4.3.7.5.1 Retos en la programación MATLAB 132
4.3.7.5.2 Retos en la programación Java 133
4.3.7.6 Análisis de la etapa 134
4.3.8 Etapa 8: Desentrelazado externo + Decodificador Reed-Solomon 137
4.3.8.1 Parámetros de entrada 137
4.3.8.2 Procesado de la señal 137
4.3.8.3 Resultados 139
4.3.8.4 Simulación 16 paquetes transporte MPEG-2 141
4.3.8.5 Retos de programación 142
4.3.8.5.1 Retos en la programación MATLAB 142
4.3.8.5.2 Retos en la programación Java 142
4.3.8.6 Análisis de la etapa 143
5 Simulador S_DVB MATLAB 145
5.1 Programación MATLAB del proyecto 145
5.1.1 Planteamiento inicial 146
5.1.2 Interfaz principal 148
5.1.2.1 Idioma 150
5.1.2.2 Sesiones 151
5.1.2.3 Ayuda del sistema completo 152
5.1.2.4 Mensajes de error 154
5.1.2.5 Salir del sistema 154
5.1.3 Estructura de las etapas 154
5.1.3.1 Interfície 155
5.1.3.2 Idioma 156
5.1.3.3 Validar 157
5.1.3.4 Procesado 158
5.1.3.5 Opciones 159
5.1.3.6 Cierra 160
5.1.3.7 Ayuda de la etapa 161
5.1.3.8 Funciones específicas de cada etapa 161
5.1.4 Funciones generales 162
5.1.4.1 Funciones gráficas 162
5.1.4.2 Funciones de ficheros 165
5.1.5 Pautas seguidas en la programación MATLAB 167
5.1.5.1 Estandarización empleada MATLAB 167
5.1.5.2 Recomendaciones de programación y optimización de código 167
5.1.6 Retos en la programación MATLAB 169
5.2 Despliegue de aplicaciones MATLAB 169
5.2.1 Aplicación standalone 169
5.2.2 Builder for JAVA 172

Apéndice A. Conceptos MATLAB del proyecto 174
6 Simulador S_DVB Java 181
6.1 Aplicativo Contenedor 181
6.1.1 Descripción 181
6.1.1.1 Arquitectura seleccionada 182
6.1.1.1.1 Interfaz de usuario: Java Swing 182
6.1.1.1.2 Ficheros de configuración XML 183
6.1.1.1.3 Diseño de gráficas: Ptolemy 185
6.1.1.2 Configuración 186
6.1.1.2.1 Multiidioma 186
6.1.1.2.2 Sistema 187
6.1.1.2.3 Etapa 189
6.1.1.2.3.1 Parámetros de entrada 189
6.1.1.2.3.2 Señales 192
6.1.1.2.3.3 Parámetros de salida 195
6.1.1.2.4 Ayudas 196
6.1.1.3 Tipos de datos 197
6.1.1.3.1 Intercambio de ficheros 197
6.1.1.3.2 Señales 199
6.1.1.4 Javadoc 200
6.1.1.5 Firma de applets 201
6.1.2 Despliegue, guía para realizar un nuevo sistema 201
6.2 Retos en la programación Java 205
6.2.1 Memoria de la consola de Java 206
6.3 Diferencias entre JAVA y MATLAB 206
Apéndice A. Lenguaje Java 208
7 Conclusiones 213
7.1 Líneas futuras de actuación 216
Acrónimos 221
Bibliografía 225

1 Introducción

En la sociedad actual llamada del conocimiento, son muchos y diversos los ámbitos donde se aplican las tecnologías de la información y de la comunicación (TIC). En el ámbito educativo, actualmente, las TIC se están utilizando de manera generalizada en todas las universidades, y están permitiendo, entre otras cosas, poder aprender y enseñar eludiendo las barreras del espacio y del tiempo que las aulas imponen.

Las aportaciones de las TIC a la enseñanza universitaria son las siguientes:

- ✓ Diluyen los itinerarios de educación rígidos y permiten la adaptación de los estudios a las necesidades de los alumnos.
- ✓ Facilitan la comunicación entre profesores y alumnos dada la eliminación de las barreras espacio-temporales.
- ✓ Generan nuevos canales de comunicación entre los alumnos.
- ✓ Posibilitan el acceso a una gran cantidad de información, con gran rapidez y con un reducido coste.
- ✓ Potencian la formación permanente.

Tecnológicamente, las TIC son el soporte de la educación virtual que se brinda a través de Internet (e-learning). El proyecto de innovación docente COMWEB [21], en el que se incluye este proyecto fin de carrera, es una iniciativa de un grupo de profesores del departamento de TSC (*Teoria de la Señal i Comunicaciones)* de la UPC (*Universidad Politécnica de Cataluña*) orientado hacia ese tipo de enseñanza. Puede observarse en la Figura 1 la página principal del portal de COMWEB (comweb.upc.edu).

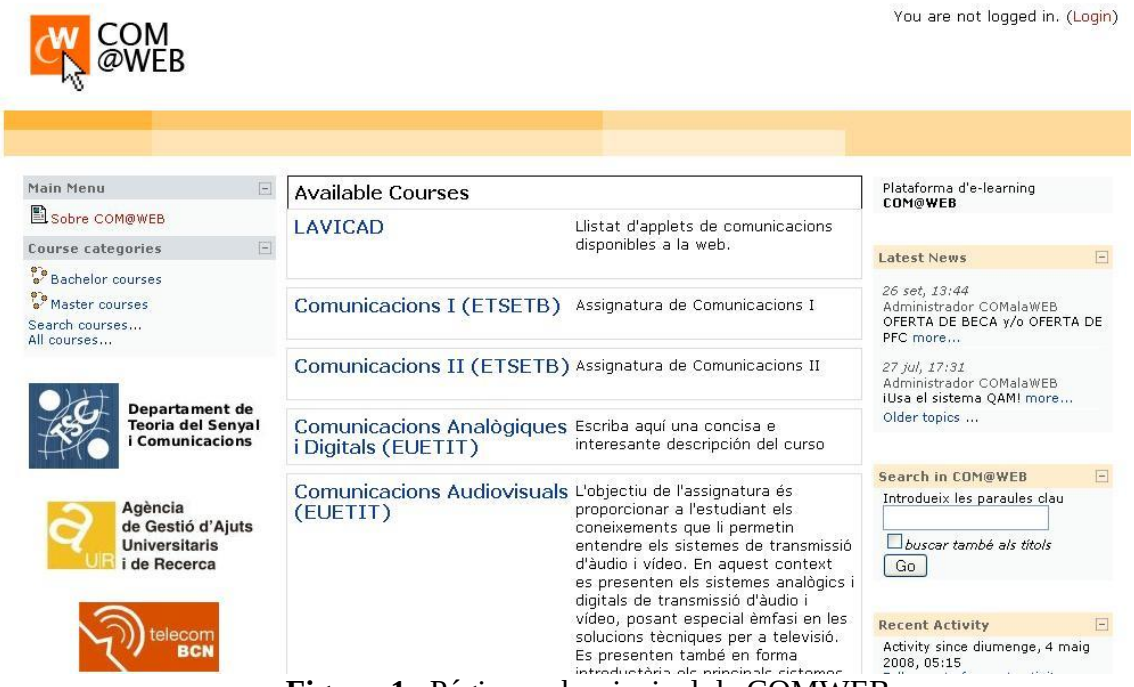


Figura 1 Página web principal de COMWEB

COMWEB significa Comunicaciones en la Web y es una plataforma de e-learning que ofrece diferentes recursos docentes on-line a los estudiantes universitarios con el objetivo de facilitar el estudio preferentemente en las temáticas de procesado de señal y comunicaciones. La plataforma es interactiva y se ha diseñado para promover la experimentación e-learning y adquisición de hábitos de estudio basados en el uso de nuevas tecnologías.

Uno de los cursos integrados en COMWEB es el Laboratorio Virtual de Comunicaciones Analógicas y Digitales LaViCAD [22], de libre difusión y acceso, y basado en la simulación de diferentes sistemas de comunicaciones que pueden ser utilizados tanto en docencia presencial como en educación a distancia. La página principal del proyecto LaViCAD se muestra en la Figura 2.

Figura 2 Página web principal de LaViCAD

El principal logro del proyecto LaViCAD, realizado como proyecto final de carrera por Carlos Vargas y Francisco Vargas bajo la dirección de Margarita Cabrera Bean, ha sido la creación de un aplicativo en lenguaje Java denominado Contenedor, mediante el cual se pueden desarrollar simuladores de cualquier sistema de comunicaciones para su libre difusión en la Red. LaViCAD es un puente entre los conocimientos teóricos de procesado de señal y dichas aplicaciones.

El proyecto a presentar consiste en un simulador de la capa física DVB-T integrado en LaViCAD, mediante el cual se desarrolla un sistema completo de comunicaciones. Dicho sistema supone un referente en la simulación de otros sistemas de comunicaciones que se puedan desarrollar dentro de la aplicación LaViCAD, como por ejemplo cualquiera de los considerados en estándares de aplicaciones de uso cotidiano, y un puente entre los conocimientos teóricos de procesado de señal y dichas aplicaciones.

1.1 Motivación

Internet, a través de las TIC, supone un salto de calidad en el ámbito de la docencia. La mayoría de los alumnos se sienten más motivados ante la posibilidad de poder utilizar como instrumentos de trabajo y formación herramientas accesibles a través de Internet. Podría afirmarse que los alumnos han crecido ya con esa aptitud y ofrecen cierta predisposición al uso de estos medios.

En este marco, se incluye el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD, en el que se integra el presente proyecto. Cabe destacar que experiencias del tipo de LaViCAD son de gran importancia en el entorno universitario actual debido a la diversidad del tipo de estudios en cuanto a presencialidad o no presencialidad y a la utilización de las TIC en las aulas y en los laboratorios docentes del entorno universitario español. Prácticamente en todas las universidades españolas se imparten algunos estudios que son semipresenciales o bien enteramente no presenciales. En las asignaturas implicadas, los alumnos deben realizar trabajos o asignaturas en su propia casa o lugar de estudio y, obviamente, en esta situación una aplicación como LaViCAD resulta de gran utilidad y soporte.

A nivel pedagógico, las ventajas que supone la utilización de un laboratorio virtual pueden resumirse en facilitar el aprendizaje de determinadas materias, actuando como un enlace entre el modelo de conocimiento basado en conceptos y teorías y su comprensión práctica, aumentando así la calidad de la enseñanza.

La motivación de este proyecto es doble. Por un lado existe la importancia de crear un sistema de comunicaciones de alto interés académico que pase a ser, junto con el sistema de modulaciones digitales QAM, el sistema más completo a nivel de capa física del laboratorio virtual LaViCAD. Por otra parte, interesa la visión más práctica enfocada al marco que engloba el sistema en una aplicación cotidiana como es la TDT, la cual revoluciona el concepto que hasta ahora se tenía de la televisión, proporcionando gran calidad de imagen y sonido, mayor poder de participación e interactividad.

Sobre el interés académico que despierta el sistema DVB-T, pueden distinguirse los amplios contenidos teóricos marcados por el estándar de la capa física *ETSI EN 300 744*, que hacen que sea un sistema muy completo para el estudio.

En el estándar, se describe el sistema de transmisión en banda base para la radiodifusión de TDT. La codificación de canal utilizada en el sistema está compuesta por la codificación externa, de tipo Reed-Solomon, y por la codificación interna, de tipo convolucional, que complementa a la anterior. Gracias a la redundancia introducida por el conjunto de los dos codificadores, se permite la corrección de errores en la recepción (*Forward Error Correction*-FEC) después de pasar por el canal de transmisión. Por otra parte, el esquema de modulación empleado en la transmisión es del tipo OFDM (*Orthogonal Frequency Division Multiplex*). El resultado, combinando el potente método de codificación para corrección de errores y la modulación multiportadora, es una transmisión COFDM (*Coded Orthogonal Frequency Division Multiplex*). La implementación de este tipo de transmisión de señal digital incrementa la motivación en la implementación del simulador del sistema DVB-T.

También cabe destacar, de cara al desarrollo de un applet integrado en un laboratorio virtual enfocado a la experimentación por parte del usuario, la flexibilidad del sistema a implementar, que permite la configuración de varios parámetros de transmisión.

La visión práctica del proyecto, focalizada en la TDT, también ha generado motivación de cara a la realización de este proyecto, debido a que ya es el presente, y será la televisión del futuro, gracias a todas las mejoras que aporta respecto a la televisión analógica. Algunas de las ventajas son las siguientes:

- ✓ *Más canales:* En el mismo espacio que ocupa un canal analógico se pueden ofrecer hasta cuatro canales digitales
- ✓ *Mejor imagen y sonido:* Se puede ver la televisión en formato panorámico, sin cortes, sin bandas negras, sin ruidos, sin interferencias ni doble imagen.
- ✓ *Más servicios:* Versión original y subtítulos, servicios interactivos de acceso a Internet, pago por visión, teletexto digital, guía electrónica de programación, visión multicámara para acontecimientos deportivos, etc.

1.2 Objetivos

El objetivo de este Proyecto Fin de Carrera consiste en el diseño e implementación del simulador de la capa física del sistema DVB-T, integrado en el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD [22] para su libre disposición en Internet.

Se trata de desarrollar un applet que simule la capa física del sistema DVB-T, siguiendo as especificaciones marcadas por el estándar *ETSI EN 300 744* [1]. Este simulador tiene como objetivo acercar al usuario el funcionamiento y especificaciones de un sistema de comunicaciones digitales utilizado en aplicaciones cotidianas, tales como la televisión digital terrestre (TDT), muy popular hoy en día.

Algunos de los applets que constituyen el soporte principal del proyecto LaViCAD, se han programado como paso previo en MATLAB. Éste es el caso del sistema DVB-T, a implementar en el presente proyecto. Así, el primer reto importante recae en la programación del applet simulador de la capa física del sistema DVB-T en MATLAB.

Un objetivo paralelo al desarrollo MATLAB del simulador es completar el *programa plantilla o contenedor* para los applets programados en MATLAB, definido en el sistema completo de modulaciones QAM (Quadrature and Amplitude Modulation) desarrollado por Margarita Cabrera, co-directora del proyecto.

El segundo gran reto del presente proyecto supone el desarrollo del sistema DVB-T como applet de Java. El interés de la implementación Java radica en que, al contrario de los applets programados en MATLAB que requieren la licencia oficial de Mathworks para su funcionamiento, un applet de Java puede ser ejecutado sin necesidad de licencia y puede colgarse en la web para ser ejecutado en modo remoto, con la única necesidad de que el usuario disponga de un navegador web con el plugin de Java instalado. Como herramienta genérica para la implementación Java, se empleará el *programa contenedor* desarrollado en el proyecto LaViCAD.

Independientemente del software sobre el que funcione la aplicación, el simulador deberá centrarse en mostrar los resultados y características más interesantes ofrecidos por la capa física del sistema DVB-T descrito en el estándar. A través de la simulación etapa a etapa, se ofrecerá al usuario una ligadura entre el procesado de señal teórico y los protocolos de comunicaciones digitales utilizados por la TDT. De este modo, se despierta el interés del usuario en profundizar sobre el procesado de señal relacionado con dichas aplicaciones y se le induce a relacionarlo con aspectos más teóricos aprendidos con enfoque académico.

Las pantallas principales de cada uno de los sistemas, MATLAB y Java, implementados se muestran en la Figura 3.

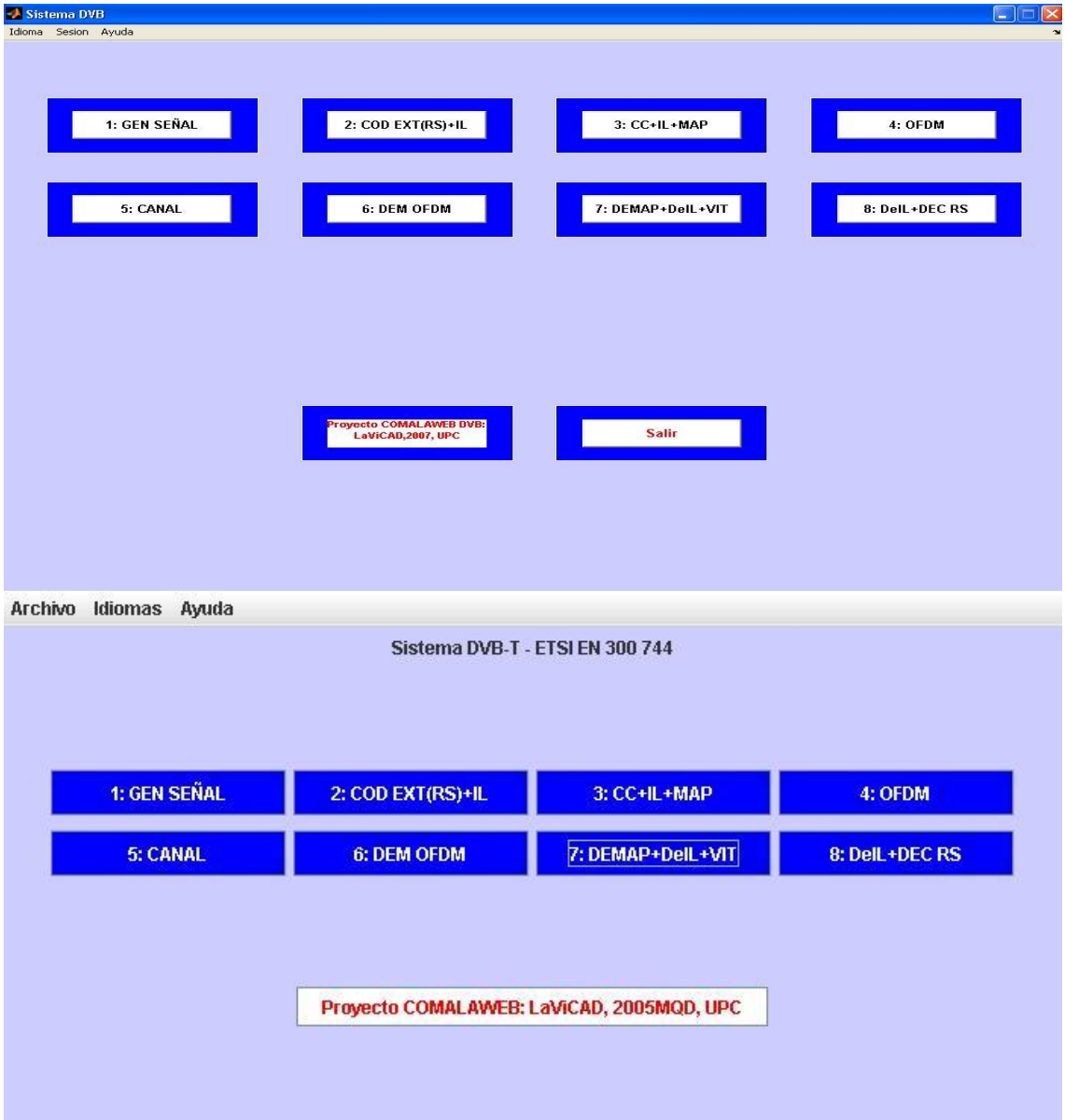


Figura 3 Pantalla inicial del sistema DVB-T. Implementación MATLAB (parte superior) e implementación Java (Parte Inferior)

1.3 Estructura de la memoria

La estructura que sigue la memoria realizada se divide en los siguientes capítulos:

En el capítulo primero, en el cual nos encontramos, se recoge la introducción del proyecto.

En el segundo capítulo se desarrolla la parte teórica del proyecto con un carácter divulgativo. Abarca desde una perspectiva muy general, hablando de la organización DVB y de los diversos tipos de estándares utilizados alrededor del mundo, hasta una perspectiva más detallada, centrándose en el estándar DVB-T, en el cual está basado el proyecto.

En el tercer capítulo se presenta el proyecto de innovación docente en el que se incluye este proyecto fin de carrera. Inicialmente se explica el proyecto global, COMWEB [21], y más adelante, el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD integrado en COMWEB.

En el cuarto capítulo se presenta la implementación del simulador de la capa física DVB-T, bajo las especificaciones del estándar *ETSI EN 300 744* [1]. En primer lugar se destacan las ventajas de la modulación OFDM y posteriormente se trata el simulador programado, describiendo etapa a etapa los parámetros de entrada, el procesado de señal teórico y los parámetros de salida. Asimismo, se ofrece la simulación de un ejemplo concreto, se presentan los retos de programación más destacables en ambos lenguajes de programación (MATLAB y Java) y se analizan ciertos aspectos relevantes de cada etapa.

En el quinto capítulo se analiza el simulador S_DVB de MATLAB. Se describe, en primer lugar, la estructura de programación empleada como *programa plantilla* o *contenedor* y, posteriormente, se explican las diferentes posibilidades para exportar la aplicación creada en MATLAB como un ejecutable o como un componente software. Al final del capítulo se proporciona un apéndice donde se presentan las características del software MATLAB empleadas.

En el sexto capítulo se analiza el simulador S_DVB de Java. Se describe, en primer lugar, el aplicativo Contenedor, parte principal del proyecto LaViCAD. Más adelante se presentan los retos afrontados en la programación Java, y ya por último se explican las diferencias principales experimentadas entre los lenguajes de programación MATLAB y Java. Al final del capítulo se proporciona un apéndice que ofrece una visión global de las características del lenguaje Java empleadas para el desarrollo del applet simulador de la capa física del sistema DVB-T.

En el séptimo capítulo se recogen las conclusiones del proyecto contrastadas con los objetivos y expectativas planteados al comienzo. También se enfatiza en las posibles futuras líneas de actuación de este proyecto.

2 Sistema DVB-T

Cuando se habla de TDT, se habla de radiodifusión digital utilizando las normas del *Digital Video Broadcasting* (DVB). En concreto, la norma empleada es la ETSI *EN 300744* [1], coloquialmente conocida como DVB-T.

Hoy en día el sistema DVB-T es muy popular, se halla en fase de implantación en muchos países y, según numerosos estudios, es una de las mejores alternativas para la televisión digital. Debido a ello, se ha considerado conveniente presentar algunos aspectos tanto de su estandarización, como de su implantación a nivel internacional.

El capítulo desarrolla, con un carácter divulgativo, la parte teórica del proyecto y presenta aspectos técnicos del estándar DVB-T. Abarca desde una perspectiva muy general, hablando de la organización DVB y de los diversos tipos de estándares utilizados alrededor del mundo, hasta una perspectiva más detallada, centrándose en el estándar DVB-T, en el cual está basado el proyecto.

Se informa al lector que, para completar la información de carácter divulgativo del capítulo, y para profundizar en las características técnicas de la capa física del estándar DVB-T, debe consultarse el capítulo 4. En él, se encuentra tanto la descripción del estándar *ETSI EN 300 744* [1], empleado por el simulador implementado, como las ventajas que aporta la modulación OFDM (*Orthogonal Frequency Division Multiplexing*) utilizada por el sistema.

Cabe mencionar, debido a que se presentan a lo largo del capítulo una serie de fechas de implantación de los estándares, que este capítulo se ha redactado a principios del año 2008, y puede quedar desfasado en unos años.

La temática del presente capítulo es la siguiente:

- En la primera sección, se habla del proyecto DVB.
- En la segunda sección, se explican los diversos estándares DVB existentes.
- Finalmente, en la sección tercera, se entra en detalle en el sistema DVB-T, el empleado en el desarrollo del proyecto, describiendo tanto sus características, como su situación a nivel nacional e internacional.

2.1 DVB Project

El estándar DVB-T forma parte de una gran familia de estándares publicados por el ETSI a partir de las especificaciones producidas por el DVB Project.

El DVB Project es una Alianza de alrededor de 250-300 empresas, inicialmente de origen europeo, pero actualmente con participantes de todo el mundo. Su objetivo es

acordar especificaciones para los sistemas de transmisión digital de datos, incluida la radiodifusión.

Hasta finales de 1990, la radiodifusión de televisión digital para el hogar se consideró impracticable y de costosa aplicación. Durante 1991, los organismos de radiodifusión y fabricantes de equipos de consumo discutieron el modo de formar una plataforma paneuropea para desarrollar la televisión digital terrestre. Hacia finales de ese año, los organismos de radiodifusión, fabricantes de electrónica de consumo y los órganos reguladores se reunieron para discutir la formación de un grupo que se encargaría de supervisar el desarrollo de la televisión digital en Europa, denominado European Launching Group (ELG).

Se redactó el memorando de entendimiento (*Memorandum of Understaning*, MoU) dónde se establecieron las normas por las que este nuevo y desafiante juego de acción colectiva se regiría. Este memorando fue firmado por todos los participantes del ELG en septiembre de 1993, y el grupo fue renombrado como el Digital Video Broadcasting Project (DVB).

Los miembros del proyecto DVB desarrollan las especificaciones que luego se envían al cuerpo de estandarización europeo para sistemas de comunicaciones, el EBU / CENELEC / ETSI Joint Technical Committee, para su aprobación. Las especificaciones, entonces, están formalmente estandarizadas por el CENELEC o, en la mayoría de los casos, por el ETSI.

- **DVB-S, DVB-C y DVB-T**

El sistema DVB-S para la radiodifusión por satélite se desarrolló en 1993. Se trata de un sistema relativamente sencillo, basado en la modulación de tipo QPSK. En las especificaciones se describieron diferentes herramientas para la codificación de canal y protección de errores que posteriormente serían utilizadas para otros sistemas de comunicaciones. El sistema fue acordado en 1994, y los primeros servicios de difusión DVB en Europa se iniciaron en la primavera de 1995 por el operador de televisión de pago Canalplus en Francia. Actualmente se utiliza en todo el mundo, aunque en algunos países, como Japón y los Estados Unidos, se utilizan otros sistemas digitales por satélite.

El sistema DVB-C para transmisión por cable se desarrolló en 1994. Se centra en el uso de la modulación 64 QAM, y en Europa puede, en caso necesario, transmitir un canal múltiplex completo de satélite en un canal de cable. La especificación DVB-CS describe una versión que puede ser utilizada para la antena de televisión por satélite. Este sistema es ampliamente utilizado en todo el mundo.

El sistema de televisión digital terrestre DVB-T es más complejo, porque inicialmente se tenía la intención de hacer frente a un ancho de banda y nivel de ruido diferente, y al fenómeno de propagación multicamino. El sistema tiene diversas posibilidades de transmisión, y el receptor tiene la obligación de adaptar su decodificación de acuerdo con la señal transmitida. El elemento clave es el uso de la modulación de tipo OFDM. Este sistema se acordó en 1997. Las primeras emisiones de DVB-T se iniciaron en Suecia y en el Reino Unido en 1998. Los servicios de DVB-T empezaron en algunas

partes de Alemania en 2002, y en el 2003 en Berlín se produjo el primer apagón analógico.

El sistema DVB-T es actualmente el menos extendido, porque el despliegue de la televisión digital terrestre en todo el mundo ha sido más lento que el del satélite y el cable. Se estima que se han distribuido más de 120 millones de receptores de DVB en todo el mundo.

- **El futuro del DVB Project**

El desarrollo de las comunicaciones se dirige hacia una mayor convergencia entre diferentes sistemas de comunicación, radiodifusión y punto a punto. El proyecto ya ha acordado un número de sistemas IP y orientados a la convergencia, por ejemplo DVB sobre redes IP. El futuro reside en la perfecta interoperabilidad entre los sistemas de telefonía digital y de radiodifusión digital, y en los entornos de red *in-home* emergentes. El Proyecto DVB debería ser capaz de utilizar su organización para ayudar a crear un mundo interoperable para sistemas convergentes.

Para más información sobre la organización DVB consultar referencia [2], donde se encuentra la dirección de su página web oficial.

2.2 Estándares DVB

Tal como se comenta en el anterior punto, el DVB Project desarrolla los estándares que posteriormente publica el ETSI. Todos estos estándares utilizan el estándar ISO MPEG-2, el cual se ha extendido para cubrir los detalles específicos del sistema y asegurar, de este modo, que está completamente especificado.

Todos estos estándares se deben agrupar por el método de transmisión utilizado.

2.2.1 ETSI

El Instituto Europeo de Estándares de Telecomunicaciones (ETSI) produce estándares aplicables a nivel mundial para Tecnologías de la Información y la Comunicación (TIC), incluyendo las tecnologías fija, móvil, radio, convergente, radiodifusión e Internet. Es oficialmente reconocido por la Comisión Europea como una Organización Europea de Estándares.

El ETSI es una organización sin fines de lucro con casi 700 organizaciones miembro procedentes de 60 países de todo el mundo.

La misión del ETSI es aprovechar las oportunidades en el desarrollo y despliegue de estándares aplicables a nivel mundial para las telecomunicaciones y otras redes de comunicaciones electrónicas y los servicios conexos, y participar en iniciativas mundiales y regionales adecuadas.

Las áreas estratégicas del ETSI, y sus funciones, son las siguientes:

- ***Productor mundial de estándares:*** proporcionar estándares para las telecomunicaciones y las redes de comunicaciones electrónicas y servicios conexos para el mercado mundial a través de la producción de productos del ETSI, especificaciones técnicas e informes para la aplicación mundial al tiempo que favorece la colaboración internacional.

- ***Organización europea de estándares (ESO):*** proporcionar estándares TIC para el mercado europeo, incluyendo la producción de estándares europeos en apoyo de la UE y la regulación EFTA y realizar su trabajo en colaboración con los demás ESOs y otros organismos europeos.

- ***Organización proveedora de servicios:*** prestación de servicios en el área de pruebas de interoperabilidad, alojamiento de foros y desarrollo de especificaciones de protocolo y testeo en apoyo de los miembros del ETSI y otras organizaciones de productores de estándares TIC para el mercado mundial.

- **Organismos técnicos**

El ETSI reconoce tres tipos de organismos técnicos:

- ***Comité técnico***
- ***Proyecto ETSI***
- ***Proyecto de asociación ETSI***

- **Los temas de trabajo (*work items*) y los resultados finales (*deliverables*)**

Cada comité técnico establece y mantiene un programa de trabajo, que consiste en los temas de trabajo. Un tema de trabajo ETSI es la descripción de una tarea de estandarización, y normalmente resulta un estándar único, un informe o un documento similar. El comité técnico aprueba cada tema de trabajo, que luego es adoptado formalmente por el conjunto de miembros (a través de un procedimiento web). En conjunto, los programas de trabajo de todas las comisiones técnicas constituyen el programa de trabajo ETSI.

Un comité técnico, por lo general, da la responsabilidad de un tema de trabajo a un pequeño grupo de expertos, dirigido por un Relator. El documento (estándar, informe, etc.) como resultado del tema de trabajo se menciona como resultado final del ETSI (ETSI Deliverable), que puede ser uno de los siguientes:

- ***Especificación técnica ETSI (TS):*** Un documento que puede contener texto normativo, es decir, texto obligatorio como “deberá”. Una TS es aprobada por el Comité Técnico de ETSI que propone el documento, y como tal, podría carecer de algunos de los de pesos asociados con las siguientes publicaciones. Una TS para el DVB es en general un paso hacia un documento(s) más estable(s).

- ***Informe técnico ETSI (TR):*** Normalmente un conjunto de directrices para la implementación de una especificación o estándar más normativo.

Un TR es aprobado por el Comité Técnico del ETSI que propone el documento.

- ***Estándar ETSI (ES):*** Documento aprobado por la totalidad de miembros del ETSI, y no sólo el Comité Técnico de la propuesta. Se trata de un documento más estable, que un TR o un TS.

- ***Guía ETSI (EG)***

- ***Estándar europeo (o Norma europea, EN):*** La publicación de más alta jerarquía ETSI aprobado por los organismos nacionales de estandarización de Europa. Un EN, es una publicación que puede estar y, a menudo, se incluye en la legislación europea y nacional.

- ***Informe especial ETSI (SR)***

- ***DVB Bluebooks:*** De vez en cuando, DVB publica documentos después de su aprobación por su Junta Directiva: la *BlueBooks*. En la práctica, son documentos comerciales, declaraciones políticas, o especificaciones técnicas que se están estandarizando.

La capacidad de producir estos diferentes tipos de documentos, permite responder a una variedad de necesidades dentro de las industrias a las que sirve. Muy de vez en cuando, el tema de trabajo no podrá dar lugar a ningún resultado final de los mencionados, en cuyo caso se le llama *Miscellaneous Item*.

Para más información consultar su página web. Ver referencia [3].

2.2.2 Estándares de transmisión

Los estándares de transmisión de DVB, con sus correspondientes documentos ETSI publicados, son los siguientes:

- ***DVB-S:***

El sistema DVB-S (*Digital Video Broadcasting by Satellite*) permite un espectacular incremento de la capacidad de transmisión de programas de televisión digital vía satélite utilizando las técnicas de compresión de video basadas en el estándar MPEG-2 para la codificación de fuente y multiplexación. La única variación entre este estándar y otros propuestos por el DVB (cable y radiodifusión terrestre), se encuentra en la capa física que estandariza el tipo de modulación y la codificación de canal empleadas. Para transmisiones vía satélite se adopta la modulación QPSK, con un flujo binario variable de 18,4 a 48,4 Mbits/s.

Para este estándar disponemos de los siguientes documentos ETSI:

- ***EN 300 421 V1.1.2:*** estructura de las tramas, codificación del canal y modulación de 11/12 GHz para servicios por satélite.

- ***TR 101 198 V1.1.1:*** implementación de la modulación BPSK para sistemas de transmisión satélite.

➢ ***DVB-S2:***

El estándar DVB-S2 (*DVB-Satélite versión 2*), constituye una evolución del estándar de satélite DVB-S e incluye una fuerte corrección contra errores basada en el empleo de dos codificadores en cascada, la denominada "*Low density Parity Check*" y la BCH, que le proporcionan una capacidad muy próxima a la fijada en el límite de Shannon.

Además, para aumentar la flexibilidad y permitir diversos servicios con diferentes velocidades binarias, se han habilitado varios esquemas de modulación (QPSK, 8PSK, 16APSK & 32APSK), varios factores de roll-off (0.2 / 0.25 / 0.35) y una adaptación flexible del flujo de entrada. El sistema es capaz de adaptar los parámetros de la capa física a las condiciones del canal de propagación aplicando técnicas adaptativas de codificación de datos (*Adaptive Coding and Modulation*-ACM).

La mejora de las capacidades de transmisión del estándar DVB-S2 sobre el estándar DVB-S se cifra alrededor de un 30%. Para lograr esta mejora, el DVB-S2 se ha beneficiado de los últimos avances en codificación de canal y modulación.

Publicaciones ETSI:

- ***EN 302 307 V1.1.2:*** estructura de las tramas, codificación de canal y modulación de sistemas de segunda generación para radiodifusión, servicios interactivos, periodismo electrónico y otras aplicaciones satélite de banda ancha.
- ***TR 102 376 V1.1.1:*** guía de usuario de sistemas de segunda generación para radiodifusión, servicios interactivos, periodismo electrónico y otras aplicaciones satélite de banda ancha.
- ***TS 102 441 V1.1.1:*** adaptación de código y modulación para aplicaciones híbridas (vía satélite y telefónica) de banda ancha

➢ ***DVB-C:***

DVB-C se refiere a la *Digital Video Broadcasting - Cable*. Es válido para cualquier tipo de red de cable.

Las características típicas de la transmisión por cable son la buena relación señal ruido, el pequeño espectro de frecuencias utilizable y rebotes y distorsión no lineal. La modulación seleccionada es la 64 QAM y el FEC (*Forward Error Correction*) es idéntico al del satélite.

Estándar europeo ETSI:

- ***EN 300 429 V1.2.1:*** estructura de las tramas, codificación de canal y modulación de sistemas de transmisión por cable.

- ***DVB-CS:***

Este estándar consiste en una adaptación entre DVB-S y DVB-C, para dar servicio a los sistemas de Satellite Master Antenna TV (SMATV).

Esta es la especificación que debe usarse en los sistemas de Antenas Colectivas (CATV) o en redes de TV por cable. Se usa para distribuir señales de televisión dentro de un mismo edificio o en edificios contiguos. Las señales son recibidas a través de la antena de recepción de satélite y se pueden combinar con las señales de TV terrestre.

El sistema SMATV representa la posibilidad de compartir los mismos recursos de varios usuarios para la recepción terrestre o por satélite. Además, permite la adaptación de las señales del satélite a las características del canal.

Documentos ETSI:

- ***EN 300 473 V1.1.2:*** Sistemas de distribución DVB *Satellite Master Antenna Television* (SMATV).
- ***TS 101 964 V1.1.1:*** Canal de control para sistemas de distribución SMATV / MATV; especificación base.
- ***TR 102 252 V1.1.1:*** Guía de usuario para el uso y la implementación del canal de control para sistemas de distribución SMATV / MATV.

- ***DVB-T:***

Es el estándar para la difusión terrenal de televisión, fue aprobado en Febrero de 1997 por el ETSI. Fue creado en base a unos requisitos del módulo Comercial Terrestre del Proyecto DVB, sus miembros contribuyeron al desarrollo técnico a través del DTTV-SA (Digital Terrestial Television- Systems Aspects).

Del mismo modo que el resto de los estándares de DVB, el DVB-T utiliza como método de codificación de audio y vídeo MPEG-2.

El sistema desarrollado en el proyecto, implementa la capa física del estándar DVB-T. En el apartado 2.3.2.1.2 del presente capítulo se presenta dicho estándar, y en el capítulo 4 de la memoria se estudia a fondo, junto al desarrollo del applet implementado.

La ETSI ha publicado la siguiente documentación:

- ***EN 300 744 V1.5.1:*** estructura de las tramas, codificación de canal y modulación para televisión digital terrestre (capa física).
- ***TR 101 190 V1.2.1:*** guía de implementación para servicios DVB-T; aspectos de transmisión.
- ***TS 101 191 V1.4.1:*** Mega-frame para sincronización SFN (Single Frequency Network).

- ***DVB-H:***

Sus siglas significan *Digital Video Broadcasting Handheld*, es un estándar abierto desarrollado por el DVB. La tecnología DVB-H constituye una plataforma de difusión IP orientada a terminales portátiles que combina la compresión de vídeo y el sistema de transmisión de DVB-T, estándar utilizado por la TDT (Televisión Digital Terrestre).

El DVB-H hace compatible la recepción de la TV terrestre en receptores portátiles alimentados con baterías. Es decir, DVB-H es una adaptación del estándar DVB-T adaptado a las exigencias de los terminales móviles.

Documentación:

- ***EN 302 304 V1.1.1:*** sistema de transmisión para terminales portátiles (*handheld*).
- ***TR 102 377 V1.2.1:*** guía de implementación para servicios DVB-H.
- ***DVB BlueBook A092r2:*** guía de implementación para servicios DVB-H (borrador de TR 102 377)
- ***TR 102 401 V1.1.1:*** Informe de validación DVB-H del grupo de refuerzo de tareas.

También se han publicado los estándares de transmisión siguientes:

- ***DVB-SH (Satellite services to Handhelds)***

- ***DVB-MDS (Multipoint Video Distribution Systems)***

- ***DVB-DSNG (Digital Satellite News Gathering)***

Todos los estándares nombrados se pueden descargar de la web oficial del ETSI (ver [3]).

2.2.3 Otros estándares

Existen muchas otras especificaciones producidas por el DVB, aparte de las mencionadas sobre los estándares de transmisión.

- ***Multiplexado:*** Especificaciones de la información de servicio (DVB-SI), difusión de datos (DVB-DATA), actualización del software del sistema (DVB-SSU), transporte y señalización de información de TV en flujos de transporte DVB (DVB-TVA) y encapsulación genérica (DVB-GSE).
- ***Codificación de la fuente:*** *G*uías de usuario sobre el DVB-MPEG.
- ***Subtítulos:*** Estándar europeo del DVB-SUB.
- ***Interactividad:*** Estándares sobre el canal interactivo para televisión digital terrestre, satélite, cable y para el resto de estándares de transmisión.
- ***Middleware:*** Se trata de un conjunto de servicios que permiten unir diferentes aplicaciones. Existe el MHP (*Multimedia Home Platform)* y el DVB-PCF (Formato de Contenido Portable).

- ***Contenido protegido:*** Para proteger el contenido se produjo el DVB-CPCM y sus elementos normativos.
- ***Interfaces:*** Se han publicado informes técnicos para diferentes redes y sistemas.
- ***Internet Protocol:*** El DVB-IPTV y el DVB-IPDC que tratan del protocolo de Internet para televisión y para envío de datos.
- ***Acceso condicional:*** Permitir el acceso a los abonados y prohibirlo a los no autorizados.
- ***Medición:*** tests y mediciones para DVB.

Se pueden descargar todos los estándares nombrados de la web oficial del ETSI (ver [3]).

2.3 DVB-T

Después de presentar el conjunto de estándares relacionados con el que se trata en este proyecto, el estándar DVB-T, a continuación se analizan en detalle sus características, su implementación alrededor del mundo (centrando la atención en España y Cataluña) y el presente y futuro del estándar.

2.3.1 Introducción al sistema DVB-T

El estándar DVB-T define una capa física y otra capa de enlace de datos de un sistema de distribución.

Los dispositivos interactúan con la capa física a través de un interfaz paralelo síncrono (SPI), un interfaz serie síncrono (SSI) o un interfaz serie asíncrono (ASI). Todos los datos se transmiten en flujos de transporte MPEG-2 con algunas restricciones adicionales (DVB-MPEG).

La capa física corresponde al nivel 1 del modelo OSI. En este nivel, se definen las características eléctricas, mecánicas y procedimentales de la comunicación en red. Es estrictamente necesaria su presencia en cualquier modelo. El nivel físico se refiere a las transformaciones que se hacen a la secuencia de bits para trasmitirlos de un lugar a otro. Es la capa de red más básica, proporcionando únicamente los medios para transmitir bit a bit sobre un enlace de datos físico conectando nodos de red.

La capa de enlace de datos es responsable de la transferencia fiable de información a través de un circuito de transmisión de datos. El nivel de enlace es el segundo nivel del modelo OSI. Recibe peticiones del nivel de red y utiliza los servicios del nivel físico. El objetivo del nivel de enlace es conseguir que la información fluya, libre de errores, entre dos máquinas que estén conectadas directamente (servicio orientado a conexión).

En la Figura 4 se puede observar el modelo de referencia de Interconexión de Sistemas Abiertos (OSI), con los 7 niveles existentes.

Figura 4 Pila OSI

2.3.2 Características del DVB-T

Se pueden encontrar todas las características completas del sistema DVB-T en su lista de estándares, publicados por la ETSI. Estos definen tanto las especificaciones del sistema de referencia en banda base (*Baseline System*), como la información de servicio (*SI*).

También se pueden estudiar los documentos que se presentan como una guía de usuario para la implementación del sistema. Estos últimos se pueden separar en aspectos de transmisión e implementación del MPEG-2.

En el resto de este capítulo se presenta un resumen de cada uno de ellos, para un estudio completo se recomiendan las referencias [1], [8], [9], [10], [11] y [12].

2.3.2.1 Especificaciones

Las especificaciones del sistema de DVB-T están recogidas en los estándares *ETSI EN 300 744* y en *ETSI EN 300 468*. En el primero de ellos está definido el sistema de referencia en banda base completo, y la información de servicio, añadida a las tramas MPEG-2 de entrada al sistema, se recoge en el segundo.

2.3.2.1.1 Estándar ETSI EN 300 468 - Codificación de fuente

Cómo en el resto de los estándares DVB, la señal de entrada normalizada es la denominada *"MPEG-2 Transport Stream (TS)"* o "Flujo de Transporte MPEG-2".

Dicho "Flujo de Transporte" (TS), obtenido mediante el procesado denominado "Codificación de Fuente" es una adaptación del estándar MPEG-2 según ISO/IEC

13818 ([9]), que se estructura multiplexando varios programas y añadiendo la "Información del Servicio" (SI) correspondiente (ver [8]).

ISO/IEC 13818 fue desarrollado en respuesta a la creciente necesidad de contar con un método de codificación genérica de imágenes en movimiento y sonido asociado para diversas aplicaciones, tales como medios de almacenamiento digital, radiodifusión televisiva y comunicación. El uso de esta especificación significa que el vídeo puede ser manipulado en forma de bits y se puede almacenar en diferentes medios de almacenamiento, transmitir y recibir sobre las redes existentes y futuras, y distribuir sobre los actuales y futuros canales de radiodifusión.

En el documento *ETSI EN 300 468* se especifica la "Información del Servicio" (SI) que forma parte de las tramas de bits DVB, con el fin de que el usuario pueda disponer de información de ayuda en la selección de servicios y / o eventos dentro de la trama de bits, y para que el Receptor Decodificador Integrado (IRD, o Set Top Box) pueda configurarse automáticamente para el servicio seleccionado. Los datos SI para configuración automática están principalmente especificados en la norma ISO / IEC 13818 ([9]) como la Información Específica de Programa (PSI).

En el documento se especifican otros datos que complementan la PSI, suministrando datos para ayudar al ajuste automático de IRDs, e información adicional destinada a mostrar al usuario. La forma de presentación de la información no está especificada en el documento, y los fabricantes de IRDs tienen libertad para elegir los métodos adecuados de presentación.

Se espera que las guías electrónicas de programas (EPG) sean una característica de las transmisiones de TV digital. Los datos que figuran dentro de la SI pueden utilizarse como base para una EPG.

En la Figura 5 se muestra un esquema de la codificación de fuente. El vídeo y audio comprimido, y las tramas de datos se multiplexan en tramas de programas (*PS, Programme Streams),* que se unen a su vez en el múltiplex de transporte para formar MPEG-2 TS (flujo de transporte MPEG-2). Ésta es la trama básica transmitida, y se recibe con el STB (Set Top Box). Este flujo es en el que viajarán un determinado número de canales de TV, de radio y servicios interactivos de forma simultánea. Además, también viaja información de señalización tal como horarios de programación o tipo de programa en emisión actual.

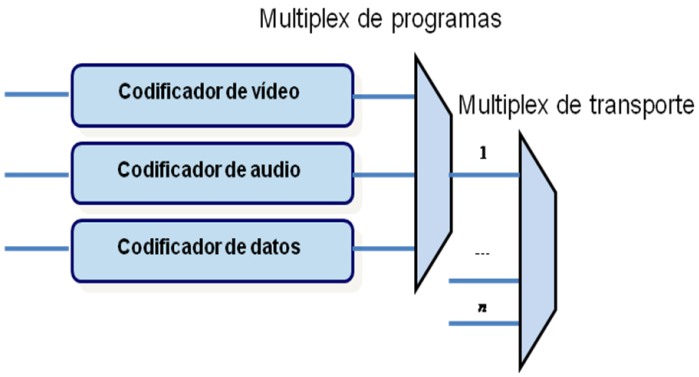


Figura 5 Codificación de la fuente y multiplexado MPEG-2

Una vez que el receptor está recibiendo un flujo TS sólo queda seleccionar un servicio en concreto, esto se realiza mediante el PID (*Packet ID*). El PID identifica un servicio dentro de un conjunto del flujo.

Las reglas para la implementación de la codificación de fuente se especifican en el informe técnico *ETSI TR 101211* ([12]).

2.3.2.1.2 Estándar ETSI EN 300 744

En este apartado se presenta el estándar de la capa física del sistema DVB-T. Dicho estándar se estudia en profundidad, junto al sistema DVB-T implementado, en la sección 4.3 de la memoria. En él, se describe un sistema en banda base de transmisión para la radiodifusión de Televisión Digital Terrestre (TDT).

En el documento se hallan las especificaciones que marcan los procesos necesarios de codificación de canal y de modulación para que, realizando el procesado pertinente de la señal en banda base, se pueda recibir la misma libre de perturbaciones cuando se usan los canales de transmisión terrestre.

El foco de estudio es el siguiente:

- ✓ Descripción general del sistema de referencia en banda base de televisión digital terrestre.

- ✓ Identifica los requisitos de realización global y las características del sistema de referencia en banda base, con el fin de satisfacer los objetivos de calidad del servicio.

- ✓ Especifica la señal modulada digitalmente con el fin de permitir la compatibilidad entre las piezas de equipos desarrollados por diferentes fabricantes. Esto se logra mediante la descripción detallada del procesado de señal en el modulador, mientras que el procesado en el receptor se deja abierto a la implementación de soluciones diferentes.

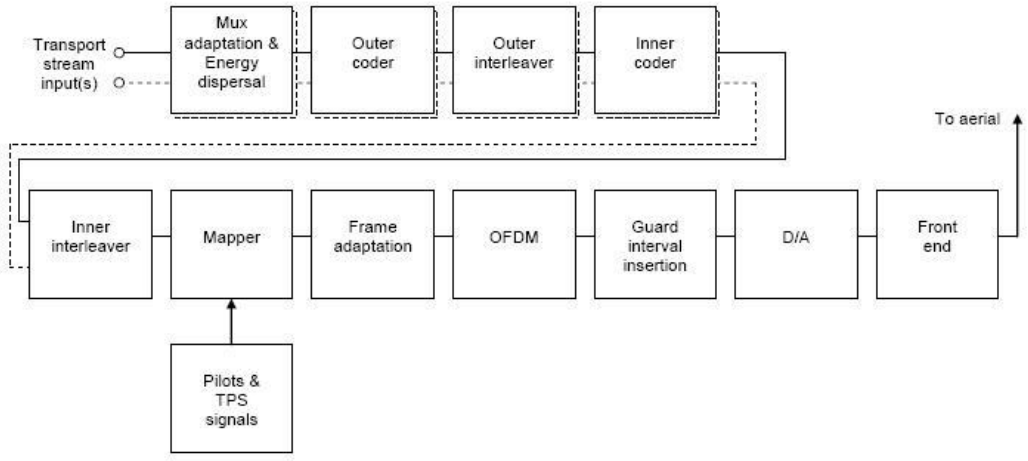


Figura 6 Adaptador de canal terrestre. Marcados con puntos los bloques funcionales para transmisiones jerárquicas. Figura extraída de [2].

En la Figura 6 se puede ver el diagrama de bloques del sistema DVB-T. Dichos bloques conforman la codificación de canal y el esquema de modulación empleados por el sistema.

La codificación de canal, es la secuencia de operaciones mediante la cual se añade suficiente redundancia y protección a la señal para hacerla más robusta con vistas a poder corregir los errores (Forward *Error Correction*-FEC) después de pasar por el canal de transmisión.

Por otra parte, el esquema de modulación usado en la transmisión es del tipo OFDM (*Orthogonal Frequency Division Multiplex*).

El resultado, combinando el potente método de codificación para corrección de errores y la modulación multiportadora, es una transmisión de tipo COFDM (*Coded Orthogonal Frequency Division Multiplex*).

El estándar de la capa física del sistema DVB-T es muy flexible, disponiéndose de una serie de opciones:

- ✓ Modos de transmisión: 2k (1.705 portadoras por símbolo OFDM); 8k (6.817 portadoras por símbolo OFDM)
- ✓ Esquemas de modulación: QPSK; 16-QAM; 64-QAM uniformes y no uniformes (parámetro de modulación α diferente de 1)
- ✓ Tasa de codificación para protección interna de errores: 1/2, 2/3, 3/4, 5/6 ó 7/8.
- ✓ Longitudes para el intervalo de guarda: 1/4, 1/8, 1/16 ó 1/32
- ✓ Modulación y codificación de canal jerárquica o no jerárquica

El sistema DVB-T permite transmisiones jerárquicas. En este caso hay 2 flujos de transporte, uno de ellos denominado de “alta prioridad” que modula las portadoras con un esquema de modulación muy robusto frente al ruido (QPSK), mientras que el segundo flujo, denominado de “baja prioridad”, combina su información con el anterior de forma que las portadoras son moduladas finalmente con un esquema más exigente en cuanto a relación señal/ruido.

El sistema estudiado, es directamente compatible con señales de televisión codificadas con MPEG-2 según ISO/IEC 13818 [9] y está especificado para canales de transmisión de 6, 7 y 8 MHz.

2.3.2.2 Implementación

A continuación se estudian las guías de usuario para la implementación de sistemas DVB-T. Primero, se presentan detallados los aspectos de transmisión a tener en cuenta para el correcto despliegue del sistema y, más tarde, el modo de implementar la señal MPEG-2.

2.3.2.2.1 Aspectos de transmisión del sistema DVB-T

La ETSI publicó el documento *ETSI TR 101 190* (ver [10]) para ayudar al correcto despliegue del sistema DVB-T dependiendo del entorno y otros aspectos diversos.

El documento ofrece las primeras directrices para la implementación de las redes de transmisión del DVB-T. Su principal intención es ser una guía para los aspectos de transmisión, mientras que los aspectos del receptor no han sido tratados. Incluye una descripción general de topologías de red para Redes de Frecuencia Única (SFN) y Múltiple (MFN), las posibilidades y limitaciones compartiendo lugares de transmisión con televisión analógica y un resumen de parámetros de planificación.

La distribución de señales a SFNs se trata a fin de dar directrices para las limitaciones particulares que la estructura de esta red implica a la distribución de señales.

- ***MPEG-2-TS multiplex***

Es la señal de entrada al transmisor, debe contener varios programas de televisión y posiblemente algunos programas de audio/datos solamente.

La especificación del estándar DVB-T (ver [1]) ofrece unas velocidades de bit dentro del rango de 4,98 a 31,67 Mbit/s. Por norma general, la mayoría de los programas no críticos pueden ser codificados satisfactoriamente con una velocidad de bit de 4 a 4,5 Mbit/s, mientras que para radiodifusiones importantes o material crítico (como los deportes) se necesita al menos 6 Mbit/s. Las mejoras en la codificación MPEG reducirán en los próximos años estos valores en un 10%. El audio estéreo debe ser codificado a 192 kbit/s al menos, y el multicanal debe estar entre 400 y 900 kbit/s.

La velocidad de bit del EPG puede ser de entre 0,25 y 0,5 Mbits, pero depende del API (Interfaz de Programación de Aplicaciones) escogido por el receptor.

Un programa de televisión comprende una o más PES (Tramas Elementales de Paquetes), conteniendo cada una de ellas una sola componente del programa codificada digitalmente, por ejemplo, vídeo o audio estéreo codificados.

En el TS también hay tablas de "Información del Servicio" (SI), dando detalles de los múltiplex y de la naturaleza de las distintas tramas elementales, información del control de acceso y canales de datos privados cuyo contenido no está especificado por MPEG (como el teletexto o los canales internos de los radiodifusores).

La TS se compone de una sucesión de paquetes, cada uno de 188 octetos, llamados "Paquetes de Transporte". Cada paquete lleva los datos relativos a una trama elemental. El MPEG especifica una protección contra errores, pero una protección adecuada como un código Reed-Solomon y un entrelazado de paquetes puede ser fácilmente aplicada a la TS para adaptarse a las características del error esperado por el medio de transporte.

La velocidad de bit de la TS está determinada por la aplicación. Un multiplexador MPEG inserta paquetes nulos, para adaptar la suma de las velocidades de bit de sus entradas a la velocidad de bit de salida requerida.

- ***MFN (Multiple Frequency Networks)***

Las redes convencionales de DVB-T consisten en transmisores con señales programadas individualmente y con radiofrecuencias individuales. Por lo tanto, se denominan redes de frecuencia múltiple (MFN). El hecho de que un número de transmisores se considera que pertenecen a una red específica es un asunto administrativo y no técnico. Con el fin de cubrir grandes zonas con una señal DVB-T se necesita un cierto número de canales de radiofrecuencia. El número de canales depende de la solidez de la transmisión, por ejemplo el tipo de modulación asociado a la tasa de transmisión del canal y el objetivo de la planificación (zona de cobertura total o la cobertura de zonas densamente pobladas solamente).

- ***SFN (Single Frequency Networks)***

En una SFN, todos los transmisores están modulados de forma sincronizada con la misma señal y radian en la misma frecuencia. Debido a la capacidad multicamino del sistema de transmisión (COFDM), las señales de varios transmisores que llegan a una antena receptora contribuyen constructivamente a la señal total deseada.

Sin embargo, el efecto limitador de la técnica SFN es el de las interferencias internas de la red. Si las señales de transmisores lejanos se demoran más de lo permitido por el intervalo de guarda, se comportan como ruido de interferencia y no como las señales deseadas. La fuerza de dichas señales depende de las condiciones de propagación, que pueden variar en función del tiempo. La interferencia interna de una SFN para un espaciado del transmisor determinado se reduce con la selección de un gran intervalo de guarda. Hay que señalar que el impacto de las señales retrasadas fuera del intervalo de guarda depende del diseño del receptor. Como regla empírica, para reducir con éxito la interferencia interna a un valor aceptable el tiempo del intervalo de tiempo de guarda debe permitir que una señal de radio se propague a lo largo de la distancia entre dos transmisores de la red.

2.3.2.2.2 Implementación del MPEG-2

A continuación se ofrece el enfoque de la guía para la implementación del MPEG-2 y de la "Información del servicio". Para un estudio detallado se recomienda consultar las referencias [11] y [12].

- ***Sistemas MPEG-2***

La guía para la implementación de los sistemas MPEG-2 presenta directrices para la codificación y decodificación usando la "Capa de Sistema" MPEG-2, codificación de vídeo y audio definidas en ISO/IEC 13818 ([9]).

Las directrices presentadas en *ETSI TR 101 154* [11] para IRDs están destinadas a representar una mínima funcionalidad que todos los IRDs de una clase particular están obligados a cumplir o superar. Es necesario especificar el mínimo de funcionalidad de un IRD para los parámetros básicos.

Los IRDs se clasifican en tres dimensiones del siguiente modo:

- ✓ "25 Hz" o "30 Hz", dependiendo de si las tasas de transmisión de las tramas de vídeo basadas en 25 ó 30 Hz son soportadas. Hay IRDs que soportan ambas tasas de transmisión.

- ✓ "SDTV" o "HDTV", definición estándar o alta definición. Si son capaces de decodificar alta definición, también lo son de decodificar definición estándar.

- ✓ "Con interfaz digital" o "*baseline*", dependiendo de si están o no destinados a ser utilizados con un dispositivo de almacenamiento, como una grabadora digital. Las capacidades de un IRD "*baseline*" son un subconjunto de las de un IRD con interfaz digital.

Para dar una definición completa de un IRD, las tres dimensiones deben ser especificadas, por ejemplo, IRD 25 Hz SDTV "baseline".

Cabe señalar que en los sistemas DVB el formato de la imagen de origen, el formato de imagen codificada y el formato de imagen de visualización no deben ser idénticos. Por ejemplo, puede difundirse material HDTV como SDTV. El IRD al recibir la imagen, podrá convertir la imagen descodificada a HDTV para su visualización en la TV de Alta Definición. Con la conversión adecuada, la calidad de la imagen de alta definición resultante puede ser muy similar a la de la fuente original.

Otra característica notable es que una sola TS puede contener programas destinados para más de un tipo de IRD.

- ➢ ***Información del servicio***

El documento que explica como implementar la "Información del Servicio" (SI) es el *ETSI TR 101 211* ([12]). Estas directrices están destinadas a ser normas altamente recomendadas para el uso de la sintaxis DVB SI, que se especifica en *ETSI EN 300 468* ([8]). Como tales, facilitan la eficiente y fiable implementación de la interacción básica por parte del usuario en el IRD.

Las normas se aplican a los organismos de radiodifusión, los operadores de redes así como los fabricantes.

La especificación de estas funciones no prohíbe a los fabricantes de IRDs incluir características adicionales. Las directrices no se refieren a características relacionadas con la interfaz de usuario o con EPG, esas cuestiones quedan al arbitrio del mercado.

2.3.3 Despliegue del DVB-T y alternativas a nivel mundial

El sistema DVB no es el único estándar de televisión digital, puesto que existen otras familias de estándares actualmente. Tres normas son las mayoritarias: el sistema europeo DVB, el estadounidense ATSC y el sistema japonés ISDB. Cada una de ellas tiene su propio estándar de televisión digital terrestre.

Además, en China se ha desarrollado la familia DMB. Inicialmente nació como un sistema de radio transmisión digital, para enviar contenido multimedia a dispositivos portátiles, como por ejemplo móviles. Actualmente ha salido un estándar para la televisión digital terrestre a raíz de esta familia, el DTMB, utilizado en China y en Hong Kong, y a través del cuál se transmitirán las Olimpiadas de Beijing 2008.

En la Figura 7 se puede observar el uso de los diferentes sistemas existentes de televisión digital terrestre alrededor del mundo.

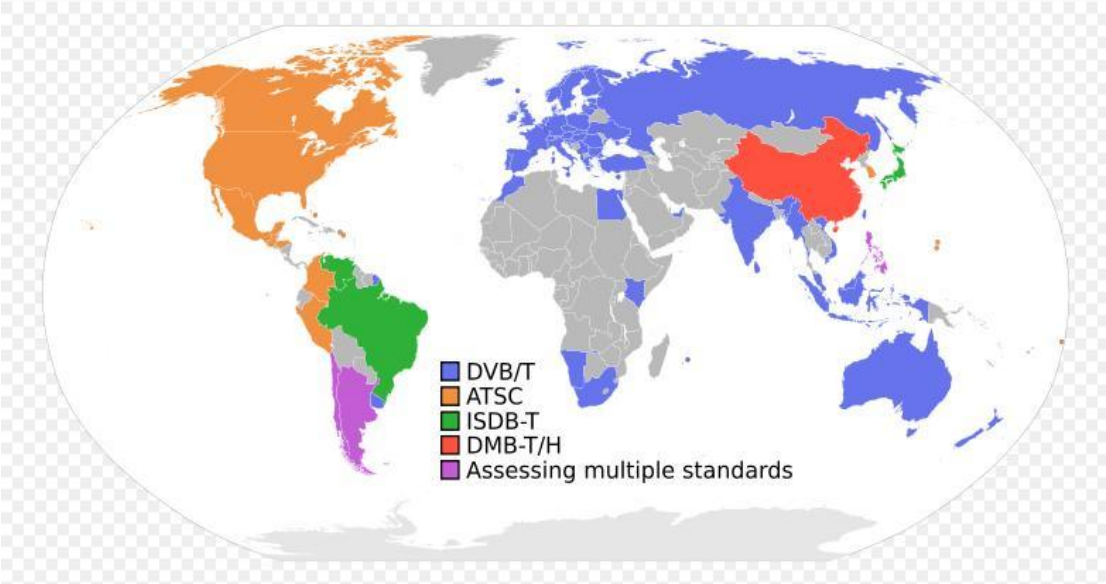


Figura 7 Estándares de televisión digital terrestre a nivel mundial. Figura extraída de [47].

En cuanto al despliegue del sistema DVB-T, en la Figura 8 se observa un mapa de Europa dónde aparecen los países que han lanzado ya la emisión de la televisión digital terrestre con el sistema DVB-T en color lila. En color gris aparecen los países que han seleccionado el estándar DVB-T, pero aún no lo han lanzado.

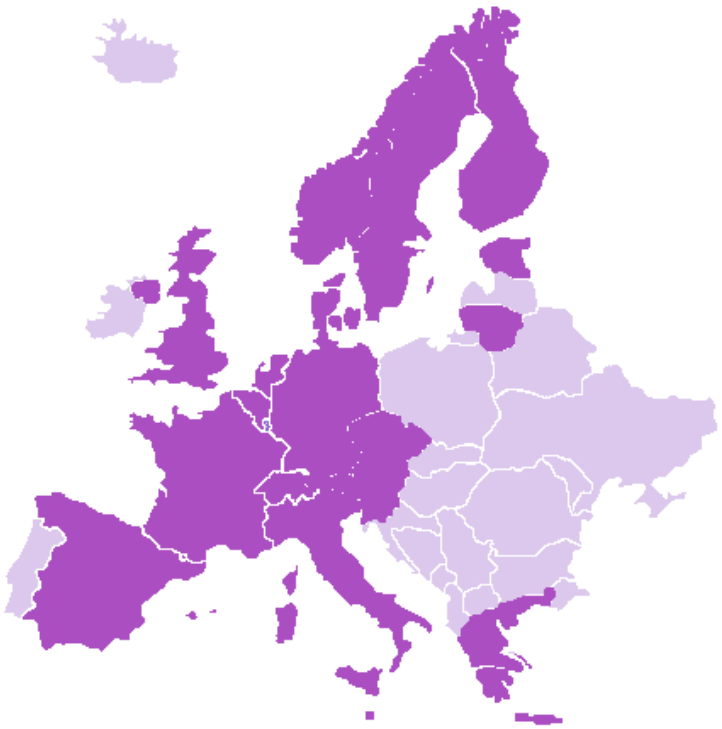


Figura 8 Mapa europeo del despliegue del DVB-T en el año 2007. Figura extraída de [13].

Esta figura ha sido extraída de la página web de DigiTAG (ver [13]), que es una organización que tiene por objetivo fomentar y facilitar la aplicación e introducción de servicios de televisión digital terrestre utilizando los estándares desarrollados por el DVB. DigiTAG cuenta con más de 50 miembros de la radiodifusión, los operadores de redes, de regulación, fabricación y organizaciones de toda Europa y más allá.

2.3.3.1 Sistemas de televisión digital terrestre

En este apartado se presentan las alternativas al sistema DVB-T a nivel mundial.

La tecnología usada es ATSC en Norte América, ISDB-T en Japón, DVB-T en Europa y Australia, y DTMB en China (incluyendo Hong Kong). El resto del mundo se mantiene indeciso.

- ***ATSC DTV***

El ATSC DTV ha sido desarrollado por el ATSC (*Advanced Television Systems Committee)*, que es una organización internacional sin ánimo de lucro en el desarrollo de normas voluntarias para la televisión digital. El ATSC crea y fomenta la implementación voluntaria de estándares y métodos recomendados para promover la radiodifusión de televisión digital terrestre, así como para facilitar la interoperabilidad con otros medios de comunicación.

El modulador del sistema ATSC DTV, de manera similar a los otros estándares, cumple dos funciones esenciales: primero, realizar la codificación de canal y segundo, la

modulación propiamente dicha. La señal de salida del modulador pasa a un conversor ascendente que la traslada a la frecuencia del canal de RF.

A la fecha actual, ha sido adoptado el estándar ATSC DTV (estándar terrestre) en los Estados Unidos, Corea del Sur, Canadá, México y Honduras. La transmisión ATSC HDTV comenzó en Costa Rica y Guatemala en 2006. Argentina, Chile, Perú, Colombia, Ecuador, la República Dominica y varios países en América Central están considerando emplear el estándar ATSC. Las condiciones para la adopción del estándar en el resto de Latinoamérica y el Caribe son excelentes. En EE.UU. y Canadá reemplazará el sistema NTSC.

También existen estándares de ATSC para transmisión por cable o por satélite. Cabe mencionar que se desarrolló inicialmente sólo para transmisión terrestre y recepción con equipos fijos, si bien en estos últimos años se viene trabajando en estándares para satélite y recepción terrestre con equipos móviles. El sistema de transmisión por satélite no se utiliza en Norte América puesto que llevan varios años utilizando el sistema DVB-S.

Los estándares de ATSC DTV incluyen la televisión de alta definición (HDTV), televisión de definición estándar (SDTV), radiodifusión de datos, audio multicanal de sonido envolvente, y televisión interactiva.

Para una información más detallada sobre el sistema ATSC y su uso a nivel mundial consultar referencias [4] y [5], que corresponden a la página web oficial de la organización ATSC, y el foro respectivamente.

- ***ISDB-T***

La Transmisión Digital de Servicios Integrados (*Integrated Services Digital Broadcasting*-ISDB) es el formato de televisión y radio digital que Japón ha desarrollado.

La ARIB (Asociación de Industrias y Negocios de Radiodifusión) es la entidad encargada de crear y mantener el ISDB, congrega a una multitud de empresas-japonesas y extranjeras-en el negocio de producir, financiar, fabricar, importar y exportar bienes de consumo relacionados con la radiodifusión. Los estándares se pueden obtener gratuitamente a través de la web de la organización japonesa DIBEG ([6]) y en ARIB ([7]).

En cuanto a la Radiodifusión Digital, el ARIB ha creado 3 estándares para su funcionamiento en Japón: El ISDB-T (televisión digital terrestre), ISDB-S (televisión digital satelital) e ISDB-C (televisión digital por cable). 1seg es el nombre de unos servicios ISDB-T para la recepción de ese tipo de señales en móviles, portátiles y vehículos. También disponen de especificaciones para difusión en la banda móvil *2.6 GHz band mobile broadcasting.*

Las transmisiones en el estándar ISDB-T comenzaron en Japón en diciembre de 2003 y tiene ciertas características que lo hacen más flexible que DVB. Brasil adoptó el estándar ISDB-T renombrándolo SBTVD-T (*Sistema Brasileiro de Televisão Digital-*

Terrestre). Éste difiere del sistema ISDB-T en que utiliza el códec de vídeo H.264 en lugar de MPEG-2.

Fuera de Brasil, hay algunos países, principalmente en el Mercosur, tal como Argentina (aunque ya había escogido el ATSC) y Venezuela que están estudiando el ISDB-T.

Los estándares de ISDB-T, al igual que ATSC DTV, incluyen la televisión de alta definición (HDTV).

- ***DTMB (Digital Terrestrial/Television Multimedia Broadcasting)***

Desarrollado en la República Popular China, aprobado en agosto de 2007, con características diferentes a los otros estándares tanto en el sistema de modulación como de codificación de canal y en el que se funden dos estándares previos también desarrollados en China, ADTB-T, similar al ATSC y desarrollado en la Universidad de Jiaotong de Shanghai y el DMB-T en la Universidad Tsinghua de Beijing.

El estándar es capaz de transmitir señales aceptables de alta definición a un receptor móvil con una velocidad de 200 km/h. El estándar también soporta el servicio en terminales portátiles, cosa que no ocurre con DVB-T ni ATSC. Además, el radio de cobertura es 10 km más grande que el europeo con DVB-T.

2.3.3.2 Comparativa DVB-T, ISDB-T y ATSC

- ✓ Los tres estándares no presentan diferencias importantes en cuanto a resoluciones de vídeo posibles, pudiendo todos operar en definición estándar y alta definición;

- ✓ Los tres estándares no presentan diferencias importantes en cuanto a la calidad de las señales de audio posibles;

- ✓ A diferencia de ATSC, ISDB-T y DVB-T tienen gran flexibilidad para configurar las transmisiones en función de tasas de datos deseadas y cobertura requerida, la que permite suavizar la transición desde TV analógica de definición estándar a TV Digital en alta definición;

- ✓ Dicha flexibilidad permite, además, que cada operador configure, en cada momento, sus transmisiones de acuerdo a su propio plan de negocios;

- ✓ A diferencia de ATSC, ISDB-T y DVB-T permiten organizar las transmisiones en flujos jerárquicos;

- ✓ Pruebas de terreno no son concluyentes en cuanto a que ATSC logra mejor cobertura que DVB-T, y que la cobertura de ISDB-T a tasas de datos comparables es levemente menor;

- ✓ DVB-T tiene la mejor inmunidad a propagación de multicamino;

- ✓ Dicha inmunidad por parte de ATSC requiere complejos diseños en los receptores;

- ✓ Mediante la introducción del estándar DVB-H, DVB-T ha resuelto sus limitaciones originales para transmisión a terminales portátiles, logrando desempeños equivalentes a ISDB-T;

- ✓ ATSC no tiene la capacidad de recepción en terminales móviles ni portátiles;

- ✓ Los tres estándares, operando en bandas de 6 MHz, presentan eficiencias espectrales similares;

- ✓ No obstante, la eficiencia espectral de DVB-T operando en bandas de 8 MHz es mayor;

- ✓ DVB-T operando en bandas de 8 MHz permite tasas de datos que pueden entregar transmisiones HDTV y SDTV simultáneas, facilitando de esta forma la introducción de HDTV;

- ✓ El tamaño del mercado mundial de equipos DVB-T es superior al de ATSC, y muy superior al de ISDB-T, garantizando gran variedad de dispositivos a precios competitivos;

- ✓ ISDB-T, además de poseer un mercado mundial pequeño, requiere de dispositivos (set-top-boxes y televisores) comparativamente más complejos que DVB-T;

2.3.4 Situación del DVB-T a nivel nacional

En este apartado se analiza el estado del sistema DVB-T en España.

Cabe mencionar que España fue un país pionero en las emisiones digitales, junto con Inglaterra. Pero por culpa de tres factores clave, se quedó en la cola en poco tiempo:

- ✓ La quiebra de la plataforma de pago Quiero TV, lo que dejó a la televisión digital terrestre sin apenas ningún producto diferenciador respecto a las emisiones analógicas.

- ✓ La falta de promoción del sistema digital, que ha tenido como resultado que la gente ignore la existencia de la tecnología y las ventajas que ofrece.

- ✓ La asociación errónea que suele hacer la población de los términos “digital” y “de pago”, cuando en realidad la TDT es gratuita.

A continuación se repasan las etapas que ha vivido el sistema cronológicamente:

- 1998 - 2000: Periodo de pruebas técnicas.

- 2000 - 2002: Periodo de actividad de Quiero TV, ofreciendo casi una veintena de canales, así como internet a través del televisor, lo cual supuso una oferta diferenciadora frente a Canal Satélite Digital y Vía Digital.
- Junio 2002 - Noviembre 2005: Tras el cierre de Quiero TV, existieron emisiones básicas con los mismos operadores nacionales que en analógico, añadiéndose a ellos Veo TV y Net TV, que en 2004 redujeron su cobertura de más del 80% al 25% de la población.
- Noviembre 2005 - Abril 2010: Reactivación de la TDT (lanzamiento de nuevos canales exclusivos, convocatoria de concursos para canales autonómicos y locales) que tiene como objetivo iniciar un periodo de transición hasta el apagón analógico.
- Desde el 3 de abril de 2010: Pleno funcionamiento de la TDT, con más del 95% de cobertura. Esta tecnología sustituirá entonces completa y definitivamente a la televisión convencional analógica, y se ampliará la oferta de canales y servicios.

Para más información sobre la situación nacional de la televisión digital terrestre, se recomienda visitar la página web del Ministerio de Industria, Turismo y Comercio dedicada a la TDT (ver [16]).

2.3.4.1 Especificaciones del sistema DVB-T

A nivel nacional, los parámetros elegidos del estándar de la capa física del sistema DVB-T son:

- ***Modo de transmisión 8K e intervalo de guarda de ¼:***
  - Permite redes de frecuencia única (SFN), siempre que la distancia entre emisores sea inferior a 68 km.
  - Problemas con la recepción en movimiento a velocidades elevadas.
- ***Cada portadora del COFDM se modula en 64QAM:***
  - Tasa binaria elevada, ya que se transmiten 6 bits por estado de modulación.
  - A cambio se necesita, para una correcta recepción, una mayor relación portadora-ruido (C/N) que si se modula en QPSK.
- ***Tasa de codificación de 2/3:***
  - Buena protección frente a errores para compensar la sensibilidad a las variaciones de fase y amplitud de la modulación 64 QAM.
- ***Modo no jerárquico:***
  - Por cada canal únicamente se transmite un múltiplex de forma que si el receptor consigue recibir, se verán todos los programas del múltiplex, y si no, no se verá ninguno.

- ***Banda UHF, canal de transmisión de 8 MHz:***

  - A diferencia de lo que ocurre con los canales de 8 MHz analógicos por donde se emite un único programa (tradicionalmente llamados canales de TV), desde el RD 2169/1998 se obliga a emitir por un canal de TDT de 8 MHz un múltiplex con 4 o más programas (o canales digitales) y sus datos asociados. Además cada programa puede estar formado por varios *streams* (vídeo, audio en varios idiomas, teletexto, subtítulos en distintas lenguas, aplicaciones interactivas, guías electrónicas de programación EPG, etc).

Con estos parámetros, se tiene una velocidad de bit útil de 19,91 Mbits/s, y se necesita una relación C/N de unos 18 dB para una recepción casi libre de errores (QEF).

2.3.4.2 Frecuencias de empleo

La extensa oferta de canales que oferta la TDT opera en frecuencias distintas a las que son utilizadas por las emisiones analógicas para no interferirse entre sí ambas tecnologías.

Actualmente en España disponemos de 5 múltiplex nacionales, y uno o dos regionales. Tanto SFN como MFN.

A continuación se muestra una división según el rango de frecuencias. Las redes de emisión dependen de este rango.

- ***830 a 862 MHz (canales 66 a 69):***

Red de frecuencia única de ámbito nacional; estas frecuencias están destinadas a albergar canales que operan en todo el país sin realizar desconexiones regionales entre Comunidades Autónomas.

En la Figura 9 se muestran las cadenas de televisión que emitirán en los canales estatales.

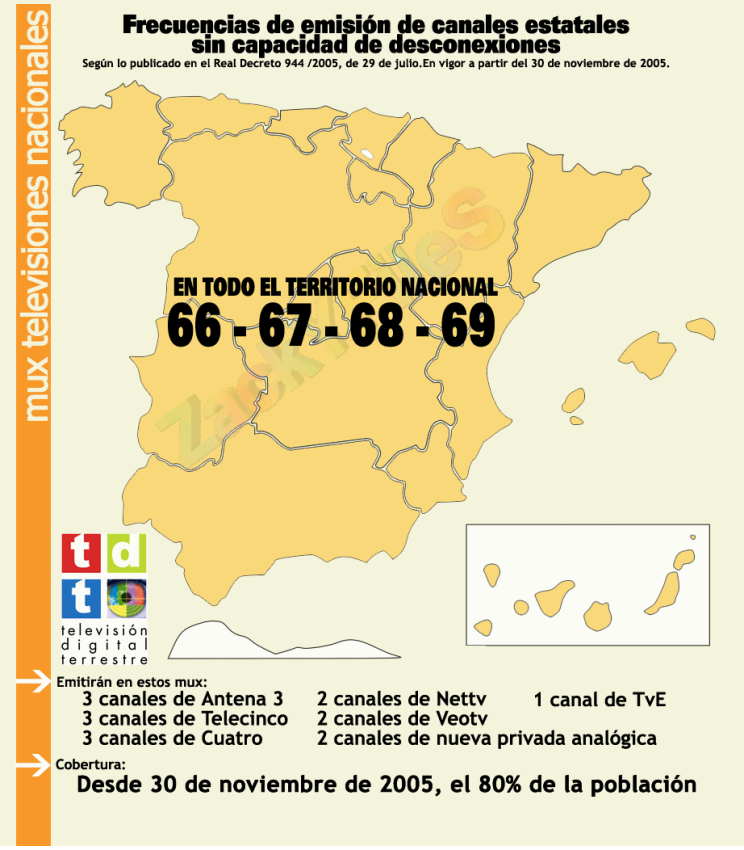


Figura 9 Frecuencias de emisión de canales estatales sin capacidad de desconexiones. Figura extraída de [45].

- ***758 a 830 MHz (canales 57 a 65):***

Redes de frecuencia única de ámbito territorial autonómico. Estos canales darán servicio con cobertura autonómica para canales de televisión autonómicos (algunos de ellos con capacidad para realizar desconexiones provinciales) y también para canales nacionales con capacidad de desconexión autonómica.

En la Figura 10 se observa la distribución de dichos canales a nivel provincial. Cada comunidad emitirá sus canales autonómicos en caso de tenerlos.

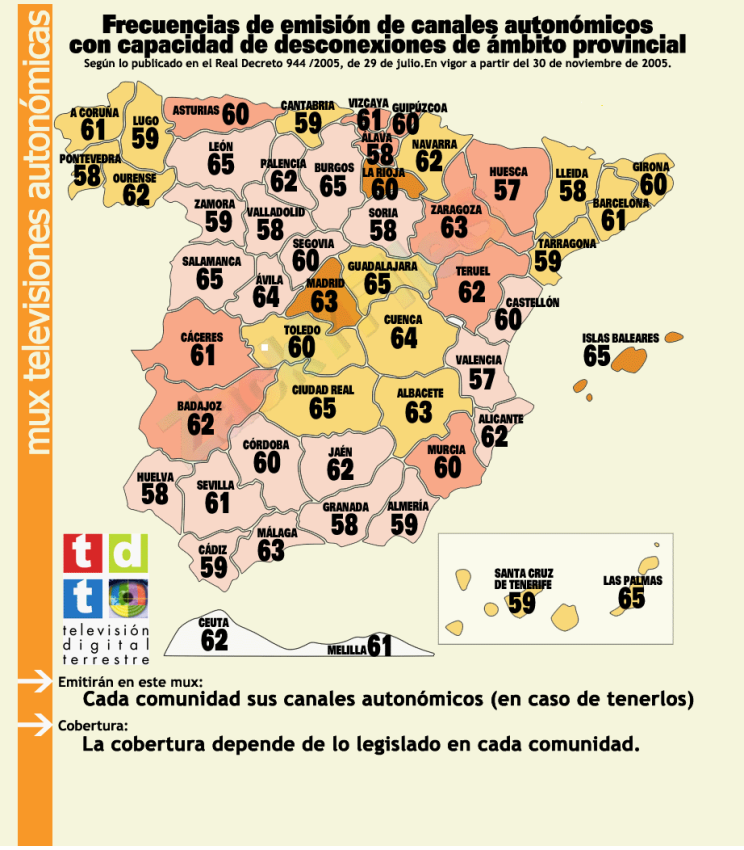


Figura 10 Frecuencias de emisión de canales estatales con capacidad de desconexiones de ámbito provincial. Figura extraída de [45].

En la Figura 11 se muestra la distribución de canales a nivel autonómico. Se trata del multiplexador de TVE.

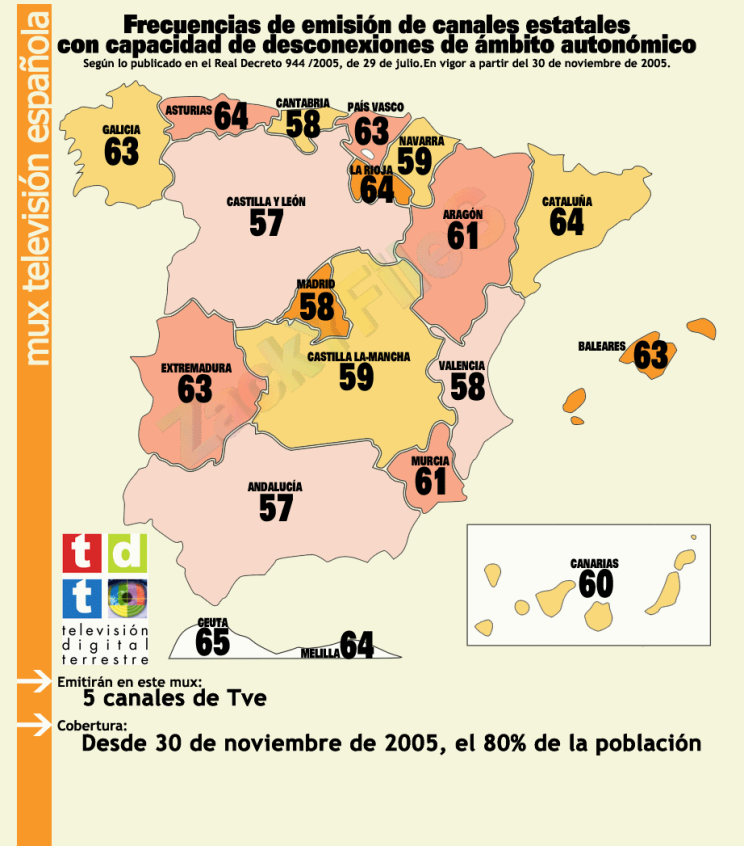


Figura 11 Frecuencias de emisión de canales estatales con capacidad de desconexiones de ámbito autonómico. Figura extraída de [45].

- ***470 a 758 MHz (canales 21 a 56):***

Red multifrecuencia y de transmisor único de cobertura local. Por estas frecuencias viajarán las televisiones locales que obtengan las licencias en las respectivas demarcaciones. Actualmente hay habilitadas también emisiones de ámbito insular, que cubren cada una de las islas que forman los archipiélagos canario y balear.

2.3.4.3 Situación en Cataluña

En Cataluña existen 30 programas de TDT agrupados en 8 múltiplex (y 1 con dos programas repetidos: TV3 y 3/24). De éstos, 21 programas son de licencia estatal española y están agrupados en 5 múltiplex. Los 9 programas restantes son autonómicos.

TVC dispone de 2 múltiplex, en uno de ellos emite señales de alta definición (43). El múltiplex 61 de Televisió de Catalunya y el 33 de Emissions Digitals de Catalunya es emitido por TRADIA.

El múltiplex 61 de TVC se emite por toda Cataluña, mientras que el 43 sólo se emite en Collserola. EDC está ampliando su cobertura.

De los 5 múltiplex estatales, uno (el de RTVE) se emite en frecuencias distintas en cada comunidad autónoma, así tiene la posibilidad de ofrecer desconexiones a nivel autonómico. Los otros (SFN) son de frecuencia única en todo el estado, por lo tanto

quiere decir que su contenido será siempre el mismo en todos los centros emisores. Los múltiplex de "Televisió de Catalunya" (TVC, canales 61 y 43) y "Emissions Digitals de Catalunya" (EDC, canal 33) tienen capacidad de ofrecer desconexiones provinciales, ya que se emiten en cada provincia por frecuencias diferentes.

Los 5 múltiplex estatales son emitidos por Abertis y sólo se radian por once centros emisores: Cabrils, Collserola, Collsuspina, Igualada, Montcada -sólo 67 y 68-, Montserrat y Sant Pere de Ribes (Barcelona); la Mussara, El Vendrell y Tortosa (Tarragona); Rocacorba y Calonge -sólo 67 y 68- (Girona) y Alpicat (Lleida).

La situación por provincias, agregando los canales locales, es la siguiente:

- Barcelona: 37 televisiones / 19 radios,
- Girona: 31 televisiones / 13 radios
- Lleida: 31 televisiones / 13 radios
- Tarragona: 31 televisiones / 13 radios

2.3.4.3.1 Frecuencias de empleo

Los canales de nivel nacional recibidos en Cataluña son el 64 (TVE), 66, 67, 68, y 69. Cada canal corresponde a una frecuencia tal como se puede ver en el apartado 2.3.4.2.

Los canales a nivel autonómico recibidos en cada provincia son:

- Barcelona: 61, 43, 33
- Girona: 60, 46, 36
- Lleida: 58, 42, 53
- Tarragona: 59, 62, 51

A continuación, se muestra un listado de los canales locales en Cataluña, viendo cuál se recibe en cada zona:

BARCELONA

- Barcelona : 26, 48
- Granollers : 40
- Hospitalet de Llobregat : 43
- Cornellá Llobregat : 36, 46
- Igualada : 37
- Manresa : 49
- Mataró : 24
- Sabadell: 39, 45
- Vic : 42
- Vilanova de Geltru : 30

GIRONA

- Blanes : 25, 42
- Figueres : 26
- Girona : 39

- Olot : 51
- Palafrugell : 25

LLEIDA

- Balaguer : 48
- Lleida : 50
- Seu Urgell : 55
- Vielha Mijaran : 51

TARRAGONA

- Reus : 56
- Tarragona : 54
- Tortosa : 34

Para más información sobre el ámbito de cobertura de cada canal múltiplex y la potencia radiada por cada uno, consultar [19].

2.3.5 Futuro del DVB-T

En el presente capítulo se analizan dos puntos clave en el futuro del DVB-T, que son el DVB-T2, cuyas especificaciones salen a la luz a fecha de Junio del 2008, y la alta definición.

Otro tema todavía pendiente es el apagón analógico. En muchos países europeos, la planificación del apagón analógico está bastante avanzada. Holanda se ha convertido en el primer país en poner fin a todos sus servicios de televisión analógica, mientras que otros países, como Alemania y Suecia, han iniciado el proceso de poner fin a los servicios analógicos a escala regional.

Se espera que la mayoría de los países de Europa el apagón tenga lugar en 2015, si no antes. La Comisión Europea ha recomendado a sus estados-miembro que completen el cierre analógico en 2012.

2.3.5.1 DVB-T2

Se trata de la segunda generación del DVB-T. La publicación de la especificación DVB-T2 se espera a finales de junio de 2008, con lo que la eficiencia en el uso del espectro mejore un 30-50% en comparación con DVB-T.

A continuación, se muestra un resumen de los requerimientos comerciales del sistema:

- ✓ DVB-T2 es diseñado para satisfacer los requerimientos de los radiodifusores después del apagón analógico.

- ✓ La prioridad será la multidifusión en HDTV, y obviamente el mejor rendimiento del espectro en multidifusión SDTV, móviles, interactividad y otros servicios personalizados.

- ✓ DVB-T2 debe ser compatible, en el uso de los sistemas de antenas de transmisión y recepción existentes.

- ✓ DVB-T2 proveerá una mayor velocidad de transmisión de manera compatible con las redes DVB-T de transmisores y antenas existentes.

- ✓ En SFN: coherentemente con la mayor velocidad de transmisión, DVB-T2 operará estas redes posibilitando mayor distancia entre repetidores.

2.3.5.2 Alta definición

Los servicios HDTV (*High Definition TeleVision*) están disponibles en muchos países de todo el mundo. Australia, Japón y Estados Unidos lideran en sus ofertas de televisión de alta definición con los servicios disponibles a través de todas las plataformas de televisión satélite, cable o terrestre.

En Europa, los servicios de televisión de alta definición han comenzado recientemente. El primer servicio se inició en 2004 por satélite y desde entonces un número creciente de operadores de satélite han puesto en marcha servicios de HDTV en varios mercados europeos, al igual que algunos de cable y operadores IPTV. Sin embargo, estos servicios han sido parte de una plataforma de pago con un limitado contenido disponible.

En este momento, los servicios de HD que están disponibles en la plataforma de televisión digital terrestre (HD / TDT) son limitados. En Suecia, está disponible en varias ciudades contenido HD del servicio público SVT. Mientras que el radiodifusor regional español, TV3, ofrece varias horas de HD al día en Barcelona. En Francia, la nueva estructura de los medios de comunicación, que fue aprobada en febrero de 2007, solicita la puesta en marcha de servicios de televisión de alta definición y el CSA (*Consejo Superior de l'Audiovisuel*) en junio de 2007 inició el proceso de invitar a los solicitantes a ofrecer dos servicios de televisión de alta definición a nivel nacional en el múltiplex digital R5. Las propuestas estaban obligadas a proporcionar un mínimo del 25% de servicios de televisión de alta definición en 2008, aumentando a un 30% en 2009 y así sucesivamente. Por Agosto, el CSA anunció que había recibido propuestas de cuatro posibles proveedores de servicios.

En muchos países, los servicios HD de TDT se han mostrado en un festival o evento deportivo como ha sido el caso en Francia, Italia, España, Suecia y el Reino Unido. Estas manifestaciones escaparate han permitido a los espectadores comprender mejor la mejora de la calidad ofrecida por la TV de Alta Definición, y han dado a los operadores y organismos de radiodifusión experiencia en el manejo y distribución de televisión de alta definición.

Los organismos de radiodifusión en Francia y el Reino Unido se han comprometido más con las pruebas HD / TDT, mientras que los organismos de radiodifusión en Italia han anunciado planes para llevar a cabo esas pruebas. En Francia, dos ensayos tuvieron lugar entre mayo y julio de 2006, y septiembre de 2006 hasta enero de 2007 en las ciudades de Lyon, Marsella y París. En ambas series de ensayos, los servicios de HD

disponibles en un múltiplex ofrecían algunos deportes y contenido francés original. En el segundo ensayo, el material 'HDTV' consistía esencialmente de programación SD (*Standard Definition*) escalada.

En el Reino Unido, los organismos de radiodifusión de servicio público - BBC, ITV, Channel 4 y Five—llevaron a cabo una prueba HD / TDT durante un período de cinco meses de junio a octubre de 2006 con 450 hogares especialmente equipados en la zona de Londres. La prueba tuvo como objetivo probar la viabilidad técnica y las consecuencias de la emisión de HD en la plataforma terrestre y evaluar la reacción a esos servicios. Además, los organismos de radiodifusión en la plataforma terrestre analógica, junto con los fabricantes y los minoristas, formaron un grupo de presión HD / TDT llamado "HDforAll". Este grupo pide que la asignación de espectro terrestre liberado como resultado del apagón analógico, se reserve para la emisión de servicios HD /TDT.

Es probable que los organismos de radiodifusión de servicio público (los órganos subsidiarios principales) se enfrenten a una presión cada vez mayor para hacer disponibles sus servicios de televisión de alta definición en todas las plataformas, incluida la terrestre. No podrá ser una situación buena ofrecer sus servicios de televisión de alta definición sólo por cable y las plataformas satelitales, especialmente en los países donde un gran porcentaje de los espectadores se basan en la plataforma terrestre.

3 Proyecto COM@WEB

En el presente capítulo, se presenta el proyecto de innovación docente en el que se incluye este proyecto fin de carrera. Inicialmente se explica el proyecto global, COMWEB [21], y más adelante, el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD integrado en COMWEB. También se realiza una breve descripción de las TICs en enseñanza universitaria, puesto que el proyecto COMWEB es una iniciativa de un grupo de profesores del departamento de TSC (*Teoria del Senyal i Comunicacions)* de la UPC (*Universitat Politècnica de Catalunya*) orientado hacia ese tipo de enseñanza.

3.1 Introducción

El proyecto COMWEB [21], que significa Comunicaciones en la Web, es una iniciativa de un grupo de profesores del departamento de TSC. Es una plataforma de aprendizaje a distancia (e-learning) que ofrece una serie de cursos a distancia a los estudiantes universitarios con el principal objetivo de facilitar una serie de recursos docentes preferentemente orientados en las temáticas del procesado de la señal y de comunicaciones. Puede observarse en la Figura 12 la página principal de COMWEB.

Figura 12 Página web inicial de COMWEB

El proyecto fue iniciado en el año 2003 y desarrollado entre otros, por profesores y alumnos de la Universidad Politécnica de Catalunya. COMWEB es una plataforma

multimedia dinámica, accesible vía web y programada sobre Moodle. Moodle es un sistema de gestión de cursos de libre distribución que ayuda a los educadores a crear comunidades de aprendizaje en línea. El proyecto conjunto tiene como objetivo permanente la oferta a los alumnos y usuarios motivados por aprender comunicaciones en general, de un conjunto de herramientas interactivas que contienen además del laboratorio virtual, contenido teórico y práctico de las asignaturas implicadas, ejercicios de autoevaluación y propuesta de itinerarios de estudio y experimentación a través de la plataforma, en función de los requerimientos del propio usuario.

Uno de los cursos integrados en COMWEB es el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD [22], de libre difusión y acceso y basado en la simulación de diferentes sistemas de comunicaciones que pueden ser utilizados tanto en docencia presencial como en educación a distancia.

El papel desempeñado por LaViCAD dentro de esta plataforma, consiste en la implementación de applets que representan sistemas o subsistemas de comunicaciones en continua interacción con otros elementos de los contenidos, como explicaciones teóricas y ejercicios tanto propuestos como resueltos principalmente.

A nivel tecnológico la estructura base consiste en una aplicación o plataforma general única denominada aplicativo Contenedor, reprogramable y configurable para emular desde sistemas muy sencillos y básicos, hasta aplicaciones más complejas y cotidianas, como por ejemplo el sistema de comunicación sin cables (WiFi) o el de televisión digital terrestre (TDT) desarrollado en este proyecto. Los diferentes sistemas de comunicaciones implementados permiten observar y analizar el avance secuencial de las señales implicadas a través de diferentes etapas. El resultado final es una aplicación en forma de applet para cada uno de los sistemas implementados. Los applets han sido programados en lenguaje JAVA y son accesibles vía web desde la plataforma COMWEB.

Es de esperar, que los estudiantes en el área de las telecomunicaciones y la ingeniería electrónica sean los usuarios más habituales de la plataforma.

3.2 Formación mediante las TICs

La docencia mediante las nuevas tecnologías de la información y comunicación es la instrucción que se imparte a través de un canal tecnológico (como por ejemplo la televisión, radio, red informática, etc).

La primera cuestión que se ha de tener en cuenta sobre la eficacia de las TICs es que repercusión, si tiene, presenta en el rendimiento del usuario final. Además se plantean algunos problemas respecto a la preparación de los estudiantes y profesorado, debido a que ambos colectivos deben disponer de una formación adecuada para la utilización de las TICs.

La formación en TICs para los educadores, presenta dos aspectos en general: formación técnica para aprender a utilizar y mantener el software y el equipamiento oportuno, y formación sobre cómo integrar el uso de las TICs a la guía de estudios, con métodos eficaces de formación y uso de aplicaciones específicas de la disciplina.

Una vez superados estos problemas, se ha demostrado de manera consistente que la docencia por medio de las TICs, que utiliza métodos pedagógicos convencionales, es tan eficaz como la instrucción cara a cara tradicional, pero con unas mejoras en las características docentes.

En el caso de la instrucción basada en la utilización del ordenador, en ciertos casos puede incluso mejorar la formación y la actitud del estudiante, debido a la motivación inherente que el uso de estas tecnologías conlleva a los estudiantes.

En los usos más perfeccionados de las TICs en las aulas, donde se fomenta que los estudiantes trabajen en entornos ricos en información y experiencia para construirse sus propias ideas, la investigación desarrollada por todo el mundo ha adoptado un panorama más claro sobre su eficacia en el soporte prestado en este tipo de enseñanza, conocido como *constructivista*.

Para más información sobre el tema de las TICs en la docencia, ver referencias [23] y [24].

3.2.1 Revolución docente

La introducción de las TICs en la docencia ha permitido, entre otros aspectos educativos: soporte a nuevos métodos pedagógicos, conectividad y ampliación de los programas educativos.

- ***Soporte a nuevos métodos pedagógicos***

Con la educación constructiva, modificamos la función de los formadores. Además, es posible que los profesores proporcionen a los estudiantes experiencias de formación basadas en problemas a un ritmo y hacia una dirección que decidan los mismos estudiantes, y también examinar a éstos desde una nueva perspectiva, más interactiva y atractiva que puede que valore mejor sus conocimientos.

- ***Conectividad***

Se puede hablar de iniciativas de bibliotecas digitales para ofrecer colecciones electrónicas accesibles mediante Internet, donde encontramos obras impresas (revistas, libros, mapas, imágenes, etc.), fotografías, películas, pinturas, software, etc. en muchos países del mundo.

Como podemos ver, se crean representaciones digitales de los recursos físicos existentes, y además se distribuyen más recursos de información sólo presentes en formato digital.

- ***Ampliación de los programas educativos***

Es un hecho que en la sociedad actual es necesaria la formación continuada para todos aquellos profesionales que se encuentren en la tesitura en que la materia sobre la cual han realizado sus estudios va variando, o mejor dicho evolucionando, a lo largo del tiempo, y es su responsabilidad la de ir actualizando sus conocimientos sobre los

diversos temas. Para todos estos preocupados profesionales, la revisión y ampliación de sus conocimientos es posible mediante la utilización de los recursos docentes que se proporcionan en las TICs.

3.2.2 Necesidad de las TICs

La instrumentación tecnológica es una prioridad en la comunicación de hoy en día, ya que las tecnologías de la comunicación suponen una importante diferencia entre una civilización desarrollada y otra en vías de desarrollo. Éstas poseen la característica de ayudar a comunicarnos ya que, a efectos prácticos, en lo que a captación y transmisión de información se refiere, desaparece el tiempo y las distancias geográficas.

3.2.2.1 ¿Por qué tenemos que integrar las TIC's?

La Era Internet exige cambios en el mundo educativo. Y los profesionales de la educación tienen múltiples razones para aprovechar las nuevas posibilidades que proporcionan las TIC para impulsar este cambio hacia un nuevo paradigma educativo más personalizado y centrado en la actividad de los estudiantes. Además de la necesaria alfabetización digital de los alumnos y del aprovechamiento de las TIC para la mejora de la productividad en general, el alto índice de fracaso escolar (insuficientes habilidades lingüísticas, matemáticas...) y la creciente multiculturalidad de la sociedad con el consiguiente aumento de la diversidad del alumnado en las aulas, constituyen poderosas razones para aprovechar las posibilidades de innovación metodológica que ofrecen las TIC para lograr una escuela más eficaz e inclusiva.

3.2.2.2 Momentos clave de aplicación práctica de las TIC's.

Los 4 momentos clave de la actuación docente en el que la utilización de las TIC puede aportar ventajas son los siguientes:

- **Fase pre-activa**: planificación, creación de materiales didácticos...
- **Fase de ejecución y evaluación de las actividades de enseñanza y aprendizaje con los alumnos:** explicaciones, autonomía de trabajo del alumno, interacciones...
- **Fase post-activa**: tutoría, gestiones administrativas
- **Formación continuada**: lecturas, cursos, jornadas, colaboración en investigaciones...

3.2.3 Las TIC's en el sistema universitario español

Todos los años se realizan en varios países estudios sobre el nivel de implantación de las TIC en las universidades. Las pioneras fueron las universidades estadounidenses, que publican anualmente un informe denominado "National Survey of Computing and Information Technology in American Higher Education" (Green, 2007). También se publica un informe similar en las universidades del Reino Unido, "Higher Education Information Technology Statistics (HEITS). Summary 2005" (HEITS, 2005). Estos estudios son muy útiles para conocer la situación actual de las Tecnologías de la

Información y las Comunicaciones (TIC) en las universidades y establecer políticas y estrategias conjuntas para la implantación de nuevas tecnologías.

Utilizando el Sistema Universitario Español (SUE) como ámbito, la CRUE [25] (Conferencia de Rectores de las Universidades Españolas) realizó en 2004 una encuesta con el objeto de conocer la situación de las TIC en las universidades españolas (Barro, 2004).

La Sectorial TIC de la CRUE diseña posteriormente un Modelo de Planificación TIC que tiene como punto de partida el establecimiento de un Catálogo de Objetivos e Indicadores TIC consensuados y comunes para todo el SUE. Tomando como base dicho catálogo, se realiza una encuesta a las universidades, conocida como UNIVERSITIC [26], cuyos resultados se recogen en un informe anual.

3.2.3.1 Servicios TIC en la UPC

La UPC (*Universidad Politécnica de Cataluña*) dispone de una página web dedicada a los servicios TIC, llamada UPCnet [27].

Presta los servicios TIC con una triple misión: conseguir y mantener unos estándares de Servicios TIC de calidad y competitivos a nivel europeo; conseguir la máxima rentabilidad de las inversiones de la UPC para la gestión de los servicios TIC a la comunidad universitaria y a otros administraciones; y facilitar a todos sus trabajadores y trabajadoras una proyección profesional ejemplar en el sector de las TIC.

A la vez, UPCnet cuenta con la colaboración tecnológica de algunos de los centros de investigación de más prestigio internacional en tecnologías como transmisión de datos, desarrollo de software, tratamiento masivo de datos, realidad virtual, seguridad, etc.

Los servicios TIC que ofrece son los siguientes:

- ***Consultoría:*** planificación y gestión estratégica y operativa de las TIC.
- ***e-soluciones:*** desarrollo, integración e implementación de las soluciones tecnológicas que mejor se adapten a las necesidades de la organización en cuestión.
- ***entorno personal:*** dar respuesta a las necesidades tecnológicas que cada usuario encuentra en su puesto de trabajo.
- ***sistemas y comunicaciones:*** gestión de infraestructuras y aplicaciones TIC con las máximas garantías de seguridad y disponibilidad.

3.3 LaViCAD

Dentro del proyecto de innovación docente COMWEB, se integra el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD basado en la simulación de diferentes sistemas de comunicaciones que pueden ser utilizados tanto en la docencia presencial como en la educación a distancia. La página principal del proyecto LaViCAD se muestra en la Figura 13.

Figura 13 Página web inicial de LaViCAD

El principal logro del proyecto LaViCAD, realizado como proyecto final de carrera por Carlos Vargas y Francisco Vargas bajo la dirección de Margarita Cabrera Bean, ha sido la creación de un aplicativo en lenguaje Java denominado Contenedor, mediante el cuál se puede desarrollar cualquier sistema de comunicaciones para su libre difusión en la Red.

3.3.1 Applets, la nueva docencia

Los applets constituyen el soporte principal utilizado en el proyecto LaViCAD con el fin de implantar una herramienta dentro de las TIC.

Las características de compilación y ejecución de Java han convertido a este lenguaje en uno de los principales sistemas para la implementación de entornos interactivos en Internet.

Básicamente, la incorporación de Java mediante los diferentes applets permite el desarrollo de aplicaciones que son ejecutadas desde páginas Web. Este modelo de diseño y programación posee un interés especial para la comunidad educativa ya que permite la integración de documentos basados en hiperenlaces (HTML) con aplicaciones interactivas integradas en los mismos documentos.

Las características principales de estos programas son:

- ✓ El ordenador se convierte en una herramienta de autoaprendizaje. Lo que aparece en la pantalla del monitor es el resultado de un diálogo interactivo entre el estudiante y el ordenador. El alumno es un participante activo, más que un observador.

- ✓ La atención del estudiante no está dirigida hacia la mera manipulación de teclas, sino a los detalles matemáticos y a la esencia del tema experimentado.

- ✓ Normalmente, cada applet viene documentado por un guión tipo texto, donde se explica el concepto que se trata de estudiar, así como las ecuaciones matemáticas que se utilizan. El objetivo es que el estudiante realice una actividad ordenada y progresiva, que conduzca a alcanzar objetivos básicos concretos.

Las principales ventajas de la utilización de los applets son:

- ✓ Posibilidad de representación de fenómenos relacionados con la comunicación (en el caso que nos ocupa), que mediante la educación tradicional resulta complicado o incluso imposible mostrar a los alumnos.

- ✓ En entornos complicados, simplificar o enfatizar en aquellos aspectos más importantes y remarcables del tema a tratar.

- ✓ Posibilidad de manipulación y control exacto de las variables que actúan en el sistema mostrado.

- ✓ La interactividad conseguida entre alumno y ordenador facilita la motivación y por tanto la docencia del usuario final.

Se piensa en los applets como una buena herramienta docente que permite al usuario final, casi sin esfuerzo, asumir una serie de conceptos básicos pero importantes sobre el tema en que versa la aplicación en cuestión.

3.3.2 Descripción funcional

Mediante LaViCAD, se emulan sistemas y sub-sistemas de diferentes sistemas de comunicaciones, tanto de estándares de actualidad como de sistemas con carácter académicos.

En el diseño de un laboratorio de comunicaciones analógicas y digitales, ya sea virtual o instrumental, en general se parte de una fuente de información, medible a través de una señal continua en tiempo (ejemplo señal de audio) o de una serie de datos (ejemplo: bits), y este tipo de información se procesa a través de diferentes sistemas: transmisor, canal y receptor. El subsistema transmisor y el subsistema receptor se hallan formados a su vez por diferentes subsistemas funcionales o etapas cada uno de ellos realizando un procesado o transformación diferente sobre la señal. En general, en un experimento virtual se emula el recorrido general que atraviesa la información.

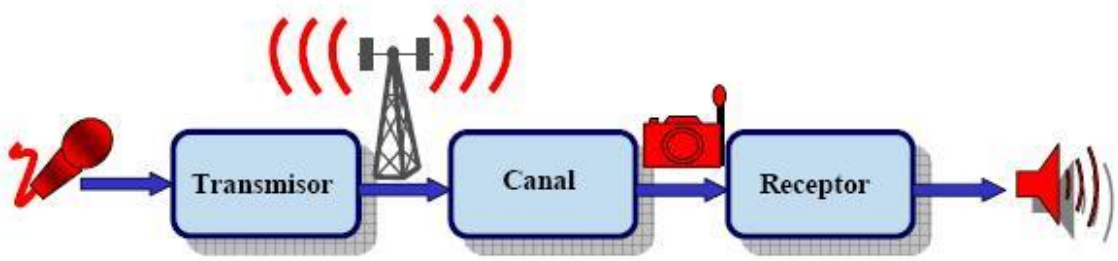


Figura 14 Sistema de comunicaciones básico

Para reproducir un sistema de comunicaciones determinado, cada uno de los bloques de la Figura 14, se desglosa a su vez en un nuevo diagrama, formado por las distintas etapas a través de las cuales se va procesando la señal. Sirva como ejemplo la pantalla del sistema DVB-T implementado en este proyecto (Figura 15):

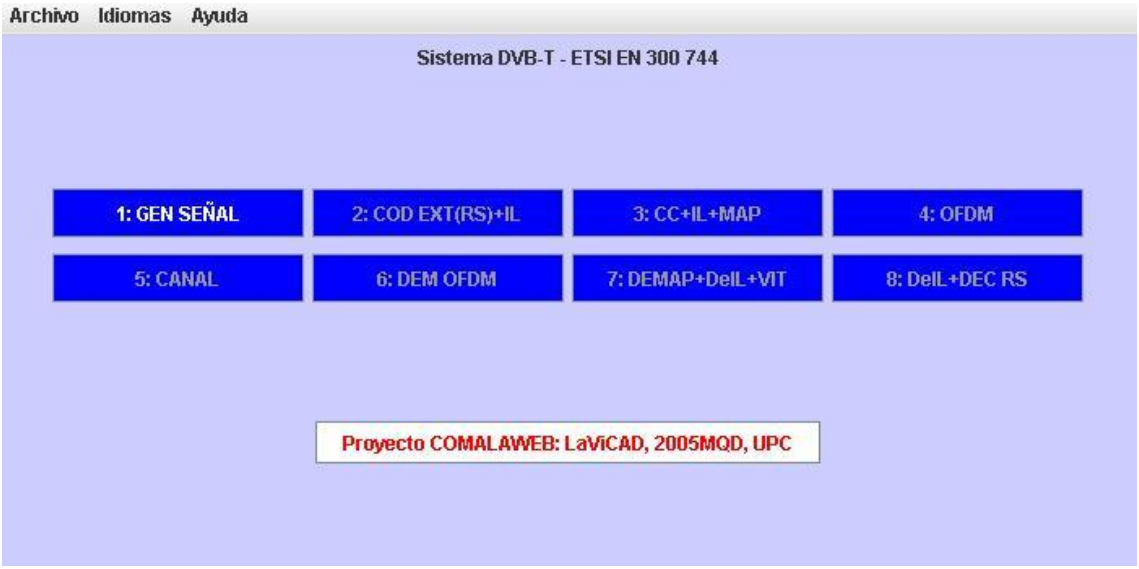


Figura 15 Pantalla principal del sistema DVB-T

Este sistema representa un sistema comunicaciones completo. Los cuatro primeros bloques o etapas corresponden al subsistema del transmisor. La etapa quinta corresponde al modelo de canal y las tres últimas etapas corresponden al subsistema del receptor. Cada etapa puede ser accionada por el usuario, siempre avanzando de forma secuencial. Con ello se abre una nueva pantalla, en la que se deben validar y/o cambiar lo diferentes parámetros de configuración de etapa y apareciendo así toda una serie de resultados correspondientes a las señales de interés de las etapas en cuestión. Estas señales pueden ser temporales, frecuenciales, histogramas, diagramas de ojo, mapas de señal, etc.

Además de visualizar resultados, el usuario dentro de cada etapa puede intercambiar la señal, pues hay opciones tanto de guardar la señal en un fichero, como de leer la señal de entrada a una de las etapas también desde un fichero. Para los ficheros de señal, se ha elegido un formato sencillo y universal que permite intercambiar señales con sistemas programados desde diferentes lenguajes de programación, o incluso con instrumentos de laboratorio, como osciloscopios y analizadores de señal que también tengan las opciones de guardar señal en fichero o bien cargar señal desde fichero. Es precisamente

este método de intercambio de señales el que enmarca a LaViCAD en una aplicación abierta a diferentes implementaciones.

3.3.3 Ejemplos de uso

Con LaViCAD se pueden realizar diferentes experiencias docentes, como las que se nombran a continuación:

- ***Resolución de un ejercicio teórico***

Mediante LaViCAD se pueden particularizar determinadas situaciones presentadas en la parte teórica de un libro de texto, como por ejemplo [29], [30] ó [31]. Entre otros análisis se pueden implementar y visualizar cálculos y gráficas de probabilidad de error, de densidad espectral y anchos de banda, de ecualización, de diagrama de ojo, de sincronismos de portadora, etc.

- ***Verificación de resultados experimentales en un laboratorio instrumental***

Una utilidad de interés, y de gran aceptación entre los estudiantes, consiste en la comparación en tiempo real de los resultados obtenidos en un laboratorio instrumental y con el laboratorio virtual. LaViCAD constituye una buena herramienta pedagógica al ofrecer la comparación de diferentes señales y medidas obtenidas en la práctica, con las funciones explicadas en clase de teoría.

- ***Experimentos que requieren programación por parte del alumno***

Otra utilidad de LaViCAD, sería la combinación de sistemas programados en diferentes plataformas, para lograr implementar una modulación especial, o cualquier experimento que exija al alumno realizar acciones entre los diferentes sistemas. Esto supondría una experiencia más creativa por parte del alumno.

3.3.4 Aplicativo Contenedor

La estrategia que se ha diseñado para la programación del aplicativo Contenedor, es capaz de agregar un número indefinido de etapas dentro de un sistema, así como un número indefinido de resultados y de parámetros de entrada y salida. En la Figura 16, se muestra el esquema general del aplicativo Contenedor:

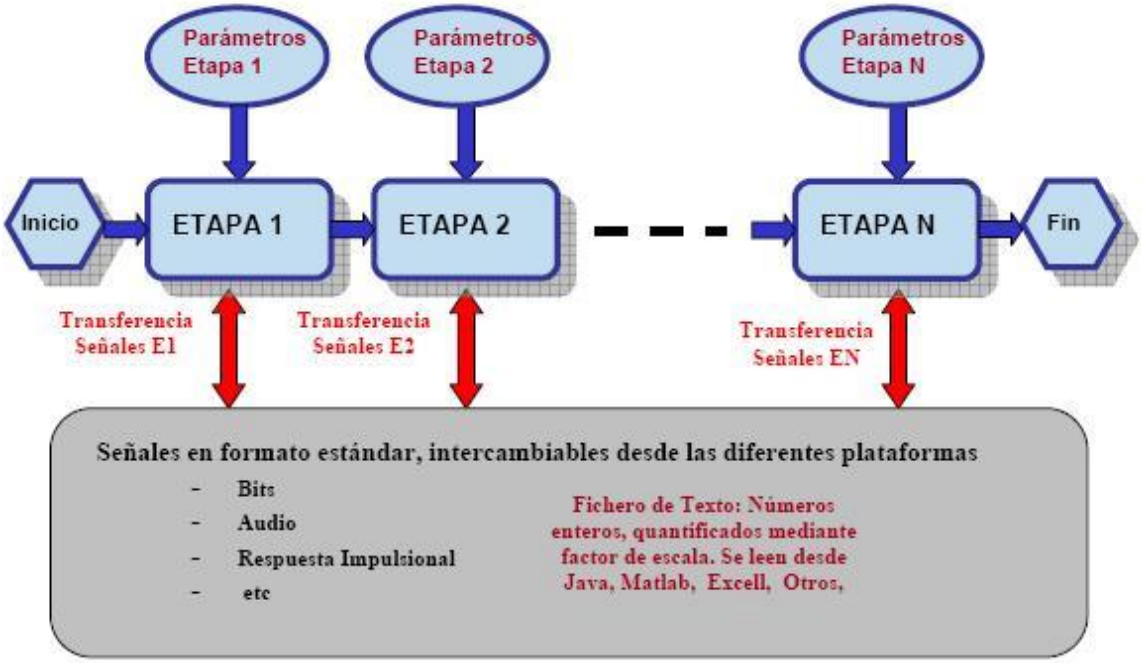


Figura 16 Estructura general del programa contenedor

El diseño ha sido realizado utilizando sólo componentes Open-Source por lo que no es necesario disponer de ningún tipo de licencia para su ejecución y su utilización. Además la programación ha sido desarrollada siguiendo unos estándares y patrones de programación (swing, xml,…) totalmente abiertos para que su posterior mantenimiento y futuras ampliaciones sean muy poco costosas.

Características del aplicativo Contenedor:

- ✓ Dispone de unos ***sistemas de ayudas*** que están formados por unas estructuras de pantallas dinámicas formadas por páginas html. El programador podrá configurar la ayuda de cada sistema sin la necesidad de modificar código fuente.

- ✓ Es ***multiidioma,*** por lo que añadir o eliminar un idioma es un proceso muy sencillo que no requiere modificar código JAVA.

- ✓ ***Define los métodos*** que el programador de un sistema concreto debe utilizar para operar con el aplicativo, de esta forma se define el procedimiento para recoger parámetros de entrada de una etapa, devolver parámetros de salida y resultados, modificar las representaciones gráficas, etc...

- ✓ El aplicativo Contenedor dispone de unos ***ficheros de configuración*** genéricos que permiten definir y configurar un determinado sistema mediante ficheros XML:

    - Número de etapas
    - Parámetros de entrada de cada etapa y sus validaciones
    - Parámetros de salida de cada etapa
    - Señales que se representarán en cada etapa
    - Señales que cada etapa permitirá guardar y/o cargar de un fichero

- Ayudas de cada etapa
- Idiomas disponibles en cada etapa

El programador debe configurar, mediante estos ficheros de configuración, el sistema que quiere implementar y una vez tiene el sistema configurado sólo debe escribir el código Java necesario para realizar el procesado de cada etapa. Dentro de este código se utilizarán los métodos que operan con el aplicativo Contendor.

En este desarrollo el programador no debe preocuparse por la interfaz gráfica que verá el usuario, validaciones, etc. Por tanto todo el conjunto del aplicativo Contenedor en Java constituye un código reaprovechable en la implementación de cualquier sistema de comunicaciones concreto.

Mediante LaViCAD se simula el funcionamiento etapa a etapa de cualquier sistema y subsistema de comunicaciones. La experiencia recogida en este proyecto abarca el desarrollo de la capa física del estándar DVB-T (estándar utilizado en la televisión digital terrestre TDT).

4 Descripción del proyecto

El proyecto desarrollado, consiste en el diseño e implementación del simulador de la capa física DVB-T tanto en MATLAB, como applet de Java. Este simulador es autocontenido, pero también puede interactuar con un laboratorio instrumental. Mediante la aplicación desarrollada, se pueden experimentar y verificar diferentes simulaciones correspondientes a temáticas propias de los estudios de ingeniería en telecomunicaciones.

Ambas versiones disponibles del simulador DVB-T, MATLAB y Java, se han programado de manera que resulten idénticas. No obstante, existe algún resultado gráfico o numérico que las diferencia. Dado el caso, se indicará en que versión está disponible el resultado en cuestión.

Como se ha descrito en el capítulo 3, el simulador está integrado en el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD [22], que es parte de un proyecto docente de propósito más general: COMWEB [21], iniciado en el año 2003 y desarrollado entre otros, por profesores y alumnos de la Universidad Politécnica de Cataluña.

El sistema DVB-T, como se ha visto en el apartado 2.3.2.1, sigue las especificaciones marcadas por el estándar *ETSI EN 300 744* [1], que marca los procesos necesarios de codificación de canal y modulación completamente imprescindibles para que, realizando el procesado pertinente de la señal en banda base, se pueda recibir la misma libre de perturbaciones cuando se usan los canales de transmisión terrestre. La señal de entrada al sistema es la denominada *MPEG-2 Transport Stream (TS)* o *Flujo de transporte MPEG-2*. Dicho *Flujo de Transporte (TS)*, obtenido mediante el proceso denominado *Codificación de Fuente*, es una adaptación del estándar MPEG-2 según ISO/IEC 13818 [9], que se estructura multiplexando varios programas y añadiendo la *Información del Servicio (SI)* correspondiente, según la referencia *ETS 300 468* [8].

La secuencia de operaciones denominada *Codificación de canal*, establece medidas de protección necesarias para robustecer el canal e incluso poder corregir errores después de pasar por el canal de transmisión - "Forward Error Correction" (FEC). También se describe el *esquema de modulación* usado en la transmisión, la modulación multiportadora OFDM, dando lugar junto al bloque codificador a una transmisión de tipo COFDM (Coded Orthogonal Frequency Division Multiplexing).

En el presente capítulo se describen los pasos seguidos en el diseño del simulador de la capa física DVB-T. A través de la simulación etapa a etapa, el sistema ofrece al usuario una ligadura entre el procesado de señal teórico y los protocolos de comunicaciones digitales utilizados por aplicaciones de uso cotidiano. De este modo se despierta el interés del usuario en profundizar sobre el procesado de señal relacionado con dichas aplicaciones y se le induce a relacionarlo con aspectos más teóricos aprendidos con enfoque académico.

Sobre el sistema DVB-T simulado puede distinguirse, aparte de los contenidos teóricos explicados, la visión más práctica enfocada al marco que engloba el sistema en una aplicación cotidiana como es el visionado de señales TDT.

La estructura que siguen los apartados que conforman este capítulo se inicia con la presentación de las ventajas de la modulación OFDM, a continuación se enumeran los requisitos que precisan las características que debe tener este sistema y por último el apartado sistema DVB-T describe todo el proceso funcional de la aplicación.

4.1 OFDM

El principio de la modulación OFDM (*Orthogonal Frequency Division Multiplexing*) consiste en distribuir los bits de información entre un gran número de portadoras de forma que cada una se transmita a una velocidad reducida con respecto a la del flujo total.

En el transmisor de un sistema DVB-T, después de realizar las etapas de procesado de señal referentes a la codificación de canal y la modulación, se deben introducir los parámetros necesarios en el receptor para lograr la sincronización, realizar la regeneración del canal y recuperar en general la información de los parámetros de transmisión. De transportar todos estos parámetros se encargan las tramas OFDM, cuya estructura se presenta más adelante en el apartado 4.3.4.2, y cuyo nombre lo adquieren por el hecho de utilizar la modulación OFDM.

Como se puede observar en la Figura 17, típicamente en OFDM la distancia espectral entre portadoras adyacentes es *1/Tu*, con lo que la posición central de las portadoras en frecuencia coincide con los ceros de las portadoras vecinas, es decir, son ortogonales entre ellas. Bajo estas condiciones, la interferencia intersimbólica (ISI) es nula.

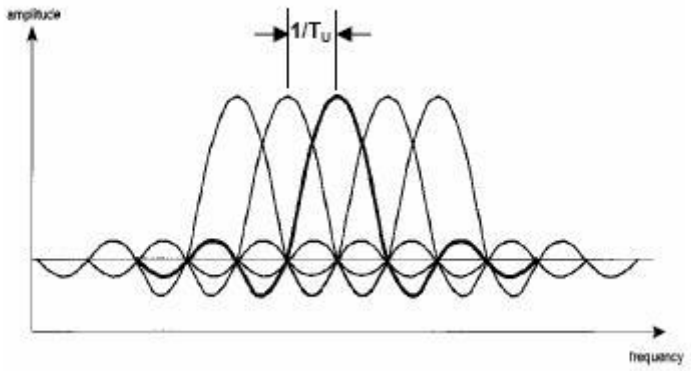


Figura 17 Espectro de portadoras adyacentes en la modulación OFDM. Figura extraída de [44].

Para potenciar más la inmunidad de la señal frente al efecto multicamino, se prolonga la duración de los símbolos añadiendo un tiempo extra llamado *intervalo de guarda.* El intervalo de guarda es una continuación cíclica de la parte útil del símbolo, el cual se

inserta delante de él. Esta prolongación del símbolo permite evitar errores al recibir la misma señal desde dos caminos distintos con un retardo entre ellos, es decir, mientras la diferencia de tiempo en que el receptor recibe la señal desde dos caminos distintos no supere el tiempo del intervalo de guarda, no habrá interferencia intersimbólica (ISI). La Figura 18 ilustra esta situación.

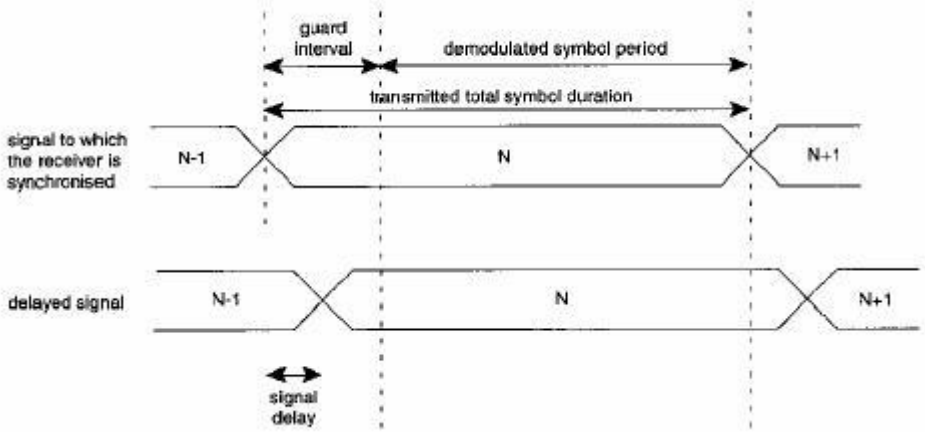


Figura 18 Multitrayecto con retardo inferior al intervalo de guarda. Figura extraída de [44].

La desventaja del intervalo de guarda, es que se produce una reducción de la eficiencia espectral, por el hecho de transmitir tramos repetidos de la señal. Por esta razón el sistema DVB-T incorpora diversas opciones de intervalo de guarda, para conseguir ajustar el mínimo intervalo de guarda necesario para evitar interferencia intersimbólica teniendo la máxima eficiencia espectral posible.

Gracias a estas características de la modulación OFDM empleada, las ventajas que aporta son:

- ✓ **Robustez frente a la interferencia intersimbólica (*ISI*) y los desvanecimientos (*fading*) producidos por la propagación multicamino.**

- ✓ **Ecualización simplificada**. Gracias a que la señal OFDM esta compuesta por una multitud de portadoras de banda estrecha ortogonales, el "*fading*" o desvanecimiento selectivo en frecuencia causado por el multicamino puede considerarse plano (constante) dentro del ancho de banda ocupado por cada subportadora OFDM. El ecualizador solo deberá multiplicar cada subportadora por un valor constante.

- ✓ **Facilita las redes SFN.** Capacidad de la tecnología DVB-T para difundir una señal en la misma frecuencia desde varios puntos emisores evitando las interferencias. La principal ventaja es la eficiencia espectral que se puede obtener, ya que un servicio compuesto por 4 ó 5 programas de televisión puede ser difundido en un área extensa o incluso a nivel nacional usando tan sólo un único canal RF. Es una de las principales ventajas que aporta la modulación OFDM al sistema DVB-T, y una de las grandes ventajas de la implantación de la televisión digital frente a la analógica. De esta manera varios repetidores pueden compartir la misma radiofrecuencia.

- ✓ **Alta eficiencia espectral.** Gracias a que la ortogonalidad entre portadoras permite el solape del espectro.

- ✓ **Eficiente implementación usando la FFT.**

- ✓ **Protección contra interferencias co-canal (CCI, CoChannel Interference).** El hecho de tener un gran número de portadoras sobre las que se distribuye la información, proporciona una protección contra interferencias que se presentan en la misma banda de frecuencias que la señal útil (por ejemplo señal analógica PAL), ya que si se pierde la información de una portadora debido a estas interferencias, se pierde una pequeña porción de información que no tiene por qué ser relevante para la calidad de la transmisión.

Para más información sobre la modulación OFDM se recomienda la lectura [32], "*Theory and applications of OFDM and CDMA*".

4.2 Especificaciones del sistema implementado

El diseño del sistema simulador DVB-T de este proyecto se ciñe a las especificaciones marcadas por el estándar *ETSI EN 300* [1], presentadas en el apartado 2.3.2.1.2. A continuación se muestran dos tablas a modo de resumen:

BLOQUE DE PROCESADO	**DESCRIPCIÓN**
Adaptación y dispersión de energía	Incorporación de un byte de sincronismo y aleatorización de la señal de entrada
Codificación externa	Codificación Reed-Solomon
Entrelazado externo	Entrelazado convolucional
Codificación interna	Codificación convolucional con *puncturing*
Entrelazado interno	Entrelazado relativo al bit y entrelazado de símbolos
Mapeo de símbolos	Modulación QPSK, 16QAM o 64QAM
Transmisión OFDM	Estructuración en tramas, inserción de portadoras piloto, inserción del intervalo de guarda, conversión D/A y transmisión a frecuencia RF

Tabla 1 Procesado de señal aplicado a la señal de entrada (MPEG-2 TS)

PARÁMETROS CONFIGURABLES	**VALORES**
Modo de transmisión	2k / 8k (1.705 / 6817 portadoras por símbolo OFDM)
Esquemas de modulación	QPSK, 16-QAM, 64-QAM uniformes y no uniformes (parámetro de modulación α diferente de 1)
Tasa de codificación interna	1/2, 2/3, 3/4, 5/6 ó 7/8
Intervalo de guarda	1/4, 1/8, 1/16 ó 1/32
Modulación y codificación de canal	jerárquica o no jerárquica

Tabla 2 Parámetros configurables de transmisión

No obstante, presenta algunos aspectos que se separan ligeramente del estándar:

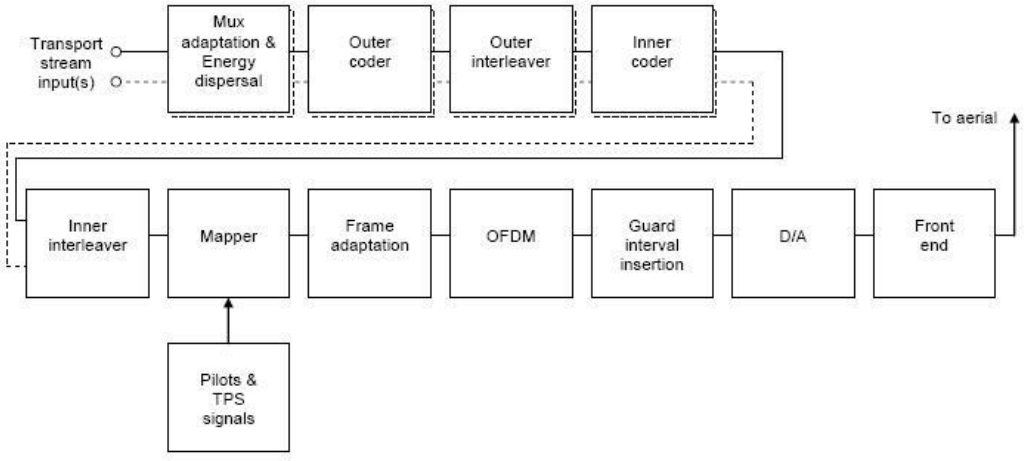
- Se ha obviado la conversión a frecuencia RF (VHF, UHF) en transmisión y la posterior conversión a banda base en recepción, así como los amplificadores a la entrada y salida del canal. El motivo de obviar estas etapas reside en el bajo interés académico que aporta a la simulación. De esta manera se facilita la representación y compresión de los datos de cara al usuario. No obstante sí se han simulado los errores que se podrían introducir en la recuperación de la portadora y el proceso de sincronización de la trama.

- El modo de transmisión implementado ha sido el 2K no jerárquico. No se ha simulado el modo 8K debido a que la simulación se haría lenta y pesada por el hecho de que en lugar de utilizar 1705 portadoras por símbolo OFDM, se usarían 6817. Tampoco se ha simulado una transmisión jerárquica porque complicaría la implementación del sistema, y no aportaría más interés académico al simulador. No obstante, sí se han simulado constelaciones no-uniformes propias de la transmisión jerárquica con el empleo del parámetro α .

- El formato de transmisión de las portadoras TPS lo ha sido adaptado a las necesidades del sistema implementado, pues para poder seguir las especificaciones del estándar se deben enviar un mínimo de 68 símbolos OFDM, lo cual implicaría el envío de más paquetes MPEG-2 en transmisión, y la consecuente ralentización del sistema completo.

Los requisitos precisados para la implementación del sistema son:

- El sistema completo debe simular el procesado de señal que engloba la capa física del estándar. El diagrama de bloques funcional descrito por el estándar es el siguiente:



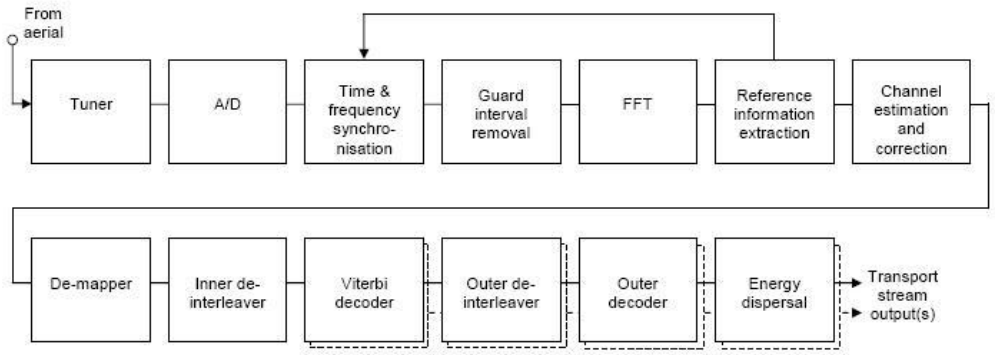


Figura 19 Diagrama de bloques del transmisor y receptor para la capa física del sistema DVB-T. Marcados con puntos los bloques funcionales para transmisiones jerárquicas. Figura extraída de [2].

El estándar describe de forma detallada del procesado de señal en el modulador, mientras que el procesado en el receptor se deja abierto a la implementación de soluciones diferentes. En cualquier caso, el receptor adopta técnicas duales a las usadas en transmisión, y en la Figura 19 se puede ver el diagrama de bloques genérico de un receptor DVB-T. En la Figura 20, puede verse el esquema completo de comunicaciones implementado por el simulador de la capa física del sistema DVB-T desarrollado en este proyecto.

- Inicialmente se plantea una división del sistema en 8 etapas:

  - Etapa1: Generación de la señal
  - Etapa2: Codificación externa (RS) + Interleaving externo
  - Etapa3: Codificador convolucional + Interleaving interno + Mapeo
  - Etapa4: OFDM
  - Etapa5: Canal
  - Etapa6: Demodulador OFDM
  - Etapa7: Demapping + Deinterleaving interno + Decodificador de Viterbi
  - Etapa8: Deinterleaving externo + Decodificador Reed-Solomon (RS)

- A su vez sobre cada etapa recaen unos requisitos enfocados a mantener la rigurosidad del estándar en el procesado de señal que aplica cada una. A continuación se muestran estos requisitos divididos en dos puntos por etapa, el primero corresponde a aquellos que son impuestos directamente por el estándar y el segundo apunta a la presentación de los resultados:

  - *Etapa1: Generación de la señal*

    - Adaptación y dispersión de energía de un número de paquetes TS MPEG-2 múltiplo de 8.
    - Un estudio de la autocorrelación de los bits generados.

  - Etapa2: Codificación externa (RS) + Interleaving externo

- Implementación del codificador Reed-Solomon y del entrelazador convolucional de Forney.
- Presentación de un paquete codificado Reed-Solomon y del proceso de entrelazado teórico (disponible en la versión MATLAB).

- Etapa3: Codificador convolucional + Interleaving interno + Mapeo

  - Implementación del codificador convolucional con puncturing para todas las tasas de codificación especificadas (1/2, 2/3, 3/4, 5/6 y 7/8), diseño del interleaving interno y del mapeador (modulaciones QPSK, 16QAM y 64QAM uniformes y no uniformes).
  - Presentar un análisis de cada bloque de procesado de la etapa.

- Etapa4: OFDM

  - Adaptación de tramas OFDM, implementación de la IFFT e inserción del intervalo de guarda.
  - Un estudio de la señal en el dominio frecuencial y temporal para transmisión en canales de 6, 7 ó 8 MHz. Parrilla de pilotos y TPS (disponible en la versión MATLAB).

- Etapa5: Canal

  - Posibilidad de generar canales tipo Rayleigh así como ruido gaussiano.
  - Un estudio de la señal en el dominio frecuencial y temporal. Análisis del canal generado.

- Etapa6: Demodulador OFDM

  - Añadir la posibilidad de simular la recepción con posibles errores de portadora. Eliminación del intervalo de guarda, implementación de la FFT, ecualización ideal opcional, demultiplexor de trama y decodificador de pilotos y TPS.
  - Presentación de la constelación de los datos, pilotos y TPS.

- Etapa7: Demapping + Deinterleaving interno + Decodificador de Viterbi

  - Implementación del decodificador de Viterbi y de los bloques inversos a sus respectivos en transmisión.
  - Un comparativa de errores antes y después del decodificador de Viterbi, y un estudio de su ubicación antes y después del deinterleaving interno.

- Etapa8: Deinterleaving externo + Decodificador Reed-Solomon (RS)

  - Simular la recepción de la señal emitida. Implementación del decodificador Reed-Solomon.

- Una comparativa de errores antes y después del decodificador Reed-Solomon, y un estudio de su posición antes y después del deinterleaving externo.

4.3 Simulador DVB-T

En esta sección se desarrolla el proceso de implementación de la capa física del simulador DVB-T bajo las especificaciones descritas en el estándar *ETSI EN 300 744* [1]. Este proceso engloba todas aquellas acciones que describen la funcionalidad y configuración de la aplicación. La implementación se encarga de la recogida de todos los parámetros de entrada y generación de todos los parámetros de salida numéricos o gráficos. Este proceso se estructura de la siguiente manera:

- ✓ Implementación secuencial y por orden de cada una de las etapas que conforma el sistema global. Dentro de cada una se organiza la implementación en el siguiente orden:
  - Parámetros de entrada. Los propios de la etapa y los necesarios de anteriores etapas.
  - Procesado: estructura teórica que sigue el programa clasificada en dos grupos. Por un lado se tiene la parte de procesado de señal que genera las variables que se traspasan entre etapas. Dentro de esta parte se estudian las especificaciones del estándar *ETSI EN 300 744* [1] que afectan al procesado de señal descrito para la etapa. Por otro lado se tiene la parte de procesado de señal enfocada únicamente a mostrar resultados gráficos numéricos pero que no sobreviven a la etapa.
  - Resultados: manual de utilización de los resultados que genera la etapa, ya sean gráficos o numéricos. También se informará de las señales que pueden guardarse en un fichero y de aquellas las cuáles pueden leerse de un fichero.
- ✓ Ejemplo generado con la aplicación: con este apartado se propone simular un ejemplo concreto, el mismo para todas las etapas, y seguir la evolución de las señales más características en cada una de ellas. El ejemplo se ha realizado con la versión MATLAB del simulador.
- ✓ Retos de programación que plantea el procesado interno de cada etapa. Se diferenciarán los retos propios de la implementación MATLAB de los retos de la implementación Java. Cabe comentar que para la implementación Java, se han utilizado los algoritmos de programación seguidos en la implementación MATLAB, así que en el apartado de retos de programación Java sólo se nombrarán los algoritmos que sean distintos a los de MATLAB.
- ✓ Análisis de diferentes aspectos teóricos y prácticos. Entre otros, se estudian las limitaciones inherentes a la programación en un PC de un sistema de comunicaciones.

Las imágenes del sistema que se presentan de aquí en adelante corresponden al simulador MATLAB del sistema, a excepción de aquellos resultados que se encuentren disponibles únicamente en la versión Java.

Siguiendo las especificaciones del apartado 4.2, se ha dividido el sistema DVB-T en 8 etapas que adoptan el siguiente esquema:

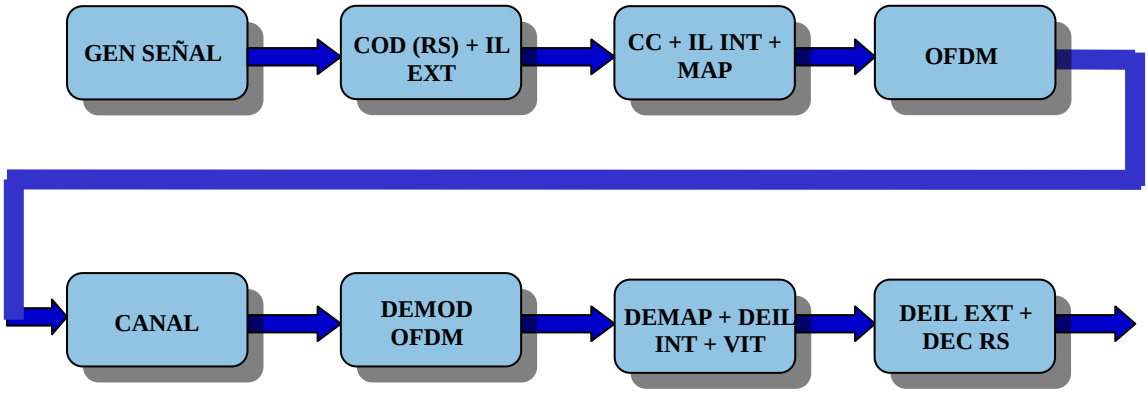


Figura 20 Esquema de las etapas que conforman el sistema DVB-T

4.3.1 Etapa 1: Generación de la señal

En la primera etapa del sistema, el usuario escoge el número de paquetes de transporte del múltiplex de entrada al sistema "Transport Stream" (TS) en formato MPEG-2 a enviar. Para proporcionar una señal de inicialización en el decodificador del receptor que permita identificar cada conjunto de 8 paquetes de transporte afectados por la aleatorización, se realiza la adaptación del flujo de transporte.

Para asegurar la dispersión de energía del espectro radiado, los paquetes de bits deben pasarse por un aleatorizador, de forma que se eviten largas series de ceros o de unos.

El procesado realizado en la etapa se esquematiza de la manera siguiente:

Figura 21 Esquema de la etapa 1 del sistema DVB-T

Como puede observarse en el esquema, la señal de salida de la etapa será $b_{scr}[n]$, que es el vector de bits a la salida del aleatorizador.

4.3.1.1 Parámetros de entrada

La primera etapa del simulador posee tres parámetros de entrada:

- **Número de paquetes:** selecciona el número de paquetes MPEG-2 a enviar entre tres posibles valores: 8, 16 y 24. Con este parámetro, fijamos la longitud binaria a generar y aseguramos tener un múltiplo de 8 paquetes MPEG-2 como indica el estándar.

- **Tipo de generación:** permite elegir entre dos opciones de generación de bits: aleatoria o introducida por fichero. Sobre las señales introducidas por fichero recaen algunas condiciones para asegurar el correcto funcionamiento de la simulación. En la versión MATLAB del simulador si se desea el tipo de generación introducida por fichero, se debe leer el fichero de bits desde el menú superior de la etapa.

- **Probabilidad bit 0:** variable que determina la probabilidad de aparición del bit '0' en caso de utilizar generación aleatoria.

4.3.1.2 Procesado de la señal

A continuación se presentan las especificaciones descritas en el estándar que aplican las operaciones de procesado de señal en la presente etapa.

➢ ***Procesado propio de la etapa***

- **Fuente de señal binaria:** La fuente de señal considera bits estadísticamente independientes entre sí. Es decir:

$$\begin{aligned} P_0 &= \Pr(b[n]=0) \\ P_1 &= \Pr(b[n]=1) \\ P_0 + P_1 &= 1 \end{aligned} \qquad (4.1)$$

- **Composición de la trama:** A partir de los bits generados por la fuente de señal binaria se componen los paquetes de transporte del multiplex de entrada en formato MPEG-2, añadiendo un byte de sincronización al principio de cada uno de ellos.

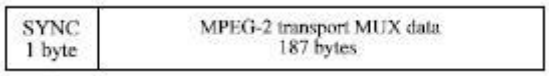


Figura 22 Formato del paquete MPEG-2. Figura extraída de [1].

Cada paquete se forma de la manera que se puede ver en la Figura 22. Los paquetes tienen una longitud de 188 bytes, siendo el primer byte el de sincronización, cuyo valor es siempre: 0x47 (01000111), y los 187 bytes restantes son los datos MPEG-2.

Para proporcionar una señal de inicialización en el decodificador del receptor que permita identificar cada conjunto de 8 paquetes de transporte afectados por la aleatorización, el byte de sincronización del primer paquete de cada conjunto de 8 está invertido, pasando de 0x47 (01000111) a 0xB8 (10111000). Este proceso es el que se conoce por *adaptación del flujo de transporte*.

- **Scrambler o aleatorizador de bits:** Es el encargado de asegurar la *dispersión de energía*. Para ello, hace que la secuencia de entrada pase a ser cuasi-aleatoria, de forma que se eviten largas series de ceros o de unos.

La aleatorización se realiza mediante el proceso que se esquematiza en la Figura 23:

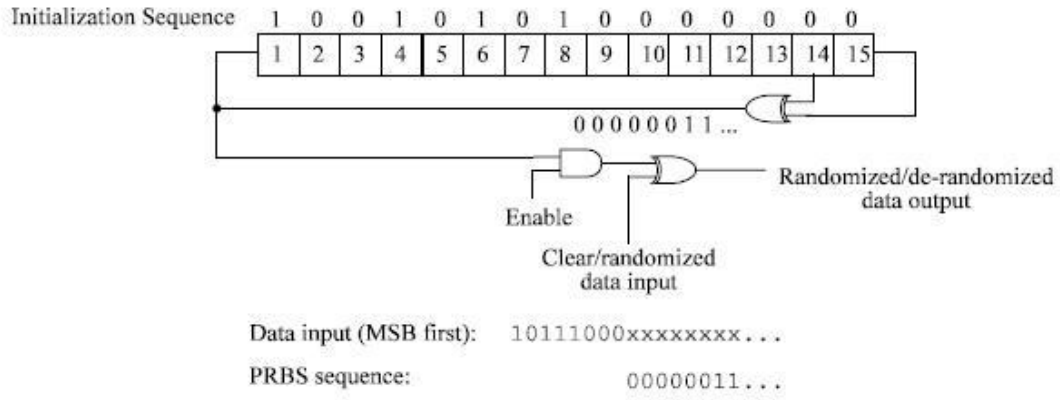


Figura 23 Generador PRBS para la dispersión de energía del Flujo de Transporte. Figura extraída de [1].

En este proceso se trata de obtener una *secuencia binaria pseudoaleatoria (PRBS)*, para lo cual se emplea un generador que usa el polinomio:

$$1 + x^{14} + x^{15} \tag{4.2}$$

Los registros del generador PRBS, que sirve tanto para desordenar en transmisión como para ordenar en recepción, tienen cargada la secuencia 100101010000000, la cual debe iniciarse al comienzo de un conjunto de 8 paquetes de transporte. Los bytes de sincronización no se ven afectados por la aleatorización.

El primer bit en la salida del generador PRBS se aplica al primer bit (el MSB) del primer byte que sigue al byte de sincronización invertido del TS (el 0xB8) y así sucesivamente con los demás bits. Para ayudar al resto de funciones de la sincronización, durante los bytes de sincronismo de los otros 7 paquetes, la generación PRBS continúa, pero está deshabilitada, dejando estos bytes sin aleatorizar. Así, el periodo de la secuencia PRBS es de 1.503 bytes (188 x 8 – 1).

El proceso de aleatorización debe permanecer activo incluso cuando no exista flujo de transporte a la entrada o cuando éste no cumpla con el formato del estándar MPEG-2.

El resultado del proceso de *Adaptación y Dispersión de Energía del Flujo de Transporte*" se indica en la Figura 24.

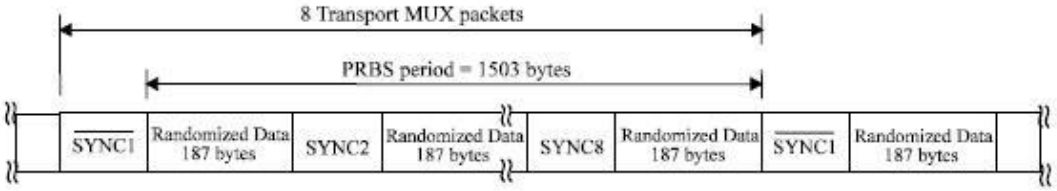


Figura 24 Resultado de la adaptación y dispersión de energía del TS MPEG-2. Figura extraída de [1].

Este resultado, es el vector de bits $b_{scr}[n]$ que pasará a la siguiente etapa del simulador.

- ***Procesado para la representación de resultados asociados a la etapa***

  - **Autocorrelación de las secuencias de bits:** Se realiza el cálculo de la autocorrelación de bits para ofrecer un estudio de las prestaciones del scrambler.

    Para mantener la media de las correlaciones a cero se consideran los siguientes valores: bit '0' = -1 y bit '1' = +1.

4.3.1.3 Resultados

En esta etapa se obtienen tanto resultados numéricos como resultados gráficos.

- ***Resultados numéricos***

Los resultados numéricos facilitados en esta etapa son de tipo teórico:

- **Byte de sincronismo:** Se trata del byte de sincronización, cuyo valor es siempre 0x47 (01000111).

- **Secuencia de scrambling:** Se trata del polinomio (4.2) utilizado para generar la secuencia binaria pseudoaleatoria (PRBS).

- ***Resultados gráficos***

Los resultados gráficos facilitados se dividen en dos grupos:

- **Secuencias de bits:** En las correspondientes gráficas se muestra un tramo correspondiente a los 200 primeros bits.

  - **Bits fuente:** Muestra los bits correspondientes a los paquetes MPEG-2 compuestos del byte de sincronismo y de los bits generados con la probabilidad de bit 0 escogida por el usuario. Ver Figura 22.

  - **Bits aleatorizados:** Muestra los bits correspondientes a la salida de etapa, después de haber sido aleatorizados. Su formato es el que puede verse en la Figura 24.

  - **Bits fuente y aleatorizados:** Solapa en una misma gráfica las dos secuencias de bits descritas anteriormente. Sólo disponible en la versión Java.

- **Autocorrelación de bits:** Estudio de las características de la correlación de los bits antes y después del paso por el scrambler.

- **Autocorrelación bits fuente:** Muestra la autocorrelación de los bits fuente descritos, antes de realizar el scrambling.

- **Autocorrelación bits aleatorizados:** Muestra la autocorrelación de los bits aleatorizados descritos, después de su paso por el scrambling.

- **Autocorrelación bits fuente y aleatorizados:** Solapa en una misma gráfica las dos autocorrelaciones de bits descritas anteriormente.

➢ ***Intercambio de ficheros***

- **Guardar:** En la presente etapa, se permite guardar en un fichero la secuencia de bits fuente generada o la secuencia de bits aleatorizados $b_{scr}[n]$.

- **Leer:** Permite leer de un fichero la secuencia de bits aleatorizados $b_{scr}[n]$ para intercambiarla por la correspondiente secuencia generada. Esta secuencia de bits debe tener la misma longitud que la generada por el simulador.

4.3.1.4 Simulación 16 paquetes MPEG-2 TS

El ejemplo que se propone en la versión MATLAB del simulador arranca en esta etapa. Se seleccionan 16 paquetes MPEG-2 TS y una probabilidad de aparición del bit 0 igual a 0,7.

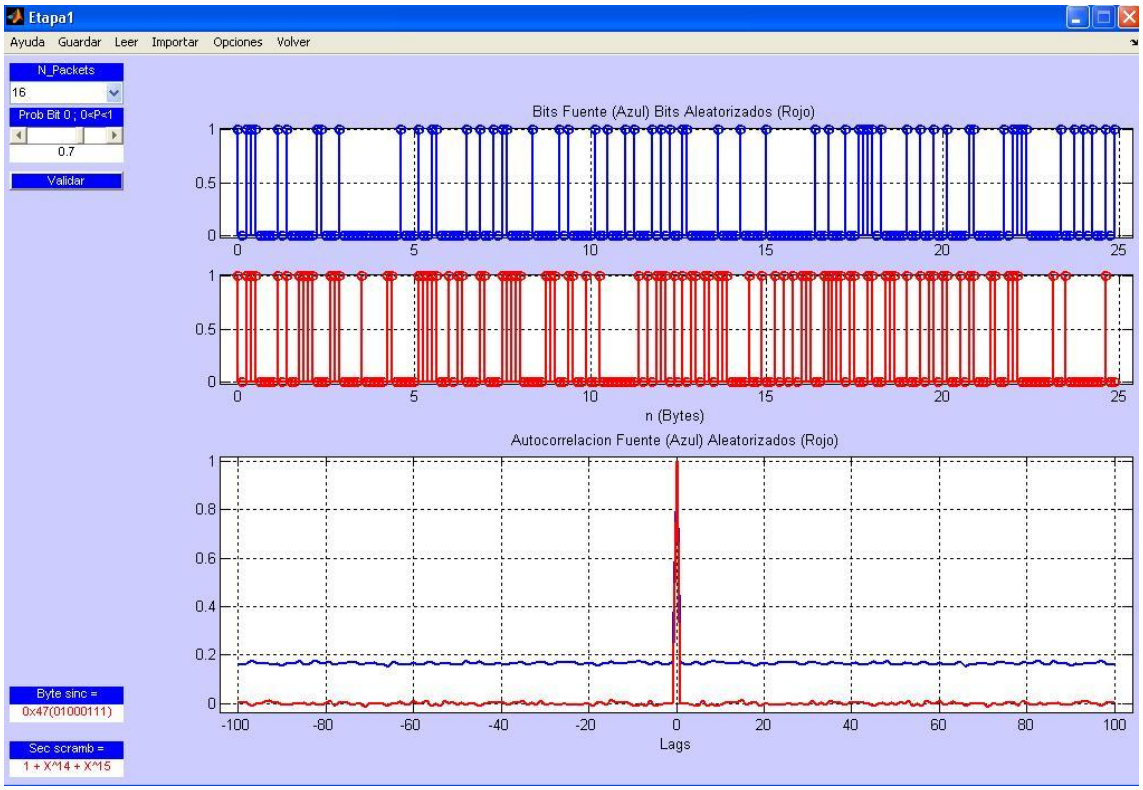


Figura 25 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 1: Generación de la señal

En la Figura 25 pueden observarse los dos grupos de gráficas descritos. En la parte superior se muestran las primeras 200 muestras tanto de los bits generados, como de los bits aleatorizados. Se puede observar que el primer byte de la secuencia queda intacto tras su paso por el aleatorizador, por tratarse del byte de sincronismo invertido.

En la parte inferior se presentan las autocorrelaciones de las secuencias de bits antes y después del paso por scrambler. Dicha gráfica ofrece un estudio de las prestaciones del scrambler. Se extrae que el scrambler se comporta de tal forma que aleatoriza las secuencias de bits de forma que secuencias generadas sin suponer equiprobabilidad de bit, de cara a la transmisión adquieren propiedades de equiprobabilidad y por tanto son más robustas. Es decir, se consigue la dispersión de energía, y con ello se logra un espectro de transmisión plano.

4.3.1.5 Retos de programación

De esta etapa no se destaca ningún reto de programación particular.

4.3.1.6 Análisis de la etapa

En la primera etapa se decide un parámetro de gran importancia en el sistema, que es el número de paquetes MPEG-2 TS a enviar.

Por el hecho de enviar 8, 16 ó 24 paquetes solamente, se obtienen unos resultados en etapas posteriores que se diferencian en ciertos aspectos de los esperados. En una transmisión real de vídeo MPEG-2 se envían muchos más paquetes, pero en el caso de esta implementación no sería viable enviar mucha cantidad de información pues se saturaría el sistema. En las etapas pertinentes se explicarán las consideraciones a tener en cuenta por este hecho.

4.3.2 Etapa 2: Codificación externa (Reed-Solomon) y entrelazado externo (Forney)

En la segunda etapa del sistema se realizan las operaciones de procesado de señal correspondientes a la codificación y entrelazado externos.

Los bits aleatorizados $b_{scr}[n]$, que tienen la estructura que puede verse en la Figura 24, pasan primero por el codificador externo (de tipo Reed-Solomon) y posteriormente por el entrelazador convolucional (basado en la aproximación de Forney). Los dos bloques de procesado operan a nivel de byte, por lo tanto se realiza una conversión de bits a bytes previa al codificador y la correspondiente conversión de bytes a bits a la salida del entrelazador.

El esquema que adopta la etapa es el siguiente:

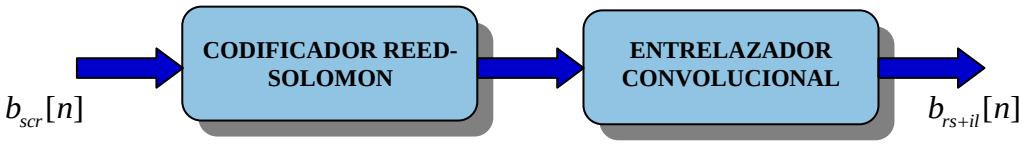


Figura 26 Esquema de la etapa 1 del sistema DVB-T

Como puede observarse en el esquema, la señal de salida de la etapa será $b_{rs+il}[n]$, que es el vector de bits a la salida del entrelazador.

4.3.2.1 Parámetros de entrada

La presente etapa del simulador posee un único parámetro de entrada:

- **Codificación externa e interleaving:** Permite elegir entre realizar codificación y entrelazado externos o, simplemente, codificación externa. Con este parámetro se deja abierta la posibilidad por parte del usuario de comprobar la influencia del entrelazado externo en la decodificación Reed-Solomon en recepción.

4.3.2.2 Procesado de la señal

A continuación se describen las operaciones de procesado de señal especificadas en el estándar para la implementación de la etapa:

- ***Procesado propio de la etapa***

  - **Codificación externa (Reed-Solomon):** La codificación externa, es el primer bloque del transmisor encargado de añadir redundancia en la estructura de los paquetes de transporte, es decir, es el primer bloque codificador de información.

La codificación externa se complementa con la codificación interna, que se estudia en la siguiente etapa. Gracias a la redundancia introducida, se permite la corrección de errores en la recepción (FEC).

La codificación usada es de tipo Reed-Solomon *RS (204, 188, t=8)*, que pertenece a una subclase de códigos BCH y es una versión acortada de la codificación original *RS (255, 239, t=8)*, mediante la cual se añaden 16 bytes de paridad a los iniciales 188 de cada paquete de transporte, resultando un total de 204 bytes. Esta codificación, que se aplica a todos los paquetes de transporte aleatorizados (incluso al byte de sincronismo), permite corregir hasta 8 bytes erróneos (t=8).

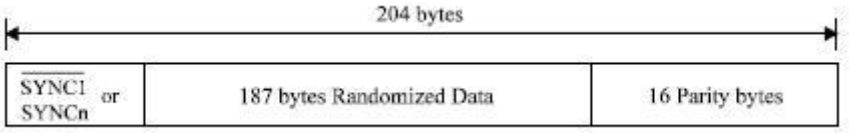


Figura 27 Paquetes protegidos contra errores RS (204, 188, t=8). Figura extraída de [1].

La implementación de este procesado de señal comienza con la conversión del vector de bits $b_{scr}[n]$ a bytes $B_{scr}[n]$, puesto que el codificador Reed-Solomon trabaja con bytes. Los símbolos del código (bytes) se definen dentro del campo de Galois $GF(2^8)$, y el polinomio generador de campo es:

$$p(x) = x^8 + x^4 + x^3 + x^2 + 1 \tag{4.3}$$

A continuación se aplica la forma acortada del codificador, añadiendo 51 bytes nulos delante de los 188 bytes de información, con lo que se obtienen 239 bytes. Al pasar por el codificador *RS (255, 239, t=8)* se añaden los 16 bytes de paridad; finalmente se eliminan los 51 bytes nulos de los 255 totales, con lo que resultan los 204 bytes para cada paquete de transporte como se observa en la Figura 27.

El polinomio generador de código para el codificador Reed-Solomon del sistema DVB-T es el siguiente:

$$g(x) = (x + \lambda^0)(x + \lambda^1)(x + \lambda^2)\cdots(x + \lambda^{15}) \tag{4.4}$$

Donde $\lambda = 02_{HEX}$.

A la salida del codificador Reed-Solomon se obtiene el vector de bytes $B_{rs}[n]$.

Para información sobre campos de Galois *GF(q)*, consultar referencia [32], y para ampliar la información sobre el codificador Reed-Solomon, consultar el apartado *retos de programación* de la etapa (4.3.2.5).

- **Entrelazado externo (Forney):** La Figura 29 muestra el esquema conceptual del entrelazado convolucional, relativo al byte y con profundidad I=12, a que se someten los datos $B_{rs}[n]$ que han sido previamente protegidos mediante la codificación externa Reed-Solomon. El resultado es la siguiente estructura de datos entrelazada:

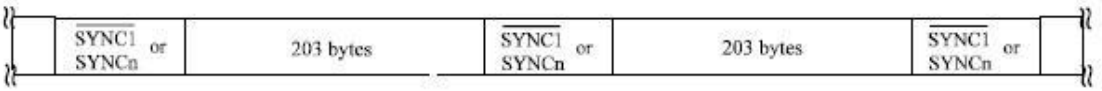


Figura 28 Estructura de la información después del entrelazador externo. Figura extraída de [1].

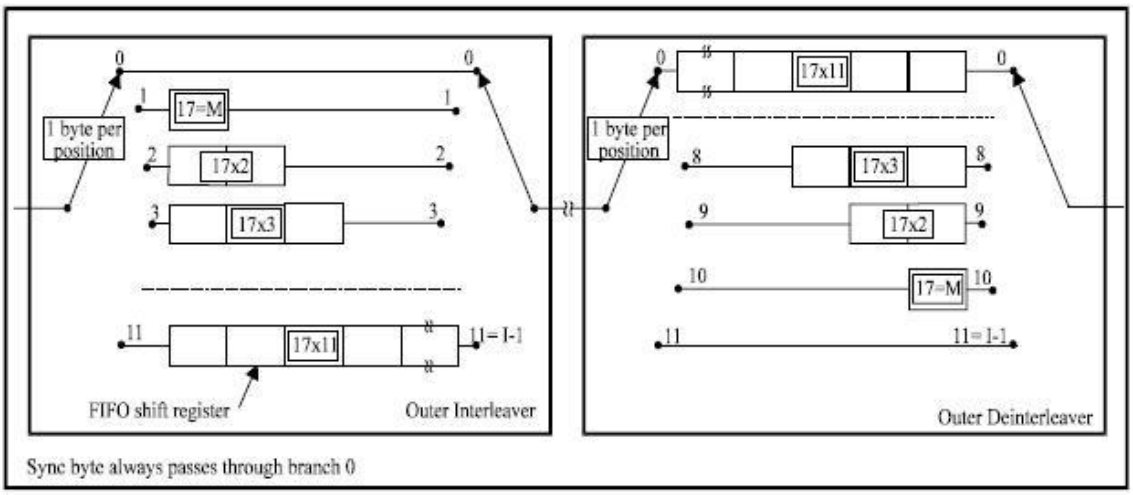


Figura 29 Diagrama conceptual del entrelazado y desentrelazado convolucional externo (Forney). Figura extraída de [1].

El proceso de entrelazado convolucional se basa en la aproximación de Forney, compatible con la aproximación de Ramsey tipo III, con I=12. Los bytes de datos entrelazados, pertenecientes a los paquetes de transporte protegidos, están delimitados por los bytes de sincronización MPEG-2 invertidos y no invertidos, que no sufren alteración alguna. El entrelazado preserva, por tanto, la periodicidad de 204 bytes de los paquetes de transporte.

El entrelazador se compone de 12 ramas (I=12), cíclicamente conectadas al flujo de bytes de entrada mediante el conmutador de entrada. Cada rama j constituye un registro de desplazamiento FIFO (First-In, First-Out) con profundidad jxM células, donde M = 17 = N/I, con N = 204.

Cada registro de desplazamiento FIFO tiene una longitud de 1 byte, y cada vez que el conmutador cambia de posición carga de nuevo el registro. Los conmutadores de entrada y salida deben estar sincronizados, y los bytes de sincronismo invertidos y no invertidos deben rutarse por la rama $j=0$ (que corresponde a retardo nulo) del entrelazador para poder ser localizados en recepción.

El resultado del proceso de entrelazado es que cada byte de los paquetes de transporte se encuentra desplazado en el tiempo un número de posiciones igual a *jx17*, con lo que los bytes originales de un paquete de transporte quedarán repartidos entre dos paquetes consecutivos.

Todo este proceso reduce los errores a ráfagas introducidos por el canal de transmisión, ya que después de la reordenación de los datos en el receptor, estos errores se habrán distribuido entre paquetes sucesivos, lo que favorecerá que no se excedan los límites en los que la codificación Reed-Solomon puede recuperar la información original (recordemos que puede corregir hasta 8 bytes erróneos consecutivos).

Tras realizar el entrelazado a nivel de byte, se finaliza el procesado de la etapa convirtiendo a bits $b_{rs+il}[n]$ el vector de bytes $B_{rs+il}[n]$. Este resultado es el vector de bits $b_{rs+il}[n]$ que pasará a la siguiente etapa del simulador.

En el apartado retos de programación, se puede analizar el algoritmo seguido en la programación del entrelazador externo.

- ***Procesado para la representación de resultados asociados a la etapa***

  - **Entrelazado externo (Forney):** En la versión MATLAB del simulador, se ha utilizado el algoritmo de entrelazado externo que se encuentra detallado en el apartado 4.3.2.5: Retos de programación, adaptándolo para realizar una presentación gráfica del entrelazado teórico mediante la función *Images_children*, la cual se estudiará en el capítulo siguiente (5.2.4.1).

    La adaptación consiste en aplicar el algoritmo de entrelazado convolucional sobre una matriz de datos teórica. Esta matriz ha sido creada de manera que cada paquete protegido Reed-Solomon sea representado por un color:

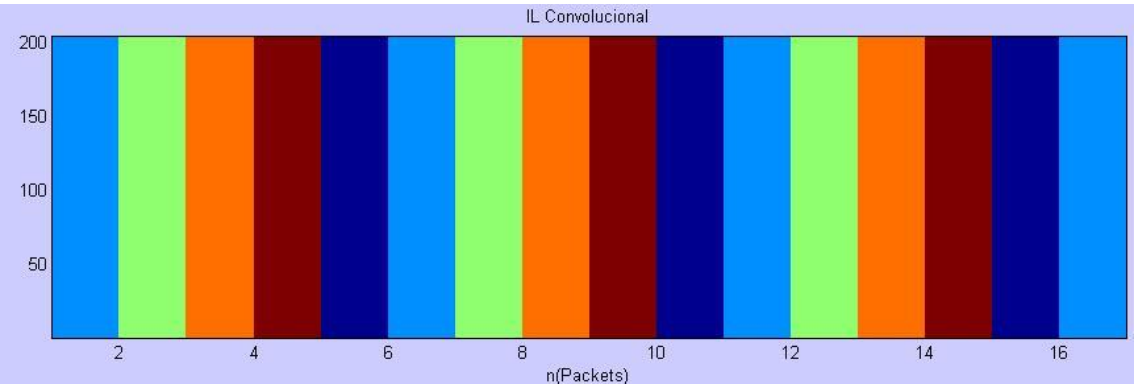


Figura 30 Representación gráfica de la matriz teórica antes del entrelazado convolucional

En la Figura 31 puede observarse el resultado del entrelazado convolucional.

4.3.2.3 Resultados

En esta etapa se obtienen dos resultados gráficos y dos resultados numéricos.

- ***Resultados numéricos***

Los resultados numéricos facilitados en esta etapa son de tipo teórico:

- **Polinomio generador de campo:** El campo de Galois de 256 elementos sobre el que opera el codificador Reed-Solomon implementado está definido por el polinomio generador de campo (4.3).

- **Polinomio generador de código:** Se trata del polinomio (4.4) utilizado para generar las palabras codificadas Reed-Solomon.

- ***Resultados gráficos***

Los resultados gráficos facilitados son dos:

- **Palabra Reed-Solomon:** Representación de la primera palabra protegida contra errores Reed-Solomon. En color azul se muestran los bytes originales y en rojo los bytes de redundancia añadidos por el codificador.

- **Entrelazado convolucional:** Estudio del entrelazador convolucional de Forney.

  - **Versión MATLAB:** Muestra un estudio teórico del entrelazado convolucional. Esta representación es el resultado de aplicar el algoritmo de entrelazado convolucional sobre la matriz representada en la Figura 30.

  - **Versión Java:** Muestra los primeros 204 bytes (longitud de la palabra Reed-Solomon) del resultado del entrelazado convolucional aplicado sobre los datos. En esta gráfica se puede observar como solo queda un byte original de cada 12 (número de ramas) de entrada al entrelazador, esto es porque la primera rama se conecta directamente a la salida (retardo nulo). También se observa que el byte de sincronismo (47_{HEX}) permanece en su posición original.

- ***Intercambio de ficheros***

- **Guardar:** Se permite guardar en un fichero los bits $b_{rs+il}[n]$ a la salida de la etapa.

- **Leer:** Permite leer de un fichero la secuencia de bits codificados y aleatorizados $b_{rs+il}[n]$ para intercambiarla por la correspondiente secuencia generada. Esta secuencia de bits debe tener la misma longitud que la generada por el simulador.

4.3.2.4 Simulación 16 paquetes transporte MPEG-2

En la presente etapa se selecciona que se realice tanto la codificación como el entrelazado externo.

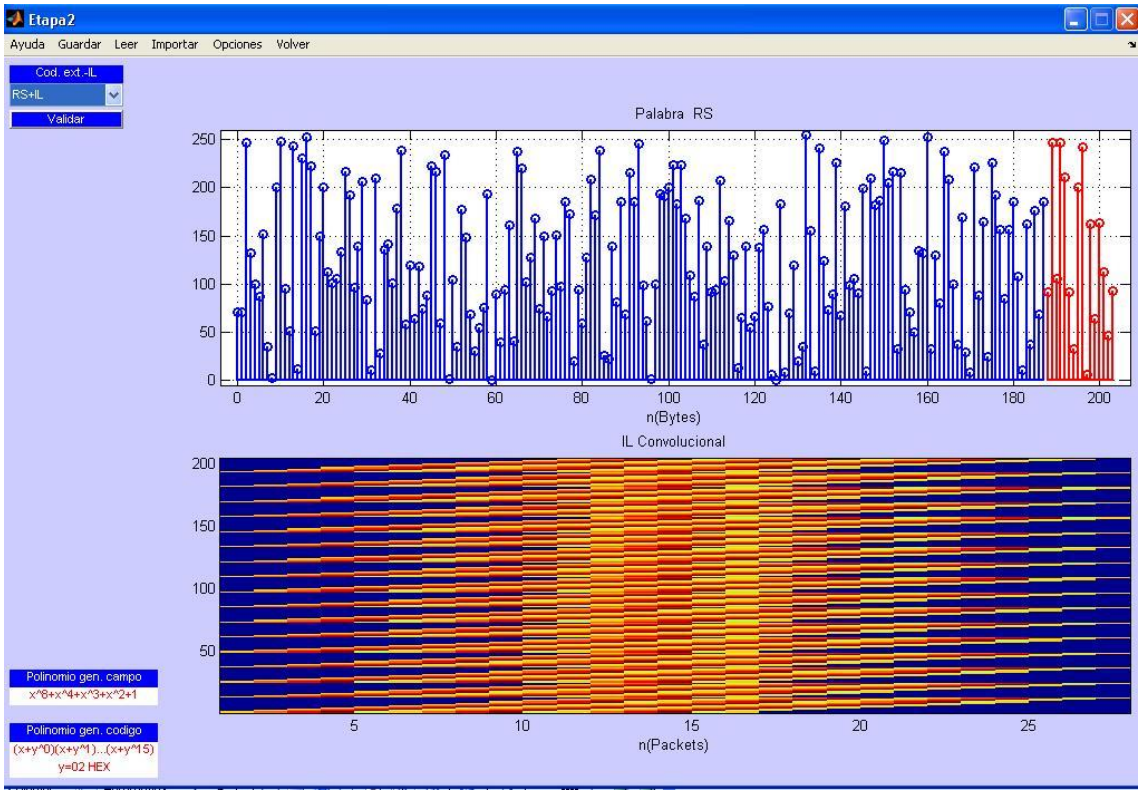


Figura 31 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 2: Codificación externa (Reed-Solomon) y entrelazado externo (Forney)

En la Figura 31 se pueden observarse dos gráficas. En la primera de ellas se muestra la primera palabra codificada Reed-Solomon. A los 188 bytes del paquete MPEG-2 TS de entrada al codificador (color azul), se han añadido los 16 bytes adicionales de redundancia (color rojo) que ayudarán a corregir errores en recepción.

En la gráfica adicional (versión MATLAB) se presenta el funcionamiento teórico del entrelazador convolucional. Dicha representación sería el resultado de aplicar el algoritmo de entrelazado sobre la matriz de datos teórica mostrada en la Figura 30. En color azul oscuro se representan los ceros agregados por el entrelazador, y los diferentes colores muestran como van entrelazándose los datos.

4.3.2.5 Retos de programación

Los retos que ha planteado la programación de esta etapa están relacionados con la implementación del entrelazador externo y el codificador Reed-Solomon.

4.3.2.5.1 Retos en la programación MATLAB

- ***Codificador Reed-Solomon***

El primer reto importante que se ha afrontado en la programación MATLAB del simulador DVB-T, ha sido el codificador Reed-Solomon que se implementa en la presente etapa.

La función de MATLAB *code=rsenc(msg,n,k,genpoly)* ha facilitado mucho la tarea, aunque se ha preparado una subrutina de procesado para la adaptación de la información a transmitir a los requerimientos de dicha función.

Las tareas que lleva a cabo la subrutina son las siguientes:

- ✓ Definir los parámetros *n* y *k*, longitud de la palabra codificada RS y la palabra a codificar respectivamente.
- ✓ Preparar el polinomio generador utilizando *genpoly=rsgenpoly(n,k,primpoly,b)*, donde *primpoly* se deja en blanco para que se use el polinomio primitivo por defecto. El parámetro *b* se comenta más adelante.
- ✓ Crear la matriz *msg'* colocando fila a fila cada paquete de símbolos a codificar.
- ✓ A partir de *msg',* y utilizando la subrutina de MATLAB *msg =gf(msg',m),* obtener la matriz de datos en el campo de Galois $GF(2^m)$, donde *m* es el número de bits por símbolo (en nuestro caso los símbolos son bytes, por lo tanto *m*=8).
- ✓ Llamar a la función *rsenc* con los parámetros definidos, la cuál devuelve *code*, que son los símbolos codificados dentro del campo de Galois.
- ✓ Por último, utilizando la función de MATLAB *double(code),* transformar la matriz *code* a formato *double* para poder operar con sus datos en las siguientes etapas.

Hay que prestar especial atención al parámetro *b* al crear el polinomio generador con la subrutina *rsgenpoly*. En la ecuación (4.5) se puede ver el formato en que MATLAB interpreta dicho polinomio, el cual está definido en [1] de la manera que se ha mostrado en la ecuación (4.4).

$$(x-\alpha^{b})(x-\alpha^{b+1})...(x-\alpha^{b+n-k-1}) \qquad (4.5)$$

Primero hay que fijarse en la selección del parámetro *b*, puesto que lo típico es que sea 1, pero en el simulador que nos ocupa se ha de seleccionar 0. Y segundo, se debe tener en cuenta que, en los campos de Galois, $-\alpha^{b}$ es exactamente lo mismo que α^{b}.

- ➢ ***Entrelazado externo (Forney)***

Se trata del *interleaving* de byte que se realiza en la etapa. Se puede consultar el apartado 4.3.2.2 para ver la definición teórica que aquí se implementa.

En la elaboración del algoritmo de *interleaving* descrito a continuación, se ha utilizado la referencia [32] para un entendimiento mayor del entrelazador convolucional definido por el estándar DVB-T. Los pasos seguidos son:

- ✓ Definir número de ramas *B* del entrelazador y el parámetro *M*, que multiplicado por el número de rama (*0…B-1)* da como resultado la profundidad del registro de desplazamiento FIFO que constituye cada una de ellas.

- ✓ Crear la matriz *A_IL* de palabras Reed-Solomon (*RS)* por columnas.

- ✓ Inicializar, mediante la función *zeros*, la matriz *B_IL* que contendrá el resultado del entrelazado. El número de filas viene definido por el número de bytes de cada palabra codificada *RS.* El número de columnas viene definido por *B + N – 1,* donde *N* es el número de paquetes de transporte transmitidos.

- ✓ Para cada valor *i1* de *1 a M* y *Delay* de *1 a B,* calcular *i_fila=(i1-1)xB+i2* y rellenar las columnas de *B_IL* que van de *Delay* a *Delay+N -1* con la *i_fila* de *A_IL*. Tras esta iteración, se obtiene la matriz *B_IL* de bytes entrelazados.

4.3.2.5.2 Retos en la programación Java

El primer reto importante en la programación Java del simulador DVB-T ha sido también el codificador Reed-Solomon.

Para implementarlo, se ha utilizado la referencia [40], tanto para el codificador como para el decodificador. También se han tenido en cuenta los detalles mencionados en la programación MATLAB del codificador RS.

- ➢ ***Codificador Reed-Solomon***

El primer paso llevado a cabo, ha sido la creación de las tablas (ROM's), que permiten convertir de notación binaria a notación exponencial dentro del campo de Galois $GF(2^8)$ definido en 4.3.2.2 y viceversa. Gracias a estas tablas, el tiempo y el número de operaciones se reducen notablemente. También se han creado con este propósito las operaciones suma y producto dentro del campo de Galois $GF(2^8)$.

Después, se define el polinomio generador de código, cuya ecuación es (4.4), tras su multiplicación, con los coeficientes definidos en notación exponencial dentro del campo de Galois $GF(2^8)$:

$$g(x) = x^{16} + \lambda^{120}x^{15} + \lambda^{104}x^{14} + \lambda^{107}x^{13} + \lambda^{109}x^{12} + \lambda^{102}x^{11} + \lambda^{161}x^{10} + \lambda^{76}x^{9} + \lambda^{3}x^{8} + \lambda^{91}x^{7} + \lambda^{191}x^{6} + \lambda^{147}x^{5} + \lambda^{169}x^{4} + \lambda^{182}x^{3} + \lambda^{194}x^{2} + \lambda^{225}x + \lambda^{120} \quad (4.6)$$

Por último se calculan los símbolos de paridad programando el resto de la división:

$$b(x) = \mathrm{Re}\,m\left\{\frac{k(x)x^{n-k}}{g(x)}\right\} \qquad (4.7)$$

Donde k(x) es la representación polinómica del vector de información de entrada, n=255 y k=239 en la especificación DVB-T.

El esquema del codificador Reed-Solomon para calcular la ecuación (4.7) es:

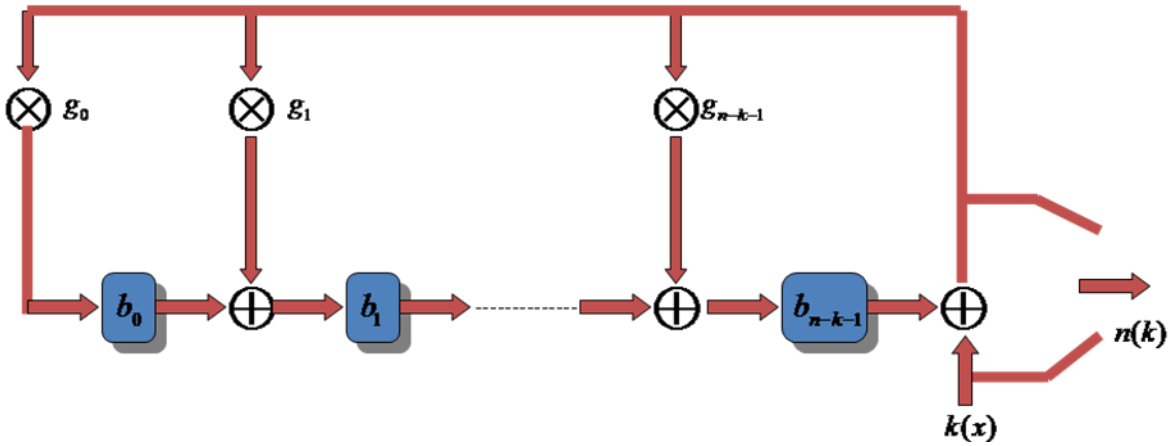


Figura 32 Codificador sistemático para códigos RS

4.3.2.6 Análisis de la etapa

En esta etapa conviene analizar el efecto del entrelazador convolucional sobre los datos enviados, considerando aspectos de programación.

El algoritmo que realiza el entrelazado añade símbolos de relleno en las posiciones donde no se envían datos; escogemos añadir ceros por simplicidad. Estos símbolos afectan en una medida considerable al simulador por el hecho de, como ya se había comentado en el análisis de la etapa 1 (4.3.1.6), no enviar una gran cantidad de información (enviamos un número pequeño de paquetes MPEG-2 TS). En las etapas 3 (4.3.3.6) y 6 (4.3.6.6) se pueden ver los efectos de estos ceros agregados por el algoritmo de entrelazado.

Cabe mencionar que los ceros agregados afectan al sistema en aspecto de potencias, tal como se detalla en el apartado 4.3.5.6. También afectarán a la energía transportada por la señal a transmitir, puesto que la aumentan. Pero en el sistema DVB-T se consideran bits útiles, puesto que se utilizan para entrelazar; además en una transmisión real afectan poco puesto que son insignificantes respecto al número total de bits enviados en formato MPEG-2 TS.

4.3.3 Etapa 3: Codificador interno (convolucional) + Entrelazado interno + Mapping de símbolos

En la presente etapa se realiza el proceso de codificación y entrelazado denominado interno, y el mapeo de los símbolos.

Los bits $b_{rs+il}[n]$, que ya han atravesado el proceso de codificación y entrelazado externo, son sometidos a los siguientes bloques de procesado de forma secuencial: codificador convolucional o codificador interno, entrelazador de bits, mapeo de los símbolos, que constituye la trama de símbolos que contiene la información y entrelazador de símbolos que, junto al entrelazador de bits, realiza el entrelazado interno.

A la salida del entrelazador de símbolos, se obtiene el vector de símbolos $s_{MAP}[n]$ que se envía a la siguiente etapa del simulador.

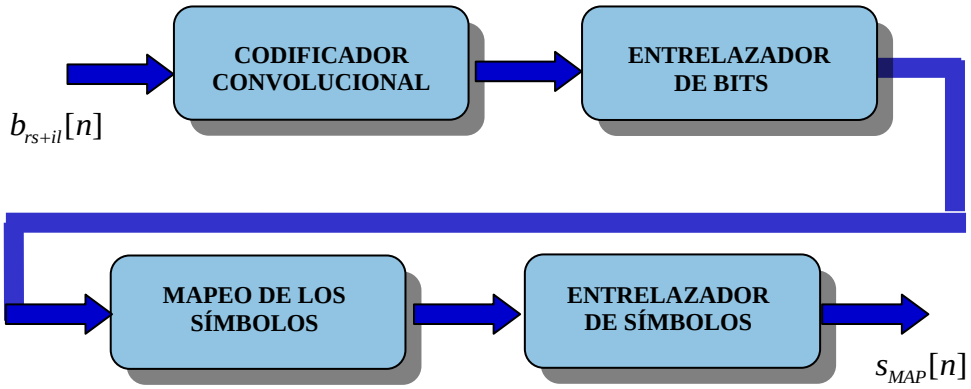


Figura 33 Esquema de la etapa 3 del sistema DVB-T

4.3.3.1 Parámetros de entrada

La presente etapa del simulador posee cuatro parámetros de entrada:

- **Code rate:** permite escoger la tasa de codificación R_c del codificador convolucional entre los cinco posibles valores que especifica el estándar [1]: 1/2, 2/3, 3/4, 5/6 y 7/8. La protección contra errores será menor cuanto mayor sea la tasa escogida, pero aumentará la capacidad del canal.

- **Codificador convolucional e interleaving interno:** permite elegir entre cuatro opciones, todas ellas tienen activos los bloques de procesado de codificación y mapeo.

  La primera opción sería sin entrelazado. La segunda activa el entrelazado de bits. La tercera activa el entrelazado de símbolos y por último la cuarta opción permite activar el entrelazado relativo al bit y el entrelazado de símbolos, que sería el procesado completo especificado por el estándar. Con este parámetro se deja abierta la posibilidad por parte del usuario de

comprobar la influencia tanto de cada entrelazado por separado, como del conjunto completo de entrelazado interno en la decodificación de Viterbi en recepción.

- **Tipo de modulación:** determina la modulación que se usará en el simulador. Se ofrecen las tres opciones que contempla el estándar: QPSK, 16QAM y 64QAM.

- **Parámetro alfa:** permite escoger entre los tres valores especificados por el estándar DVB-T: modulación uniforme con α=1, y modulaciones no uniformes con α=2 y α=4. El parámetro de modulación α indica la distancia que existe entre dos puntos adyacentes de distintos cuadrantes.

  Cabe comentar que, gracias a este parámetro α, se han simulado constelaciones no-uniformes propias de la transmisión jerárquica. El tipo de transmisión jerárquica no ha sido implementada para el simulador DVB-T desarrollado, tal como se explica en la sección 4.2.

4.3.3.2 Procesado de la señal

A continuación se describen los cálculos de procesado de señal que se implementan en esta etapa:

- ***Procesado propio de la etapa***

  - **Codificador convolucional:** Después de la codificación y entrelazado externos, el vector de bits $b_{rs+il}[n]$, que transporta los datos, se somete a un nuevo proceso de codificación y entrelazado, denominados internos. La codificación interna es de tipo convolucional y se complementa con un *puncturing* o *perforado*, obteniéndose diferentes secuencias transmitidas dependiendo del *puncturing* escogido.

    Los datos que provienen de la etapa 2 deben ser codificados con un codificador convolucional de tasa R_c=1/2, 2/3, 3/4, 5/6 ó 7/8, en función de la elección del usuario. Los polinomios generadores utilizados son: $G1 = 171_{OCT}$ y $G2 = 133_{OCT}$ para la tasa R_c=1/2, el resto de tasas más altas derivan de ésta utilizando puncturing. La Figura 34 muestra el esquema del codificador convolucional de relación 1/2 con 64 estados (K=7 tomas), en que se basa el mecanismo:

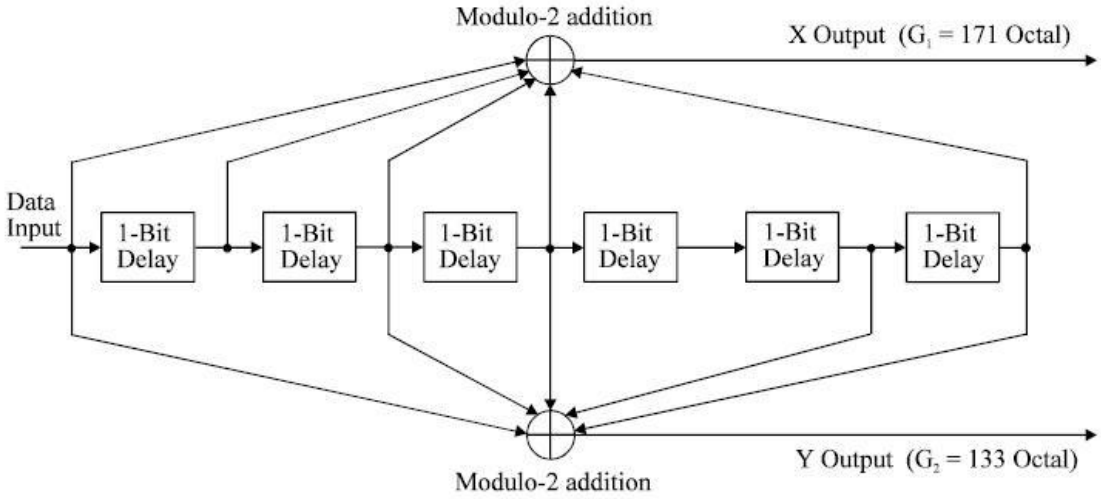


Figura 34 Codificador convolucional de relación 1/2. Figura extraída de [1].

Este codificador está orientado al bit y distribuye en dos salidas (X e Y) el flujo de datos original a base de combinar (sumas módulo-2) los datos de entrada con los obtenidos en las tomas situadas detrás de una serie de registros de desplazamiento. Se suma a los datos de entrada el dato de la toma correspondiente cuando el valor del polinomio generador (expresado en binario) en la toma en cuestión corresponde a "1" y no se usa el dato de la toma cuando el valor es "0".

Para la tasa de codificación R_c=1/2, se logra una potente protección contra errores aleatorios, pero se baja a la mitad la capacidad del canal. Para no limitar tanto la capacidad del canal, el sistema permite seleccionar para la transmisión sólo algunos de los datos obtenidos en las salidas X e Y, los cuales son posteriormente convertidos a secuencia en serie; es el proceso de puncturing.

Code Rates r	Puncturing pattern	Transmitted sequence (after parallel-to-serial conversion)
1/2	X: 1 Y: 1	$X_1 Y_1$
2/3	X: 1 0 Y: 1 1	$X_1 Y_1 Y_2$
3/4	X: 1 0 1 Y: 1 1 0	$X_1 Y_1 Y_2 X_3$
5/6	X: 1 0 1 0 1 Y: 1 1 0 1 0	$X_1 Y_1 Y_2 X_3 Y_4 X_5$
7/8	X: 1 0 0 0 1 0 1 Y: 1 1 1 1 0 1 0	$X_1 Y_1 Y_2 Y_3 Y_4 X_5 Y_6 X_7$

Figura 35 Plantilla de puncturing y secuencia transmitida para los posibles R_c. Figura extraída de [1].

El codificador convolucional empleado en la versión Java del simulador, ha sido implementado por Pedro Espinosa Fricke para el departamento de Teoría de la Señal y Comunicaciones de la Universidad Politécnica de Cataluña [28].

- **Entrelazador de bits:** Se trata del primer proceso de entrelazado que, junto al entrelazador de símbolos, compone el entrelazado denominado interno.

  Inicialmente, el único flujo de bits de salida del bloque codificador x_i, se demultiplexa en *v* sub-flujos, siendo *v* el número de bits por símbolo de la modulación empleada. El esquema de demultiplexado de los bits x_i en los resultantes $b_{i,j}$ es el siguiente:

  Para QPSK:

  $$x_0 \rightarrow b_{0,0}$$
  $$x_1 \rightarrow b_{1,0}$$

  Para 16QAM:

  $$x_0 \rightarrow b_{0,0}$$
  $$x_1 \rightarrow b_{2,0}$$
  $$x_2 \rightarrow b_{1,0}$$
  $$x_3 \rightarrow b_{3,0}$$

  Y para 64QAM:

  $$x_0 \rightarrow b_{0,0}$$
  $$x_1 \rightarrow b_{2,0}$$
  $$x_2 \rightarrow b_{4,0}$$
  $$x_3 \rightarrow b_{1,0}$$
  $$x_4 \rightarrow b_{3,0}$$
  $$x_5 \rightarrow b_{5,0}$$

  Después del demultiplexado, cada sub-flujo se procesa por separado en un entrelazador de bits. Hay por tanto *v* entrelazadores de bits (hasta 6) que se numeran desde *I0* hasta *I5*. Los entrelazadores *I0* e *I1* se usan para QPSK, *I0* a *I3* para 16-QAM y finalmente *I0* a *I5* para 64-QAM.

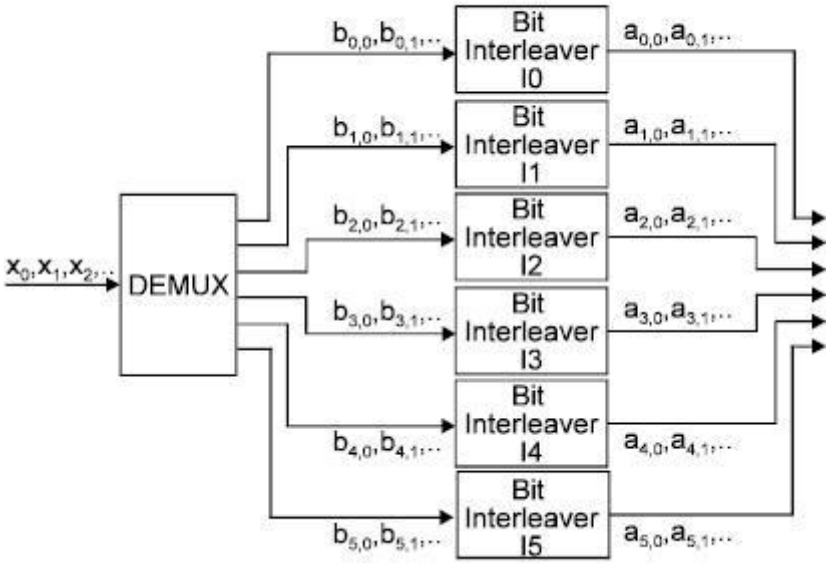


Figura 36 Entrelazado interno para modo "no-jerárquico". Figura extraída de [1].

El entrelazado de bits se realiza por bloques. El tamaño de estos bloques es de 126 bits para todos los entrelazadores, aunque la secuencia de entrelazado es diferente de unos a otros. Dicho tamaño implica que este proceso se repite un número exacto de veces por cada símbolo OFDM, 12 veces en el caso de la transmisión 2k implementada.

Tal como se puede ver en la Figura 36, los bloques de bits de entrada a cada entrelazador están compuestos por:

$$B(e) = (b_{e,0}\ b_{e,1}\ b_{e,2}\ \ldots\ b_{e,125}) \text{ donde "e" va desde 0 hasta v-1}$$

A la salida la expresión de los bits queda:

$$A(e) = (a_{e,0}\ a_{e,1}\ a_{e,2}\ \ldots\ a_{e,125})$$

Vector que es definido por:

$$a_{e,w} = b_{e,H_e(w)} \text{ , donde "w" va desde 0 hasta 125}$$

$H_e(w)$ es una función de permutación diferente para cada entrelazador:

I0: $H_0(w) = w$

I1: $H_1(w) = (w + 63) \bmod 126$

I2: $H_2(w) = (w + 105) \bmod 126$

I3: $H_3(w) = (w + 42) \bmod 126$

I4: $H_4(w) = (w + 21) \bmod 126$

$$\textbf{\textit{I5:}}\ H_5(w) = (w + 84) \bmod 126$$

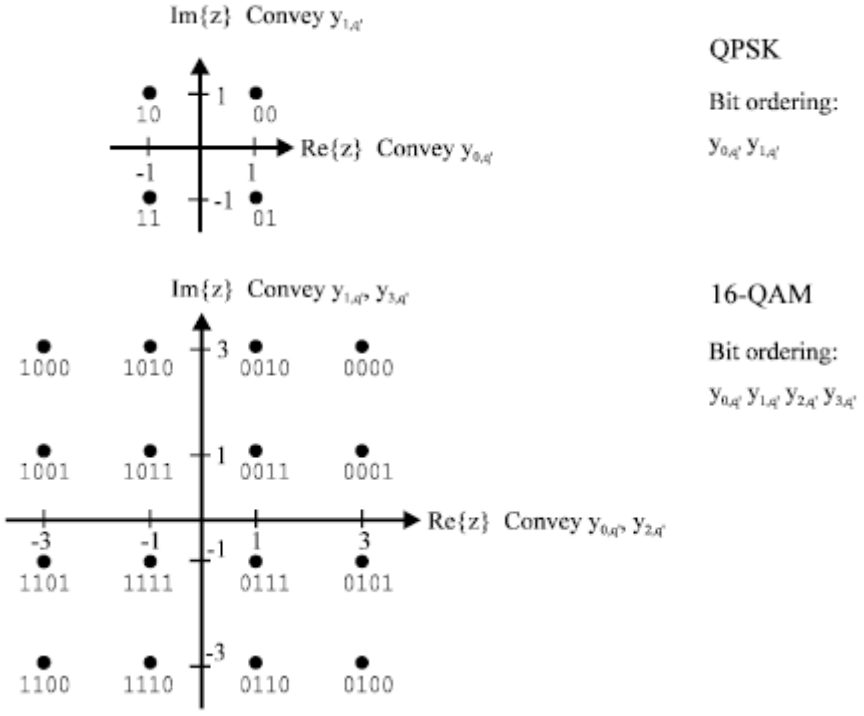
Las salidas de los "v" entrelazadores se agrupan para formar símbolos, de tal manera que cada símbolo de "v" bits consiste en un único bit de la salida de cada entrelazador, correspondiendo el bit más significativo a la salida de ***I0***, es decir:

$$y'_w = (a_{0,w}, a_{1,w}, ..., a_{v-1,w})$$

En el apartado 4.3.3.5: Retos de programación, se puede analizar el algoritmo seguido en la programación del entrelazador relativo al bit.

- **Mapeo de los símbolos:** En el estándar de la capa física del sistema DVB-T, se propone realizar el entrelazado de símbolos previamente al mapeo. No obstante, el simulador desarrollado realiza el mapeo antes que el entrelazado para simplificar el procesado de señal, manteniendo exactamente el mismo funcionamiento que se especifica en el estándar.

  Cada símbolo y'_w a la salida del entrelazador de bits consta de "v" bits. Estos "v" bits de modulan con un esquema de modulación correspondiente al valor de "v". Las modulaciones empleadas son QPSK, 16QAM, 64QAM, 16QAM no uniforme y 64QAM no uniforme. Las constelaciones, y los detalles del mapeo de Gray aplicado pueden verse en la Figura 37:



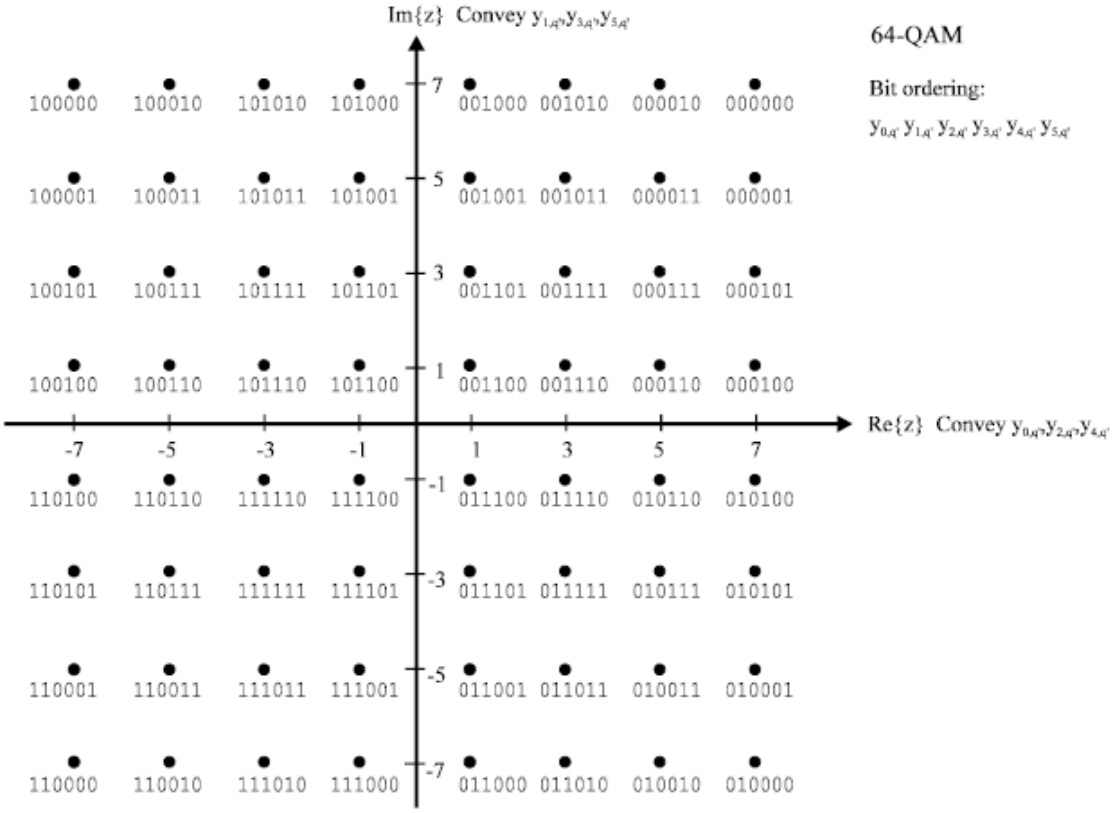


Figura 37 Constelaciones QPSK, 16QAM y 64QAM y los correspondientes patrones binarios. Figura extraída de [1].

Las proporciones exactas de las constelaciones, dependen del parámetro α descrito en 4.3.3.1, que puede tomar 3 valores distintos: 1, 2 ó 4. Por lo tanto la Figura 37, que corresponde a α=1, debería modificar sus constelaciones para los casos 16QAM y 64QAM si el valor de α fuera 2 ó 3. En el estándar [1] se muestran las constelaciones para todos los valores de α.

Los símbolos obtenidos tras el mapeo de los bits seguirán la siguiente ecuación: $s_i = I_i + jQ_i$, donde I_i y Q_i son la componente en fase y en cuadratura respectivamente de cada símbolo.

- **Entrelazador de símbolos:** El entrelazado de símbolos se realiza sobre los símbolos mapeados, de forma que a la salida del entrelazador queden agrupados en bloques cuyo tamaño está calculado para que los datos se puedan distribuir directamente entre las 1.512 (modo 2k) portadoras activas que tiene cada símbolo OFDM.

  Se puede representar cada bloque de 1512 símbolos a la salida del mapeo por:

$$Y' = (y'_0\ y'_1\ y'_2\ ...y'_{1511})$$

  Estos vectores Y′ a la entrada del entrelazador de símbolos se convierten en los vectores entrelazados Y a su salida:

$$Y = (y_0\ y_1\ y_2\ ...\ y_{Nmax-1}) \text{ siendo } N_{max}=1512 \text{ (modo 2k)}$$

Donde:

$y_q = y'_{H(q)}$ desde q = 0 hasta q = Nmax-1 para símbolos OFDM impares
$y_{H(q)} = y'_q$ desde q = 0 hasta q = Nmax-1 para símbolos OFDM pares

$H(q)$ es una función de permutación que se puede generar mediante el algoritmo cuyo diagrama de bloques esquemático se muestra en la Figura 38 para el modo 2k.

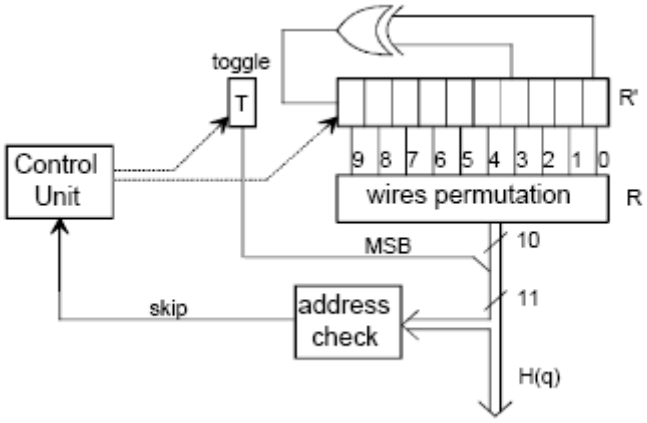


Figura 38 Algoritmo generador de la función de permutación H(q) para modo 2k. Figura extraída de [1].

Por último, puesto que en la modulación OFDM las portadoras que contienen datos se transmiten con nivel de potencia normalizado (potencia de las portadoras de datos igual a 1), se realiza la normalización para enviar a la siguiente etapa del simulador (etapa OFDM) los símbolos normalizados $s_{MAP}[n]$.

Modulation scheme		Normalization factor
QPSK		c = z/√2
16-QAM	α = 1	c = z/√10
	α = 2	c = z/√20
	α = 4	c = z/√52
64-QAM	α = 1	c = z/√42
	α = 2	c = z/√60
	α = 4	c = z/√108

Figura 39 Factores de normalización para las portadoras de datos. Figura extraída de [1].

4.3.3.3 Resultados

En la presente etapa se obtienen diversos resultados gráficos y dos resultados numéricos.

- ***Resultados numéricos***

Los resultados numéricos facilitados en esta etapa son de tipo teórico:

- **Polinomio generador X:** Es el polinomio generador de la salida X del codificador convolucional, 171_{OCT} .

- **Polinomio generador Y:** Es el polinomio generador de la salida Y del codificador convolucional, 133_{OCT} .

➢ ***Resultados gráficos***

Los resultados gráficos facilitados son diferentes en las versiones MATLAB y Java.

Por una parte se presentan, en un primer grupo de señales, los siguientes resultados de la implementación:

- **Secuencia de entrada al codificador convolucional:** Se representa un tramo de 100 bits correspondientes al vector de bits $b_{rs+il}[n]$ de entrada a la presente etapa.

- **Secuencia de salida X del codificador convolucional:** Se representa un tramo de la salida X del codificador convolucional, generada por el polinomio 171_{OCT} , que se corresponde con el tramo de bits representados en la secuencia de entrada al codificador. En el caso de que se seleccione una tasa de codificación superior a ½, el usuario podrá observar en dicha salidas los bits que han sido eliminados por el puncturing asociado al codificador.

- **Secuencia de salida Y del codificador convolucional:** Se representa un tramo de la salida Y del codificador convolucional, generada por el polinomio 133_{OCT} , que se corresponde con el tramo de bits representados en la secuencia de entrada al codificador. En el caso de que se seleccione una tasa de codificación superior a ½, el usuario podrá observar en dicha salidas los bits que han sido eliminados por el puncturing asociado al codificador.

- **Secuencia de salida total del codificador convolucional:** Se representa un tramo de la salida del codificador convolucional tras aplicarse el puncturing, que se corresponde con el tramo de bits representados en la secuencia de entrada al codificador. Sólo disponible en la versión Java.

- **Secuencia de salida del entrelazador relativo al bit:** Se representa un tramo de la salida del entrelazador de bits, que se corresponde con el tramo de bits representados en la secuencia de salida del codificador. Sólo disponible en la versión Java.

- **Símbolos en fase de la señal mapeada:** Se representa un tramo del vector de símbolos en fase de la señal mapeada, que se corresponde con el tramo de bits representados en la secuencia de salida del entrelazador de bits. Sólo disponible en la versión Java.

- **Símbolos en cuadratura de la señal mapeada:** Se representa un tramo del vector de símbolos en cuadratura de la señal mapeada, que se corresponde con el tramo de bits representados en la secuencia de salida del entrelazador de bits. Sólo disponible en la versión Java.

- **Símbolos en fase tras entrelazador de símbolos:** Se representa un tramo del vector de símbolos en fase tras el entrelazado, que se corresponde con la secuencia de símbolos en fase de la señal mapeada. Sólo disponible en la versión Java.

- **Símbolos en cuadratura tras entrelazador de símbolos:** Se representa un tramo del vector de símbolos en cuadratura tras el entrelazado, que se corresponde con la secuencia de símbolos en cuadratura de la señal mapeada. Sólo disponible en la versión Java.

Por otra parte, se presentan los resultados gráficos teóricos siguientes:

- **Representaciones para los interleavers:** Para cada entrelazador empleado en la etapa, se muestra una representación gráfica:

  - **Permutaciones de interleaver de bit:** Muestra la función de permutación $H_e(w)$, que es diferente para cada entrelazador. El subíndice "e" va de 0 a 1 en el caso QPSK, de 0 a 3 en el 16QAM y de 0 a 5 en 64QAM.

  - **Función de permutación de símbolos H(q):** Se trata de la función de permutación empleada para el entrelazado de símbolos, y cuyo algoritmo generador para el modo 2k puede verse en la Figura 38.

- **Constelación de la señal:** Muestra la constelación utilizada para realizar el mapeo de los bits.

➢ ***Intercambio de ficheros***

- **Guardar:** Se permite guardar en un fichero los símbolos $s_{MAP}[n]$ normalizados a la salida de la etapa.

- **Leer:** Permite leer de un fichero los símbolos normalizados $s_{MAP}[n]$ para intercambiarlos por la correspondiente secuencia generada. Esta secuencia de símbolos debe tener la misma longitud que la generada por el simulador.

4.3.3.4 Simulación 16 paquetes transporte MPEG-2

Se configura la etapa para emplear un codificador convolucional con tasa de codificación $2/3$, una constelación de señal 16QAM con $\alpha = 1$(uniforme) y una implementación completa del procesado de señal de la etapa, activando el conjunto completo de entrelazado interno.

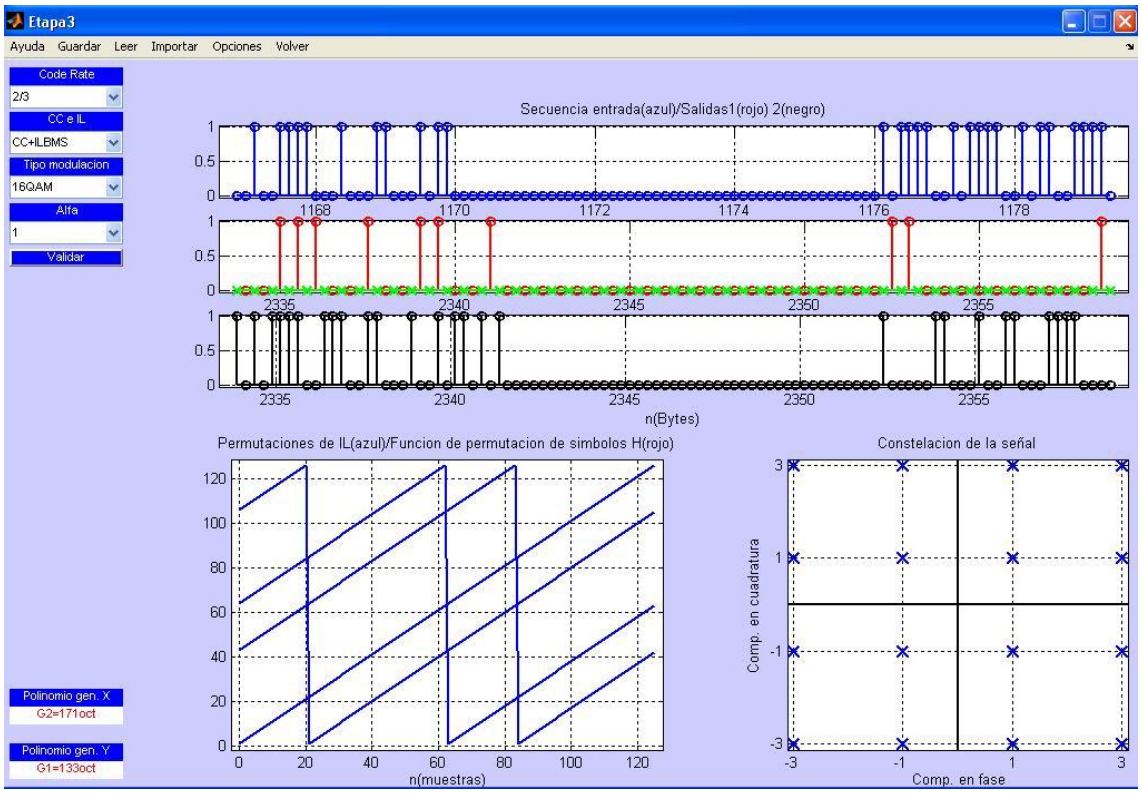


Figura 40 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 3: Codificador interno (convolucional) + Entrelazado interno + Mapping de símbolos

En la Figura 40 se puede observar, en el grupo de gráficas superior, el puncturing realizado sobre la secuencia de entrada (cruces verdes). A partir de estas representaciones, se deduce que para la tasa de codificación $2/3$, el polinomio de salida Y (color negro) no realiza puncturing sobre su salida original (correspondiente a la tasa $1/2$).

En las representaciones teóricas, por un lado se estudian las cuatro diferentes funciones de permutación $H_e(w)$, implementadas por el entrelazador de bits. Puede observarse que, al seleccionarse 16QAM, se dispone de cuatro entrelazadores de bits (*I0* a *I3*). Por otro lado se estudia la constelación de la señal empleada para realizar el mapeo de símbolos.

4.3.3.5 Retos de programación

Entre los retos de programación que los cálculos de procesado de señal han planteado en esta etapa, pueden destacarse el codificador convolucional, el entrelazado interno y el mapeo de la señal.

4.3.3.5.1 Retos en la programación MATLAB

- ***Puncturing***

En la realización del *puncturing*, que complementa al codificador convolucional, se siguen los siguientes pasos (cogiendo un R_c de $3/4$ por ejemplo):

- ✓ Hacer que el vector de bits *v*, de salida del codificador convolucional, sea múltiplo de 6, añadiendo ceros en caso necesario.

- ✓ Ordenar, con la función *reshape* de MATLAB, el vector de bits *v* en una matriz *A* de 6 filas. Cabe mencionar que esta función *reshape* va colocando los bits por columnas, tal como interesa.

- ✓ Seleccionar las filas que se mantienen tras el *puncturing*, en el ejemplo que nos ocupa de *code rate* $3/4$:

A1=A([1,2,4,5],:);

- ✓ Por último, leer por columnas la matriz *A1* y obtener el resultado del *puncturing* $3/4$. Para el resto de casos se realiza de manera análoga.

➢ ***Entrelazado interno***

El *"entrelazado interno"* se compone de un primer proceso de entrelazado relativo al bit que va seguido de un entrelazado de símbolos. Ambos, el relativo al bit y el entrelazado de símbolos están basados en bloques. En el apartado 4.3.3.2 se han explicado ambos entrelazadores. A continuación, se definen los algoritmos seguidos para su programación:

- ***Interleaving de bit***

- ✓ Definir el número de subtramas en el que se demultiplexa la trama de bits. Dicho número será 2, 4 ó 6 dependiendo si se ha escogido modulación QPSK, 16QAM ó 64 QAM respectivamente. Será el número de filas de la matriz *bits_Ilb*.

- ✓ Definir el vector *Index_suma*, que contendrá los índices empleados posteriormente para realizar las permutaciones definidas por el estándar DVB-T.

Index_suma=[0,63,21,84,105,42]

Se puede observar que los índices se han definido con los índices de la función $H_e(w)$, definida en el estándar, realizando el opuesto en módulo 126 de cada uno de ellos.

- ✓ Para cada fila de la matriz *bits_Ilb*, con la función *reshape* de MATLAB, crear una matriz *A* de 126 filas. A continuación se aplica la subrutina de MATLAB *circshift*, que desplaza los valores de una matriz de forma circular, del siguiente modo:

A=circshift(A,Index_suma(i))

Donde *i*, es la fila de *bits_Ilb* correspondiente. Por último, se guarda en la fila *i* de la matriz *bits_Ilb* el contenido de *A*, leyendo la matriz por columnas.

- ✓ El resultado, guardado en *bits_Ilb* al finalizar las anteriores operaciones para cada fila, son los bits entrelazados.

- ***Interleaving de símbolo***

- ✓ Primero, crear la función de permutación definida en el estándar [1] *H(q)*, siguiendo los pasos indicados por el mismo.

- ✓ No es necesario realizar las agrupaciones de los bits, por el hecho de realizar el entrelazado de símbolos después del mapeo. Así que en este punto ya se dispone de símbolos.

- ✓ Recorrer los símbolos preparados la modulación OFDM (1512 portadoras por símbolo), y para los símbolos impares en MATLAB (pares en el estándar, porque MATLAB comienza sus vectores en 1 en lugar de 0) hacer el cálculo $y_{H(q)} = y'_q$. Para los símbolos OFDM pares en MATLAB (impares en el estándar) calcular $y_q = y'_{H(q)}$.

  En la realización del entrelazador de símbolos, se ha prestado especial atención a que, al hacer referencia a *"* $y_{H(q)} = y'_q$ para símbolos pares o impares de *q=0…1512"* en el estándar [1], se hace referencia a la paridad del número de símbolo OFDM que llega al entrelazador, y no a si la posición *q* del vector *y* o *y'* es par o impar, como puede interpretarse por error.

- ***Mapeo de la señal***

Para desarrollar el mapeo de la señal llevado a cabo en el sistema DVB-T, se ha seguido una programación pensada para aprovechar la potencia de MATLAB en el cálculo vectorial, y de esta manera conseguir un código eficiente.

Los pasos seguidos son los siguientes:

- ✓ Definir el número de subtramas *N_bits* en el que se demultiplexa la trama de bits. *N_bits* será 2, 4 ó 6 dependiendo si se ha escogido modulación QPSK, 16QAM ó 64 QAM respectivamente. Es el número de filas de la matriz *E3_simbolos* que contiene los bits a mapear.

- ✓ Crear las matrices *simbolos_R* y *simbolos_I,* que contienen las filas impares y pares respectivamente de *E3_simbolos.* Cada columna de estas matrices se utilizará para obtener la parte en fase y cuadratura de la señal modulada respectivamente.

- ✓ Definir las tablas de las modulaciones (QPSK, 16QAM y 64QAM) observando las constelaciones y las pautas de distribución de bits del estándar DVB-T, de manera que la traducción a decimal, tanto de los bits en fase (*simbolos_R)* como

en cuadratura (*simbolos_I),* coincida con la posición del vector que contiene la tabla de la modulación correspondiente.

- ✓ Convertir los bits de las columnas de *simbolos_R* y *simbolos_I* a decimal, obteniendo como resultado dos vectores, *simbolos_R_d* y *simbolos_I_d.*

- ✓ Por último, obtener los símbolos mapeados con la siguiente instrucción:

  *E3_simb_map=tabla_X(simbolos_R_d+1)+i*tabla_X(simbolos_I_d+1)*

  Donde X puede ser QPSK, 16QAM o 64QAM. Se puede ver como, accediendo a la posición de la tabla correspondiente a la traducción a decimal de los bits, se obtiene el símbolo en cuestión.

4.3.3.5.2 Retos en la programación Java

- ➢ ***Codificador convolucional***

El procesado de señal referente al codificador convolucional así como al algoritmo de Viterbi queda enmarcado dentro del trabajo titulado *Algoritmo de Viterbi Implementado en Java* [28]. Dicha labor ha sido desarrollada por Pedro Espinosa Fricke para el departamento de Teoría de la Señal y Comunicaciones de la Universidad Politécnica de Cataluña. El trabajo realizado consiste en la implementación de las etapas codificadoras y decodificadoras especificadas tanto en el estándar 802.11a como en el estándar DVB-T con la finalidad de ser integradas adecuadamente en el sistema OFDM del proyecto LaViCAD y en el presente proyecto.

De forma simplificada, en el método que realiza la codificación convolucional de la etapa, se ejecutan los siguientes pasos:

- ✓ Generación de los polinomios que describen el codificador.

- ✓ Generación de la trama a codificar a partir de los bits.

- ✓ Llamada al codificador convolucional.

- ✓ Llamada al proceso de puncturing en caso necesario.

- ✓ Adecuar la variable retornada al formato de salida deseado.

Se recomienda consultar la referencia [28] con el fin de profundizar en el procesado interno del codificador.

4.3.3.6 Análisis de la etapa

En la presente etapa cabe destacar que, en apoyo a la implementación de la misma, se ha usado el Anexo C del estándar *ETSI EN 300 744*, gracias al cual se comprueba el correcto funcionamiento del conjunto completo de entrelazado interno y el mapeo.

Por otra parte, como puede observarse en el grupo superior de gráficas de la Figura 40, se ven reflejados los efectos que se anunciaban en el apartado 4.3.2.6. Se aprecia una gran cantidad de ceros a la entrada y salida del codificador convolucional, por el hecho de haber realizado el entrelazado externo en la etapa 2, el cual agrega una gran cantidad de bits 0 a la señal transmitida, la longitud de la cual es pequeña en relación a una señal transmitida en una implementación real.

4.3.4 Etapa 4: OFDM

En la presente etapa se prepara la señal para su transmisión, estructurando en tramas OFDM la señal de entrada normalizada $s_{MAP}[n]$.

El procesado que se realiza en la etapa es el propio de la modulación OFDM, presentada en la sección 4.1 junto a sus ventajas. Para cada grupo de 1512 portadoras normalizadas (modo 2k) contenido en la señal de entrada $s_{MAP}[n]$, se insertan las portadoras piloto dispersas, las portadoras piloto continuas y las portadoras TPS. A continuación, se realiza la transformación al dominio temporal mediante el algoritmo IFFT. Por último, se añaden adecuadamente los intervalos de guarda. La etapa se esquematiza de la siguiente manera:

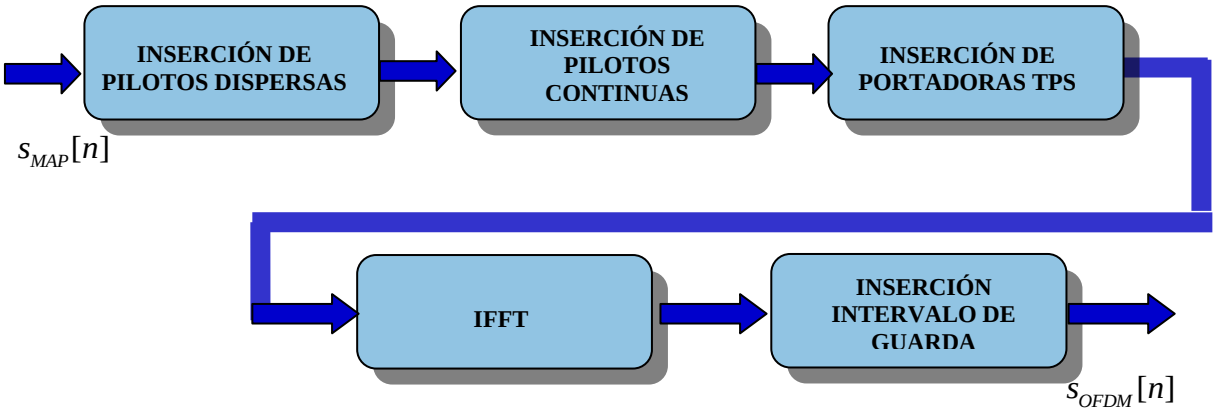


Figura 41 Esquema de la etapa 4 del sistema DVB-T

A la salida de la etapa se tiene el vector de símbolos OFDM transmitidos $s_{OFDM}[n]$, que se envía a la siguiente etapa del simulador.

4.3.4.1 Parámetros de entrada

Esta etapa cuenta con dos parámetros de entrada, configurables por el usuario:

- **Ancho de banda del canal:** selección del ancho de banda del canal radioeléctrico empleado para la transmisión de la señal, entre los tres valores que contempla el estándar *ETSI EN 300 744* [1] para transmisiones terrestres: 6, 7 y 8 MHz.

- **Relación intervalo de guarda-tiempo útil de símbolo:** permite elegir, de entre cinco opciones, la duración del intervalo de guarda añadido al símbolo útil OFDM. Los cinco valores posibles son: 0, 1/4, 1/8, 1/16 y 1/32. De estos valores dependerá la duración total del símbolo OFDM transmitido.

  Cabe comentar que el valor 0 no está especificado por el estándar, pero se ha añadido para simular una transmisión sin intervalo de guarda.

4.3.4.2 Procesado de la señal

A continuación se describen los cálculos de procesado de señal que se implementan en esta etapa:

- ***Procesado propio de la etapa***

A la presente etapa llega el vector $s_{MAP}[n]$, que contiene grupos de 1512 portadoras de datos útiles moduladas (modo 2k).

Sin embargo, un símbolo OFDM está compuesto por conjuntos de 1705 portadoras. Esto significa que la señal transmitida incluye, además de las portadoras para datos, otras portadoras (pilotos dispersas, pilotos continuas y TPS) con diferente utilidad.

La incorporación de estas portadoras en número y distribución adecuados exige organizar la señal transmitida en *tramas*. Cada trama consiste en 68 símbolos OFDM, que se numeran de 0 a 67. Una *super-trama* está formada por 4 *tramas*.

El número de portadoras piloto dispersas por símbolo OFDM es 131, 45 el número de portadoras piloto continuas y 17 el de portadoras TPS; en total 1705 portadoras por símbolo OFDM.

Las portadoras que contienen datos se transmiten con nivel de potencia normalizado ($s_{MAP}[n]$ está normalizado), de manera que cumplen: $E[c \cdot c^*] = 1$, correspondiendo "c" a los distintos puntos $c_{m,\ell,k}$ de la constelación de la portadora "k" (0 a 1704 en modo 2k) del símbolo número "ℓ" (0 a 67) en la trama número "m".

En cambio, las portadoras piloto continuas y piloto dispersas se transmiten con nivel de potencia reforzado (*boosted*), cumpliendo: $E[c \cdot c^*] = 16/9$.

Finalmente, las portadoras TPS se transmiten con el nivel de potencia normalizado, es decir, cumpliendo: $E[c \cdot c^*] = 1$.

Las portadoras piloto dispersas y las portadoras piloto continuas se modulan con la denominada "información de referencia". Cada portadora piloto continua coincide con una portadora piloto dispersa cada 4 símbolos, y la información transmitida por ambos tipos de portadoras se deriva de una secuencia binaria pseudoaleatoria ("*Pseudo Random Binary Sequence" PRBS*) que se genera de acuerdo con el esquema mostrado en la Figura 42:

El PRBS se inicializa de forma que el primer bit de salida coincide con la primera portadora activa. Con cada portadora, ya sea piloto o no, se genera un nuevo valor.

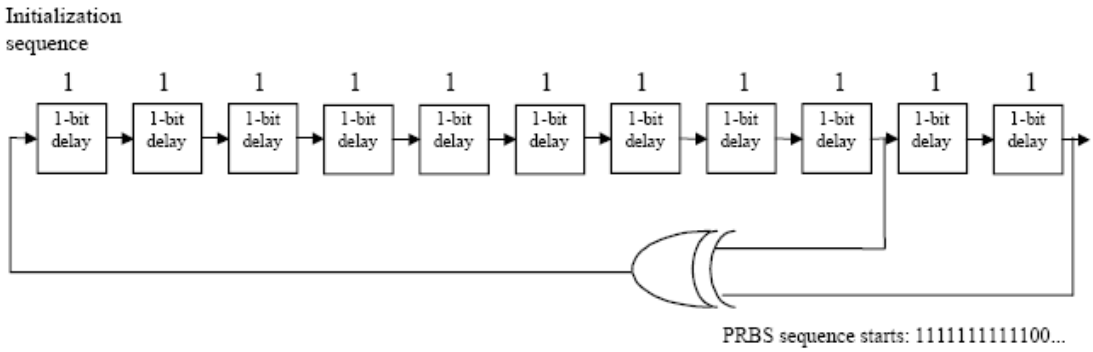


Figura 42 Generador PRBS para las portadoras piloto. Figura extraída de [1].

El polinomio generador de la secuencia PRBS es:

$$x^{11} + x^2 + 1 \tag{4.8}$$

- **Inserción de las portadoras piloto dispersas:** Se utilizan para la sincronización del receptor en frecuencia y fase.

  Para la inserción de las portadoras piloto dispersas, primero se modula la información de referencia, cogida de la secuencia PRBS, y posteriormente se inserta en las posiciones correspondientes:

  - **Modulación:** El resultado del generador PRBS, denominado w_k, aplicable a la portadora número *k*, modula las portadoras dispersas con el siguiente esquema:

$$\begin{aligned} \mathrm{Re}\{c_{m,l,k}\} &= 4/3 \times 2(\frac{1}{2} - w_k) \\ \mathrm{Im}\{c_{m,l,k}\} &= 0 \end{aligned} \tag{4.9}$$

    Llamando *m* al número de la trama, ℓ al número del símbolo dentro de cada trama ($0 \leq \ell \leq 67$) y *k* al índice de la portadora dentro de cada símbolo OFDM (*0 ≤ k ≤1.704*, modo 2k*).* Corresponde a un nivel de potencia *boosted*, como ya se ha indicado.

  - **Localización:** Para el símbolo número ℓ de una determinada trama, los índices *k* que corresponden al subconjunto de portadoras piloto dispersas, vienen dados por:

$$k = 3x(\ell \bmod 4) + 12\,\mathrm{p} \tag{4.10}$$

    Siendo *p* un número entero $p \geq 0$ que debe cumplir la condición de que el valor resultante para *k* no exceda el valor 1705.

- **Inserción de las portadoras piloto continuas:** Se utilizan para la regeneración del canal en amplitud y fase en el receptor.

Para la inserción de las portadoras piloto continuas, se realiza el mismo proceso que para las portadoras piloto dispersas:

- **Modulación:** Emplean la misma modulación que las portadoras piloto dispersas, a partir de la secuencia de referencia, como puede verse en la ecuación (4.9).

- **Localización**: A diferencia de lo que ocurre con las portadoras piloto dispersas, las posiciones que ocupan dichas portadoras piloto continuas en cada símbolo OFDM son siempre las siguientes:

0	48	54	87	141	156	192	201	255
279	282	333	432	450	483	525	531	618
636	714	759	765	780	804	873	888	918
939	942	969	984	1050	1101	1107	1110	1137
1140	1146	1206	1269	1323	1377	1491	1683	1704

Tabla 3 Posiciones de las portadoras piloto continuas (índice *k*)

- **Inserción de las portadoras TPS:** Las portadoras TPS sirven para señalizar los parámetros correspondientes al modo de transmisión empleado, es decir, para informar de todo lo relativo a la codificación de canal y a la modulación usadas en la transmisión.

  Estas portadoras se transmiten en paralelo, ocupando 17 posiciones de cada símbolo OFDM (modo 2k). La tabla siguiente recoge los índices de las posiciones que ocupan las portadoras TPS en cada símbolo para el modo 2k simulado en este proyecto:

34	50	209	346	413	569	595	688	790
901	1073	1219	1262	1286	1469	1594	1687	

Tabla 4 Posiciones de las portadoras TPS (índice *k*)

Por el hecho de, como se ha explicado en el apartado 4.3.1.6, transmitir solamente 8, 16 ó 24 paquetes MPEG-2 TS, el número de símbolos OFDM transmitidos no supera los 38 (en el mejor de los casos). Por tanto no se ha podido simular la transmisión TPS propuesta por el estándar, que necesita los 68 símbolos OFDM componentes de una trama, y en la simulación nunca se llega a enviar una trama completa.

Según especifica el estándar [1], todas las portadoras TPS de un mismo símbolo OFDM transportan el mismo bit de información, y se agrupan en bloques de 68 símbolos consecutivos coincidentes con una trama. Así pues, cada bloque de portadoras TPS contiene 68 bits, los cuales se distribuyen de la forma siguiente:

- 1 bit de inicialización.
- 16 bits de sincronización.
- 37 bits de información.
- 14 bits redundantes para protección contra errores.

Sin embargo, en el simulador se asigna un máximo de 28 bits con información TPS, rellenando de ceros en caso de transmitir más de 28 símbolos OFDM. Se hace uso de la siguiente tabla:

Número de bit	**Valor (Formato)**	**Objetivo**
s_0	Según secuencia prbs	Bit de inicialización
$s_1 - s_2$	QPSK (0 0) 16QAM (0 1) 64QAM (1 0)	Constelación
$s_3 - s_5$	1 (0 0 1) 2 (0 1 0) 4 (0 1 1)	Parámetro alfa
$s_6 - s_8$	1/2 (0 0 0) 2/3 (0 0 1) 3/4 (0 1 0) 5/6 (0 1 1) 7/8 (1 0 0)	Tasa de codificación interna
$s_9 - s_{11}$	0 (1 0 0) 1/32 (0 0 0) 1/16 (0 0 1) 1/8 (0 1 0) 1/4 (0 1 1)	Intervalo de guarda
$s_{12} - s_{27}$	0011010111101110	Palabra de sincronización

Tabla 5 Asignación de portadoras TPS por el simulador implementado

El formato de los bits se hace de acuerdo a las especificaciones del estándar [1] para cada caso. Excepto en el caso del intervalo de guarda, donde se añade 1 bit para poder informar de la transmisión sin dicho intervalo, no contemplada por el estándar.

Cada portadora TPS está modulada con un esquema tipo BPSK diferencial (DBPSK). Esta modulación se inicializa al principio de cada bloque TPS, es decir, al principio de cada trama, para lo cual, las portadoras TPS pertenecientes al primer símbolo OFDM ($\ell=0$) de una trama llevan el bit *s0* de inicialización.

La modulación de este bit s_0 de inicialización, se deriva de la secuencia de referencia w_k obtenida del generador PRBS (Figura 42) usado para las portadoras piloto, y es como sigue:

$$\begin{aligned} \mathrm{Re}\{c_{m,l,k}\} &= 2\left(\frac{1}{2} - \mathrm{w_k}\right) \\ \mathrm{Im}\{c_{m,l,k}\} &= 0 \end{aligned} \qquad (4.11)$$

Para la modulación diferencial del resto de portadoras TPS, para la portadora de índice *k* del símbolo ℓ ($\ell>0$) en la trama *m*:

$$\begin{aligned} &\text{Si el bit } s_l\text{=0: } \mathrm{Re}\left\{c_{m,\ell,k}\right\} = \mathrm{Re}\left\{c_{m,\ell-1,k}\right\}; \mathrm{Im}\left\{c_{m,\ell,k}\right\} = 0 \\ &\text{Si el bit } s_l\text{=1: } \mathrm{Re}\left\{c_{m,\ell,k}\right\} = -\mathrm{Re}\left\{c_{m,\ell-1,k}\right\}; \mathrm{Im}\left\{c_{m,\ell,k}\right\} = 0 \end{aligned} \quad (4.12)$$

Como ya se había comentado, las portadoras TPS se transmiten con nivel de potencia normalizado, es decir, $E[c \cdot c^*] = 1$.

- **Transformada de Fourier inversa (IFFT):** El algoritmo utilizado para realizar la transformada inversa de Fourier es el de la IFFT. El objetivo es convertir al dominio temporal las subportadoras construidas.

  Cada símbolo OFDM contiene 1705 portadoras (modo 2k), y la IFFT se realiza sobre un número de elementos que ha de ser potencia de 2, así que se realiza sobre 2048 elementos. Este número de elementos, debe ser bastante mayor que el número de portadoras a transmitir, para asegurar una transmisión fiable. Para lograr este fin, deben añadirse ceros en las bandas del símbolo a transmitir. En el caso de nuestra implementación, se añaden 172 ceros a la izquierda de cada símbolo, y 171 a la derecha, obteniendo las 2048 portadoras.

  El último paso consiste en colocar la componente central del símbolo OFDM en la frecuencia cero, así se consigue la transmisión en banda base objetivo de este simulador. En este momento ya se puede aplicar el algoritmo de la IFFT:

$$s_k = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X_n e^{j\frac{2\pi n}{N}k} \quad (4.13)$$

  Con esto, se obtiene la señal a transmitir.

- **Inserción del intervalo de guarda:** Tras la IFFT, cada símbolo OFDM se compondrá de 2048 muestras en el dominio temporal. La inserción del intervalo de guarda consiste en repetir delante de cada uno de los símbolos OFDM las últimas M muestras tal y como se ha explicado en la sección 4.1.

  El símbolo OFDM preparado para la transmisión quedará compuesto por 2560, 2304, 2176, 2112 ó 2048 muestras para los valores 1/4, 1/8, 1/16, 1/32 ó 0 del parámetro de entrada "Relación intervalo de guarda-tiempo útil del símbolo" respectivamente.

  Con todo, la señal transmitida será $s_{OFDM}[n]$.

➢ ***Procesado para la representación de resultados asociados a la etapa***

- **Cálculo del tiempo de muestreo:** Para la representación de los resultados de la presente etapa tanto en el dominio temporal como en el frecuencial, se ha calculado el tiempo de muestreo.

La frecuencia de muestreo es el resultado de dividir las 2048 muestras del símbolo útil, entre la duración del mismo.

$$f_m = \frac{2048}{T_U} \tag{4.14}$$

Según especifica el estándar, para canales de 8 MHz $T_U = 224\mu s$, de lo cual se deduce $f_m = \frac{2048}{224\mu s} = 9{,}1429 MHz$. Tm es el inverso, $T_m = \frac{1}{f_m} = 109{,}37 ns$. Esto coincide con el período elemental para 8 MHz definido por el estándar de $7/64\mu s$.

Una vez se obtiene el tiempo de muestreo para canales de 8 MHz, se extrapola para canales de 6 ó 7 MHz.

- **Representación símbolo OFDM en el dominio frecuencial:** Para la representación del símbolo OFDM, antes de su paso por la IFFT, en el dominio frecuencial, se siguen los siguientes pasos:

  - ✓ Se utiliza una matriz que contiene en cada fila las 1705 portadoras correspondientes a cada símbolo OFDM antes de realizar la IFFT.
  - ✓ Se añaden ceros a ambos lados de cada símbolo, hasta completar 2048 portadoras.
  - ✓ Se calcula el módulo al cuadrado de cada símbolo OFDM de 2048 portadoras.
  - ✓ A continuación, se calcula la media de todos los elementos, obteniendo un único símbolo de 2048 muestras resultante.

  Utilizando la frecuencia de muestreo calculada, y el símbolo resultante de los cálculos expuestos, se obtiene el espectro teórico del símbolo OFDM.

- **Representación densidad espectral:** Para el cálculo de la densidad espectral simulada, se realiza el cálculo del periodograma, tal como se detalla en el apartado 4.3.4.5: Retos de programación.

- **Parrilla de pilotos:** Para la representación de la parrilla de pilotos dispersos y continuos, y TPS en la versión MATLAB del simulador, se ha simulado una transmisión de 68 símbolos OFDM (1 trama completa).

  Una vez se tiene la matriz de 68 filas y 1705 columnas rellena, se representa con la función *Images_children*, la cual se estudiará en el siguiente capítulo en el apartado 5.2.4.1.

- **Cálculo del peak to average ratio (PAR):** Para calcular el PAR de la señal *s(t)* se ha utilizado la siguiente fórmula:

$$PAR(s(t)) = \frac{\max\left|s(t)\right|^2}{E\left\{s(t)^2\right\}} \quad (4.15)$$

Donde *E{.}* es la esperanza. El resultado se expresa en *dB.*

4.3.4.3 Resultados

Para mostrar al usuario los resultados de la etapa, se cuenta con diversas gráficas distribuidas en dos grupos y cuatro resultados numéricos.

- ***Resultados numéricos***

En esta etapa se facilitan tanto resultados de tipo teórico como propios de la simulación en curso:

- **Peak to average ratio:** El primer resultado mostrado es calculado a raíz de la señal temporal OFDM transmitida. Muestra la relación entre el valor de pico y el valor medio de la señal, mediante la ecuación (4.15). Se expresa en *dB.*

- **Modo de transmisión:** Indica al usuario que la simulación implementa el modo de transmisión 2k.

- **Número de símbolos OFDM:** Se trata del número de símbolos OFDM transmitidos en la simulación. Este número estará entre 2 y 38 símbolos OFDM, dependiendo de los parámetros escogidos por el usuario.

- **Polinomio generador de la secuencia PRBS:** Se trata del polinomio (4.8) utilizado para generar la secuencia binaria pseudoaleatoria (PRBS) que modula las portadoras continuas y dispersas.

- ***Resultados gráficos***

Los resultados gráficos facilitados se dividen en tres grupos. En el primero de ellos se muestran las gráficas correspondientes a representaciones espectrales de la señal OFDM:

- **Densidad espectral en escala logarítmica:** Representación de la densidad espectral de potencia de la señal transmitida $s_{OFDM}[n]$ en *dB* $(10 \cdot \log_{10})$.

- **Densidad espectral en escala lineal:** Representación de la densidad espectral de potencia de la señal transmitida $s_{OFDM}[n]$ en escala lineal.

- **Densidad espectral lineal símbolo OFDM:** Representación de la densidad espectral de potencia del símbolo OFDM antes de su paso por la IFFT y de la inserción del intervalo de guarda, en escala lineal.

- **Densidad espectral lineal símbolo OFDM teórico y transmitido:** Solapa en una misma gráfica las dos densidades espectrales, en escala lineal, descritas anteriormente.

El segundo grupo de señales muestra la señal $s_{OFDM}[n]$ en el dominio temporal:

- **Componentes IQ símbolo OFDM + Marca intervalo de guarda:** Representación de la componente real e imaginaria de la señal temporal $s_{OFDM}[n]$ transmitida, y de la marca del intervalo de guarda.

  A raíz de esta gráfica, el usuario puede visualizar el resto de señales temporales:

  - **Componente I símbolo OFDM + Marca intervalo de guarda**
  - **Componente Q símbolo OFDM + Marca intervalo de guarda**
  - **Componente IQ símbolo OFDM**
  - **Componente I símbolo OFDM**
  - **Componente Q símbolo OFDM**

Por último, en la versión MATLAB del simulador, el programa ofrece la visualización de una trama OFDM (68 símbolos OFDM) que muestra la posición de las portadoras piloto dispersas y continuas, las portadoras TPS y los datos a lo largo del índice frecuencial *K* (*Kmin=0, kmax=1704)*.

- ***Intercambio de ficheros***

  - **Guardar:** Se permite guardar en un fichero la señal $s_{OFDM}[n]$ a la salida de la etapa.

  - **Leer:** Permite leer de un fichero la señal $s_{OFDM}[n]$ para intercambiarla por la correspondiente secuencia generada. Esta señal debe tener la misma longitud que la generada por el simulador.

4.3.4.4 Simulación 16 paquetes transporte MPEG-2

La simulación del ejemplo 16 paquetes MPEG-2 TS continúa en esta etapa realizando la inserción de un intervalo de guarda de 512 muestras, puesto que la "relación intervalo de guarda-tiempo útil de símbolo" escogida es de $1/4\,(T_G/T_U = 512/2048 = 1/4)$. Para los resultados gráficos, se escoge un ancho de banda de 8 MHZ para el canal de transmisión.

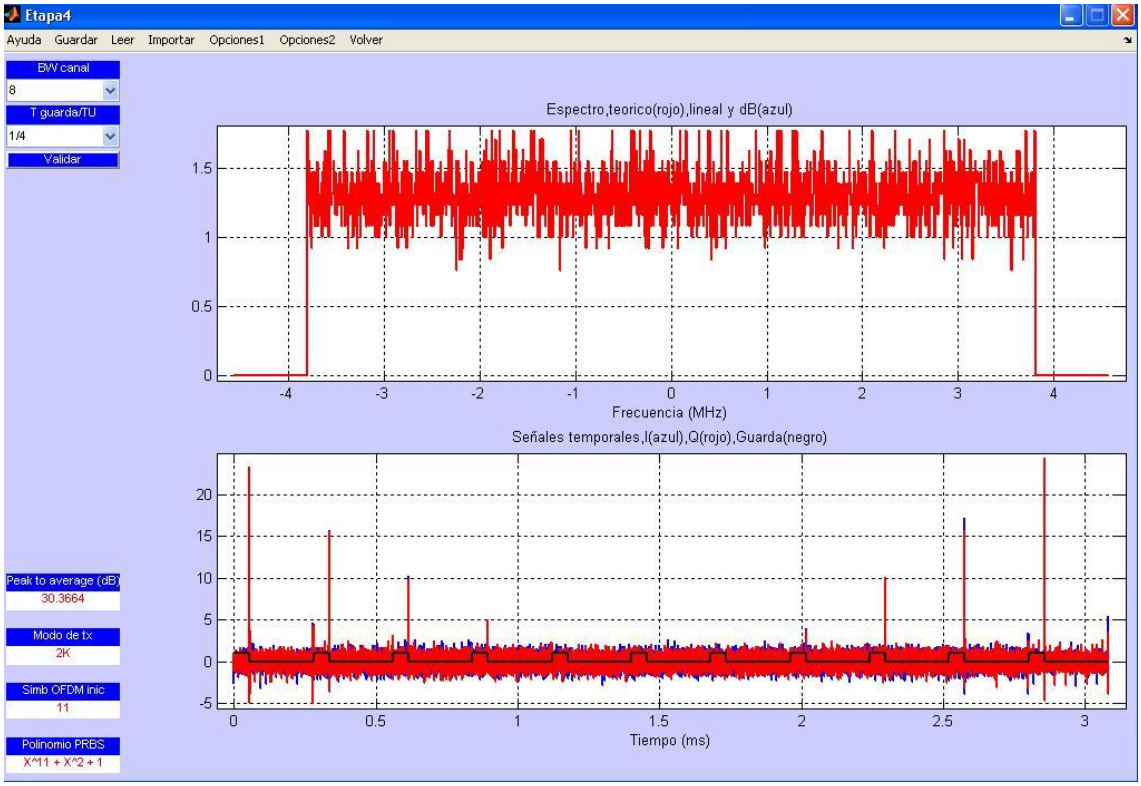


Figura 43 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 4: OFDM

Los resultados más significativos representados en la Figura 43 que pueden extraerse de esta etapa son:

- A partir del estudio de las funciones en el dominio frecuencial, puede comprobarse que la señal de salida está compuesta por las diferentes subportadoras descritas a lo largo del apartado. También puede comprobarse que el espectro de la señal transmitida coincide con el espectro teórico representado en el estándar del sistema DVB-T [1] para el modo de transmisión 2k y los parámetros escogidos en la etapa.

- A partir del estudio de las funciones temporales, pueden evidenciarse las características de la señal transmitida $s_{OFDM}[n]$ descritas en la Figura 43:

  - ✓ Presencia del prefijo cíclico. Visualizando las marcas de prefijo cíclico, puede comprobarse cómo las 512 muestras ($56\,\mu s$ de duración) que conforman el prefijo son idénticas a las últimas 512 del símbolo OFDM.

  - ✓ Duración de la señal transmitida en *ms*.

  - ✓ Peak to average ratio. Puede comprobarse el resultado numérico proporcionado a raíz de la representación gráfica de la señal temporal.

4.3.4.5 Retos de programación

Los retos de programación que se han resuelto a lo largo de esta etapa son:

4.3.4.5.1 Retos en la programación MATLAB

- ***Densidad espectral OFDM***

En esta etapa, se ha utilizado el siguiente algoritmo para la representación de la densidad espectral de la señal temporal OFDM, *s(t)*:

- ✓ Partición de la señal en bloques de *NFFTPC* muestras, que es igual a 2048 muestras más las muestras del prefijo cíclico.
- ✓ Cálculo del periodograma de cada uno de los segmentos.
- ✓ Promedio de los diferentes periodogramas obtenidos.

Se puede observar que el resultado final es el periodograma:

$$S_p(f) = \frac{1}{NFFTPC} \sum_{i=1}^{N} S^{(i)}(f) \tag{4.16}$$

Este mismo proceso se ha seguido para el cálculo del resto de densidades espectrales calculadas a partir de sus respectivas señales temporales.

4.3.4.5.2 Retos en la programación Java

De esta etapa no se destaca ningún reto de programación particular en la programación Java del simulador.

4.3.4.6 Análisis de la etapa

En la presente etapa se continúan observando los efectos comentados en el apartado 4.3.1.6, propios de una implementación práctica de la capa física del sistema DVB-T.

Al transmitir solamente 8, 16 ó 24 paquetes MPEG-2 TS, el número de símbolos OFDM transmitidos oscila entre 2 y 38. Por tanto no se ha podido simular la transmisión TPS propuesta por el estándar, que necesita los 68 símbolos OFDM componentes de una trama. Por este motivo se ha realizado una transmisión TPS adaptada a las necesidades de nuestro simulador.

4.3.5 Etapa 5: Canal

En la presente etapa se implementa la transmisión de la señal a través del modelo de canal. El esquema que se sigue es el siguiente:

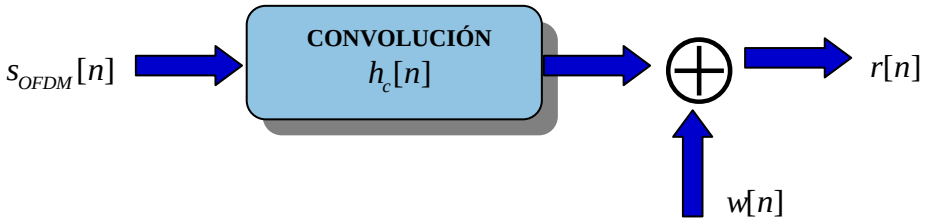


Figura 44 Esquema de la etapa 5 del sistema DVB-T

Para modelar el canal hay que tener en cuenta que el simulador permite la elección de canales Rayleigh. Por el hecho de utilizar distribuciones de Rayleigh aleatorias, el canal es variante y puede sufrir desvanecimientos. No obstante, en el simulador (y en muchos estándares *wireless*) se considera que el canal es invariante a lo largo de la transmisión de la ráfaga de *N* símbolos OFDM, puesto que se suponen desvanecimientos lentos (el cambio del canal se da en un tiempo mayor que la duración de la ráfaga de *N* símbolos). Por lo tanto, en este simulador el canal se modela como un sistema lineal e invariante.

Se trabaja con la respuesta impulsional del canal normalizada, de forma que la señal transmitida no aumente el nivel de potencia. El ruido generado es ruido aditivo gaussiano y blanco. La potencia del ruido se genera de acuerdo al cociente de energías: energía media transmitida por bit a densidad espectral de ruido, $\frac{E_b}{N_0}$.

A la salida de la etapa se obtiene $r[n]$, el vector de símbolos OFDM $s_{OFDM}[n]$ tras su paso por el canal y la adición de ruido, que se envía a la siguiente etapa del simulador.

4.3.5.1 Parámetros de entrada

La etapa posee dos grupos de parámetros, el primero reúne la configuración relativa a la generación del ruido, el segundo configura el tipo de canal a utilizar en esta etapa:

- **Generación de ruido:**
  - E_b/N_0 **:** Se solicita el cociente $10\log_{10}\left(\frac{E_b}{N_0}\right)$. Es el cociente entre la energía media transmitida por bit (E_b) y la densidad espectral de ruido (N_0). Se trata del valor en dB de este parámetro, y se permite el rango de valores entre -10 y 20 dB.
- **Tipo de canal:**

- Ideal: Mediante la opción ideal, como su nombre indica se genera una respuesta impulsional ideal, de atenuación igual a 0 dB y retardo nulo.

- Rayleigh: En el caso de seleccionar canal Rayleigh, se generará un canal considerado invariante durante la transmisión de los símbolos OFDM y selectivo en frecuencia. Se despliegan las siguientes opciones:

  - DS/T_U : Factor que determina el perfil exponencial que seguirá el canal Rayleigh. En el numerador se dispone del intervalo de tiempo en que la respuesta impulsional del canal es considerablemente mayor que cero (*Delay Spread*, DS). En el denominador está la duración útil de símbolo (T_U). Los valores permitidos son: 1/1024, 1/512, 1/256, 1/128, 1/64, 1/32, 1/16, 1/8 y 1/4.

  - Rice factor: Línea de visión directa, LOS (Line of Sight), también denominado factor de Rice: se trata del valor en dB que toma la primera muestra del perfil exponencial que adopta la respuesta impulsional del canal Rayleigh. Su valor está limitado entre –100 y 30 incluidos.

- Fichero: Respuesta impulsional arbitraria introducida mediante un archivo.

4.3.5.2 Procesado de la señal

A continuación se presenta el procesado de señal teórico que se implementa en la presente etapa.

- ***Procesado propio de la etapa***

  - **Tipos de canal:** además de un canal libre introducido por fichero, la etapa permite seleccionar dos tipos de canal definidos:

    - Canal ideal:

$$h_c[n] = \delta[n] \tag{4.17}$$

    - Canal Rayleigh: Permite seleccionar un canal Rayleigh, que se modela en el simulador como un canal lineal e invariante (durante la transmisión de los N símbolos OFDM) y selectivo en frecuencia. La respuesta impulsional del canal Rayleigh se modela a partir de la siguiente expresión en tiempo discreto utilizando muestras espaciadas el tiempo de muestreo:

$$h_c[n] = (k_{LOS} + c[0]) \cdot \delta[n] + \sum_{i=1}^{L} c[i] \cdot e^{-i/\sigma} \cdot \delta[n-i] \qquad (4.18)$$

La expresión se compone de:

- $e^{-i/\sigma}$: representa el perfil exponencial que sigue la respuesta impulsional, dicho perfil queda determinado por el *delay spread*, σ .

- $c[i]$: es una variable aleatoria gaussiana compleja, multiplicativa al perfil exponencial. Se genera con perfil de potencia 1 para todos sus coeficientes.

- K_{LOS} : es una constante denominada línea de visión directa o factor de Rice que simula una pérdida o ganancia de señal en el rayo principal existente entre el receptor y el emisor. Esta constante multiplicativa sólo afecta, por tanto, a la primera muestra de la respuesta impulsional.

- L: número de taps a generar hasta que la potencia del canal cae 20 dB. Se calcula a partir del *delay spread*, σ, el cual indica con cuantos taps la potencia del canal queda dividida por el número e, que equivale a 4,34 dB.

Nota: se está asumiendo que el retardo de un trayecto es un entero por el tiempo de muestreo. Si no es así existen teorías desarrolladas para sacar el canal equivalente.

Puede consultarse el apartado 4.3.5.5: Retos de programación, con el fin de profundizar en el algoritmo de programación utilizado para el cálculo de la respuesta impulsional de los canales Rayleigh.

- **Potencia de la señal:** La potencia de la señal s_{OFDM} a la entrada de esta etapa se calcula a partir de la esperanza de la señal, esto es:

$$P_S = E[{s_{OFDM}}^2[n]] \qquad (4.19)$$

- **Convolución de la señal con el canal:** A continuación se hace la convolución de la señal $s_{OFDM}[n]$ con el canal generado $h_c[n]$.

$$s_{h_c}[n] = s_{OFDM}[n] * h_c[n] \qquad (4.20)$$

- **Generación del ruido:** En el presente punto se genera el ruido aditivo gaussiano y blanco.

  La potencia de ruido, que es igual a la varianza al cuadrado, se calcula a raíz de la siguiente ecuación:

$$\sigma^2{}_{COFDM} = \left(\frac{E_b}{N_0}\right)^{-1} \cdot \frac{P_s}{b} \cdot \frac{1}{R_c \rho_f \rho_t} \tag{4.21}$$

La expresión se compone de:

- E_b/N_0 : Se trata del valor en dB introducido por el usuario. Es el cociente entre la E_b transmitida por el sistema y la densidad espectral de ruido. Se da en lineal.

- P_S : es la potencia de la señal

- R_c : Es la tasa de código o *code rate.* Se calcula a partir de la tasa de codificación externa del codificador Reed-Solomon ($R_{RS} = 188/204$) y la tasa de codificación interna del codificador convolucional:

$$R_c = R_{cc} \cdot R_{RS} \tag{4.22}$$

- ρ_f : Es el cociente:

$$\rho_f = \frac{N_d}{N_d + N_p} \tag{4.23}$$

  Donde N_d es el número de portadoras de datos por símbolo OFDM, y N_p es el número de portadoras piloto.

- ρ_t : Es el cociente:

$$\rho_t = \frac{T_U}{T_U + T_{CP}} \tag{4.24}$$

  Donde T_U es la duración útil del símbolo OFDM, y T_{CP} es la duración del prefijo cíclico insertado.

- b : Es el número de bits por símbolo.

Una vez calculada la potencia del ruido, se genera la secuencia $w[n]$ de la siguiente manera:

$$w[n] = \sqrt{\frac{\sigma^2{}_{COFDM}}{2}} \cdot (v[n] + j \cdot c[n]) \tag{4.25}$$

Donde $v[n]$ y $c[n]$ son secuencias de variables aleatorias gaussianas complejas.

- **Adición del ruido:** El último paso a realizar es la adición del ruido generado a la señal convolucionada $s_{h_c}[n]$:

$$r[n] = s_{h_c}[n] + w[n] \tag{4.26}$$

4.3.5.3 Resultados

Para mostrar al usuario los resultados de la etapa, se cuenta con diversas gráficas distribuidas en dos grupos y cinco resultados numéricos.

➢ ***Resultados numéricos***

Todos los resultados que se facilitan en la presente etapa son propios de la simulación en curso:

- E_b/N_0 *útil* **:** El primer resultado informa al usuario de la E_b/N_0 útil que transportan los bits de información a raíz de la E_b/N_0 escogida para la transmisión. Se obtiene a raíz de:

$$E_{b-util} = R_c \rho_f \rho_t \cdot E_{b-Tx} \tag{4.27}$$

  Se expresa en dB.

- **Peak to average ratio:** Se calcula a raíz de la señal de salida de la etapa $r[n]$. Muestra la relación entre el valor de pico y el valor medio de la señal, mediante la ecuación (4.15). Se expresa en *dB*.

- **Duración** $h_c[n]$**:** Indica al usuario la duración de la respuesta impulsional del canal $h_c[n]$**.** Se expresa en μs .

- **Duración** $h_c[n]/T_{CP}$ **:** Indica al usuario la relación entre la duración de la respuesta impulsional del canal $h_c[n]$ y la duración del prefijo cíclico T_{CP}**.** Si la duración del prefijo cíclico es mayor que la duración del canal, se puede eliminar la selectividad en frecuencia del canal en la etapa de ecualización. En el caso contrario, existirá una interferencia con el símbolo OFDM adyacente que no se podrá eliminar con la ecualización.

- **Duración** $h_c[n]/T_U$ **:** Indica al usuario la relación entre la duración de la respuesta impulsional del canal $h_c[n]$ y la duración del símbolo útil OFDM T_U **.**

➢ ***Resultados gráficos***

Los resultados gráficos facilitados se dividen en tres grupos. En el primero de ellos se muestran las gráficas correspondientes a representaciones espectrales:

- **Densidad espectral señal en escala logarítmica:** Representación de la densidad espectral de potencia de la señal tras la convolución con el canal $s_{h_c}[n]$ en dB $(10 \cdot \log_{10})$.

- **Densidad espectral señal + ruido en escala logarítmica:** Representación de la densidad espectral de potencia de la señal tras la convolución con el canal $s_{h_c}[n]$ y del ruido $w[n]$ en dB $(10 \cdot \log_{10})$.

- **Densidad espectral señal + H(f) en escala logarítmica:** Representación de la densidad espectral de potencia de la señal tras la convolución con el canal $s_{h_c}[n]$ y del canal $h_c[n]$ en dB $(10 \cdot \log_{10})$.

  También se representan las correspondientes densidades espectrales en escala lineal.

El segundo grupo de señales muestra la señal transferida a la siguiente etapa, $r[n]$, en el dominio temporal:

- **Componentes IQ símbolo OFDM + Marca intervalo de guarda:** Representación de la componente real e imaginaria de la señal temporal $r[n]$, y de la marca del intervalo de guarda.

  A raíz de esta gráfica, el usuario puede visualizar el resto de señales temporales:

  - **Componente I símbolo OFDM + Marca intervalo de guarda**
  - **Componente Q símbolo OFDM + Marca intervalo de guarda**
  - **Componente IQ símbolo OFDM**
  - **Componente I símbolo OFDM**
  - **Componente Q símbolo OFDM**

Por último, se muestran dos gráficas relacionadas con el canal $h_c[n]$ en el dominio temporal:

- **Respuesta impulsional del canal + Marca intervalo de guarda:** Representación de la componente real e imaginaria del canal $h_c[n]$, y de la marca del intervalo de guarda. A través de esta gráfica se puede deducir el valor $h_c[n] \big/ T_{CP}$.

- **Forma exponencial del canal:** Representación de la forma exponencial del canal generado. Si se trata de una generación Rayleigh, se calculará a raíz de la media de 500 muestras del canal $h_c[n]$, generado con los parámetros escogidos por el usuario.

➢ ***Intercambio de ficheros***

- **Guardar:** Se permite guardar en un fichero la señal $r[n]$, sin el ruido añadido, a la salida de la etapa o el canal $h_c[n]$ generado.

- **Leer:** Permite leer de un fichero la señal $r[n]$, sin el ruido añadido, para intercambiarla por la correspondiente secuencia generada. Esta señal debe tener la misma longitud que la generada por el simulador.

4.3.5.4 Simulación 16 paquetes transporte MPEG-2

Continuando con la simulación del ejemplo 16 paquetes MPEG-2 TS, se ha seleccionado en esta etapa un canal Rayleigh caracterizado por una relación DS/T_U de 1/256 y línea de visión directa de 10 dB. La potencia de ruido se ajustará de forma que la E_b/N_0 transmitida en el sistema sea de 9,5 dB.

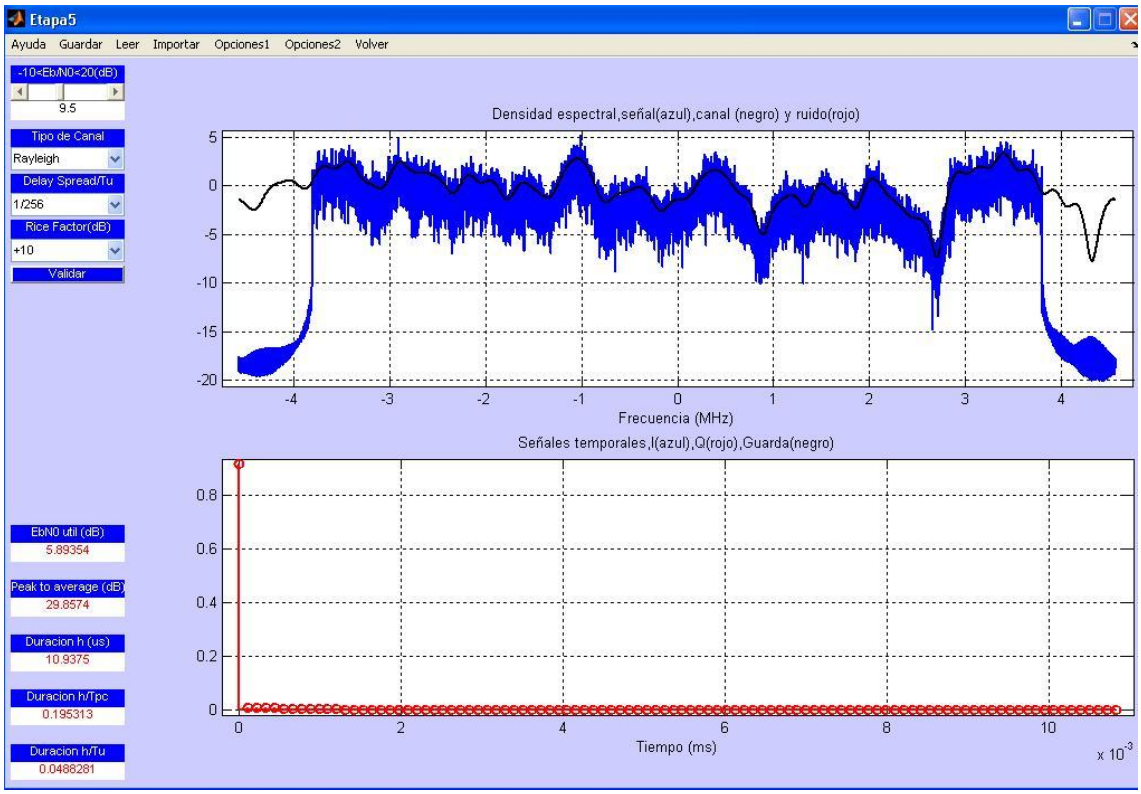


Figura 45 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 5: CANAL

De los resultados que muestra la Figura 45, pueden resaltarse los siguientes aspectos:

✓ Observando la gráfica mostrada la parte superior (densidad espectral) puede percibirse como se manifiesta el efecto del canal sobre el espectro de la señal. Se observa que este canal es selectivo en frecuencia puesto que se degenera

excesivamente algunas subportadoras no siendo así para otras. Este efecto se traduce en ráfagas de errores que el receptor podrá corregir en la etapa de ecualización, puesto que la duración del canal es menor que la duración del intervalo de guarda.

- ✓ A partir de la gráfica temporal inferior, se puede observar la forma exponencial del canal Rayleigh generado.

- ✓ A través de los resultados numéricos, se observa que para una E_b/N_0 transmitida de 9,5 dB, la correspondiente E_b/N_0 para los bits útiles baja hasta 5,89 dB.

4.3.5.5 Retos de programación

Los retos de programación que se han resuelto a lo largo de esta etapa son:

4.3.5.5.1 Retos en la programación MATLAB

- ➢ ***Canal Rayleigh***

En la presente etapa se plantea la generación de canales Rayleigh, caracterizados a partir del valor de ciertos parámetros. Se ha creado un método capaz de devolver la respuesta impulsional compleja y normalizada del canal Rayleigh a partir de los parámetros de entrada. El proceso utilizado para calcular la respuesta impulsional de estos canales es el siguiente:

- ✓ Cálculo del número de coeficientes *Delay_norm* para que la potencia del canal quede dividida por el número *e (*caiga 4,34 dB). Es el resultado de dividir la duración en segundos del *delay spread* por el tiempo de muestreo *Tm*.

- ✓ Cálculo del número de coeficientes *N_rayos* en que la potencia del canal cae 20 dB, es decir, se tienen en cuenta los ecos que llegan con una energía significativa. Se calcula a partir de *Delay_norm*.

- ✓ Generación del perfil exponencial:

$$PDP[n] = e^{-n/Delay_norm},\ n=0,1,\ldots,N_rayos$$

- ✓ Normalización del perfil exponencial:

$$PDP[n] = e^{-n/Delay_norm} \Big/ sum(PDP[n])$$

- ✓ Generación de la parte aleatoria del canal, *al_canal[n]*, con la función de MATLAB *randn*, que crea variables gaussianas aleatorias.

- ✓ Generación de la respuesta impulsional del canal, a partir del producto de la parte aleatoria con el perfil exponencial:

$$h_canal[n] = \sqrt{PDP[n]} \cdot al_canal[n]$$

- ✓ Suma del valor lineal del factor de Rice, a la primera muestra de la respuesta impulsional del canal $h_canal[1] + K_{LOS}$.

- ✓ Normalización del canal (potencia de canal=1) para que no aumente la potencia de la señal transmitida.

4.3.5.5.2 Retos en la programación Java

- ➢ ***Convolución de la señal con el canal***

Para implementar la convolución de la señal con el canal en la presente etapa, se ha empleado la función programada por Carlos y Francisco Vargas para el sistema OFDM en el proyecto LaViCAD.

La solución adoptada por dicha función para implementar la convolución, consiste en combinar el algoritmo suma y solapa [43], basado en la división de la señal en bloques que se procesan de forma independiente con el algoritmo de filtrado FIR definido en la ecuación (4.28):

$$y[n] = \sum_{k=-\infty}^{+\infty} x[k] \cdot h[n-k] \qquad (4.28)$$

Se utilizará el algoritmo de filtrado FIR siempre que se de cualquiera de las siguientes condiciones:

- ✓ Una de las dos señales implicadas no posea una longitud de al menos el doble que la otra.

- ✓ La señal con menos longitud contenga más de 1024 muestras.

- ✓ En el resto de los casos se emplea el algoritmo de suma y solapa descrito en [43].

4.3.5.6 Análisis de la etapa

En la etapa que se trata, cabe destacar algunos detalles. El primero es que cuando la duración del canal $h_c[n]$**,** bien generado por el simulador o bien introducido por fichero, no alcanza las 100 muestras, se añaden ceros hasta llegar a ese número. Por lo tanto debe tenerse en cuenta para los resultados numéricos y gráficos de la etapa que la duración mínima será de 100 muestras, o lo que es lo mismo, de 10,9375 μs.

El segundo de ellos es comentar que la P_s teórica debería resultar:

$$P_s = (N_d \cdot P_d + N_{tps} \cdot P_{tps} + N_p \cdot P_p) \cdot \frac{Nc}{N_T} = 0.8994 \qquad (4.29)$$

Donde $N_d = 1512$ es el número de portadoras de datos, $P_d = 1$ su potencia, $N_{tps} = 17$ el número de portadoras TPS, $P_{tps} = 1$ su potencia, $N_p = 176$ el número de portadoras piloto, $P_d = 16/9$ su potencia, $N_C = 1705$ el número de portadoras por símbolo OFDM y $N_T = 2048$ el número de portadoras transmitidas ($N_T - N_C = 343$ portadoras nulas).

Calculando la potencia de la señal de la manera descrita en 4.3.5.2, se obtiene dicho resultado teórico en el caso QPSK. En el caso 16QAM o 64QAM se obtiene algo más de potencia por el hecho de que la normalización aplicada sobre las portadoras de información en la etapa 3 del simulador, se realiza según las indicaciones del estándar [1], que presupone símbolos equiprobables. En cambio, por el hecho de añadir símbolos de relleno (ceros) en el entrelazador de bytes de la etapa 2 del simulador, ya no se tienen símbolos equiprobables, sino que habrá algún símbolo (*3+3i* en el caso 16QAM y *7+7i* en el caso 64QAM) que se dará más que otro, y esto provocará que la potencia no sea exactamente igual a 1 para las portadoras de datos. En la siguiente etapa se podrá ver en la constelación de la señal recibida este fenómeno.

Por último cabe destacar la forma en que se alcanza una E_b/N_0 prefijada por el usuario. En este simulador a partir de la E_b/N_0 deseada, se calcula la potencia a asignar al ruido, cosa que no es posible en una transmisión real, donde el ruido es el parámetro que está fijado, y ha de buscarse la manera para obtener la E_b/N_0 deseada.

4.3.6 Etapa 6: Demodulación OFDM

En la primera etapa de recepción se realizan las operaciones inversas a las realizadas en el transmisor, esto es: eliminación de los intervalos de guarda, cálculo de la FFT, ecualización y extracción de las portadoras de datos, pilotos y TPS por separado. En la etapa se simula, además, la sincronización de la señal recibida.

El esquema que se adopta en la etapa es:

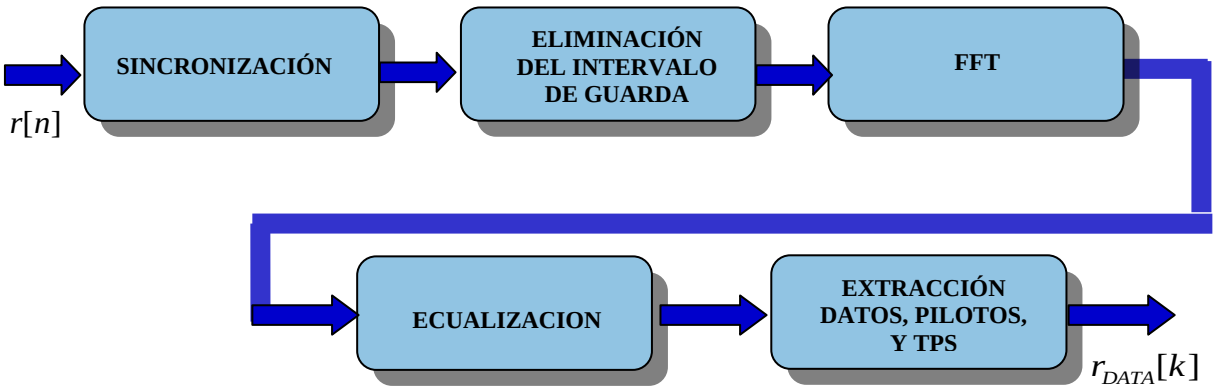


Figura 46 Esquema de la etapa 6 del sistema DVB-T

A la salida de la etapa se obtiene $r_{DATA}[k]$, que es el vector que contiene las portadoras de datos que se envían a la siguiente etapa.

4.3.6.1 Parámetros de entrada

La etapa posee tres parámetros de entrada, los dos primeros permiten introducir errores de sincronismo que afectan a la frecuencia portadora y a la fase de la señal portadora. El tercero permite escoger entre ecualizar o no ecualizar la señal recibida:

- **Error de fase normalizado:** Selecciona el error de fase ε_φ que se desea introducir a la portadora utilizada en recepción dentro de un rango de –1 a 1. Este error está normalizado a π.

- **Error de frecuencia normalizado a la frecuencia de muestreo:** añade un error ε_f a la frecuencia de la portadora utilizada para demodular en recepción dentro de un rango de –0,5 a 0,5. Este error está normalizado a la frecuencia de muestreo.

- **Ecualización:** Permite escoger entre realizar una ecualización ideal de la señal, o no ecualizar.

4.3.6.2 Procesado de la señal

El procesado de señal realizado en la etapa es el que sigue:

- ***Procesado propio de la etapa***

  - **Sincronización:** Las operaciones de procesado comienzan con la simulación de errores de sincronismo sobre la señal recibida. De forma configurable, se pueden suponer errores en la fase y en la frecuencia de la portadora. En conjunto, este error se representa mediante la portadora compleja:

$$c_e[n] = \cos(2\pi\varepsilon_f n + \pi\varepsilon_\varphi) + j\sin(2\pi\varepsilon_f n + \pi\varepsilon_\varphi) \tag{4.30}$$

    donde ε_f es el error de frecuencia normalizado y ε_φ es el error de fase normalizado a π.

    La señal recibida $r[n]$, se multiplica por $c_e[n]$ y se entrega al siguiente bloque de procesado.

  - **Eliminación del intervalo de guarda:** A partir de la señal recibida, esta operación elimina las primeras muestras de cada símbolo OFDM, correspondientes al prefijo cíclico. Tras este procesado, cada símbolo OFDM se compondrá por 2048 muestras en el dominio temporal.

  - **Transformada de Fourier (FFT):** El algoritmo utilizado para realizar la transformada de Fourier es el de la FFT. El objetivo es devolver al dominio frecuencial las subportadoras construidas en transmisión. En este punto, cada símbolo OFDM constará de 2048 muestras, por lo que la FFT utilizada trabajará con ese número de muestras. El algoritmo de la FFT es el que sigue:

$$X_n = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x_k e^{-j\frac{2\pi n}{N}n} \tag{4.31}$$

  - **Ecualización:** La técnica de ecualización empleada se basa en la multiplicación, en el dominio frecuencial, de cada uno de los símbolos OFDM (2048 muestras) por la función inversa de la función de transferencia del canal $H_c[k]$. Esta función de transferencia se calcula a partir de la FFT del canal empleado en la etapa 5: Canal, por lo tanto se realiza una ecualización ideal.

    Tras la ecualización, se eliminan las 343 portadoras nulas por símbolo OFDM introducidas en la etapa 4, y se transfieren al siguiente bloque de procesado de señal los símbolos OFDM con las 1705 portadoras de información.

  - **Extracción de las portadoras de datos, pilotos y TPS:** Esta operación de procesado de señal recupera por separado para cada símbolo OFDM (1705

portadoras), todas las subportadoras con datos (1512), pilotos continuas (45) o dispersas (131) y TPS (17). Se trata del *demultiplexor* de trama.

Tras recuperar las subportadoras, se realiza la desnormalización de todas ellas.

Como señal de salida se obtiene $r_{DATA}[k]$, que será la señal transmitida a la siguiente etapa del simulador. El resto de señales, $r_{PILOT}[k]$ y $r_{TPS}[k]$ se utilizan para cálculos numéricos y gráficos de la etapa.

- ***Procesado para la representación de resultados asociados a la etapa***

  - **Detección de las portadoras de datos:** De cara a representar la constelación de los datos de la señal recibida, y para medir la probabilidad de error, se realiza la detección de la información correspondiente a los datos transmitidos.

    Para realizar la detección de las portadoras de datos $r_{DATA}[k]$, dependiendo de la modulación y el parámetro α escogidos en la transmisión, se realiza una cuantificación de los símbolos recibidos, dando como resultado los símbolos detectados. En la cuantificación, se calcula por separado la parte real e imaginaria del símbolo detectado a partir de la parte real e imaginaria del símbolo recibido.

    El proceso de cuantificación devuelve, para cada símbolo recibido, el símbolo del espacio de señal escogido en la etapa 3 con la mínima distancia euclídea al mismo.

  - **Detección portadoras piloto:** De cara a representar la constelación de la información piloto de la señal recibida y para medir la probabilidad de error, se realiza la detección de la información correspondiente a los pilotos transmitidos.

    Para la detección de las portadoras piloto, basta con usar la función *signo* sobre la parte real de las portadoras piloto recibidas $r_{PILOT}[k]$, para detectar si se trata de un +1 ó un -1, puesto que la modulación que emplean es BPSK.

  - **Detección portadoras TPS:** De cara a representar la constelación de la información TPS de la señal recibida, y para medir la probabilidad de error, se realiza la detección de la información correspondiente a las portadoras TPS transmitidas.

    Cada símbolo OFDM transporta 17 bits TPS modulados DBPSK (± 1), que transportan el mismo bit de información.

    Para la decodificación de las portadoras TPS se sigue el siguiente algoritmo:

    - ✓ A partir de la señal de portadoras TPS recibidas $r_{TPS}[k]$, se obtiene una matriz $R_{TPS}[N][L]$ donde cada fila contiene la información TPS

de un símbolo OFDM, es decir, $L=17$ portadoras TPS y $N=$ número de símbolos OFDM transmitidos.

- ✓ A continuación se realiza la operación:

$$M_{DBPSK}[N-1][L]=sign\left(\mathrm{Re}\left\{\sum_{n=0}^{N-2}\sum_{l=0}^{L-1}R_{TPS}[n][l]\times R_{TPS}[n+1][l]\right\}\right) \quad (4.32)$$

Donde se realiza, para cada portadora *l* de un símbolo OFDM *n*, el producto de su valor por el valor de la misma portadora *l* del símbolo siguiente *n+1*. Así se obtiene una matriz M_{DBPSK} que contiene en cada fila $L=17$ valores que indican si el valor TPS ha variado respecto al símbolo OFDM anterior (-1), o si se mantiene igual (+1).

- ✓ En este momento, se define el umbral de decisión $\eta=8$. Puesto que cada símbolo OFDM (fila de M_{DBPSK}) contiene 17 valores, si más de la mitad de esos valores son +1, se decide que el valor detectado para ese símbolo OFDM es el +1. Con esto se obtiene el vector $V_{DBPSK}[N-1]$, donde *N* es el número de símbolos OFDM.

- ✓ Puesto que la modulación que emplean es DBPSK, como se ha explicado en 4.3.4.2, se ha decodificar el valor -1 por 1, y el valor 1 por 0 dentro del vector $V_{DBPSK}[N-1]$, obteniendo el vector $r_{TPS}[N-1]$ de portadoras TPS decodificadas. Como puede observarse, se decodifica el valor TPS de todos los símbolos OFDM menos del primero, puesto que es el que transporta el bit de inicialización.

- ✓ A partir de los bits del vector $V_{DBPSK}[N-1]$, se recupera la información transmitida por las portadoras TPS vista en la Tabla 5.

4.3.6.3 Resultados

Para mostrar al usuario los resultados de la etapa, se cuenta con 3 grupos de gráficas y 7 resultados numéricos.

- ➢ ***Resultados numéricos***

  - **Prefijo cíclico decodificado:** Se trata prefijo cíclico decodificado a partir de la decodificación TPS. Sólo se obtiene cuando se transmiten 12 o más símbolos OFDM.

  - R_c **decodificado:** Se trata de la tasa de codificación del codificador convolucional de la etapa 3, que se decodifica a partir de las portadoras TPS. Sólo se obtiene cuando se transmiten 9 o más símbolos OFDM.

- **Parámetro alfa decodificado:** Se trata del parámetro α utilizado para crear las constelaciones de señal de la etapa 3, que se decodifica a partir de las portadoras TPS. Sólo se obtiene cuando se transmiten 6 o más símbolos OFDM.

- **Modulación decodificada:** Se trata del tipo de modulación utilizado para mapear la señal en la etapa 3, que se decodifica a partir de las portadoras TPS. Sólo se obtiene cuando se transmiten 3 o más símbolos OFDM.

- **SER datos:** Es la probabilidad de error de símbolo sobre los símbolos de datos presentes a la salida de la etapa $r_{DATA}[n]$.

- **SER pilotos:** Es la probabilidad de error de símbolo sobre los símbolos pilotos $r_{PILOT}[k]$ extraídos en el último bloque de procesado de la etapa**.** Al estar modulados BPSK, es equivalente hablar de BER.

- **BER TPS:** Es la probabilidad de error de bit sobre los bits TPS $r_{TPS}[k]$ extraídos en el último bloque de procesado de la etapa**.**

➢ ***Resultados gráficos***

- **Espacio de señal de datos:** Este grupo consta de una gráfica que muestra en un espacio de señal todas las muestras recibidas que conforman la señal $r_{DATA}[n]$ de salida de la etapa. Se superpone el mapa de señal correspondiente a la constelación inicialmente transmitida. Se ofrecen dos versiones:

  - **Mapa de señal datos sin ruido:** representa las muestras de la señal $r_{DATA}[n]$ sin considerar el ruido añadido en la etapa 5.

  - **Mapa de señal datos con ruido:** representa las muestras de la señal $r_{DATA}[n]$ considerando también el ruido añadido en la etapa 5.

- **Espacio de señal de pilotos:** Este grupo consta de una gráfica que muestra en un espacio de señal todas las muestras recibidas que conforman la señal $r_{PILOT}[k]$ obtenida en el último bloque de procesado de la etapa. Se superpone el mapa de señal BPSK, correspondiente a la constelación inicialmente transmitida. Se ofrecen dos versiones:

  - **Mapa de señal pilotos sin ruido:** representa las muestras de la señal $r_{PILOT}[k]$ sin considerar el ruido añadido en la etapa 5.

  - **Mapa de señal pilotos con ruido:** representa las muestras de la señal $r_{PILOT}[k]$ considerando también el ruido añadido en la etapa 5.

- **Espacio de señal de TPS:** Este grupo consta de una gráfica que muestra en un espacio de señal todas las muestras recibidas que conforman la señal

$r_{TPS}[k]$ obtenida en el último bloque de procesado de la etapa. Se superpone el mapa de señal BPSK, correspondiente a la constelación inicialmente transmitida. Se ofrecen dos versiones:

- **Mapa de señal pilotos sin ruido:** representa las muestras de la señal $r_{PILOT}[k]$ sin considerar el ruido añadido en la etapa 5.

- **Mapa de señal pilotos con ruido:** representa las muestras de la señal $r_{PILOT}[k]$ considerando también el ruido añadido en la etapa 5.

➢ ***Intercambio de ficheros***

- **Guardar:** En la presente etapa, se permite guardar en un fichero la secuencia de símbolos a la salida de la etapa $r_{DATA}[n]$, sin el ruido añadido en la etapa 5.

- **Leer:** Permite leer de un fichero la secuencia de símbolos a la salida de la etapa, $r_{DATA}[n]$, sin el ruido añadido en la etapa 5, para intercambiarla por la correspondiente secuencia generada. Esta secuencia de símbolos debe tener la misma longitud que la generada por el simulador.

4.3.6.4 Simulación 16 paquetes transporte MPEG-2

Para la simulación del ejemplo en esta primera etapa de recepción, se utilizará un demodulador que considere una portadora sin errores de fase y frecuencia. También se ecualizará la señal en el dominio de la frecuencia a partir de la función de transferencia del canal utilizado en la etapa 5 (ecualización ideal).

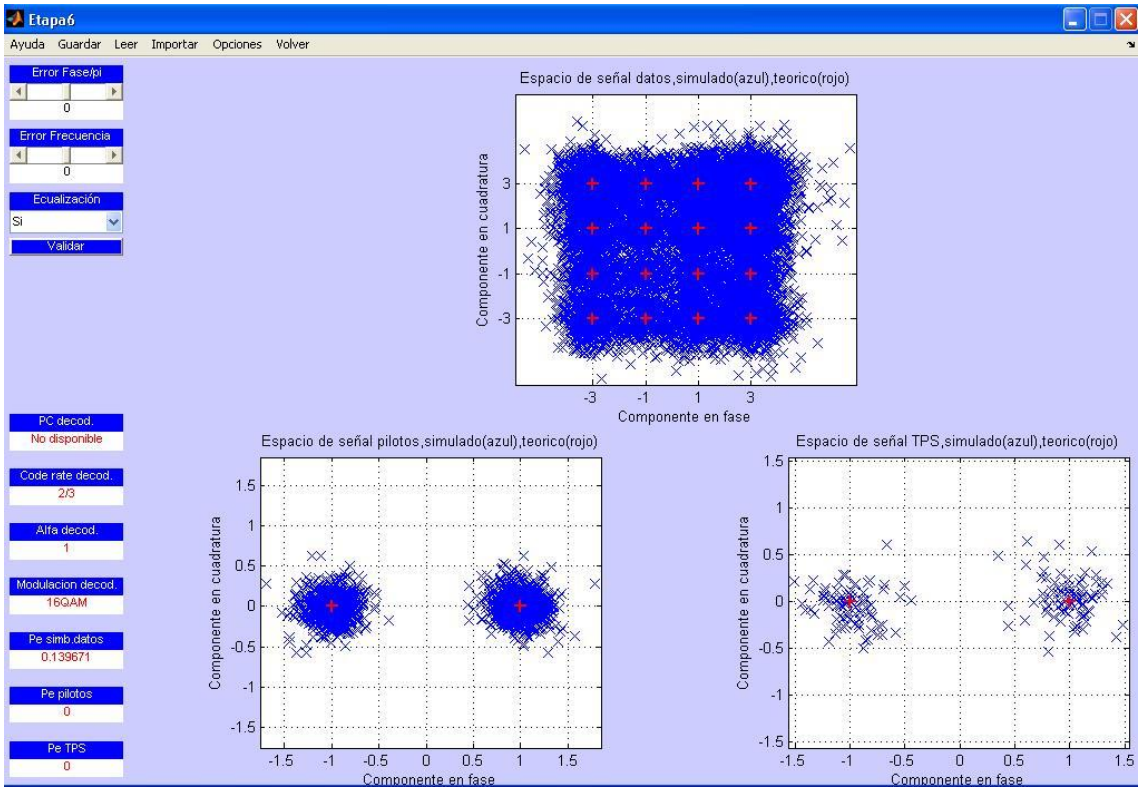


Figura 47 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 6: Demodulación OFDM

De los resultados tanto gráficos como numéricos representados en la Figura 47, pueden resaltarse los siguientes análisis:

- ✓ A partir de la observación de los símbolos de datos ecualizados comparados con su mapa de señal transmitido, puede obtenerse un análisis por inspección acerca de la calidad de la señal $r_{DATA}[n]$ a la salida de la etapa. Este análisis se puede reforzar con el resultado numérico que indica la SER de los datos.

- ✓ Observando el espacio de señal de las portadoras piloto, se puede analizar la calidad de la señal $r_{PILOT}[n]$, obtenida en el bloque demultiplexador de información. También se refuerza dicho análisis con la SER de los pilotos.

- ✓ Por último, a raíz del espacio de señal de las portadoras TPS, se estudia la calidad de la señal $r_{TPS}[n]$, obtenida en el bloque demultiplexador de información. Gracias a este estudio, reforzado con el resultado numérico que indica la BER de las portadoras TPS, se deduce la fiabilidad del resto de resultados referentes a la información TPS ofrecidos por la etapa (prefijo cíclico, R_c, parámetro α y modulación).

4.3.6.5 Retos de programación

De esta etapa, no se destaca ningún reto de programación particular.

4.3.6.6 Análisis de la etapa

En la etapa que se ha tratado se evidencia la adición de ceros de relleno por el entrelazador convolucional de la etapa 2, tal como se comenta en el apartado 4.3.2.6.

En el espacio de señal de los datos $r_{DATA}[n]$ de la Figura 47, se observa como en la posición ***3+3i*** hay una mayor cantidad de símbolos que en cualquier otra posición, y es dónde ocurren más errores.

Por otra parte, cabe comentar que inicialmente en el simulador se planteó hacer la decisión de los símbolos por distancia euclídea mínima del siguiente modo:

- ✓ Cálculo de la distancia euclídea entre el símbolo recibido y cada uno de los símbolos del mapa de señal.

- ✓ Detección del símbolo que ha generado la mínima distancia euclídea.

Dicha implementación era inviable, por la cantidad de recursos que se necesitaban para el cálculo. Por esto, se ha programado un algoritmo cuantificador, que obtiene el símbolo con mínima distancia euclídea al recibido de manera más eficiente.

4.3.7 Etapa 7: Demapping + Desentrelazador interno + Decodificador de Viterbi

En la penúltima etapa del sistema se realizan las operaciones de procesado de señal desde el desentrelazado de los símbolos hasta la recuperación de los bits $b_{dec}[n]$, que se envían a la última etapa del simulador.

Los símbolos recibidos atraviesan de forma secuencial los siguientes procesos: desentrelazador de símbolos, demapping de símbolos (incluye el decisor de símbolos), desentrelazador de bits y decodificador de Viterbi. Antes de cada bloque de procesado, se calcula la probabilidad de error de la correspondiente señal recibida. El esquema que adopta la etapa es el siguiente:

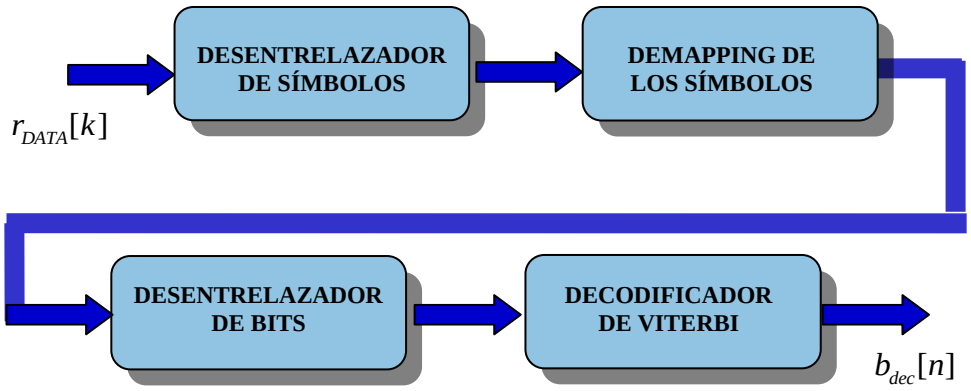


Figura 48 Esquema de la etapa 7 del sistema DVB-T

4.3.7.1 Parámetros de entrada

La presente etapa del simulador posee un único parámetro de entrada:

- **Tipo de decodificación:** permite elegir entre decodificación hard o soft para el decodificador de Viterbi.

  En la versión MATLAB del simulador se puede escoger entre estos tres tipos de decodificadores: ***hard, soft-2bits o soft-3bits*.** En la versión Java, simplemente se escoge decodificación ***hard* o *soft***. En el apartado 4.3.7.2 se explica la diferencia existente en cada tipo de decodificación.

4.3.7.2 Procesado de la señal

Las operaciones de procesado de señal que se desarrollan en esta etapa pueden dividirse en 4 subetapas:

➢ ***Procesado propio de la etapa***

- **Desentrelazador de símbolos:** El procesado de señal de la etapa comienza con el desentrelazador de símbolos, que opera sobre la señal de entrada

$r_{DATA}[k]$. El desentrelazador ejecuta el cálculo inverso al entrelazador de símbolos detallado en la etapa 3.

- **Demapping de los símbolos:** Una vez se tiene la señal con los símbolos desentrelazados, se pasa a detectar los símbolos y demapearlos (decodificarlos) en bits.

  El bloque de procesado de señal de demapping funciona de diferente modo dependiendo si se escoge una decodificación *hard* o *soft.* En todos los casos se decodifica la parte real y la parte imaginaria del símbolo recibido por separado, puesto que al ser cuadrada la constelación, y gracias a las propiedades del código Gray, la componente en fase no interfiere a la componente en cuadratura, y viceversa.

  En el caso que el usuario seleccione decodificación ***hard***, se trabaja de la siguiente manera:

  - ✓ Para realizar la detección, dependiendo de la modulación y el parámetro α escogidos en la transmisión, se realiza una cuantificación de los símbolos recibidos, dando como resultado los símbolos detectados. Como ya se ha comentado, en la cuantificación se calcula por separado la parte real e imaginaria del símbolo detectado, a partir de la parte real e imaginaria del símbolo recibido.

    El proceso de cuantificación nos devuelve, para cada símbolo recibido, el símbolo del espacio de señal escogido en la etapa 3 con la mínima distancia euclídea al mismo.

  - ✓ Una vez detectados, se procede a demapear (decodificar) los símbolos. Esto consiste en asignar a cada uno de los símbolos detectados la serie de "b" bits inversamente a como se haya realizado la codificación de símbolo en la transmisión. Puede consultarse la Figura 37, hallada en el punto "Mapeo de los símbolos" del procesado de señal de la etapa 3 del simulador, donde se encuentra la asignación de bits en símbolos para cada constelación considerada en este sistema.

  En el caso contrario, cuando el usuario seleccione decodificación ***soft***, se debe diferenciar la versión Java de la versión MATLAB del simulador, puesto que el decodificador de Viterbi trabaja con un tipo distinto de código de entrada en cada caso, números enteros en MATLAB y probabilidades en Java. En ambos casos, la decisión para la decodificación *soft* debería hacerse en base a la distancia euclídea entre el símbolo recibido y los símbolos más cercanos pero, debido a la complejidad de las constelaciones, se ha optado por implementar una solución más sencilla subóptima.

  En la versión MATLAB del simulador, tal como puede verse en detalle en el apartado 4.3.7.5: Retos de programación, se detectan y decodifican los símbolos bit a bit, usando el conocimiento del código Gray aplicado a cada uno de ellos en transmisión, y estableciendo unos umbrales de detección

entre los símbolos de la constelación más cercanos, cuyo bit i-ésimo a decidir valga 0 ($b_i = 0$) y 1 ($b_i = 1$) respectivamente. Nótese que la solución adoptada es una simplificación de la detección teórica, pero mucho más eficiente para una implementación práctica.

Finalmente se obtiene, a partir de los símbolos recibidos, una secuencia de valores de 0 a $2^{nsdec-1}$, donde el 0 representa que el bit 0 más fiable, y $2^{nsdec-1}$ el 1 más fiable. Esta secuencia de valores representa, entonces, a la secuencia de bits en que se decodifican los símbolos recibidos. El valor *nsdec* es 2 en el caso *soft-2bits*, y 3 en el caso *soft-3bits*.

En la versión Java del simulador, el demapeo de símbolos para la demodulación *soft*, cuya realización ha sido implementada por Pedro Espinosa Fricke para el departamento de TSC [28], el proceso que se sigue es el siguiente:

- ✓ De cada símbolo se conoce la secuencia de bits asociada, puesto que es la que se ha fijado en la etapa de modulación.

- ✓ Se desea obtener la probabilidad de que un bit en concreto valga uno o cero. En función de la varianza del ruido, la probabilidad de que el símbolo recibido sea uno u otro se puede modelar con el valor de una gaussiana convenientemente centrada.

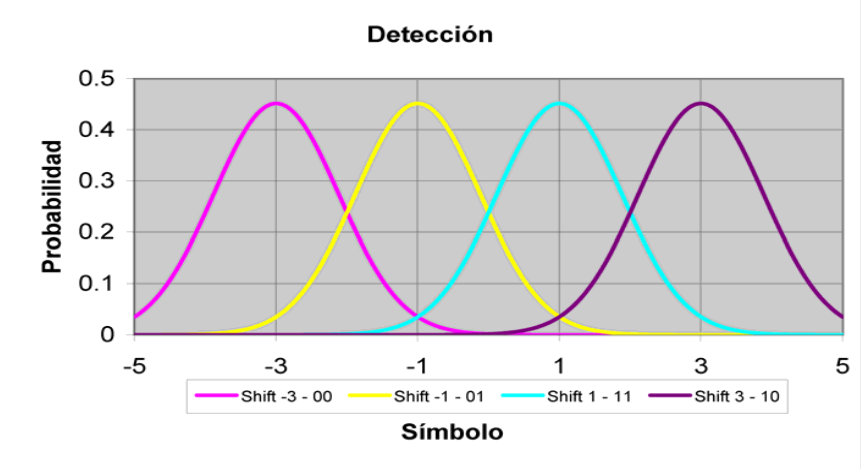


Figura 49 Detección soft para modulación 16QAM simulador Java. Figura extraída de [28]

- ✓ Finalmente, para conocer la probabilidad de que, por ejemplo, el primer bit sea cero, deberá evaluarse la probabilidad de toda aquella gaussiana cuya asignación de bits tenga un cero en la primera posición. Puesto que esta evaluación no es necesariamente una probabilidad, es necesario ponderarla con la probabilidad de que el mismo bit valga uno. Así, en la 16-QAM de la Figura 49, donde se tienen hasta cuatro gaussianas centradas en los valores -3, -1, 1 y 3, se calcularía de la siguiente manera:

$$P(b_1 = 0 \mid x) = \frac{1}{2} f(x \mid -3) + \frac{1}{2} f(x \mid -1) = p_0$$
$$P(b_1 = 1 \mid x) = \frac{1}{2} f(x \mid 3) + \frac{1}{2} f(x \mid 1) = p_1 \qquad (4.33)$$
$$p_0 = \frac{p_0}{p_0 + p_1}; p_1 = \frac{p_1}{p_0 + p_1}$$

Como puede notarse, en la versión Java del demodulador *soft* se tienen en cuenta todos los símbolos posibles de la constelación en el momento de calcular la probabilidad, mientras en la versión MATLAB solo se tienen en cuenta los dos símbolos más cercanos en la constelación, en los cuales el bit a decidir sea igual a 0 en uno de ellos, y 1 en el otro.

- **Desentrelazador de bits:** El procesado de señal del bloque desentrelazador de bits, opera sobre la señal que simboliza los bits a la salida del demapping. El desentrelazador ejecuta el cálculo inverso al entrelazador de bits detallado en la etapa 3.

  Si se observa la Figura 36, contenida dentro del punto "Entrelazador de bits" del procesado de señal de la etapa 3 del simulador, después de realizar el desentrelazado de bits, se debe deshacer el **demultiplexado** realizado en transmisión (DEMUX) para recuperar los bits de salida del codificador convolucional.

- **Decodificador de Viterbi:** El decodificador empleado en la etapa se basa en el algoritmo de Viterbi. Cabe destacar que el decodificador es capaz de decodificar tramas con R_c de 1/2, 2/3, 3/4, 5/6 ó 7/8, y comienza en el estado todo ceros igual que el codificador convolucional de la etapa 3.

  El decodificador de Viterbi trabaja con los llamados diagramas de Trellis. El concepto que persiguen los mismos es trazar un camino a través de todos los estados posibles que cumpla ser de máxima probabilidad (*maximun likelihood*) en base a una métrica por determinar.

  La capacidad correctora de este tipo de códigos se sustenta en que la propia codificación restringe el rango de valores posibles que pueden tomar los bits de la trama. De esta manera, es posible detectar los errores producidos en la transmisión. Si se trazan todos los caminos posibles en el diagrama de Trellis, al finalizar existirá uno que, en base a la métrica, sea el que implique un número mínimo de errores y, por ello, sea la decodificación con mayor probabilidad de ser correcta. Dicha decodificación se puede deducir de forma instantánea para cada camino. De esta manera se va formando una propuesta de trama decodificada para cada camino.

  En el desarrollo del Trellis se producen convergencias entre los diversos caminos obtenidos. Cuando esto sucede, debe determinarse cuál de ellos es el que supone un menor número de errores y descartar el otro.

Para calcular las métricas, es necesario diferenciar el proceso según el tipo de decodificación, puesto que en modo *hard* debe calcular la distancia de Hamming y para el modo *soft* debe maximizar la probabilidad.

La distancia de Hamming se utiliza para evaluar en cuánto difiere la pareja de bits recibidos de las transiciones posibles y se reduce a la suma exclusiva de la pareja recibida y la transición.

La probabilidad máxima, utilizada para el caso *soft*, se basa en la probabilidad de que el bit de una posición determinada valga cero. Esta información se empleará para tratar de afinar la decodificación. En este caso se trata de averiguar cuál de las dos transiciones es la más probable.

Finalmente se obtiene $b_{dec}[n]$, que es la secuencia de bits a la salida del decodificador de Viterbi, que se pasa a la última etapa del simulador.

El decodificador de Viterbi de la versión Java del simulador, ha sido implementado por Pedro Espinosa Fricke para el departamento de TSC [28].

- ***Procesado para la representación de resultados asociados a la etapa***

  - **Cálculo de la probabilidad de error de bit (BER):** Los cálculos de probabilidad de error de bit que se determinan en esta etapa se enfocan a evidenciar la capacidad correctora del decodificador de Viterbi. Para ello se comparan las secuencias de bits antes y después del paso por el decodificador con las secuencias equivalentes en transmisión.

    También se muestra la probabilidad de error de bit antes y después del desentrelazado interno, que obviamente es la misma. Para este cálculo es preciso realizar un demapeo de la señal para poder comparar con su secuencia correspondiente en transmisión a nivel de bit.

    La probabilidad de error de bit se contabiliza como el número de errores detectados dividido entre el número total de bits.

4.3.7.3 Resultados

Para mostrar los resultados de la etapa, se cuenta con 2 comparaciones gráficas y 4 resultados numéricos.

- ***Resultados numéricos***

  - **BER antes del desentrelazado interno:** Es la probabilidad de error de bit sobre la señal de entrada a la etapa $r_{DATA}[k]$. Como dicha señal es una secuencia de símbolos, se debe realizar una detección de los símbolos recibidos, y posterior decodificación a bits. Esta misma operación debe realizarse sobre la secuencia equivalente en transmisión. Tras estos cálculos, se halla la probabilidad de error de bit.

- **BER después del desentrelazado interno:** Se trata de la probabilidad de error de bit tras el desentrelazado interno (desentrelazador de símbolos y desentrelazador de bits). Este resultado coincide con el anteriormente calculado, debido a que solamente se han cambiado de sitio los símbolos y se han pasado a bits.

- **BER antes del decodificador de Viterbi:** es el cálculo de la probabilidad de error de bit sobre los bits presentes a la entrada del decodificador de Viterbi. Este resultado coincide con los dos anteriores. Sirve para comprobar el correcto funcionamiento del simulador.

- **BER después del decodificador de Viterbi:** es el cálculo de la probabilidad de error de bit sobre los bits de datos presentes a la salida del decodificador de Viterbi. Con este valor se comprobará la eficacia del decodificador de Viterbi.

➢ ***Resultados gráficos***

- **Comparación desentrelazado interno:** Este grupo consta de dos gráficas que marcan los errores existentes en la secuencia de bits antes y después del desentrelazado interno (desentrelazador de símbolos + desentrelazador de bits). Gracias a esta comparación, puede verse como cambia la posición de los errores, que pasan a aleatorizarse. Dicha gráfica representa la presencia o ausencia de error para cada uno de los bits, así pues se representa un punto por bit que puede adoptar los siguientes valores:

  - El valor cero indica que el bit se ha recibido correctamente.
  - El valor uno indica bit erróneo.

  Las gráficas que se presentan son:

  - **Errores antes del desentrelazado interno**

  - **Errores después del desentrelazado interno**

- **Comparación decodificador de Viterbi:** Este grupo consta de dos gráficas que marcan los errores existentes en la secuencia de bits antes y después del decodificador de Viterbi. Gracias a esta comparación, puede verse como disminuye el número de errores. Dicha gráfica representa la presencia de error (valor 1) o ausencia de error (valor 0) para cada uno de los bits.

  Las gráficas que se presentan son:

  - **Errores antes del decodificador de Viterbi**

  - **Errores después del decodificador de Viterbi**

➢ ***Intercambio de ficheros***

- **Guardar:** En la presente etapa, se permite guardar en un fichero la secuencia de bits a la salida de la etapa $b_{dec}[n]$.

- **Leer:** Permite leer de un fichero la secuencia de bits a la salida de la etapa, $b_{dec}[n]$, para intercambiarla por la correspondiente secuencia generada. Esta secuencia de bits debe tener la misma longitud que la generada por el simulador.

4.3.7.4 Simulación 16 paquetes transporte MPEG-2

Se escoge una decodificación *soft-3bit* para lograr corregir el máximo número de errores posible.

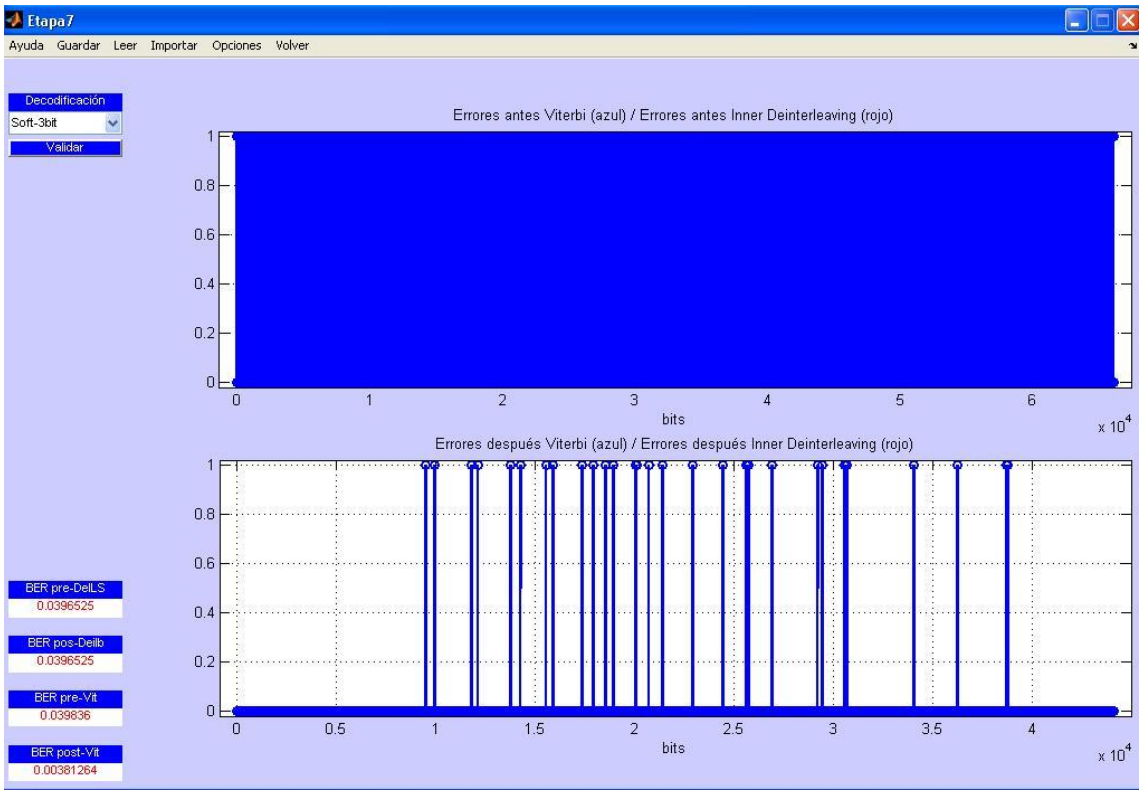


Figura 50 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 7: Demapping + Desentrelazador interno + Decodificador de Viterbi

De la observación de los resultados presentados por la Figura 50, pueden extraerse las siguientes conclusiones:

- ✓ Por un lado, las gráficas evidencian la presencia de una gran cantidad de bits erróneos a la entrada del decodificador de Viterbi. Tras el paso por el decodificador, existe una clara disminución de bits erróneos.

- ✓ El cálculo numérico de la probabilidad de error de bit evalúa la reducción de la BER que proporciona el decodificador de Viterbi. Ésta pasa de estar cerca de un 4% a un 0,4%, reduciéndose considerablemente (hasta 10 veces). Gracias a esta

reducción en la BER, el decodificador Reed-Solomon podrá corregir los errores todavía existentes en la señal recibida.

4.3.7.5 Retos de programación

Entre los retos de programación que los cálculos de procesado de señal han planteado en esta etapa, se destaca el decodificador de Viterbi.

4.3.7.5.1 Retos en la programación MATLAB

- ***Demodulación soft para decodificador de Viterbi***

El objetivo del demodulador *soft* es obtener, a partir de los símbolos recibidos, una secuencia de valores de 0 a $2^{nsdec-1}$, donde el 0 representa que el bit 0 más fiable, y $2^{nsdec-1}$ el 1 más fiable. Esta secuencia de valores representa, entonces, la secuencia de bits en que se decodifican los símbolos recibidos. El valor *nsdec* es 2 en el caso *soft-2bits*, y 3 en el caso *soft-3bits.*

Para la implementación, se ha consultado [34], tanto para el tema de los umbrales de decisión como para el número de niveles de cuantificación. Se parte de la idea de una demodulación BPSK, donde se representa la probabilidad de la señal recibida condicionada al hecho de haber enviado un 0 $f(x|0)$ y la probabilidad al hecho de haber enviado un 1 $f(x|1)$.

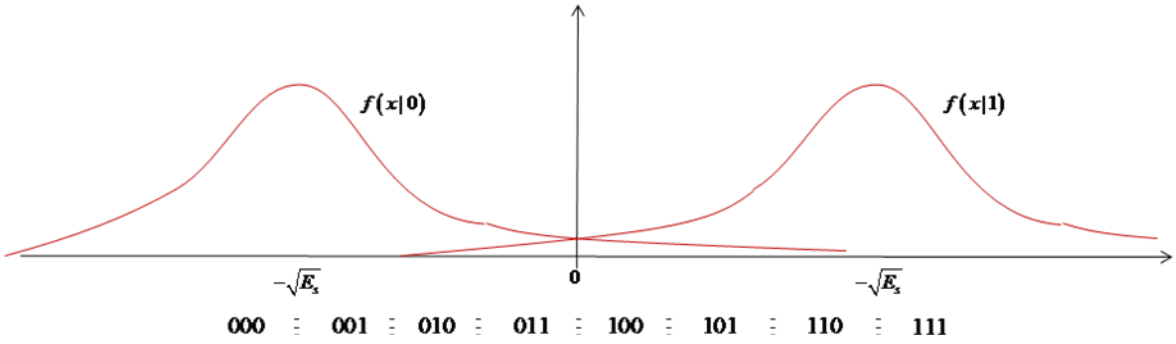


Figura 51 Decisiones del demodulador soft-3 bits

Para este modelo se proponen los umbrales de decisión que se pueden ver en la Figura 51. A partir de éstos, se cuantifica una graduación entre los dos bits '0' y '1'. Así, la decisión de si un símbolo es '1' o '0' se suaviza (decisión *soft*). En la Figura 51, el símbolo '000' se interpreta como un '0' con máxima probabilidad y el símbolo '111' como un '1' con máxima probabilidad. En medio se encuentran el resto de símbolos.

Con la idea planteada para BPSK, se adapta la secuencia de símbolos recibida, modulada en QPSK, 16QAM o 64QAM, para poder decodificarla usando el modelo expuesto. Esta adaptación es posible, ya que gracias a conocer el código Gray usado en transmisión, se puede decodificar bit a bit, y al ser simétricas las constelaciones, se pueden realizar todos los cálculos para la cuantificación en la parte positiva del espacio

de señal y usar el modelo bpsk. Con esto, se está simplificando la decodificación *soft* óptima, puesto que está reduciendo la decisión a los dos símbolos más cercanos al umbral que diferencia el bit de 0 a 1 en cada caso. Los umbrales empleados en el simulador para el demodulador *soft-3bits* son los siguientes:

Figura 52 Umbrales de detección para demodulador *soft-3bits*

Como puede verse en la Figura 52, se han fijado unos umbrales. No obstante cabe comentar que la programación que se ha llevado a cabo en este apartado está preparada para ser fácilmente cambiada por el programador para usar otra plantilla de umbrales, teniendo únicamente que cambiar los valores de la misma, puesto que el código es genérico. Según [34], la ganancia por usar más niveles de cuantificación que los 8 que han sido implementados es muy poca.

Esta demodulación se hace de manera subóptima, para así tener un programa más eficiente y veloz. Lo óptimo sería hacerlo calculando probabilidades con las funciones de densidad…o cambiando los umbrales por unos que dependan de la forma de la gaussiana (hay tablas que regulan estos umbrales). Pero pensando que pasar de una decodificación de tipo *soft-3bits* a una decodificación de tipo *soft-4bits* la mejora es insustancial, quiere decir que pasar de mover el umbral un poco a la derecha o a la izquierda no aporta una mejora de prestaciones, por lo cual se ha decidido establecer unos umbrales fijos (que además el programador los puede cambiar fácilmente cuando lo desee).

En la realización del algoritmo de Viterbi para la versión MATLAB del simulador se ha utilizado la función *vitdec*, subrutina del propio programa, cuya secuencia de entrada a decodificar ha de estar en el formato que prepara el demodulador *soft* implementado.

4.3.7.5.2 Retos en la programación Java

- ***Decodificador mediante algoritmo de Viterbi***

El procesado de señal referente al algoritmo de Viterbi, tal como se ha comentado, queda enmarcado dentro del trabajo titulado *Algoritmo de Viterbi Implementado en Java* [28].

De forma simplificada en este método se ejecutan los siguientes pasos:

- ✓ Generación de los polinomios que describen el codificador.
- ✓ Generación de la trama a decodificar a partir de los bits.

- ✓ Llamada al decodificador de Viterbi. El proceso de depuncturing se incluye en este paso.

- ✓ Adecuar la variable retornada al formato de salida deseado.

Se recomienda consultar la referencia [28], con el fin de profundizar en el procesado de señal interno del decodificador.

4.3.7.6 Análisis de la etapa

En la presente etapa se destacan varios aspectos sobre la interpretación de los resultados presentados y sobre el funcionamiento de la misma:

- ➢ ***Cálculo de probabilidad de error de símbolo a probabilidad de error de bit***

El primero punto destacable es que, a diferencia de la anterior etapa, se realizan cálculos de la BER en lugar de la SER. Entonces, se evidencia una clara diferencia entre la SER de los datos $r_{DATA}[n]$a la salida de la etapa 6, con la BER de los mismos datos a la entrada de la etapa 7. La diferencia es del orden del factor de número de bits por símbolo, porque se pasa de calcular error de símbolo a error de bit. Para el caso de 1 bit erróneo por símbolo erróneo: en QPSK la BER sería la mitad que la SER, una cuarta parte en 16QAM o una sexta parte en 64QAM.

- ➢ ***Decodificación de Viterbi versión MATLAB frente a la versión Java***

Otro aspecto que se ha analizado es la capacidad correctora que presenta el decodificador de Viterbi tanto en la versión MATLAB como en la versión Java del simulador.

En el caso de la decodificación *hard* las prestaciones de ambas versiones son las mismas, puesto que ambas operan del mismo modo. No obstante, en el caso de decodificación *soft*, el decodificador de Viterbi de la versión Java ofrece una capacidad correctora mayor que el decodificador *soft* tanto de *2* como de *3-bits* de la versión MATLAB para todas las modulaciones. Esto es debido a que la versión Java del simulador utiliza un demapeo de bits para la demodulación *soft* más cercano al óptimo que la versión MATLAB, teniendo en cuenta todos los símbolos posibles de la constelación, y evaluando gaussianas convenientemente centradas en función de la varianza de ruido. Aparte, la demodulación Java obtiene probabilidades, las cuales tienen una precisión mayor que los valores decimales obtenidos en la versión MATLAB (de 0 a 7 en *soft-3bits*).

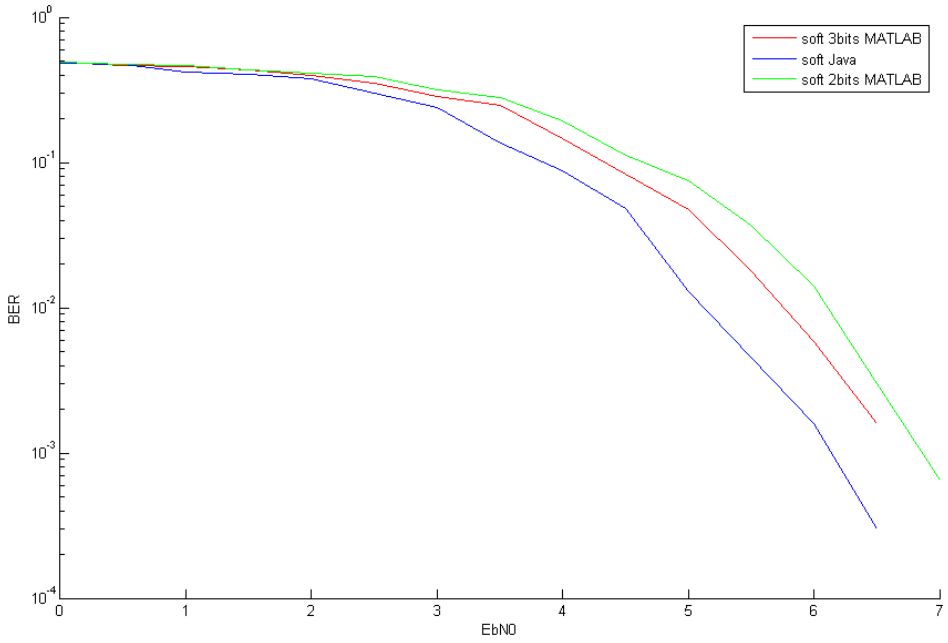


Figura 53 Curvas de rendimiento decodificador Viterbi *soft* BER vs EbN0. Modulación empleada: 16QAM. $R_c = 1/2$

En la Figura 53 se puede observar el mejor rendimiento del decodificador *soft* empleado en la versión Java del simulador. La simulación se ha realizado con la transmisión de 16 paquetes MPEG-2 TS usando modulación 16QAM y una tasa de codificación interna de $2/3$.

Por otra parte, en la versión MATLAB del simulador, se aprecia una mayor capacidad correctora en el decodificador *soft* para el caso de modulación QPSK respecto a los casos 16QAM o 64QAM, siendo con esta última modulación con la que se aprecia una menor capacidad correctora. Esto es debido a que la demodulación *soft* implementada en la versión MATLAB del simulador es una mejor aproximación de la demodulación *soft* ideal para el caso QPSK que para los casos 16QAM o 64QAM. En la versión Java ofrece una capacidad correctora muy semejante para todas las modulaciones.

- ***Análisis de las gráficas representadas***

Al principio la presente etapa, se ha hecho un estudio teórico de lo que se esperaba visualizar en las gráficas representadas. Después de realizar la etapa, se ha comprobado el correcto funcionamiento.

Así, en la gráfica de errores antes del desentrelazador interno, se esperan muchos errores. Si el canal utilizado es el ideal, dichos errores se deben presentar en posiciones aleatorias; si el canal no es ideal (Rayleigh), los errores deben aparecer a ráfagas en los desvanecimientos del canal.

En la gráfica después del desentrelazado interno, que coincide con la gráfica antes del decodificador de Viterbi, los errores mencionados pasan a estar en posiciones aleatorias.

Finalmente, en la gráfica de errores después del decodificador de Viterbi, aparecen menos errores y se presentan a ráfagas.

4.3.8 Etapa 8: Desentrelazado externo + Decodificador Reed-Solomon

En la última etapa del sistema se realizan las operaciones de procesado de señal correspondientes al desentrelazado y la decodificación externos.

Los bits tras la decodificación de Viterbi $b_{dec}[n]$, pasan primero por el desentrelazador convolucional (basado en la aproximación de Forney) y posteriormente por el decodificador externo (de tipo Reed-Solomon). Los dos bloques de procesado operan a nivel de byte, por lo tanto se realiza una conversión de bits a bytes previa al desentrelazador y la correspondiente conversión de bytes a bits a la salida del decodificador.

El esquema que adopta la etapa es el siguiente:

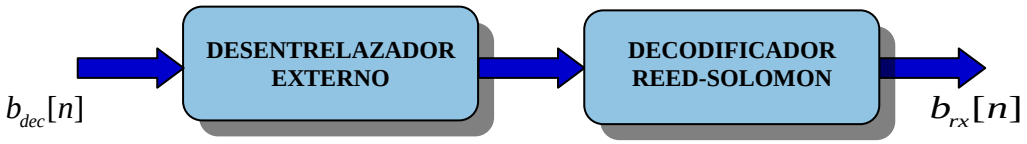


Figura 54 Esquema de la etapa 8 del sistema DVB-T

Como puede observarse en el esquema, la señal de salida de la etapa será $b_{rx}[n]$, que es el vector de bits a la salida del simulador.

4.3.8.1 Parámetros de entrada

La presente etapa no dispone de ningún parámetro de entrada configurable por el usuario.

4.3.8.2 Procesado de la señal

A continuación se describen las operaciones de procesado de señal especificadas en el estándar para la implementación de la etapa:

- ***Procesado propio de la etapa***

  - **Desentrelazado externo (Forney):** La implementación de este procesado de señal comienza con la conversión del vector de bits $b_{dec}[n]$ a bytes $B_{dec}[n]$, puesto que el desentrelazador externo trabaja a nivel de byte.

    El desentrelazador, cuyo esquema puede verse en la Figura 29, tiene un principio similar al entrelazador, salvo que los índices de las ramas están invertidos, es decir: *j=0* se corresponde con el máximo retardo. En la práctica, la sincronización se obtiene rutando el primer byte de sincronismo reconocido hacia la rama *0*. En el simulador no es necesaria la sincronización.

En estas condiciones, en el desentrelazador cada byte se retarda *(11-j)x17* posiciones, con lo que el retardo total entre emisión y recepción es de *(j+11-j)x17=187*, valor que por ser idéntico para todos los bytes permite recuperar en la recepción el orden original.

Todo este proceso, como ya se ha comentado en la correspondiente etapa de transmisión, reduce los errores por ráfagas introducidos por el canal (errores que afectan a varios bytes consecutivos) y así se ayudará al decodificador Reed-Solomon a recuperar la información original.

- **Decodificación externa (Reed-Solomon):** El último bloque de procesado de señal del sistema realiza la decodificación Reed-Solomon.

  Los pasos seguidos por dicho decodificador para recuperar la información originalmente enviada son los siguientes:

  - ✓ Calcular los *2t* síndromes, donde *t=8* es la capacidad correctora. Si todos los síndromes son iguales a 0, no existen errores y, por lo tanto, no se necesita ninguna corrección de errores.

  - ✓ Si algún síndrome es distinto de cero, quiere decir que ha ocurrido algún error, por lo que se deben localizar los errores.

    Para encontrar el polinomio localizador de errores, se usa el algoritmo de Berlekamp-Massey.

  - ✓ Posteriormente, se debe encontrar la magnitud del error para cada posición con error.

    Para encontrar la magnitud del error, se utiliza el algoritmo de Forney.

  - ✓ Finalmente, tras obtener el polinomio de errores *e(x)*, se realiza la corrección de los mismos añadiendo *e(x)* a *r(x)* para obtener el originalmente transmitido *c(x)*.

$$c(x) = r(x) + e(x) \tag{4.34}$$

En la versión MATLAB, se ha utilizado la subrutina de procesado *rsdec* (función del *toolbox* de MATLAB), teniendo en cuenta los aspectos descritos en el apartado 4.3.2.5: Retos de programación de la etapa 2. Si el decodificador no es capaz de corregir el paquete de datos entrante, lo mantiene intacto e informa de que ha habido un error en la decodificación. Con esta información se puede sacar por pantalla el número de paquetes que se mantienen erróneos tras el decodificador y el número de paquetes corregidos por el mismo.

En el apartado 4.3.7.5: Retos de programación, se detalla la implementación del decodificador Reed-Solomon empleado en la versión Java del simulador,

el cual intenta corregir cada paquete de datos, aunque el número de errores sea mayor que la capacidad correctora.

Después de realizar la decodificación Reed-Solomon, se realiza la conversión de la secuencia de bytes $c(x)$ a la secuencia de bits $b_{rx}[n]$. Así, como salida de la etapa se tiene $b_{rx}[n]$, que es la señal transmitida correspondiente a la señal de salida de la etapa 1 $b_{scr}[n]$.

- ***Procesado para la representación de resultados asociados a la etapa***

  - **Cálculo de la probabilidad de error de byte (P_e):** Los cálculos de probabilidad de error de byte que se determinan en esta etapa se enfocan a evidenciar la capacidad correctora del decodificador Reed-Solomon. Para ello se comparan las secuencias de bytes antes y después del paso por el decodificador con las secuencias equivalentes en transmisión.

    También se muestra la probabilidad de error de byte antes y después del desentrelazado externo, la cual varía en el caso de haber escogido realizar el entrelazado externo en transmisión. Esto es debido a que al hacer el desentrelazado se eliminan los 2244 ceros de relleno añadidos por el entrelazador.

    La probabilidad de error de byte se contabiliza como el número de errores detectados dividido entre el número total de bytes.

4.3.8.3 Resultados

Para mostrar los resultados de la etapa, se cuenta con 2 comparaciones gráficas y varios resultados numéricos.

- ***Resultados numéricos***

  - P_e **antes del desentrelazado externo:** Es la probabilidad de error de byte sobre la señal de entrada a la etapa $b_{dec}[n]$, tras su conversión a bytes.

  - P_e **después del desentrelazado externo (antes del decodificador Reed-Solomon):** Se trata de la probabilidad de error de byte tras el desentrelazado externo. También es la probabilidad de error de byte a la entrada del decodificador Reed-Solomon. Este resultado varía con el anteriormente calculado, debido a que no solamente se han cambiado de sitio los bytes, sino que además se han eliminado los ceros de relleno añadidos en el entrelazador de la etapa 2. Sirve para comprobar el correcto funcionamiento del simulador.

  - P_e **después del decodificador Reed-Solomon:** es el cálculo de la probabilidad de error de byte presente a la salida del decodificador Reed-Solomon. Con este valor se comprobará la eficacia del decodificador.

- **BER después del decodificador Reed-Solomon:** es el cálculo de la probabilidad de error de bit presente a la salida del sistema. Se calcula tras la conversión de bytes a bits a la salida del decodificador Reed-Solomon.

- **Número de paquetes erróneos:** En la versión MATLAB del simulador significa el número de paquetes que han llegado erróneos al decodificador RS y no han podido ser corregidos por el mismo. En la versión Java del simulador, significa el número de paquetes que llegan erróneos al decodificador.

- **Número de paquetes corregidos:** Es el número de paquetes que el decodificador RS ha logrado corregir. Solo disponible en la versión MATLAB.

➢ ***Resultados gráficos***

- **Comparación desentrelazado externo:** Este grupo consta de dos gráficas que marcan los errores existentes en la secuencia de bytes antes y después del desentrelazado externo. Gracias a esta comparación, puede verse como cambia la posición de los errores, que pasan a aleatorizarse. Dicha gráfica representa la presencia de error (valor 1) o ausencia de error (valor 0) para cada uno de los bytes.

  Las gráficas que se presentan son:

  - **Errores antes del desentrelazado externo**

  - **Errores después del desentrelazado externo**

- **Comparación decodificador Reed-Solomon:** Este grupo consta de dos gráficas que marcan los errores existentes en la secuencia de bytes antes y después del decodificador Reed-Solomon. Gracias a esta comparación, puede verse como disminuye el número de errores. Dicha gráfica representa la presencia de error (valor 1) o ausencia de error (valor 0) para cada uno de los bytes.

  Las gráficas que se presentan son:

  - **Errores antes del decodificador Reed-Solomon**

  - **Errores después del decodificador Reed-Solomon**

➢ ***Intercambio de ficheros***

- **Guardar:** En la presente etapa, se permite guardar en un fichero la secuencia de bits a la salida de la etapa $b_{rx}[n]$.

- **Leer:** En esta etapa, por ser la última, no se permite leer ningún fichero.

4.3.8.4 Simulación 16 paquetes transporte MPEG-2

En la última etapa del sistema no hay ningún parámetro de entrada configurable por el usuario.

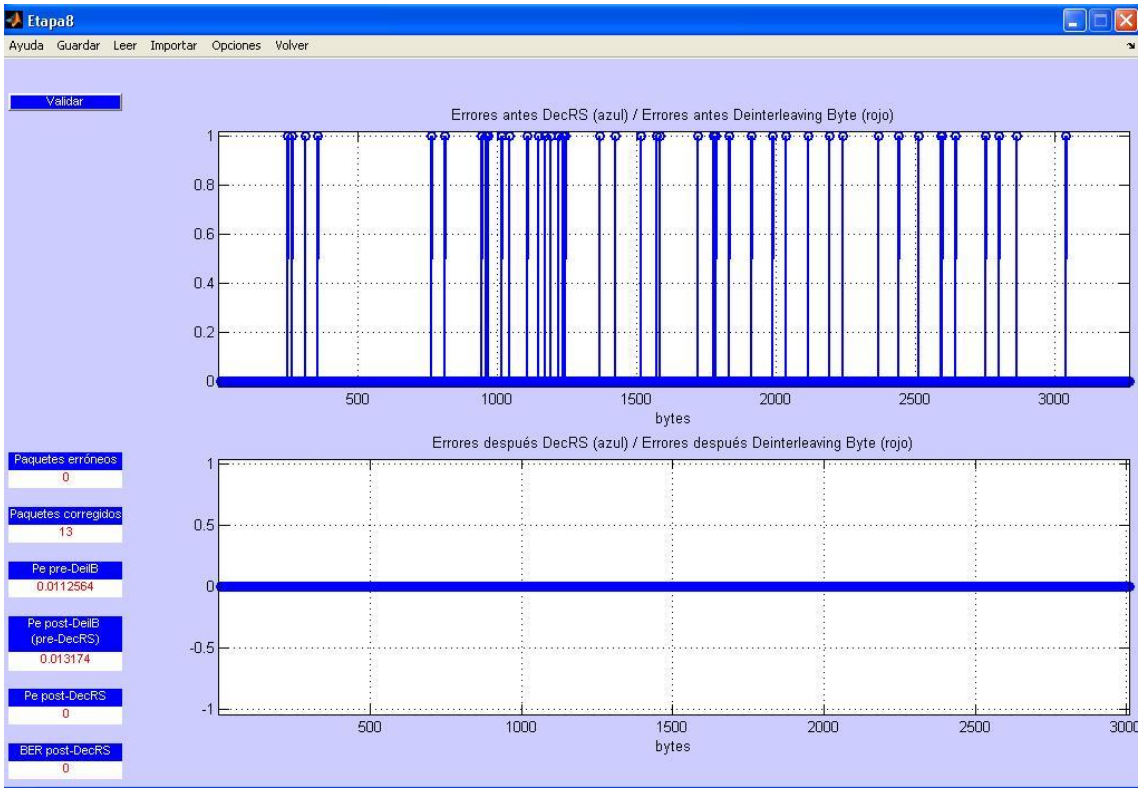


Figura 55 Ejemplo 16 paquetes MPEG-2 TS simulado en la etapa 8: Desentrelazado interno + Decodificador Reed-Solomon

De la observación de los resultados presentados en la Figura 55, pueden extraerse las siguientes conclusiones:

- ✓ Por un lado, las gráficas evidencian la presencia de errores aleatorizados a la entrada del decodificador Reed-Solomon. Tras el paso por el decodificador, se recupera la secuencia original sin errores. Se deduce que el número de errores presente en cada paquete recibido RS es menor que la capacidad correctora *t=8* bytes.

- ✓ El cálculo numérico de la probabilidad de error de byte evalúa la reducción de la P_e que proporciona el decodificador Reed-Solomon. En el ejemplo se observa como corrige por completo la secuencia de bytes.

- ✓ El resultado numérico del número de paquetes corregidos y erróneos, informa de la llegada al decodificador de 13 paquetes erróneos de los 16 transmitidos, y evidencia la corrección de todos ellos.

4.3.8.5 Retos de programación

Entre los retos de programación que los cálculos de procesado de señal han planteado en esta etapa, se destaca el decodificador Reed-Solomon.

4.3.8.5.1 Retos en la programación MATLAB

No se destaca ningún reto de programación en especial. Tal como se ha comentado, para la decodificación Reed-Solomon, se emplea la subrutina de procesado *rsdec* (función del *toolbox* de MATLAB) teniendo en cuenta los aspectos descritos en el apartado 4.3.2.5: Retos de programación de la etapa 2).

4.3.8.5.2 Retos en la programación Java

- ***Decodificador mediante Reed-Solomon***

El decodificador Reed-Solomon es un proceso largo y complejo que debe ser desarrollado en varios pasos. La idea principal es localizar la posición de los errores y sus valores, y con esta información reconstruir la palabra transmitida librándola de los mismos.

Los pasos seguidos por el algoritmo programado son los siguientes:

- Calcular las ecuaciones del síndrome

  Estas ecuaciones se obtienen evaluando la palabra codificada recibida con las raíces del polinomio generador. Con estas ecuaciones se detectan los errores.

- Usar el algoritmo de Berlekamp-Massey que encuentra el polinomio localizador.

  Es un polinomio con ceros en las posiciones inversas de error. Con este polinomio es posible localizar la posición de los símbolos erróneos.

- Encontrar, con el algoritmo de Forney, las magnitudes de los símbolos erróneos.

- Obtener el polinomio de errores y recuperar la información originalmente transmitida.

$$r(x) = n(x) + e(x) \tag{4.35}$$

En la ecuación (4.35) *r(x)* es la información recibida y *e(x)* el polinomio de errores.

El decodificador implementado, al contrario que la versión MATLAB, intenta corregir errores siempre, aunque haya más de los que pueda corregir.

Todos los pasos descritos se han programado siguiendo la referencia [40], pero se ha adaptado el cálculo del síndrome y el algoritmo de Forney por el hecho que dicha

referencia considera *b*=1, y no *b*=0 como es el caso del sistema DVB-T en la ecuación (4.5).

Para la adaptación se han consultado las referencias [41] y [42], de donde se obtiene la fórmula:

$$Y_j = X_j^{1-b} \frac{\Omega(X^{-1}_j)}{\Delta'(X^{-1}_j)} \tag{4.36}$$

Donde se destaca el parámetro X_j^{1-b}, que es la posición del error elevado a *1-b*, con lo que se deduce que si *b* es cero, como en el caso del sistema DVB-T implementado, se tiene la posición del error X_j multiplicando en el numerador.

4.3.8.6 Análisis de la etapa

En la última etapa del simulador, se destacan algunos aspectos sobre los resultados obtenidos.

El decodificador Reed-Solomon logra corregir, en ambas versiones del simulador, hasta 8 bytes (capacidad correctora). Ahora bien, si se producen más errores de los que puede corregir, en la versión MATLAB se deja el paquete erróneo intacto, mientras en la versión Java se intenta corregir. Esto provoca que en la versión Java se aumente, en la mayoría de los casos, el número de errores en el caso de que éste sea mayor que 8 a la entrada del decodificador.

Sobre el resultado numérico que muestra la probabilidad de error de byte después del desentrelazado externo, se observa que es un poco diferente de la P_e a la entrada del desentrelazador. Esto es debido a que, tras el desentrelazador de bytes, se eliminan los ceros de relleno añadidos en transmisión para realizar el entrelazado, y entonces se calcula la P_e sobre una secuencia de bytes de menor longitud.

- ***Análisis de las gráficas representadas***

Al principio de la presente etapa, igual que en la etapa anterior, se ha hecho un estudio teórico de lo que se esperaba visualizar en las gráficas representadas. Después de realizar la etapa, se ha comprobado el correcto funcionamiento.

Así, en la gráfica de errores antes del desentrelazador externo, se esperan algunos errores a ráfagas. En la gráfica después del desentrelazado externo, que coincide con la gráfica antes del decodificador RS, los errores mencionados pasan a estar en posiciones aleatorias.

Finalmente, en la gráfica de errores después del decodificador RS, aparecen menos errores. Si el número de errores de cada paquete a la entrada es menor que la capacidad correctora, se corrigen todos.

5 Simulador S_DVB MATLAB

Algunos de los applets que constituyen el soporte principal del proyecto LaViCAD presentado en el capítulo 3, se han programado como paso previo en MATLAB. Este es el caso del sistema DVB-T implementado en el presente proyecto.

MATLAB, que es el nombre abreviado de "MATrix LABoratory", es un lenguaje de computación técnica de alto nivel y un entorno interactivo para desarrollo de algoritmos, visualización de datos, análisis de datos y cálculo numérico. Con MATLAB, se pueden resolver problemas de cálculo técnico más rápidamente que con lenguajes de programación tradicionales, tales como C, C++ y FORTRAN, a cambio de mayor tiempo de computación.

La estructura que sigue el capítulo es la siguiente:

- ✓ Se describe, en primer lugar, la estructura de programación empleada como *programa plantilla* o *contenedor*, similar al empleado para Java (LaViCAD), para el desarrollo del simulador de la capa física del sistema DVB-T. En esta primera sección, también se dan las pautas a seguir en la programación MATLAB.

- ✓ En la siguiente sección, se explican las diferentes posibilidades para exportar la aplicación creada en MATLAB, como un ejecutable o como un componente software, con la finalidad de permitir su ejecución a usuarios que no disponen de MATLAB.

- ✓ En el apéndice A, al final del capítulo, se presentan las características del software MATLAB que han sido empleadas en el desarrollo del proyecto.

5.1 Programación MATLAB del proyecto

El primer reto importante del proyecto ha sido la programación del applet simulador de la capa física del sistema DVB-T en MATLAB. El resultado final se presenta en la Figura 56, en la que puede verse el desglose del sistema en distintas etapas a través de las cuales se va procesando la señal.

Figura 56 Pantalla principal del applet DVB-T programado en MATLAB (simulación finalizada)

Como se ha estudiado en el capítulo 4, en el sistema de comunicaciones implementado los cuatro primeros bloques o etapas corresponden al ***transmisor***. La etapa quinta corresponde al ***canal*** y las tres últimas etapas corresponden al ***receptor***. En la Figura 56 se muestra la pantalla principal del sistema tras el paso de los bits por todos los bloques.

Tal como se detalla más adelante en esta primera sección, cada etapa puede ser accionada por el usuario, y se avanza de forma secuencial. Una vez accionada, se muestra una nueva pantalla en la que se deben introducir diferentes parámetros de configuración de la etapa y validarlos, apareciendo así toda una serie de resultados correspondientes a las señales de interés de la etapa.

A continuación, se detalla la programación seguida en la implementación MATLAB del simulador DVB-T, en la que se usan las posibilidades de MATLAB que se presentan en el apéndice A del capítulo, y se estudia tanto la estructura del sistema programado, como las pautas seguidas para su realización.

5.1.1 Planteamiento inicial

La estructura de programación del sistema DVB-T en MATLAB se ha diseñado siguiendo el estilo del sistema completo de modulaciones QAM (Quadrature and Amplitude Modulation) desarrollado por Margarita Cabrera, co-directora del proyecto. Dicho estilo se puede utilizar como *programa plantilla o contenedor*, similar al creado para Java (LaViCAD), tanto para la simulación de un sistema completo de comunicaciones como para la programación de las pantallas de cada etapa.

La estructura principal puede verse en la Figura 57.

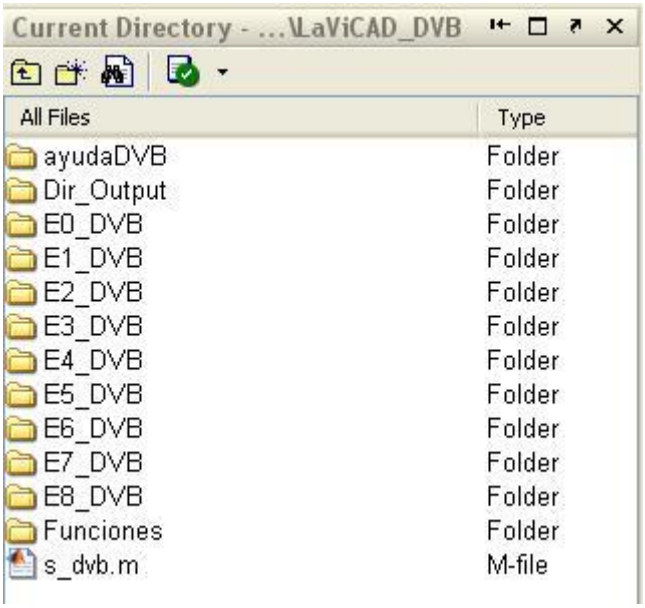


Figura 57 Estructura de programación S_DVB MATLAB

Se puede observar que se ha seguido una estructura de carpetas, acompañadas del fichero *s_dvb.m*, el cual es el script encargado de cargar la aplicación. La carpeta E0_DVB es la que contiene la interfaz principal, y el resto de carpetas E1_DVB,…, E8_DVB contienen las etapas del sistema. Por último están las carpetas Dir_Output, para guardar sesiones y señales de nuestras simulaciones, Funciones, donde se ubican las funciones comunes de todas las etapas, y ayudaDVB donde se guardan todos los archivos que proporcionan la ayuda del sistema.

Para realizar la carga del sistema, basta con llamar al script desde la línea de comandos de MATLAB con la instrucción *s_dvb*. Éste llama a su vez a la interfaz principal del simulador, *DVB_Interficie*, disponible en la carpeta E0_DVB. Previamente a esta llamada, se prepara todo lo necesario para comenzar una nueva ejecución del simulador (limpieza del *workspace*, cierre de *figures* y creación de *paths).*

```
if exist('fig_DVB','var')==0
clear
close all
Dir_Principal=cd;

path(path,[cd,'\Funciones']);
path(path,[cd,'\E0_DVB']);
path(path,[cd,'\E1_DVB']);
path(path,[cd,'\E2_DVB']);
…
[V_estados,fig_DVB]=DVB_Interficie;
end
```

Figura 58 Aspecto del script s_dvb

5.1.2 Interfaz principal

La interfaz principal del simulador S_DVB programado en MATLAB, se halla en la carpeta E0_DVB que puede verse en la Figura 57. Dicha carpeta contiene ficheros *.m encargados de la interacción del usuario con esta interfaz principal, y de la representación de la misma.

Para la representación de ésta, se han usado las posibilidades que ofrece MATLAB para crear interfaces que se explican en la sección A.2 del apéndice A del capítulo5.2.2A.2. En la Figura 59 se puede ver el principal código de la función DVB_Interficie, encargada de la apariencia del sistema.

```
function  [V_estados,fig_DVB]=DVB_Interficie
fig_DVB=figure('Color',[0.8 0.8 1],...
   'Name','Sistema DVB',...
   'Numbertitle','off',...
   'MenuBar','none',...
   'Pointer','arrow',...
   'Resize','off',...
   'Units','Normalized',...
   'position',[0.002 0.04 0.996 0.91]);

uicontrol('Style','Frame',...
   'Units','Normalized','Position', [coordenadas],...
    'BackgroundColor',[0 0 1],...
    'ForegroundColor','w',...
    'HorizontalAlignment','Center');

Etapa1_Input=uicontrol('Style','Pushbutton',...
    'Units','Normalized',...
    'Position',[Ini_hor,Ini_vert,Ancho_horiz,Ancho_vert],...
    'Background','w',...
    'FontSize',10,...
    'FontWeight','Bold',...
    'ForegroundColor','k',...
    'CallBack',...
    ['if exist(''Variables_E1'')==1,',...
        'E1_Interficie(Variables_E1),',...
        'else,',...
        'E1_Interficie,',...
        'end']);
…
uicontrol('Style','Frame',...
    'Units','Normalized',...
    'Position',  [coordenadas],...
    'BackgroundColor',[0 0 1],...
    'ForegroundColor','w',...
    'HorizontalAlignment','Center');

Exit_DVB=uicontrol('Style','Pushbutton',...
    'Units','Normalized',...
    'Position',[Ini_hor,Ini_vert,Ancho_horiz,Ancho_vert],...
    'String','EXIT',...
    'Background','w',...
    'ForegroundColor','r',...
    'FontSize',10,...
    'FontWeight','Bold',...
    'CallBack','Close_DVB');
```

Figura 59 Función DVB_Interficie

En este código se puede observar la configuración de la *figure* principal, así como de las sucesivas etapas hasta llegar al *uicontrol* mediante el cual se tiene la opción de salir del simulador.

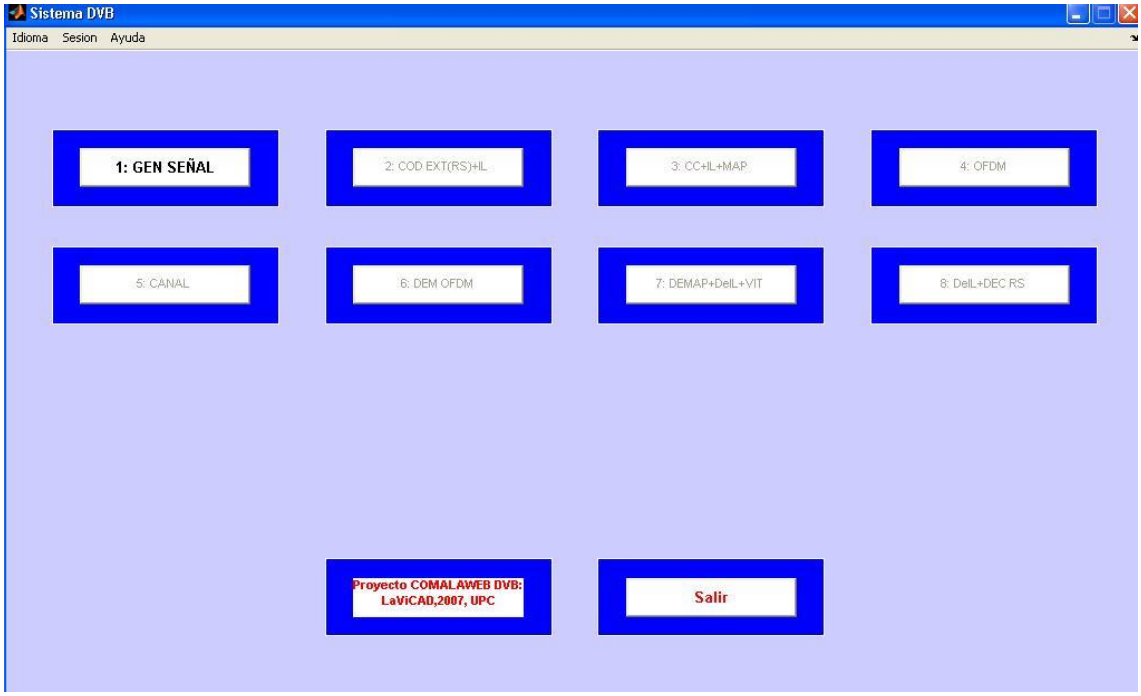


Figura 60 Pantalla principal del applet DVB-T programado en MATLAB (comienzo de la simulación)

Cabe recordar la filosofía que siguen los applets de LaViCAD, ya sean programados en Java o MATLAB. Como se ha visto en el capítulo 4 en el caso del simulador DVB-T implementado, cada sistema se halla formado por una concatenación de etapas a través de las cuales se va procesando la señal. Cada etapa se ejecuta en tres partes: configuración de parámetros, operaciones de procesado de señal y presentación de resultados. Este tipo de procesado de señal secuencial permite reutilizar los algoritmos de la aplicación en general y el software en particular para cada uno de los lenguajes de programación utilizados.

En la Figura 60 se puede ver la pantalla principal del programa. Se muestran inactivas las etapas a las cuales aún no se ha accedido. En negrita se pueden observar las etapas ya simuladas, siendo la última etapa resaltada la siguiente a simular. En éste caso el flujo de ejecución del simulador nos llevaría a simular la etapa 1: GEN SEÑAL. Una vez situados en este punto, el usuario tiene a su disposición diversas opciones para interactuar con el simulador.

5.1.2.1 Idioma

Una de las opciones visibles en el menú superior de nuestro simulador, es la selección del idioma. Actualmente se dispone del sistema en castellano, catalán e inglés. Dentro del fichero *DVB_Interficie.m* se halla, entre otros, el código de la Figura 61.

```
Id_Language=uimenu('Label','Idioma');

uimenu(Id_Language,'Label','Castellano','CallBack','Idioma_DVB(1)');

uimenu(Id_Language,'Label','Catala','CallBack','Idioma_DVB(2)');

uimenu(Id_Language,'Label','English','CallBack','Idioma_DVB(3)');
```

Figura 61 Menús de la User Interface de MATLAB para los idiomas (A.2.3)

Se puede ver que tiene la propiedad de multiidioma, es decir, está preparado para poder incorporar un número indeterminado y configurable de idiomas. Simplemente bastaría con añadir tantos elementos *uimenu* como idiomas se deseen.

Al final del fichero *DVB_Interficie.m*, se hace una llamada a la función *Idioma_DVB*, pasándole como idioma el castellano, que es el que saldrá por defecto.

```
function Idioma_DVB(Idioma)

if Idioma==1
    Text_Etapa1='1: GEN SEÑAL';
    …
    Simulador_name='Sistema DVB';
    Text_Sesion='Sesion';
    …
    Text_Language='Idioma';
end
if Idioma==2
    …
end
…
```

Figura 62 Función Idioma_DVB

En la Figura 62, se pueden ver las primeras líneas de código de la función *Idioma_DVB* encargada de asignar el texto de las etiquetas de todas las etapas y demás elementos de la interfaz principal del simulador S_DVB.

5.1.2.2 Sesiones

La segunda opción del menú superior del simulador, corresponde al manejo de las sesiones. El usuario, tiene la opción de cargar una sesión previamente guardada, o bien, puede guardar una sesión para cargarla más adelante.

En la sección A.2 se explica, entre otras características de MATLAB, el uso de los comandos *save* y *load* para guardar y cargar sesiones. Éstos han sido de gran utilidad para la programación de esta característica del simulador.

```
Id_Session=uimenu('Label','Sesion');
Id_L=uimenu(Id_Session,'Label','Cargar','CallBack','Load_Session');
Id_S=uimenu(Id_Session,'Label','Guardar','CallBack','Save_Session');
```

Figura 63 Menús de la User Interface de MATLAB para las sesiones

En la Figura 63, se puede ver la parte de *DVB_Interficie* encargada de mostrar el menú con el cual puede interactuar el usuario para cargar o guardar sesiones.

A partir de la selección del usuario, se hace la llamada al script *Load_Session.m*, o bien, *Save_Session.m*. Ambos scripts utilizan la función "*nom_session*", alojada en la carpeta Funciones para *cargar* o *guardar* la sesión, respectivamente.

Los dos casos se resumen a continuación:

- ***Guardar sesión***

Cuando el usuario selecciona la opción guardar sesión, el sistema solicita al usuario que seleccione un nombre y ubicación donde desea guardar el fichero de sesión y guarda en el ordenador del usuario dicho fichero.

- ***Cargar sesión***

Cuando se selecciona esta opción, el sistema solicita al usuario que seleccione el fichero que desea cargar y sitúa al usuario en la etapa correspondiente de la simulación previamente guardada.

En la realización Java del proyecto, explicada en el capítulo 6 de la memoria, también se ha utilizado esta funcionalidad del aplicativo Contenedor programado por Carlos y Francisco Vargas en el proyecto LaViCAD. La diferencia es que en Java se guardan los parámetros de entrada de cada etapa y se procede a la ejecución del sistema con dichos parámetros cuando se carga una etapa, así como en MATLAB directamente se cargan todos los parámetros y variables que se habían guardado sin necesidad de ejecutarse el sistema etapa a etapa.

5.1.2.3 Ayuda del sistema completo

La tercera y última opción del menú principal del simulador es la ayuda del sistema. El usuario podrá acceder desde la página principal a una ayuda del sistema global donde se explicará el uso general del sistema DVB-T, o del sistema que corresponda, y temas específicos del sistema implementado en cada caso.

El sistema de ayudas del simulador se halla implementado mediante archivos HTML. Se tiene un archivo HTML principal para el sistema global (*E0_Castellano.html*), y luego uno por etapa (*E1_Castellano.html*). Con lo cual, estas ayudas estarán tanto a nivel de sistema global como a nivel de etapa individual y cada una podrá tener una estructura diferente.

Cada archivo general llamará a su vez a subarchivos HTML. En el caso del sistema global estos subarchivos serán *E0_Castellano_acercade.html*, *E0_Castellano_Menu.html* y *E0_Castellano_Sistema.html*. Aparte podrán tener archivos HTML específicos del sistema en cuestión. En el caso de las etapas los subarchivos serán *E1_Castellano_Menu.html*, *E1_Castellano_Parametros.html*, *E1_Castellano_Resultados.html* y *E1_Castellano_Teoria.html*.

Cada archivo de los nombrados tendrá su equivalente en catalán e inglés. El formato será igual, pero se cambiará el idioma del contenido. Todos estos archivos, se guardan en la carpeta ayudaDVB que se puede ver en la Figura 57. Además de estos archivos HTML, dentro de la carpeta ayudaDVB se aloja un archivo llamado *estilo.css* que se encarga de aspectos de la representación de los archivos de ayuda, y también se aloja una carpeta llamada *img* que contiene todas las imágenes que aparecen en las páginas de ayuda.

El código que implementa el menú seleccionable de ayuda se encuentra, como los demás, en la función *DVB_Interficie*.

```
Id_Help=uimenu('Label','Ajut','CallBack','En_Ayuda');
```

Figura 64 Menú ayuda del sistema DVB

Cuando el usuario selecciona la ayuda, el sistema llama al archivo E0_Ayuda. Su estructura se puede ver en la Figura 65. Dependiendo del idioma escogido por el usuario se llamará a una página HTML u otra.

```
function En_Ayuda

Idioma=get(gcf,'UserData');
Idioma=Idioma(11);

if Idioma==1
    web 'E0_Castella.html';
end
if Idioma==2
    …
```

Figura 65 En_Ayuda

La página de ayuda que el usuario verá, es como la que se ve en la Figura 66 para el caso del sistema QAM.

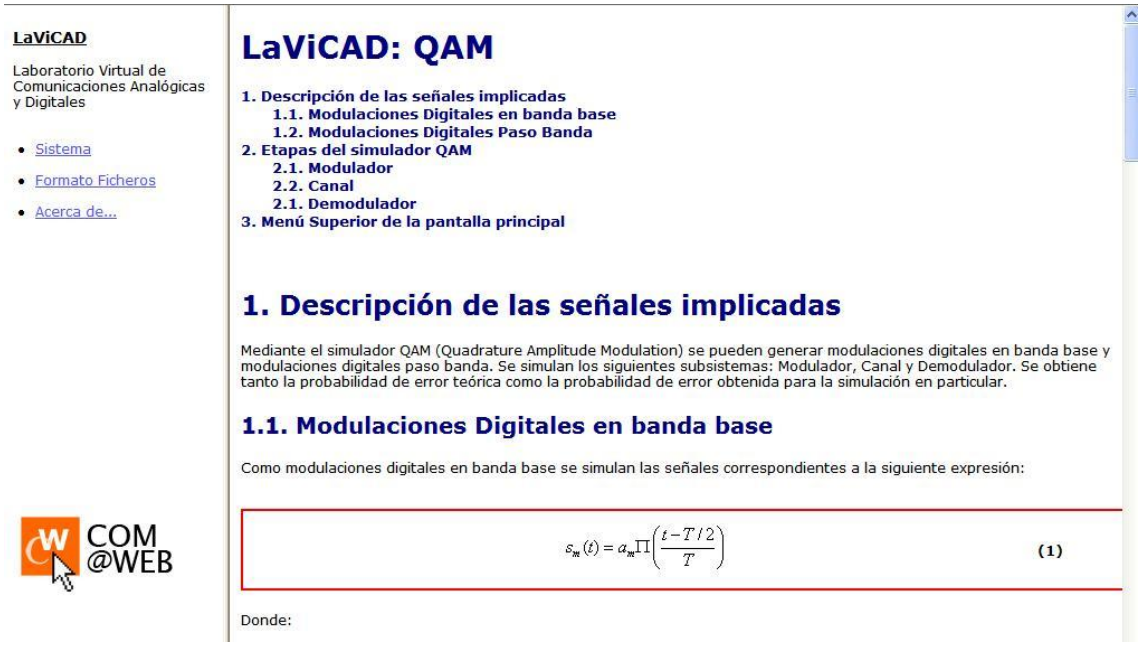


Figura 66 Ayuda sistema QAM

Como puede verse en la Figura 66, en la ayuda del sistema global se puede encontrar información del sistema implementado (en este caso QAM), acerca de…y casos particulares del sistema como el formato de los ficheros intercambiables.

5.1.2.4 Mensajes de error

Otra de las funciones alojadas en la carpeta E0_DVB es la de las alertas de error.

```
function Mensajes_Error(Idioma)
if Idioma==1
mensaje='Error en parametros de entrada, se han modificado
valores';
    figname='Atencion';
end
if Idioma==2
   …
uiwait(msgbox(mensaje,figname,'modal'));
```

Figura 67 Mensajes_Error.m

En la Figura 67, se puede ver la estructura de la función *Mensajes_Error*, la cual se puede llamar desde cualquier parte del programa con el idioma correspondiente y dará una alerta al usuario de que alguna interacción que ha realizado no se ha efectuado correctamente.

Los casos en los que puede saltar una alerta de error sería en la lectura incorrecta de una señal, la carga de una sesión que no se guardó correctamente, intentar cargar una señal en la etapa que no corresponde, etc.

5.1.2.5 Salir del sistema

Por último, se puede encontrar el script encargado de la salida del sistema, *Close_DVB.m.*

En la Figura 59, se puede ver el código de *DVB_Interficie* encargado de mostrar el botón Salir. Una vez pulsado ese botón el sistema ejecuta el script *Close_DVB.m*. Este script cierra la *figure* principal, elimina los *paths* de las etapas y limpia las variables creadas durante la ejecución de S_DVB.

5.1.3 Estructura de las etapas

La simulación se desarrolla de forma secuencial en 8 etapas. La simulación completa se realiza accionando desde la ventana o interfaz de usuario inicial (Sistema DVB) secuencialmente cada una de las 8 etapas. En las páginas de ayuda correspondientes a cada una de las etapas se facilita más información detallada.

Para la ejecución de cada etapa, se abre una ventana o interfície específica. Una vez generada la ventana se deben configurar los parámetros de etapa, cambiando los que se proponen por defecto o dejando los propuestos por el simulador. Una vez elegidos, mediante el botón "Validar" se da paso a los algoritmos de procesado específicos de la etapa y aparecen en la misma ventana los resultados tanto gráficos, como numéricos correspondientes. A partir de este punto, por el menú superior de la ventana se permiten realizar diferentes acciones, como las que se destacan a continuación.

- Visualizar más resultados gráficos.
- Ampliar una gráfica en una ventana independiente.
- Guardar Señales Generadas.
- Cambiar las Señales Generadas por otras que se hayan almacenado previamente en un fichero, según el formato requerido.
- Salir de la etapa en particular y volver al sistema general.

La estructura de las etapas sigue un esquema común para cada una de ellas. Para cada etapa (de E1_DVB a E8_DVB) se dispone de los siguientes archivos:

5.1.3.1 Interfície

Para la etapa n, el archivo sería *En_Interficie.m*. Dentro de este archivo, se halla programada la interfaz gráfica de la etapa en cuestión.

En el flujo de ejecución del sistema, cuando un usuario pulsa el botón de una etapa (*1: GEN SENYAL*, por ejemplo) se realizan los siguientes pasos:

- ✓ Se invoca a la función *En_Interficie*, para la etapa n.
- ✓ Ésta llama a su vez a *En_Idioma* (5.1.3.2), para cargar las etiquetas de la etapa en ejecución.
- ✓ Una vez cargado el idioma, *En_Interficie* presenta la pantalla inicial de la etapa n.

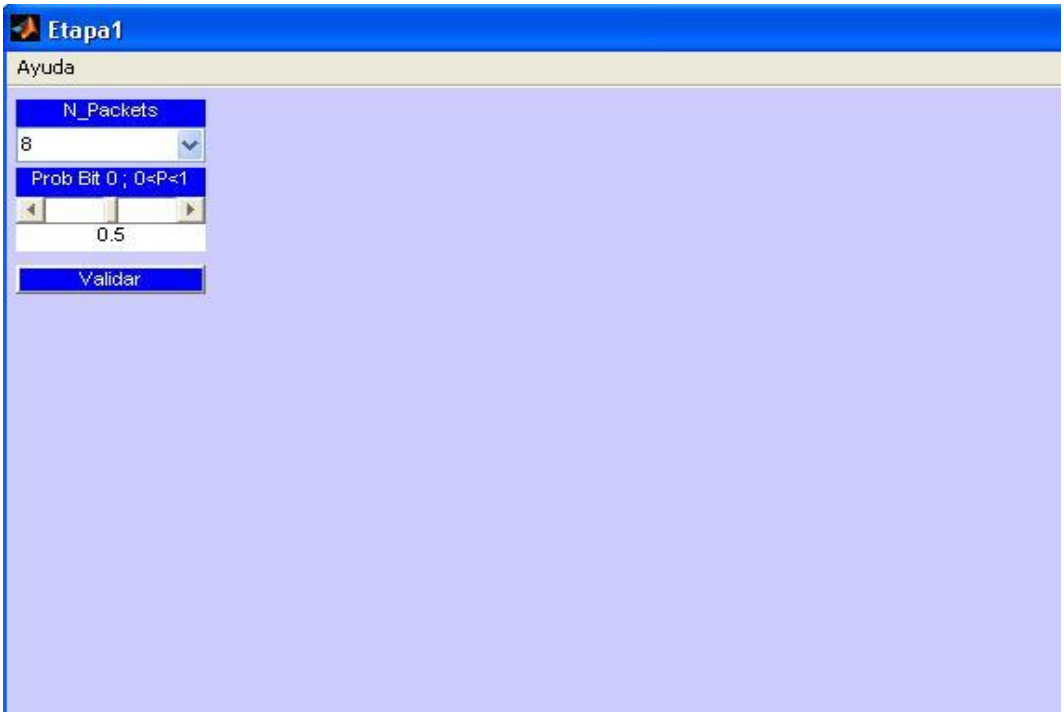


Figura 68 Interfaz principal de la etapa 1 del sistema DVB-T

Tal como se puede ver en la Figura 68, esta interfaz muestra los parámetros de entrada y el botón *validar* para proceder a la ejecución de la etapa.

Para los parámetros de entrada se han definido previamente unos valores por defecto, que son los valores que ve el usuario cuando entra en una etapa. Una vez aquí, el usuario puede modificar a su gusto dichos valores. Si el usuario vuelve a una etapa ya ejecutada, el sistema pasará a mostrar los valores que había escogido el usuario previamente. Esto se consigue gracias a la variable *nargin*, explicada en A.2.5.

Aparte, en *En_Interficie* se han preparado los elementos correspondientes a los textos de resultados, los ejes de las gráficas y el menú superior de la etapa en ejecución. Inicialmente, en la primera ejecución son declarados como invisibles, para posteriormente hacerlos visible con la función *Validar*, explicada en 5.1.3.3, que se ejecuta cuando el usuario pulsa el botón Validar de la etapa.

5.1.3.2 Idioma

Para la etapa n, el archivo sería *En_Idioma.m*, es invocado por *En_Interficie* para cargar las etiquetas de la etapa en el idioma escogido por el usuario.

Tal como se ve en la Figura 69, la función se encarga de devolver los parámetros definidos por el programador en el idioma pasado por parámetro.

```
function [Nom_fig,Nom_NPackets,…]=E1_Idioma(Idioma)

if Idioma==1
    Nom_fig='Etapa1';
    Nom_NPackets='N_Packets';
    …
end
if Idioma==2
…
```

Figura 69 Función E1_Idioma

5.1.3.3 Validar

Para la etapa n, el archivo sería *En_Validar*. El nombre ha sido escogido porque se encarga de validar los parámetros de entrada, viendo que estén dentro del rango correcto y maneja el flujo de ejecución de la etapa.

Se puede invocar la función *En_Validar* con dos objetivos principales. Uno, para actualizar parámetros de entrada, como el movimiento de una *scroll* que simboliza una probabilidad de error o el dejar visibles ciertos parámetros de entrada dependientes de otros.

El segundo motivo por el cual es invocado, es para proceder a la ejecución de la etapa cuando el usuario clica en el botón Validar. Se realizan los siguientes pasos:

- ✓ El usuario hace click en el botón Validar.
- ✓ La función *En_Validar* es invocada con los parámetros definidos en *En_Interficie.*
- ✓ Se recuperan los parámetros de entrada escogidos por el usuario, y las variables de etapas anteriores.
- ✓ A continuación se hacen visibles tanto los resultados de la etapa como el menú superior.
- ✓ En este punto se realiza la llamada a la función *En_Procesado*, con algunos cálculos previos propios de la etapa simulada.
- ✓ Las variables que devuelve *En_Procesado* se guardan en *Variables_En*, que son las variables que se mantienen mientras se realiza la etapa n.

El aspecto de una etapa cualquiera, después de pulsar el botón Validar, es el mostrado en la Figura 70.

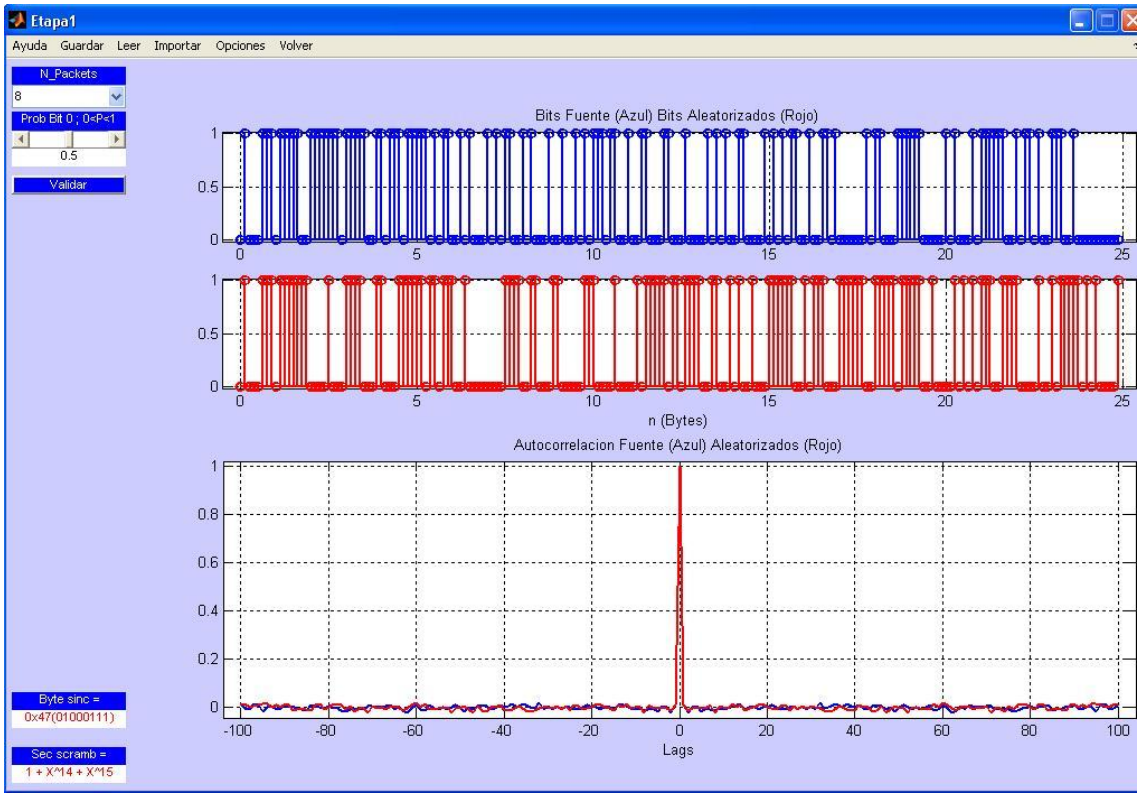


Figura 70 Etapa 1 sistema DVB-T

5.1.3.4 Procesado

Para la etapa n, el archivo sería *En_Procesado.* Su función consiste principalmente en realizar el procesado de la señal para la etapa correspondiente.

Su llamada siempre se hace desde la función *En_Validar*, la cual pasa a la función de procesado los parámetros necesarios para el tratamiento de la señal que se va a realizar. Estos parámetros suelen ser la señal de salida de la etapa anterior con los parámetros de entrada de la etapa correspondiente. Aparte de estos parámetros, harán falta otros dependiendo de la etapa (parámetros de entrada de la etapa anterior, señal intermedia de otra etapa para cálculo de errores…).

La función de procesado, a su vez, puede llamar a ciertas funciones específicas de cada etapa (5.1.3.8), las cuales se encargarán de realizar una subrutina de procesado.

Después de realizar el procesado de la señal, la función se encarga de presentar las gráficas y los resultados numéricos de la etapa.

En lo referente a las gráficas, la función de procesado prepara todas las gráficas que están disponibles en la etapa, dejando visibles las que se desean ver inicialmente, e invisibles el resto. También se tiene la opción de dejar deshabilitadas ciertas opciones del menú superior de la etapa dependiendo de los parámetros escogidos por el usuario para la simulación.

En la Figura 70 se pueden ver posibles gráficas que pueden representarse. La posición de las gráficas y las etiquetas a representar, se preparan dentro de *En_Interficie*, y los

ejes “x” e “y” se dibujan en *En_Procesado*, haciendo uso de funciones propias de MATLAB, como *line*, y otras que se han preparado para este tipo de sistemas (5.1.4.1).

En cuanto a los resultados numéricos, la función de procesado calcula el parámetro en cuestión y lo muestra en su etiqueta correspondiente. En la Figura 70, se pueden ver estos resultados en la parte inferior izquierda. Pero, en este caso, al tratarse de resultados que no requieren ningún cálculo por parte del procesado de señal de la etapa en cuestión, los valores se rellenan directamente dentro de *En_Interficie*.

Después de realizar todas sus funciones, la subrutina de procesado devuelve a *Validar* la señal a pasar a la siguiente etapa, los *handles* de las gráficas para tratarlos en *Opciones* según se explica en el siguiente apartado, y demás señales y datos de interés para ser guardados.

5.1.3.5 Opciones

Para la etapa n, el archivo sería *En_Opciones.* Se llama desde el menú superior para ejecutar todas las correspondientes funcionalidades.

Su función consiste en hacer visibles e invisibles las gráficas preparadas por la rutina de procesado dependiendo de la elección del usuario, a través de los *handles* (identificadores) de las mismas guardados por la función Validar como propiedad de la gráfica correspondiente a la etapa actual.

Su llamada se realiza desde el menú superior *opciones*, que se puede ver en la Figura 70, con el cual el usuario puede escoger la gráfica que desea visualizar. En el caso de haber una cantidad considerable de opciones, se habilitan dos elementos opciones en el menú superior, como se puede ver en la Figura 72. En este caso, se usa Opciones1 para las gráficas del dominio frecuencial, representadas en la parte superior. Opciones2 se utiliza para seleccionar la gráfica del dominio temporal que deseamos visualizar.

Dicho menú se crea desde *En_Interficie*, con la siguiente estructura:

```
Id_opciones=uimenu('Label',Nom_Id_opciones,'Visible','Off');
Id_opciones_1=uimenu(Id_opciones,'Label',Nom_Id_opciones_1,'Visible'
,'Off','CallBack','En_Opciones(1)');
…
```

Figura 71 Menú superior Opciones

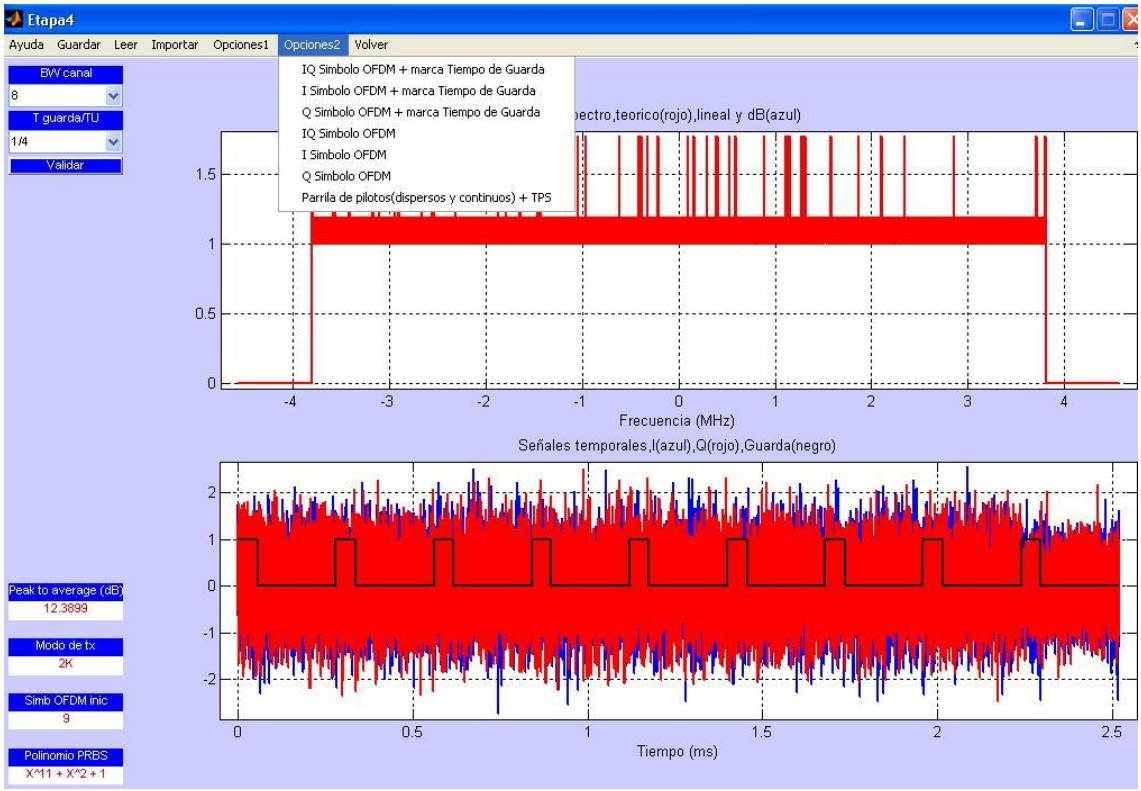


Figura 72 Etapa 4:OFDM sistema capa física de DVB-T

5.1.3.6 Cierra

Para la etapa n, el archivo será *En_Cierra*. Se ejecuta al salir de etapa, devuelve a la interfaz principal y gestiona transferencia de variables entre etapas.

Su llamada la realiza el usuario cuando desea salir de la etapa simulada para pasar a la siguiente, y selecciona *volver* en el menú superior de la etapa.

```
Id_close=uimenu('Label',Nom_Id_close,'Visible','Off','Callback','En_Cierra');
```

Figura 73 Menú superior volver

Como se puede ver en la Figura 73, se realiza la llamada a *En_Cierra*, script que realiza las siguientes tareas:

- ✓ Elimina todas las ventanas que se hayan creado en la simulación de la etapa en curso al hacer importación de gráficas.
- ✓ Guarda las variables que contienen los parámetros de entrada de la etapa en curso para poderlas cargar en caso de volver a entrar en la etapa.
- ✓ Elimina las variables que ya no se necesitan.
- ✓ Hace visible la interfaz principal del sistema, situando la atención del usuario en la siguiente etapa a simular.

5.1.3.7 Ayuda de la etapa

Para la etapa n, el archivo será *En_Ayuda*. El usuario podrá acceder desde una etapa concreta a su ayuda donde se explicará el uso particular de dicha etapa.

El código que implementa el menú de ayuda se encuentra en *En_Interficie.* Una vez el usuario pulsa en el botón, la etapa llama a la función *En_Ayuda*, la cual abre una nueva ventana con su ayuda. Dentro de esta ventana el usuario podrá navegar por una serie de páginas de ayuda.

El funcionamiento es el mismo que el de la ayuda del sistema global explicado en 5.1.2.3. En cuanto a la manera de crear el menú superior de ayuda y la estructura de *En_Ayuda*, se puede ver su forma en la Figura 64 y Figura 65 respectivamente.

A continuación, puede verse la ayuda de la etapa 1 del sistema QAM.

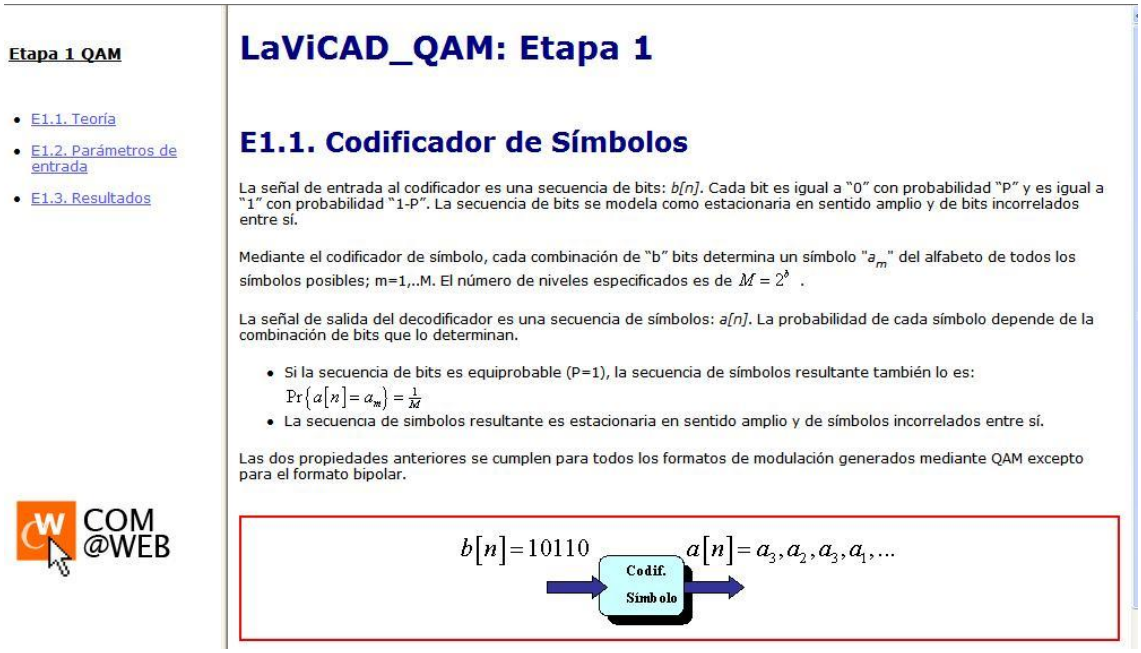


Figura 74 Ayuda etapa 1 sistema QAM

Como puede verse en la Figura 74, la página de ayuda de una etapa tendrá información de la teoría de la etapa (procesado de señal), de los parámetros de entrada de la etapa y también ayudará a interpretar los resultados de la etapa a simular.

5.1.3.8 Funciones específicas de cada etapa

Tal como se ha comentado anteriormente, cada etapa dispone de funciones específicas que solo se utilizan en dicha etapa. Por lo tanto, junto a los archivos *Validar.m, Interficie.m, Procesado.m…*en cada etapa se hallarán ciertas funciones como *c_rayleigh.m, g_ruido.m, hq.m…*cuya función consiste en realizar una subrutina de la función *En_Procesado* de la etapa en cuestión.

Estas funciones, reciben los parámetros correspondientes para realizar el tratamiento de la señal necesario, y devuelven los resultados de dicho tratamiento. Cabe mencionar que ha sido de gran utilidad la programación de estas subrutinas aparte de la subrutina de procesado principal de la etapa, puesto que permiten dejar dicha función menos cargada de código y más legible de cara al programador. De este modo, al visualizar la subrutina de procesado, el programador puede analizar rápidamente que procesado de señal principal se realiza en la etapa.

5.1.4 Funciones generales

Volviendo a la Figura 57, donde se puede ver la estructura seguida en la programación MATLAB, solo falta por estudiar la carpeta Funciones. Dentro de esta carpeta, se encuentran las funciones comunes de todas las etapas del sistema simulado. Estas se pueden separar según su ámbito de uso en funciones gráficas y de ficheros.

5.1.4.1 Funciones gráficas

Dentro de esta categoría, se hace referencia a todas las funciones generales del sistema que se utilizan para la representación y manipulación de las gráficas de las etapas.

- ***Importa_grafico***

Esta función se encarga de importar en una ventana nueva la gráfica escogida por el usuario.

Su llamada se realiza desde el menú superior "Importar", que se puede ver en la Figura 70. Dicho menú es implementado, como los demás, en la función *En_Interficie*, y su forma en la siguiente:

```
Id_importar=uimenu('Label',Nom_Id_importar,'Visible','Off');
Id_importar_G1=uimenu(Id_importar,'Label',Nom_Id_Importar_1,'Visible
','Off','CallBack',...
 ['if exist(''Borrador'')==0,',...
        'Borrador=[];,',...
        'end,',...
        'Borrador=importa_grafico(1,1,Borrador);,']);
Id_importar_G2=…
```

Figura 75 Menú superior "Importar"

La cabecera de la función es *Borrador=importa_grafico(grafica,alfa,Borrador).* Los parámetros de entrada son:

i. *Gráfica*: número identificador de la gráfica a importar.
ii. *Alfa*: parámetro empleado para definir la posición de la ventana nueva con la gráfica importada.
iii. *Borrador*: array Borrador, la cual contiene todas las importaciones de gráficas realizadas en la etapa.

Como parámetro de salida, la función *Importa_grafico* devuelve la array Borrador, que consta del parámetro Borrador enviado, añadiéndole la nueva gráfica importada.

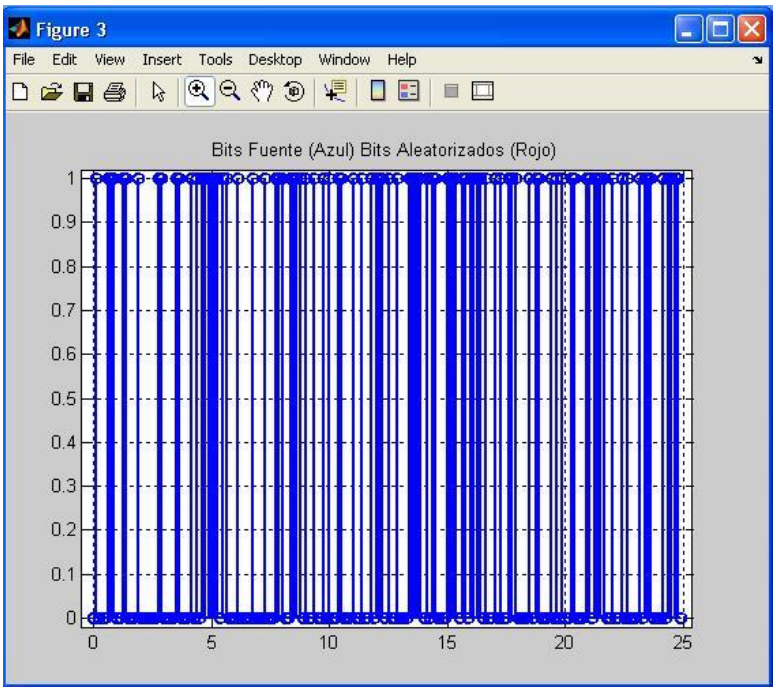


Figura 76 Ventana de gráfico importado

Tal como se puede ver en la Figura 76, una vez importada una gráfica se pueden realizar una serie de acciones (zoom, copia, edición, etc) proporcionadas por el menú de la nueva ventana.

- ***Stem_children***

Se trata de la función empleada para el dibujo de secuencias, de bits o de bytes, como la que se ve en la Figura 76.

Su cabecera es *Children=stem_children(Eje,yy,color,symbol,size).* Los parámetros de entrada son:

i. *Eje*: vector de abscisas
ii. *yy*: vector de ordenadas
iii. *Color*: color de la gráfica
iv. *Symbol*: símbolo elegido para el dibujo
v. *Size*: tamaño de los símbolos

Como parámetro de salida, la función *Stem_children* devuelve el handle o identificador de la gráfica.

- ***Images_children***

Función utilizada para gráficas especiales, como la parrilla de pilotos de la etapa OFDM, o el *interleaver* convolucional de la etapa 2. Rellena el gráfico con un sistema de colores, dependiente de la matriz que tiene como parámetro de entrada. Cada número de la matriz de entrada, se asociará a un color.

Su cabecera es *Children=images_children(A).* El parámetro de entrada *A,* es una matriz preparada específicamente para esta representación.

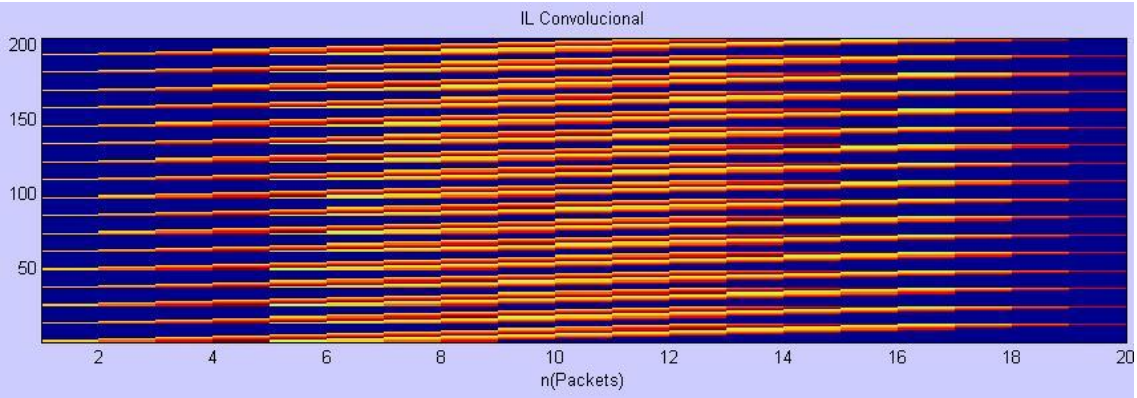


Figura 77 Representación con images_children del interleaver convolucional

Esta función utiliza la subrutina de MATLAB *patch*.

- ***Camlimx***

Función utilizada para cambiar los límites de abscisas de una gráfica, para ajustarlos a lo que hay representado con una pequeña ampliación.

Su cabecera es *camlimx(h,fac).* Los parámetros de entrada son:

i. *h*: *handle* de la gráfica, por defecto *gca (*handle de la gráfica actual). También puede ser un vector.
ii. *fac*: factor de exceso en el eje x. Recomendado 0.02 para dejar un poco de margen a ambos lados.

Normalmente se combina el uso de esta función con el de *camlimy*.

Función programada por el profesor del departamento de Teoría de la Señal y Comunicaciones (TSC) Antoni Gasull.

- ***Camlimy***

Función utilizada para cambiar los límites de ordenadas de una gráfica para ajustarlos a lo que hay representado con una pequeña ampliación.

Su cabecera es *camlimy(h,tipo,visi,fac).* Los parámetros de entrada son:

i. *h*: *handle* de la gráfica, por defecto *gca (*handle de la gráfica actual). También puede ser un vector.
ii. *Tipo*: 1, sólo límite inferior; 2, sólo límite superior y 3, hace la ampliación de los límites independientemente de las curvas representadas
iii. *Visi*: 1, sólo considera las líneas de las gráficas que son visibles. Por defecto las considera todas.

iv. *Fac*: factor de exceso en el eje y. Por defecto 0.02.

Normalmente se combina el uso de esta función con el de *camlimx*.

Función programada por el profesor del departamento de Teoría de la Señal y Comunicaciones (TSC) Antoni Gasull.

- ***Lim_se***

Función utilizada para cambiar los límites de ordenadas de una gráfica para hacer una pequeña ampliación (2%) y forzar que sea igual escala x e y. Sólo se utiliza para representar el espacio de señal.

Su cabecera es *lim_se(h).* El parámetro de entrada es *h, handle* de la gráfica, por defecto *gca (*handle de la gráfica actual).

Función programada por el profesor del departamento de Teoría de la Señal y Comunicaciones (TSC) Antoni Gasull.

5.1.4.2 Funciones de ficheros

En esta clasificación se encuentran todas las funciones comunes que intervienen en la manipulación de ficheros.

Para cargar y guardar sesiones, se dispone de la siguiente función:

- ***Nom_session***

Esta función es invocada tanto por Load_Session como por Save_Session, para cargar y guardar una sesión respectivamente.

Su cabecera es *File_name=nom_session(i_session).* El parámetro de entrada es *i_session,* el cual se utiliza para diferenciar si se desea cargar o guardar sesión. En el caso de que dicho parámetro valga 1, se carga la sesión que escoja el usuario. En el caso de que no tenga parámetro de entrada, se guarda la sesión actual en el directorio indicado por el usuario.

En cuanto a la lectura y escritura de señales se dispone de las siguientes funciones, las cuales siguen el formato de escritura e intercambio de ficheros descrito en el capítulo siguiente (6.1.1.3.1):

- ***Lee_signal***

Función utilizada para cargar una señal previamente guardada. Con esta función se guarda, como señal de salida de la etapa en cuestión, la señal cargada mediante fichero, siempre y cuando las longitudes coincidan con la simulación realizada. Si no coinciden (se ha elegido simulación de 8 paquetes MPEG-2 TS, y el fichero contiene 16), saltará un mensaje de error.

Su llamada se realiza desde el menú superior "Leer" que se puede ver en la Figura 70. Dicho menú es implementado, como los demás, en la función *En_Intercicie*, y su forma en la siguiente:

```
Id_load=uimenu('Label',Nom_Id_load,'Visible','Off');
Id_load_S1=uimenu(Id_load,'Label',Nom_Id_Load_1,'Visible','Off','Cal
lBack','lee_signal(1)');
…
```

Figura 78 Menú superior "Leer"

Su cabecera es *lee_signal(I_Signal).* El parámetro de entrada es *I_Signal*, número identificador de la gráfica.

Esta función utiliza *lee_file* para la lectura del fichero que contiene la señal a cargar.

- ***Lee_file***

Función para leer datos numéricos desde fichero. Dichos datos son números enteros a reescalar.

Su cabecera es *[Signal,N_long]=lee_file.* Los parámetros de salida son:

i. *Signal:* vector numérico.
ii. *N_long*: longitud del vector.

- ***Guarda_signal***

Función utilizada para guardar una señal resultante de la simulación realizada.

Su llamada se realiza desde el menú superior "Guardar", que puede verse en la Figura 70. Dicho menú es implementado, como los demás, en la función *En_Intercicie*, y su forma en la siguiente:

```
Id_save=uimenu('Label',Nom_Id_Save,'Visible','Off');
Id_save_S1=uimenu(Id_save,'Label',Nom_Id_Save_1,'Visible','Off','Cal
lBack','guarda_signal(1)');…
```

Figura 79 Menú superior "Guardar"

Su cabecera es *guarda_signal(I_Signal).* El parámetro de entrada es *I_Signal*, número identificador de la gráfica.

Esta función utiliza *guarda_file* para la correcta escritura del fichero donde se guarda la señal.

- ***Guarda_file***

Función para guardar datos numéricos desde variable a fichero. Dichos datos son previamente convertidos a enteros a reescalar.

Su cabecera es *guarda_file(Signal)*. El parámetro de entrada, *Signal*, es el vector numérico que contiene la señal que desea guardar el usuario.

5.1.5 Pautas seguidas en la programación MATLAB

Aunque MATLAB es un lenguaje de programación de alto nivel y muy directo, es conveniente marcar unas pautas de programación tanto para el sistema DVB-T, como para los diferentes sistemas de LaViCAD que se deseen implementar en MATLAB.

5.1.5.1 Estandarización empleada MATLAB

Lo primero que se ha tenido en cuenta para comenzar a programar en MATLAB, después de definir una estructura como la que se ha presentado en la Figura 57, ha sido estandarizar el nombre de etapas, funciones y scripts.

A continuación se muestra una lista con los formatos escogidos:

- *Formato para nombrar las etapas*: En_DVB, es decir, en mayúsculas nombre del proyecto y E de etapa, además separados con un subguión. N es el número de la etapa.

- *Formato para nombrar funciones comunes y específicas*: por ejemplo *genera_trama*, es decir, primera letra en minúscula, separamos ambas palabras con subguión y la primera letra de la segunda palabra en minúscula.

- *Formato para nombrar funciones principales*: En_Opciones, es decir, E de etapa en mayúsculas, separamos con subguión ambas partes, y siempre empezando por mayúscula el nombre. N es el número de etapa.

- En la etapa 0 o generadora de la interfaz general, los nombres serán empezando por mayúsculas y separados con un subguión.

Se ha de poner especial atención a los nombres que se dan a las funciones creadas, porque usar nombres que utilice MATLAB para funciones propias daría problemas de ejecución.

5.1.5.2 Recomendaciones de programación y optimización de código

Después de programar con MATLAB, se van adquiriendo unas costumbres de actuación que ayudan a desarrollar nuestra aplicación de manera más eficiente. A continuación se muestran en una lista para que puedan servir de ayuda a futuros programadores de sistemas de la plataforma LaViCAD en MATLAB:

- Utilizar subrutinas de MATLAB básicas, también denominadas funciones intrínsecas por estar incorporadas al propio código ejecutable de MATLAB, evitando en la medida de lo posible ficheros *.m que no hayan sido creados específicamente dentro de LaViCAD. Mediante esta estrategia se consigue

obtener programas con mayor grado de supervivencia y compatibilidad ante nuevas versiones de MATLAB.

- Programación directa de las propiedades de los objetos gráficos que ofrece MATLAB con el fin de controlar al máximo la ejecución resultante y hacerla robusta a instalación en diferentes ordenadores, con diferentes resoluciones gráficas, ubicación, etc...

- Las funciones vectoriales de MATLAB son mucho más rápidas que sus contrapartidas escalares. En la medida de lo posible es muy interesante vectorizar los algoritmos de cálculo, es decir, realizarlos con vectores y matrices, y no con variables escalares dentro de bucles. Éstos solo se deben usar en el caso de un recorrido pequeño.

- Aunque los vectores y matrices pueden ir creciendo a medida que se necesita, es mucho más rápido reservarles toda la memoria necesaria al comienzo del programa. Se puede utilizar para ello la función *zeros*. Además de este modo la memoria reservada es contigua.

- Conviene desarrollar los programas incrementalmente, comprobando cada función o componente que se añade. De esta forma siempre se construye sobre algo que ya ha sido comprobado y que funciona: si aparece algún error, lo más probable es que se deba a lo último que se ha añadido, y de esta manera la búsqueda de errores está acotada y es mucho más sencilla.

- En este mismo sentido, puede decirse que pensar bien las cosas al programar (sobre una hoja de papel en blanco, mejor que sobre la pantalla del PC) siempre es rentable, porque disminuye más que proporcionalmente el tiempo de depuración y eliminación de errores.

- Otro objetivo de la programación debe ser mantener el código lo más sencillo y ordenado posible. Al pensar en cómo hacer un programa o en cómo realizar una determinada tarea es conveniente pensar siempre primero en la solución más sencilla, y luego plantearse otras cuestiones como la eficiencia.

- Finalmente, el código debe ser escrito de una manera clara y ordenada, introduciendo comentarios, utilizando líneas en blanco para separar las distintas partes del programa, sangrando las líneas para ver claramente el rango de las bifurcaciones y bucles, utilizando nombres de variables que recuerden al significado de la magnitud física correspondientes, y en nuestro caso el número de etapa en el que se utilizan, etc.

  - ***Depuración de código***

Aparte de todo lo mencionado, gracias a que MATLAB aparte de informar de los errores en la programación informa a través de *warnings* de partes del código que pueden provocar un funcionamiento poco eficiente del programa, se ha depurado el código inicialmente programado. A continuación se muestran algunos ejemplos:

- Usar de la función *exist* del siguiente modo: "*exist ('Borrador','var') ==1*", al añadirle un segundo parámetro limita la búsqueda y es más eficiente.

- Para averiguar si una variable está vacía, usar "*if isempty(nombre_variable)==0*", en lugar de usar "*if (length > 0)*".

- Usar indexación lógica en lugar de *find*. Por ejemplo, realiza el mismo cálculo *A (find (B>0))* que poner *A (B>0)* directamente, es decir, se estaba añadiendo redundancia.

5.1.6 Retos en la programación MATLAB

A lo largo de la implementación MATLAB del simulador DVB-T se han afrontado retos intrínsecos al procesado de señal interno de cada etapa. Todos ellos se han explicado en la sección 4.3 de la memoria.

5.2 Despliegue de aplicaciones MATLAB

MATLAB ofrece la posibilidad de exportar sus aplicaciones como un ejecutable o como un componente software, con la finalidad de permitir su ejecución a usuarios que no disponen de MATLAB. Esto se consigue gracias a la *toolbox* MATLAB Compiler.

Los ejecutables y las librerías creadas con MATLAB Compiler usan un motor de ejecución en tiempo real llamado MATLAB Compiler Runtime (MCR). El MCR se suministra con MATLAB y se puede distribuir libremente junto a la aplicación creada en MATLAB. Para que se puedan ejecutar dichas exportaciones, el usuario debe tener instalado el MCR.

Entre estas posibilidades de exportación, en el desarrollo del proyecto se han estudiado las dos siguientes:

- ✓ Aplicaciones *standalone*
- ✓ Compilación para Java

Para más información sobre la ejecución e instalación en las diferentes plataformas, y los problemas experimentados con la compilación para Java, se recomienda la referencia [35], documento desarrollado durante la realización de este proyecto.

5.2.1 Aplicación standalone

Se creará un programa autónomo que el usuario podrá ejecutar con un sencillo doble clic. Se explican en este apartado los pasos a seguir para crear una aplicación "standalone" a partir de una programación en MATLAB. Siguiendo los pasos descritos, se ha desarrollado la aplicación standalone del simulador de la capa física del sistema DVB-T.

1. Teclear "*deploytool*" en la ventana de comandos. Se observará la siguiente ventana:

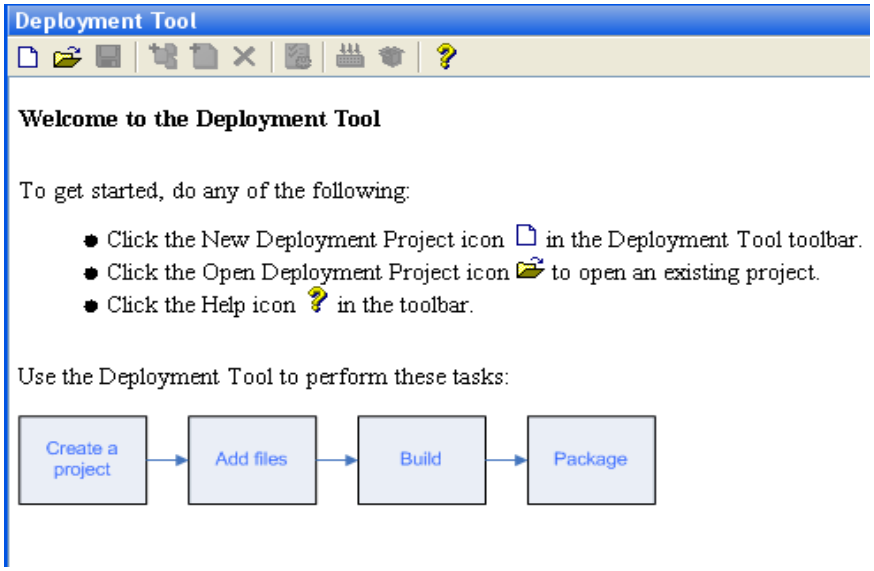


Figura 80 Ventana Deploytool

2. Clicar en "Create a new deployment project".

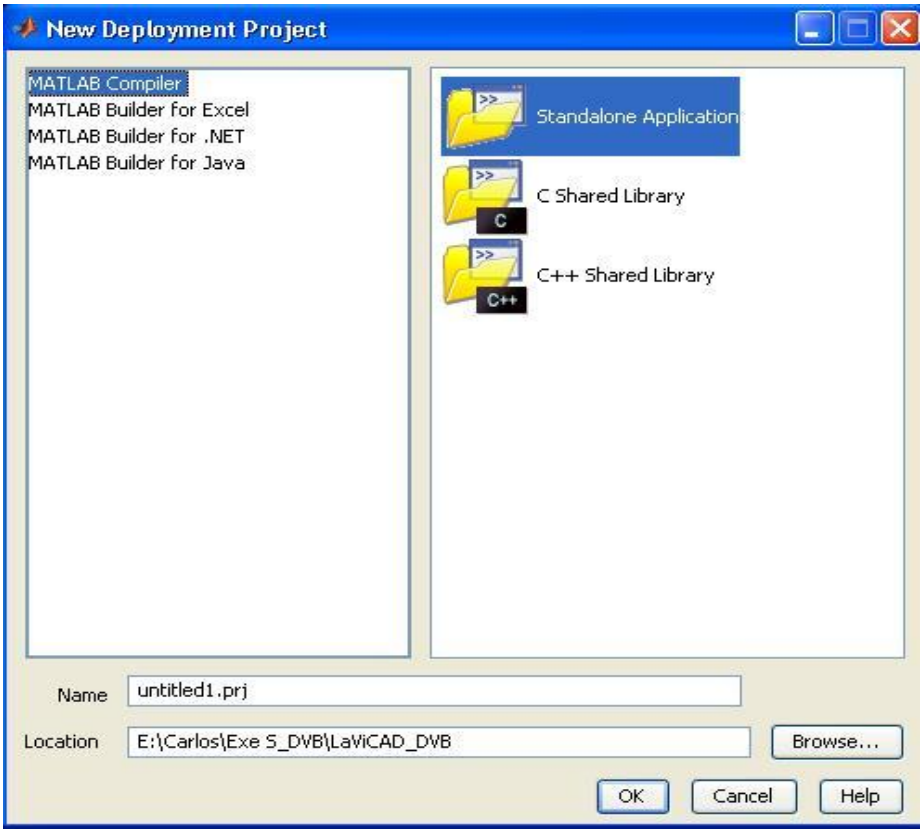


Figura 81 Crear nuevo proyecto Standalone

3. En este punto se debe escoger "*Matlab Compiler*", "*Standalone Application*", el nombre de nuestro proyecto y hacer clic en "*ok*". Entonces se pasa a observar la siguiente imagen:

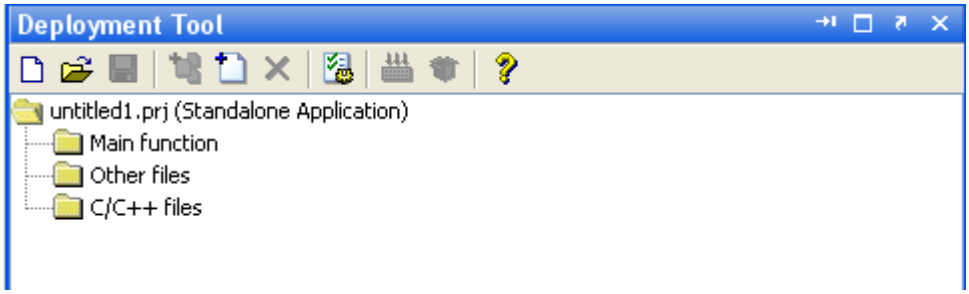


Figura 82 Ventana del proyecto a crear

En ella se pueden observar las carpetas a rellenar por nuestros scripts y funciones.

4. Llegados a este punto, escribir "*mbuild –setup*" en la ventana de comandos. Con esta acción se escoge el compilador que utilizará MATLAB para crear la aplicación. Se puede utilizar el compilador que ofrece MATLAB. (*Lcc- win32 C 2.4.1*). Solo es necesario ejecutar esta instrucción la primera vez que se utiliza la aplicación.

5. Una vez seleccionado el compilador, se deben mover todas las funciones y ficheros *.m de nuestro programa a las carpetas situadas en la ventana "*Deployment Tool*".

    a. Primero se debe mover la función principal de nuestro programa a la carpeta "*Main function*".

    Hay que tener cuidado en este punto, puesto que debe ser una función y no un script. De este modo si en nuestro programa el "*.m file*" al que se llama para ejecutarlo no es una función, se deberá transformar en una escribiendo "*function nombre_función*" en la primera línea del "*.m file*".

    Puesto que las variables declaradas en scripts son globales, y las de las funciones son locales, las variables declaradas en nuestro "*.m file*" convertido a función ya no aparecerán en el "*workspace*", y por tanto no se podrán usar en "*scripts*" posteriores de nuestro programa. La solución es declarar estas variables como globales dentro de nuestro "*.m file*" principal, y volver a declararlas una vez más como global dentro del script que se necesite. Una vez realizados estos pasos, ya estarán disponibles para utilizarlas en cualquier script de nuestro programa sin necesidad de volverlas a declarar.

    b. El siguiente paso será pasar todas las funciones y scripts de nuestro programa a la carpeta "*Other files*".

    En resumen, las modificaciones que hemos realizado a nuestro software de LAVICAD para que sea adecuado son:

    - Hacer que el fichero de llamada sea una función.
    - Declarar las variables generadas en el anterior fichero como globales.
    - Eliminar las instrucciones para actualizar explícitamente el *path*.

6. Una vez realizados todos los pasos anteriores, escoger en "*Settings…*" nuestras preferencias. En el apartado "*General*", decidir si se incluye el "*MCR*" en el "*Package*" que se creará. Marcando esta opción, junto al standalone, aparecerá todo el software que necesita instalar el usuario para poder ejecutarlo. También se podrán añadir carpetas adicionales a todos los ficheros que componen el simulador de MATLAB que en particular se esté convirtiendo, que sean necesarias y no se hayan incluido previamente.

7. Hacer clic en el icono "*Build the project*" de la ventana "*Deployment Tool*", y MATLAB se pondrá a compilar. Si no hay errores indicará "*Compilation Complete*", y se habrá creado una carpeta en nuestro directorio con el nombre del proyecto. Esta carpeta consta de dos subcarpetas, "*distrib*" y "*src*". La carpeta "*src*" es el directorio intermedio de salida, y "*distrib*" es el directorio final. Éste constará de un archivo ejecutable (*"nombre_proyecto".exe*) y un archivo (*"nombre_proyecto".ctf*) que contiene todas las funciones encriptadas de nuestro programa MATLAB. Los archivos creados pueden variar en función de la versión de MATLAB empleada.

8. Adicionalmente, se puede crear el paquete de distribución clicando en el icono "*Package the project*" dentro de la ventana "*Deployment tool*".

9. Finalmente, se puede ejecutar el programa (*"nombre_proyecto".exe*) que empleará la información contenida en el archivo *.ctf* para la ejecución.

Una vez realizado todo el proceso, se puede observar que para compartir nuestro programa solo se necesitan los archivos *.exe* y *.ctf*. Aparte, esta aplicación necesitará que se instale previamente en el PC donde se desee utilizar el MATLAB Component Runtime (MCR). Este archivo se puede incluir en nuestro paquete a distribuir seleccionándolo como se explica en el apartado 6.

El MCR, que es necesario tener instalado para ejecutar ocupa 146MB. Esto supone los siguientes tiempos de descarga para el usuario:

- 20 min con una conexión de 1Mbps
- 6 min con una conexión de 3Mbps

5.2.2 Builder for JAVA

Builder for Java es un toolbox de MATLAB que permite encapsular las funciones de MATLAB en una clase Java. Con ello se podrán emplear las funciones encapsuladas desde el propio código de Java.

Durante el desarrollo de este proyecto se asistió a una conferencia virtual de The Mathworks acerca de este tema, por el interés que despertó al ver la posibilidad de crear el applet de Java a partir de la aplicación MATLAB. Más adelante se encontraron diversas dificultades que provocaron desestimar esta opción, y se realizó un estudio paralelo llevado a cabo por el estudiante de la ETSETB Pedro Espinosa Fricke para el departamento de TSC. A continuación se explican los resultados del estudio, y se pueden ampliar en el documento [35].

- ***Estado actual del estudio***

Dado que no es posible una integración directa de código MATLAB en un applet de Java a fecha de hoy, se propuso como alternativa la implementación de un *servlet*. Dicho *servlet* se ejecutará en un servidor UNIX y su tarea será ejecutar las funciones MATLAB que le pidan los applets mediante paso de mensajes. Esta solución permite conservar la estructura actual del applet y ahorra la descarga por parte del usuario del software MCR de MATLAB, necesario para ejecutar el código MATLAB.

El único requisito que necesita la máquina que va a ejecutar el applet es tener instalado el MCR.

- ***Builder for Java***

Es importante diferenciar dos tipos de programas de Java, en función de dónde se lance el programa:

- Si el programa se lanza desde la consola, se trata de una aplicación normal de Java. En este caso, los pasos para crear una clase de Java a partir de una función MATLAB son muy similares a los que se siguen para la creación de una aplicación Standalone. De hecho, se trata de seleccionar la opción *Builder for Java > Java Package* y continuar del mismo modo que para crear un standalone. Si el programa de MATLAB se ejecuta mediante un script, es necesario convertirlo a una función antes de efectuar el *build*.

- Si el programa se lanza a través de un navegador de Internet, se le conoce como applet. En este caso el proceso es más complejo: debido a cuestiones de seguridad ineludibles, no es posible crearlo de una forma tan directa como en el caso anterior. Para poder integrar funciones de MATLAB en un applet deberán exportarse por separado e integrar el resultado en el código Java del applet.

En caso de querer integrar la función de MATLAB en un applet de LaViCAD, se recomienda que la función únicamente realice operaciones de procesado dejando la interacción con el usuario para el applet. Esto requiere eliminar toda la parte correspondiente a la interfaz gráfica como *plots*, ventanas y *uicontrols* de dicha función de MATLAB.

En ambos casos, en la carpeta distrib aparecerá un archivo *nombre_proyecto.jar* que contiene las clases Java que encapsulan la función MATLAB: si se trata del primer caso contendrá el programa completo y si se trata del segundo contendrá un fragmento del mismo. El siguiente paso es integrar estas clases en el código Java que se esté desarrollando.

Una vez compilado el programa Java ya estará listo para ser ejecutado.

Apéndice A. Conceptos MATLAB del proyecto

En este apéndice al capítulo 5 de la memoria del proyecto se describen los conceptos MATLAB que han sido empleados para programar el applet en dicho software.

Para más información sobre The Mathworks, y MATLAB consultar su página web [33].

A.1 El programa MATLAB

MATLAB se puede utilizar en una amplia gama de aplicaciones que incluyen procesamiento de señales e imágenes, comunicaciones, diseño de sistemas de control, sistemas de prueba y medición, modelado y análisis financiero y biología computacional. Los conjuntos de herramientas complementarios (colecciones de funciones de MATLAB para propósitos especiales, que están disponibles por separado, llamados *toolboxes*) amplían el entorno de MATLAB permitiendo resolver problemas especiales en estas áreas de aplicación.

Además, MATLAB contiene una serie de funciones para documentar y compartir las aplicaciones. Puede integrarse el código MATLAB con otros lenguajes y aplicaciones, y distribuir los algoritmos y aplicaciones desarrollados usando MATLAB.

- ***Características principales***

- Lenguaje de alto nivel para cálculo técnico
- Entorno de desarrollo para la gestión de código, archivos y datos
- Herramientas interactivas para exploración, diseño y resolución de problemas iterativos
- Funciones matemáticas para álgebra lineal, estadística, análisis de Fourier, filtraje, optimización e integración numérica
- Funciones gráficas bidimensionales y tridimensionales para visualización de datos
- Herramientas para crear interfaces gráficas de usuario personalizadas
- Funciones para integrar los algoritmos basados en MATLAB con aplicaciones y lenguajes externos, tales como C/C++, FORTRAN, Java, COM y Microsoft Excel.

A.2 Desarrollo de una aplicación

MATLAB proporciona un lenguaje de alto nivel y herramientas de desarrollo con los que es posible desarrollar y analizar algoritmos y aplicaciones rápidamente.

A.2.1 El lenguaje MATLAB

El lenguaje MATLAB incluye operaciones vectoriales y matriciales que son fundamentales para resolver los problemas científicos y de ingeniería. Agiliza tanto el desarrollo como la ejecución.

Con el lenguaje de MATLAB, se pueden programar y desarrollar algoritmos más rápidamente que con los lenguajes tradicionales porque no son necesarias tareas de bajo nivel, tales como declarar variables, especificar tipos de datos y asignar memoria. En muchos casos, MATLAB elimina la necesidad de bucles "for". En consecuencia, una línea de código de MATLAB generalmente reemplaza a varias líneas de código C o C++.

Al mismo tiempo, MATLAB ofrece todas las características de los lenguajes de programación tradicionales, que incluyen operadores aritméticos, control de flujo, estructuras de datos, tipos de datos, programación orientada a objetos (OOP) y depuración. Cabe mencionar que la programación orientada a objetos es una característica muy nueva de MATLAB, y por lo tanto no se tuvo en cuenta al iniciar el desarrollo del simulador DVB-T.

MATLAB permite ejecutar comandos o grupos de comandos uno a uno, sin compilar ni enlazar, y repetir su ejecución hasta lograr la solución óptima.

Para ejecutar rápidamente cálculos matriciales y vectoriales complejos, MATLAB utiliza bibliotecas optimizadas para el procesador. Para cálculos escalares de aplicación general, MATLAB genera instrucciones en código máquina utilizando su tecnología JIT (Just-In-Time). Gracias a esta tecnología, que está disponible para la mayoría de las plataformas, las velocidades de ejecución son mucho más rápidas que las de los lenguajes de programación tradicionales.

A.2.2 Herramientas de desarrollo

MATLAB incluye herramientas de desarrollo que ayudan a implementar los algoritmos eficientemente. Como por ejemplo:

- **MATLAB Editor** - Funciones de edición y depuración estándar, como establecimiento de puntos de interrupción y simulaciones paso a paso.
- **M-Lint Code Checker** - Analiza el código y recomienda modificaciones para mejorar el rendimiento y mantenimiento.
- **MATLAB Profiler** - Registra el tiempo que tarda en ejecutarse cada línea de código.
- **Directory Reports** - Explora todos los archivos de un directorio y crea informes sobre la eficiencia del código, las diferencias entre los archivos, las dependencias de los archivos y la cobertura del código.

A.2.3 Guide

Puede usarse la herramienta interactiva GUIDE (Graphical User Interface Development Environment) para diseñar y editar interfaces de usuario. Esta herramienta permite incluir listas de selección, menús desplegables, botones de pulsación, botones de opción y deslizadores, así como diagramas de MATLAB y controles ActiveX. También pueden crearse interfaces gráficas de usuario por medio de programación usando las funciones de MATLAB.

En el desarrollo del proyecto en estudio, se ha creado la interfaz gráfica por medio de programación, prescindiendo de la herramienta GUIDE. Esta interfaz se ha diseñado siguiendo el estilo del sistema completo de modulaciones QAM (Quadrature and Amplitude Modulation) desarrollado por Margarita Cabrera, co-directora del proyecto.

Dicho diseño se basa en que los gráficos de MATLAB tienen una estructura jerárquica, formada por objetos de distintos tipos. Esta jerarquía tiene forma de árbol, con el aspecto mostrado en la Figura 83.

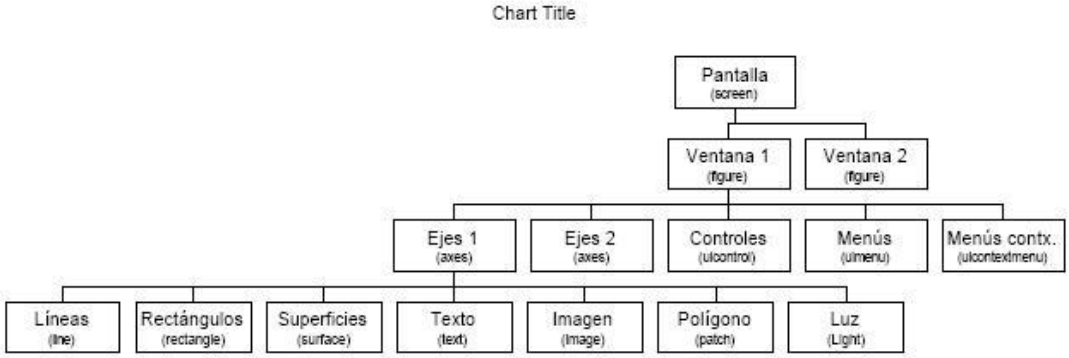


Figura 83 Jerarquía gráfica de MATLAB

Según se muestra en la Figura 83, el objeto más general es la pantalla (*screen*). Este objeto es la raíz de todos los demás y sólo puede haber un objeto pantalla. Una pantalla puede contener una o más ventanas (*figures*). A su vez cada una de las ventanas puede tener uno o más ejes de coordenadas (*axes*) en los que representar otros objetos de más bajo nivel. Una ventana puede tener también controles (*uicontrol*) tales como botones, barras de desplazamiento, botones de selección o de opción, etc) y menús (*uimenu*). Finalmente, los ejes pueden contener seis tipos de elementos gráficos: líneas (*line*), rectángulos (*rectangle*), polígonos (*patches*), superficies (*surface*), imágenes bitmap (*image*) y texto (*text*).

La jerarquía de objetos mostrada en la Figura 83 indica que en MATLAB hay objetos padres e hijos. Por ejemplo, todos los objetos ventana son hijos de pantalla, y cada ventana es padre de los objetos ejes, controles o menús que están por debajo. A su vez los elementos gráficos (líneas, polígonos, etc.) son hijos de un objeto ejes, y no tienen otros objetos que sean sus hijos.

Cuando se borra un objeto de MATLAB automáticamente se borran todos los objetos que son sus descendientes. Por ejemplo, al borrar unos ejes, se borran todas las líneas y polígonos que son hijos suyos.

Cada uno de los objetos de MATLAB tiene un identificador único (*handle*). Algunos gráficos tienen muchos objetos, en cuyo caso tienen múltiples *handles*. El objeto raíz (*pantalla*) es siempre único y su identificador es el cero. El identificador de las ventanas es un entero, que aparece en la barra de nombre de dicha ventana. Los identificadores de otros elementos gráficos son números *float*, que pueden ser obtenidos como valor de retorno y almacenados en variables de MATLAB.

A.2.4 Ficheros *.m

En el desarrollo del proyecto se ha utilizado una estructura de carpetas por etapas, con una serie de ficheros *.m en cada una de ellas.

Existen dos tipos de ficheros *.m, los ficheros de comandos (scripts) y las funciones. Los primeros contienen simplemente un conjunto de comandos que se ejecutan sucesivamente cuando se teclea el nombre del fichero en la línea de comandos de MATLAB. Un fichero de comandos puede llamar a otros ficheros de comandos. Si un fichero de comandos se llama desde de la línea de comandos de MATLAB, las variables que crea pertenecen al espacio de trabajo base de MATLAB, y permanecen en él cuando se termina la ejecución de dicho fichero.

Las funciones permiten definir funciones enteramente análogas a las de MATLAB, con su nombre, sus argumentos y sus valores de retorno. Los ficheros *.m que definen funciones permiten extender las posibilidades de MATLAB; de hecho existen bibliotecas de ficheros *.m que se venden (toolkits) o se distribuyen gratuitamente (a través de Internet). Las funciones definidas en ficheros *.m se caracterizan porque la primera línea (que no sea un comentario) comienza por la palabra *function*, seguida por los valores de retorno (entre corchetes [ ] y separados por comas, si hay más de uno), el signo igual (=) y el nombre de la función, seguido de los argumentos (entre paréntesis y separados por comas).

Un fichero *.m puede llamar a otros ficheros *.m, e incluso puede llamarse a sí mismo de forma recursiva. Los ficheros de comandos se pueden llamar también desde funciones, en cuyo caso las variables que se crean pertenecen al espacio de trabajo de la función. El espacio de trabajo de una función es independiente del espacio de trabajo base y del espacio de trabajo de las demás funciones. Esto implica por ejemplo que no puede haber colisiones entre nombres de variables: aunque varias funciones tengan una variable llamada A, en realidad se trata de variables completamente distintas (a no ser que A haya sido declarada como variable global).

A.2.5 Funciones con número variable de argumentos

En la realización de este proyecto, se ha utilizado en cada etapa una forma de pasar a una función un número variable de argumentos por medio de la variable *varargin*. Ésta contiene tantos elementos como sean necesarios para poder recoger en dichos elementos todos los argumentos que se hayan pasado en la llamada. De forma análoga, una función puede tener un número indeterminado de valores de retorno utilizando varargout. Las variables *nargin* y *nargout* indican el número de argumentos y de valores de retorno con que ha sido llamada la función.

A.2.6 Guardar variables de sesión

En el proyecto que nos ocupa, y en infinidad de casos, puede resultar interesante interrumpir el trabajo con MATLAB y poder recuperar la sesión más tarde, en el mismo punto en el que se dejó (con las mismas variables definidas, con los mismos resultados

intermedios, etc.). Hay que tener en cuenta que al salir del programa todo el contenido de la memoria se borra automáticamente.

Para guardar el estado de una sesión de trabajo en el directorio actual existe el comando save. Si se teclea *"save",* antes de abandonar el programa, se crea un fichero binario llamado matlab.mat (o matlab) con el estado de la sesión (excepto los gráficos, que por ocupar mucha memoria hay que guardar aparte). Dicho estado puede recuperarse la siguiente vez que se arranque el programa con el comando "*load*". Esta es la forma más básica de los comandos save y load.

Se pueden guardar también matrices y vectores de forma selectiva y en ficheros con nombre especificado por el usuario. Por ejemplo, el comando "save filename A, x, y" guarda las variables A, x e y en un fichero binario llamado filename.mat (o filename). Para recuperarlas en otra sesión basta teclear "load filename". Si no se indica ningún nombre de variable, se guardan todas las variables creadas en esa sesión.

El hecho de poder guardar sesiones fácilmente, ha resultado ser de gran utilidad para el proyecto desarrollado, puesto que ha permitido la posibilidad de guardar simulaciones hasta cierta etapa, para posteriormente poder volver al mismo punto donde se había pausado, de manera sencilla y rápida. Esta característica del simulador desarrollado, se ha explicado en el apartado 5.1.2.2.

A.3 Análisis y acceso a datos

MATLAB soporta todo el proceso de análisis de datos, desde la adquisición de datos de dispositivos externos y bases de datos, pasando por el preprocesado, visualización y análisis numérico, hasta la producción de resultados con calidad de presentación.

Para el análisis de datos, MATLAB proporciona herramientas interactivas y funciones de línea de comandos. Algunas de las usadas en el desarrollo del proyecto son las siguientes:

- Extracción de secciones de datos, escalado y promediado
- Correlación, análisis de Fourier y filtrado
- Búsqueda de picos unidimensionales, valles y ceros
- Estadística básica y ajuste de curvas
- Análisis matricial

Hay varias formas de importar y exportar datos en MATLAB. En el proyecto en estudio se ha escogido el formato de fichero. Así que mediante las funciones de bajo nivel *fopen, fread y otras* se puede tanto leer como escribir datos en un fichero. Estos ficheros serán intercambiables entre las versiones JAVA y MATLAB del simulador DVB-T.

- ***Lectura y escritura de ficheros***

MATLAB dispone de funciones de lectura/escritura análogas a las del lenguaje C (en las que están inspiradas), aunque con algunas diferencias.

Las funciones ***fopen y fclose*** sirven para abrir y cerrar ficheros, respectivamente. La función fopen tiene la forma siguiente: *[fi,texto] = fopen('filename','c')*, donde *fi* es un valor de retorno que sirve como identificador del fichero, *texto* es un mensaje para caso de que se produzca un error, y *c* es un carácter (o dos) que indica el tipo de operación que se desea realizar.

Las opciones más importantes son las siguientes:

- *'r'* lectura (de read)
- *'w'* escritura reemplazando (de write)
- *'a'* escritura a continuación (de append)
- *'r+'* lectura y escritura

Después de realizar las operaciones de lectura y escritura deseadas, el fichero se puede cerrar con la función close en la forma siguiente: *st = fclose(fi),* donde *st* es un valor de retorno para posibles condiciones de error. Si se quieren cerrar a la vez todos los ficheros abiertos, puede utilizarse el comando: *st = close('all').*

Las funciones *fscanf y fprintf* permiten leer y escribir en ficheros ASCII, es decir, en ficheros formateados. La forma general de la función fscanf es la siguiente: *[var1, var2,...] = fscanf (fi,'cadena de control', size),* donde *fi* es el identificador del fichero (devuelto por la función *fopen*), y *size* es un argumento opcional que puede indicar el tamaño del vector o matriz a leer. La *cadena de control* va encerrada entre apóstrofos simples, y contiene los especificadores de formato para las variables:

- *%s* para cadenas de caracteres
- *%d* para variables enteras
- *%f* para variables de punto flotante
- *%lf* para variables de doble precisión

Finalmente, la función *fprintf* dirige su salida formateada hacia el fichero indicado por el identificador. Su forma general es: *fprintf(fi,'cadena de control',var1,var2,...).*

A.4 Visualización de datos

Todas las funciones gráficas necesarias para visualizar datos de ingeniería y científicos están disponibles en MATLAB. Se incluyen funciones de representación de diagramas bidimensionales y tridimensionales, visualización de volumen tridimensional, herramientas para crear diagramas en forma interactiva y la posibilidad de exportar los resultados a los formatos gráficos más conocidos.

- ***Gráficos bidimensionales***

En el proyecto se han graficado todas las señales en dos dimensiones. MATLAB facilita mediante sus funciones la graficación de diagramas de líneas, de barras y de sectores, la representación de histogramas, y demás gráficas.

A raíz de estas funciones, se han creado *images_children* y *stem_children*, de mayor complejidad, para situaciones concretas de nuestro simulador. Estas funciones se han explicado en el apartado 5.1.4.1.

Aparte, MATLAB ofrece posibilidades de realizar gráficos tridimensionales, característica no utilizada en el desarrollo del proyecto.

Por último, MATLAB incluye herramientas interactivas para el diseño y modificación de gráfico. En el proyecto, el zoom está disponible desde la interfaz principal de cada etapa, y una vez importado un gráfico, queda a nuestra disponisición toda una barra de menús.

A.5 Despliegue de aplicaciones

Puede integrarse el código MATLAB con otros lenguajes y aplicaciones, y distribuir los algoritmos y aplicaciones de MATLAB como programas autónomos o módulos de software.

MATLAB proporciona funciones para integrar código C y C++, código FORTRAN, objetos COM y código Java en sus aplicaciones. Puede llamar a DLLs, clases de Java y Controles ActiveX. Utilizando la biblioteca *engine* de MATLAB, también puede llamar a MATLAB desde código C, C++ o FORTRAN.

- ***Distribución de aplicaciones***

Puede crearse un algoritmo en MATLAB y distribuirlo a otros usuarios de MATLAB como código M. Usando el MATLAB Compiler (MCR), puede distribuirse el algoritmo, ya sea como aplicación autónoma o como módulo de software incluido en un proyecto, a los usuarios que no tengan MATLAB.

Estas características de MATLAB se han analizado en la sección 5.2.

6 Simulador S_DVB Java

En este capítulo se estudia el desarrollo del simulador de la capa física del estándar DVB-T (estándar utilizado en la televisión digital terrestre TDT) sobre el entorno de desarrollo integrado de código abierto Eclipse [36].

El desarrollo del applet de Java se llevó a cabo una vez terminada la aplicación MATLAB. Por este motivo, la programación MATLAB de los diferentes procesados de señal de las etapas del simulador, se ha utilizado de ayuda para la programación del applet de Java. Así mismo, se ha utilizado la aplicación MATLAB, por medio de simulaciones, para comprobar el correcto funcionamiento de las diferentes etapas del applet de Java.

En el presente capítulo, la estructura que se sigue es la siguiente:

- ✓ Se describe, en primer lugar, el aplicativo Contenedor, parte principal del proyecto LaViCAD implementado por Carlos y Francisco Vargas bajo la dirección de Margarita Cabrera Bean. En esta primera sección, también se dan las pautas para realizar un nuevo sistema con el aplicativo.

- ✓ En la siguiente sección, se presentan los retos afrontados en la programación Java.

- ✓ Más adelante, se explican las diferencias principales experimentadas entre los lenguajes de programación MATLAB y Java.

- ✓ En el apéndice A, al final del capítulo, se pretende dar una visión global de las características del lenguaje Java que han sido empleadas para el desarrollo del applet simulador de la capa física del sistema DVB-T.

6.1 Aplicativo Contenedor

En el desarrollo del applet de Java simulador de la capa física del sistema DVB-T se ha empleado el aplicativo Contenedor, principal logro del proyecto LaViCAD implementado por Carlos y Francisco Vargas.

En esta sección se describe el aplicativo, estudiando todos aquellos conceptos necesarios para la realización del applet, y posteriormente se dan las pautas a seguir para crear un nuevo sistema.

6.1.1 Descripción

En este apartado se realiza una descripción del aplicativo, explicando la manera que tiene de trabajar, la configuración y demás detalles que se han estudiado para desarrollar

el applet DVB-T utilizando este Contenedor. Para un análisis más profundo se recomienda [22].

6.1.1.1 Arquitectura seleccionada

A continuación, se enumeran las tecnologías software empleadas en el desarrollo del aplicativo Contenedor. En los siguientes puntos se describen algunas de las tecnologías desde un punto de vista general y se presentan las características más relevantes de éstas. En el apéndice A del capítulo se explican algunas de ellas.

- ✓ Orientación a objetos
- ✓ Lenguaje de programación: Java.
- ✓ Diseño de aplicación: Applet
- ✓ Diseño interfaz usuario: Swing
- ✓ Diseño ficheros de configuración: XML
- ✓ Diseño gráficas: Ptolemy

6.1.1.1.1 Interfaz de usuario: Java Swing

Para la realización de la interfaz gráfica de usuario (GUI) se ha utilizado la librería de Java Swing.

Swing es una biblioteca gráfica que apareció en la versión 1.2 de Java y está compuesta por un amplio conjunto de componentes de interfaces de usuario que funcionan en el mayor número posible de plataformas. Anteriormente a Swing existía la librería AWT, pero se ha seleccionado para la realización de la interfaz de usuario de la aplicación la librería Swing debido a las siguientes ventajas y mejoras que se le han introducido comparando con AWT:

- ✓ **Amplia variedad de componentes**: En general las clases que comiencen por "J" son componentes que se pueden añadir a la aplicación. Por ejemplo: *JButton*.

- ✓ **Aspecto modificable (*look and feel*)**: Se puede personalizar el aspecto de las interfaces o utilizar varios aspectos que existen por defecto (Metal Max, Basic Motif, Window Win32).

- ✓ **Arquitectura Modelo-Vista-Controlador**: Esta arquitectura da lugar a todo un enfoque de desarrollo muy arraigado en los entornos gráficos de usuario realizados con técnicas orientadas a objetos. Cada componente tiene asociada una clase de modelo de datos y una interfaz que utiliza. Se puede crear un modelo de datos personalizado para cada componente, con sólo heredar de la clase *Model*.

- ✓ **Gestión mejorada de la entrada del usuario**: Se pueden gestionar combinaciones de teclas en un objeto *KeyStroke* y registrarlo como componente. El evento se activará cuando se pulse dicha combinación si está siendo utilizado el componente, la ventana en que se encuentra o algún hijo del componente.

- ✓ **Objetos de acción (*action objects*)**: Estos objetos cuando están activados (*enabled*) controlan las acciones de varios objetos componentes de la interfaz. Son hijos de *ActionListener*.

- ✓ **Contenedores anidados**: Cualquier componente puede estar anidado en otro. Por ejemplo, un gráfico se puede anidar en una lista.

- ✓ **Escritorios virtuales**: Se pueden crear escritorios virtuales o "interfaz de múltiples documentos" mediante las clases *JDesktopPane* y *JInternalFrame*.

- ✓ **Bordes complejos**: Los componentes pueden presentar nuevos tipos de bordes. Además el usuario puede crear tipos de bordes personalizados.

- ✓ **Diálogos personalizados**: Se pueden crear multitud de formas de mensajes y opciones de diálogo con el usuario, mediante la clase *JOptionPane*.

- ✓ **Clases para diálogos habituales**: Se puede utilizar *JFileChooser* para elegir un fichero, y *JColorChooser* para elegir un color.

- ✓ Componentes para tablas y árboles de datos: Mediante las clases *JTable* y *JTree*.

- ✓ **Potentes manipuladores de texto**: Además de campos y áreas de texto, se presentan campos de sintaxis oculta *JPassword*, y texto con múltiples fuentes *JTextPane*. Además hay paquetes para utilizar ficheros en formato HTML o RTF.

- ✓ **Capacidad para "deshacer"**: En gran variedad de situaciones se pueden deshacer las modificaciones que se realizaron.

- ✓ **Soporte a la accesibilidad**: Se facilita la generación de interfaces que ayuden a la accesibilidad de discapacitados, por ejemplo en *Braille*.

6.1.1.1.2 Ficheros de configuración XML

El aplicativo Contenedor ha sido diseñado de manera que pueda configurarse rápida y fácilmente por cualquier programador que quiera implementar un nuevo sistema, como el caso del simulador DVB-T que tratamos. Por este motivo se ha seleccionado para crear todos los ficheros de configuración que componen el aplicativo el lenguaje XML (eXtensible Markup Language ("lenguaje de marcas extensible")).

Habitualmente se utilizan ficheros configuración propietarios que cada aplicación define a su modo y que frecuentemente sólo son entendibles por la propia aplicación que los utiliza. Mediante XML se pretende sustituir este problema de compresión y utilizar un lenguaje que sea fácilmente entendible por cualquier usuario y que tenga una gran flexibilidad para adaptarse a cualquier estructura de datos que se pueda necesitar.

XML es un metalenguaje extensible de etiquetas desarrollado por el Word Wide Web Consortium (W3C), que se utiliza como estándar para el intercambio de información

estructurada entre diferentes plataformas y para la definición de estructuras de datos como puede ser un fichero de configuración con una jerarquía compleja de objetos.

A continuación se enumeran varias de las características más importantes de XML:

- ✓ Asigna una estructura lógica a la información permitiendo crear código más ligero.
- ✓ Es extensible. Permite crear etiquetas propias, por lo que cada usuario codifica el fichero XML con los nombres que considera más entendibles en el contexto en el que se está trabajando.
- ✓ Facilita el intercambio de documentos entre usuarios y aplicaciones.
- ✓ Permite validar el documento contra una gramática, por lo que evita poder cometer errores en el proceso de escribir el fichero de configuración.
- ✓ Si un tercero decide usar un documento creado en XML, es sencillo entender su estructura y procesarlo.
- ✓ Es independiente de la plataforma sobre la que trabaja, con lo que facilita el intercambio de datos.

La tecnología XML, por lo tanto, soluciona el problema de poder expresar información estructurada de la manera más abstracta y reutilizable posible. Que la información sea estructurada quiere decir que ésta se compone de partes bien definidas, a estas partes se les llaman *elementos* y se identifican mediante *etiquetas*.

Una *etiqueta* consiste en una marca hecha en un documento, que identifica una porción de éste como un *elemento*, una parte de información con un sentido claro y definido. Las etiquetas tienen la siguiente forma <*nombre*> donde *nombre* es el identificador del elemento que se está señalando.

Una de las ventajas de los documentos XML es que deben seguir unas reglas de sintaxis definidas por la especificación de XML, por lo que si nos equivocamos en su formación el documento estará *mal formado* y no podrá ser leído por ningún aplicativo. Las herramientas de desarrollo normalmente ya verifican esta sintaxis por lo que facilita enormemente su construcción y evita errores que repercutirían en la ejecución o carga de datos del documento XML. A los documentos que cumplen las siguientes reglas básicas de formato se les llama "*bien formados"*:

- ✓ Los documentos han de seguir una estructura estrictamente jerárquica con lo que respecta a las etiquetas que delimitan sus elementos. Una etiqueta debe estar correctamente incluida en otra, es decir, las etiquetas deben estar correctamente anidadas. Los elementos con contenido deben estar correctamente cerrados.
- ✓ Los documentos XML sólo permiten un elemento raíz del que todos los demás sean parte, es decir, sólo pueden tener un elemento inicial.
- ✓ Los valores atributos en XML siempre deben estar encerrados entre comillas simples o dobles.

- ✓ Si se inserta una etiqueta vacía debe terminarse con '/>' o se debe hacer que no sea vacía añadiéndole una etiqueta de fin.

- ✓ El XML es sensible a mayúsculas y minúsculas. Existe un conjunto de caracteres llamados espacios en blanco (espacios, tabuladores, retornos de carro, saltos de línea) que los procesadores XML tratan de forma diferente en el marcado XML.

- ✓ Es necesario asignar nombres a las estructuras, tipos de elementos, entidades, elementos particulares, etc. No se pueden crear nombres que empiecen con la cadena "xml", "xML", "XML" o cualquier otra variante. Las letras y rayas se pueden usar en cualquier parte del nombre. También se pueden incluir dígitos, guiones y caracteres de punto, pero no se puede empezar por ninguno de ellos.

- ✓ Las construcciones como etiquetas, referencias de entidad y declaraciones se denominan marcas; son partes del documento que el procesador XML espera entender. El resto del documento entre marcas son los datos. Las marcas en un documento XML son aquellas partes que empiezan con "<" y que acaban con ">".

Que un documento esté "*bien formado*" solamente quiere decir que su estructura sintáctica básica es correcta, es decir, que se compone de elementos, atributos y comentarios tal como el metalenguaje XML define que se deben escribir. Ahora bien, cada documento XML definido con esta tecnología, necesitará especificar cuál es exactamente la relación que debe verificarse entre los distintos elementos presentes en el documento.

Esta relación entre elementos se especifica en un documento externo o definición expresada como DTD (*Document Type Definition = Definición de Tipo de Documento*). DTD es un estándar que nos permite definir una gramática que deben cumplir nuestros documentos XML para considerarlos válidos. Una definición DTD para n documentos XML especifica:

- qué elementos pueden existir en un documento XML
- qué atributos pueden tener éstos
- qué elementos pueden o deben aparecer contenidos en otros elementos y en qué orden.

Los programas que son capaces de validar documentos con DTD leen esos documentos y el DTD asociado. En caso de que el documento XML no cumpla los requerimientos que le impone el DTD, nos advertirán del error y no validarán el documento.

6.1.1.1.3 Diseño de gráficas: Ptolemy

En el aplicativo Contenedor se seleccionó una librería para la representación gráfica de funciones diseñada por EECS (Electrical Engineering and Computer Sciences) de la universidad UC Berkeley en su proyecto Ptolemy [39], proyecto cuyos objetivos son el estudio del modelado, simulación y diseño de sistemas concurrentes en tiempo real. Como parte de este enorme proyecto, se incluyen múltiples aplicaciones, entre ellas, las

originadas a partir del uso de Java (Ptolemy II), con el objetivo de diseñar un software orientado a la investigación académica. Dentro de este paquete se encuentra PtPlot que se distribuye de modo gratuito como el resto de software utilizado en el proyecto.

PtPlot es un sistema de componentes que permite dibujar señales en dos dimensiones escrito en Java con las siguientes características:

- ✓ Se puede incrustar en applets o aplicaciones
- ✓ Dispone de funciones de auto-rangos
- ✓ Etiquetado automático o manual de ejes
- ✓ Marcado automático o manual de las señales
- ✓ Ejes logarítmicos
- ✓ Gráficas animadas
- ✓ Zoom infinito
- ✓ Varios estilos de gráficas: líneas conectadas, barras,…
- ✓ Varios estilos de puntos en las gráficas: puntos, pixels,…
- ✓ Múltiples señales en un gráfica y leyendas
- ✓ Gráficas en color o blanco y negro
- ✓ Barras de error
- ✓ Gráficas editables
- ✓ Lenguaje XML para la especificación de las gráficas
- ✓ Representación de histogramas

Este paquete contiene un gran número de clases Java, documentación y una amplia variedad de aplicaciones de ejemplo que permiten la posibilidad de, entre otras cosas, crear nuevas clases derivadas de las proporcionadas para realizar y personalizar las gráficas. De esta manera se puede controlar completamente la colocación de las gráficas en cualquier punto de la pantalla y escribir el código Java que define los datos que se representarán, como interesa en el aplicativo Contenedor.

La clase base para la construcción de las gráficas será la clase *Plot*, que a su vez se extiende de la clase *PlotBox*. Esta clase permitirá realizar zooms sobre la gráfica, compresión o expansión de ejes, escoger el color…y todo tipo de posibilidades que se han aprovechado en el proyecto del aplicativo Contenedor.

6.1.1.2 Configuración

El aplicativo Contenedor está formado por una serie de ficheros de configuración que permiten diseñar y construir un sistema utilizándolo como plataforma.

Gracias al metalenguaje XML con el que están definidos todos los ficheros de configuración se ha hecho mucha más sencilla la realización del sistema DVB-T en applet de Java. A continuación se hace un resumen de los ficheros que componen el aplicativo Contenedor y que se han tenido que estudiar y rellenar para la realización del simulador. Para un estudio a fondo se recomienda la lectura [22].

6.1.1.2.1 Multiidioma

El aplicativo Contenedor es multiidioma, es decir está disponible inicialmente en los idiomas: Castellano, Catalán e Inglés, pero puede ser ampliable a cualquier otro idioma y no es necesario de ningún tipo de programación adicional, sólo habrá que traducir los literales utilizados.

La configuración de los idiomas se realiza mediante un fichero XML integrado en la propia aplicación y que es de la siguiente forma:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE idiomas SYSTEM "idiomas.dtd">
<idiomas>
<!—Castellano -->
<idioma id="es" nombre=”Castellano” defecto="true" />
<!-- Catalán -->
<idioma id="ca" nombre=”Català” />
<!-- Inglés -->
<idioma id="en" nombre=”English” />
</idiomas>
```

Tabla 6 Fichero XML de configuración de los idiomas del sistema

El fichero XML está definido mediante su fichero DTD: idiomas.dtd que indica como debe confeccionarse y que valores pueden tener sus distintos elementos.

En la realización del proyecto que se trata, se han utilizado los tres idiomas definidos por defecto, por lo tanto no se ha tenido que modificar el fichero que se observa en la Tabla 6.

Para cada idioma que aparezca en el fichero XML de configuración, existe un fichero de recursos donde se encuentran los textos que el sistema debe mostrar para ese idioma. Por lo que hay tantos ficheros como idiomas definidos en el fichero XML, en nuestro caso tres.

El nombre de estos ficheros de recursos siempre tiene el siguiente formato: *textos_idIdioma.properties.* Por ejemplo: *textos_en.properties* para inglés. El contenido de este fichero tendrá una estructura del tipo: *Identificador_texto=Valor_texto.* Donde el *Identificador_texto* es la referencia que se debe utilizar siempre que se quiere escribir un texto en la aplicación. Estos identificadores estarán definidos en el resto de ficheros de configuración XML de la aplicación. El *Valor_texto* es el texto que se le desea dar al correspondiente identificador en el idioma al que pertenece el fichero de recursos.

6.1.1.2.2 Sistema

El aplicativo Contenedor está formado por un conjunto de etapas, por lo que tiene un fichero de configuración XML mediante el cual se pueden configurar todas las etapas que contiene el sistema a implementar, ocho etapas en nuestro caso.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE sistema SYSTEM "sistema.dtd">
<sistema nombre="sistema.nombre.ejemplo" ayuda="ayudaSistema.xml">
<etapa class="sistema.ejemplo.etapa.EtapaEjemplo" id="ejemplo1"
nombre="etapa.ejemplo1.nombre" conf="etapaEjemplo1.xml"/>
<etapa class="sistema.ejemplo.etapa.EtapaEjemplo2" id="ejemplo2"
nombre="etapa.ejemplo2.nombre" conf="etapaEjemplo2.xml"/>
</sistema>
```

Figura 84 Ejemplo de fichero de configuración de un sistema.

En la Figura 84 se puede ver una configuración para el caso de un sistema de dos etapas.

El fichero de configuración XML está definido por su fichero de definición o sintaxis DTD: *sistema.dtd*, que marca como debe confeccionarse y qué valores pueden tener todos sus elementos.

El elemento principal es "sistema" y está formado por uno o más elementos "*etapas*". Cada elemento "etapa" definirá los parámetros que debe tener una etapa dentro de un sistema.

En el proyecto realizado, se han utilizado estos dos elementos para definir el sistema DVB-T y sus ocho etapas. Para ello se han definido los atributos de ambos teniendo en cuenta las definiciones de cada uno de ellos.

Atributo	**Descripción de los atributos del elemento "*sistema*"**
nombre	Es la etiqueta del nombre del sistema que se muestra en la pantalla inicial del Applet. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties*' en el directorio *'recursos'*).
ayuda	Es el nombre del fichero de configuración de la ayuda que tendrá el sistema. Este atributo no es obligatorio y si no se informa el sistema no tendría disponible ninguna ayuda.

Tabla 7 Descripción del elemento sistema

Atributo	**Descripción de los atributos del elemento "*etapa*"**
class	Es el nombre completo de la clase Java que implementa dicha etapa. Esta clase debe ser una extensión de la clase: "*sistema.Etapa*".
id	Es el identificador de la etapa, debe ser único entre todas las etapas configuradas en el sistema. Debe comenzar siempre por una letra y sólo permite los siguientes caracteres: letras, números, puntos, guiones, subrayados y los dos puntos.
nombre	Es la etiqueta del nombre de la etapa que se muestra en cada botón que activa la etapa en la pantalla inicial del Applet. Esta etiqueta debe que tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties*' en el directorio '*recursos*').
conf	Es el nombre del fichero de configuración de la etapa que define la etapa, sus parámetros, sus resultados y su visualización. El fichero que se cree debe tener la estructura del siguiente dtd: "*etapa.dtd*".
ayuda	Es el nombre del fichero de configuración de la ayuda que tiene la etapa. Este atributo no es obligatorio y si no se informa la etapa no tiene disponible ninguna ayuda.

Tabla 8 Descripción del elemento etapa

6.1.1.2.3 Etapa

Cada etapa de un sistema está definida por un fichero de configuración XML, en el que se definen los parámetros de entrada que el usuario puede introducir, los resultados que se generarán para el resto de etapas y las gráficas que se visualizarán cuando se ejecute.

Se puede definir que el fichero de configuración XML de una etapa estará formado por tres elementos: "*entrada*", "*senyales" y "salida"*.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etapa SYSTEM "etapa.dtd">
<etapa>
      <entrada>
      .....
      </entrada>
      <senyales>
      .....
      </senyales>
      <salida>
      .....
      </salida>
</etapa>
```

Figura 85 Ejemplo de fichero de configuración de una etapa

El fichero de configuración XML está definido por su fichero de definición o sintaxis DTD: *etapa.dtd*, que marca como debe confeccionarse y qué valores pueden tener todos sus elementos.

6.1.1.2.3.1 Parámetros de entrada

Dentro del elemento "*entrada*" se define como son los parámetros que el usuario puede introducir en la etapa, si hay algún tipo de validación automática de dichos parámetros y si unos dependen de otros.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etapa SYSTEM "etapa.dtd">
<etapa>
  <entrada>
    <parametros nombre="parametros.ejemplo1.nombre">
      <parametro        id="param1"        nombre="param1.ejemplo.nombre"
tipo="TextField" defecto="1000">
        <validador tipo="obligatorio"></validador>
        <validador tipo="entero" maximo="100" minimo="0"></validador>
      </parametro>
      <parametro        id="param2"        nombre="param2.ejemplo.nombre"
tipo="ComboBox" defecto="1">
               <valor codigo="1" descripcion="valor1.ejemplo.id"/>
               <valor codigo="F" descripcion="Archivo.id"/>
      </parametro>
    </parametros>
  </entrada>
  <senyales>
  .....
  </senyales>
  <salida>
  .....
  </salida>
</etapa>
```

Figura 86 Ejemplo de un fichero de configuración de una etapa con parámetros de entrada

- ***Elemento parámetros:***

El elemento "*entrada*" está formado por elementos "*parametros*" o puede estar vacío si no tiene elementos. Los parámetros de entrada están agrupados dentro de un grupo de parámetros a través del elemento "*parametros*" que puede tener el siguiente atributo:

Atributo	**Descripción de los atributos del elemento "*parametros*"**
nombre	Es la etiqueta del nombre del grupo de parámetros que se muestra al inicio de los parámetros que contenga este grupo. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties*' en el directorio *'recursos'*). Este atributo no es obligatorio y si no se informa no se pone ningún nombre antes de los parámetros.

Tabla 9 Descripción del elemento parámetros

- ***Elemento parámetro:***

El elemento "*parametro*" está formado por ninguno o más elementos "*validador*", por ninguno o más elementos "*dependencia*" y por ninguno o más elementos "*valor*". Con los siguientes atributos se define el tipo de parámetro que verá el usuario para entrar los datos de la etapa:

Atributo	**Descripción de los atributos del elemento "*parametro*"**
id	Es el identificador del parámetro, debe ser único entre todos los parámetros de la etapa. Debe comenzar siempre por una letra y sólo permite los siguientes caracteres: letras, números, puntos, guiones, subrayados y los dos puntos. Mediante este identificador el programador puede recoger o cargar

	el valor del parámetro.
nombre	Es la etiqueta del nombre del parámetro que se muestra en la etapa. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties*' en el directorio *'recursos'*).
tipo	Es el tipo de parámetro que se muestra al usuario. Puede tener los siguientes valores: 1. *TextField*: Campo de texto en el que el usuario puede introducir un valor. 2. *ComboBox*: Lista desplegable de valores en el que el usuario puede elegir un valor. (Ver la descripción del elemento "*valor*") 3. *ScrollBar*: Barra deslizante con la que el usuario puede moverse entre un valor mínimo y un valor máximo.
defecto	Es el valor por defecto que tiene el parámetro cuando se carga la primera vez la etapa. NOTA: Si el valor es decimal se debe emplear el '.' como separador decimal.
maximo	Este atributo sólo tiene importancia cuando el tipo de parámetro es *'ScrollBar'*, entonces será el valor máximo que puede tener este parámetro.
minimo	Este atributo sólo tiene importancia cuando el tipo de parámetro es *'ScrollBar'*, entonces será el valor mínimo que puede tener este parámetro.
escala	Este atributo sólo tiene importancia cuando el tipo de parámetro es *'ScrollBar'*, entonces será el valor de la escala que se quiera aplicar para moverse una unidad con el scroll. Este valor siempre debe ser un entero.

Tabla 10 Descripción del elemento parámetro

- ***Elemento valor:***

El elemento "*valor*" es la lista de valores que puede tener un parámetro tipo *'ComboBox'*. Por lo que sólo tiene sentido y sólo se tendrán en cuenta si el parámetro es un *'ComboBox'*. Tendrá los siguientes atributos:

Atributo	**Descripción de los atributos del elemento "*valor*"**
codigo	Es el valor que tiene este elemento de la lista, debe ser único entre todos los valores de un parámetro. Existe la posibilidad de que mediante un elemento de la lista se desee que el usuario introduzca un fichero con una señal, si este es el caso hay que colocar como valor del atributo "F".
descripcion	Es la etiqueta del nombre del elemento de la lista que ve el usuario. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties*' en el directorio *'recursos'*).

Tabla 11 Descripción del elemento valor

- ***Elemento validador:***

El elemento "*validador*" sirve para que un parámetro de entrada se valide automáticamente cuando el usuario clicke el botón *'Validar'* de la etapa. Si se produce algún error en algún parámetro el sistema devuelve un mensaje de error y corrige el valor del parámetro que tiene el error.

Atributo	**Descripción de los atributos del elemento "*validador*"**
tipo	Es el tipo de validación que se realiza sobre el parámetro. Puede tener los siguientes valores: 1. *obligatorio*: Verifica que el usuario haya introducido un valor en el parámetro sino pone el valor por defecto. 2. *entero*: Verifica que el valor del parámetro introducido sea entero si no es así y lo puede redondear a entero lo redondea, sino pone el valor por defecto del parámetro. 3. d*ecimal*: Verifica que el valor del parámetro introducido sea decimal y tenga el número de decimales indicados en el atributo *'num_decimales'*. Si no es así y puede redondearlo lo hace y sino pone el valor por defecto.
num_decimales	Este atributo sólo tiene importancia cuando el tipo de validador es *'decimal'*, entonces será el número de decimales que se validarán. Este valor siempre debe ser un entero.
maximo	Este atributo sólo tiene importancia cuando el tipo de validador es *'entero'* o *'decimal'*, entonces será el valor máximo que puede tener el valor de este parámetro.
minimo	Este atributo sólo tiene importancia cuando el tipo de validador es *'entero'* o *'decimal'*, entonces será el valor mínimo que puede tener el valor de este parámetro.

Tabla 12 Descripción del elemento validador

- ***Elemento dependencia:***

El elemento "*dependencia*" sirve para que un parámetro de entrada sea visible o no dependiendo del valor de otro parámetro de la etapa. De esta manera podemos mostrar automáticamente al usuario unos parámetros u otros según el valor que va seleccionando en los parámetros.

Atributo	**Descripción de los atributos del elemento "*dependencia*"**
id_parametro	Es el identificador del parámetro (atributo *'id'* del elemento '*parametro'*) del que dependerá si se visualiza o no el parámetro con esta dependencia.
valor	Es el valor del parámetro del que depende si se visualiza o no el parámetro con esta dependencia.

Tabla 13 Descripción del elemento dependencia

6.1.1.2.3.2 Señales

Una vez se han definido los parámetro de entrada de la etapa que se va a configurar, se han de implementar las señales que se van a mostrar en la pantalla de la etapa.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etapa SYSTEM "etapa.dtd">
<etapa>
   <entrada>
   .....
   </entrada>
   <senyales>
       <grupo_senyales        tipo="RECTANGULAR"        defecto="senyal1"
orden_carga="1">
          <senyal          class="sistema.ejemplo.senyales.SenyalEjemplo"
id="senyal1"
            nombre="senyalEjemplo1.nombre"
titulo="titulo.senyalEjemplo1"
           etiquetaX="xlabel.senyalEjemplo1"
etiquetaY="ylabel.senyalEjemplo1" conectada="true"
           num_graficas="5" num_puntos="400">
         </senyal>
      </grupo_senyales>
   </senyales>
   <salida>
    .....
   </salida>
</etapa>
```

Figura 87 Ejemplo de un fichero de configuración de una etapa con señales.

- ***Elemento grupo_senyales:***

El elemento *"senyales"* está formado por ninguno o más elementos *"grupo_senyales"*. Un grupo de señales agrupa un conjunto de señales que se muestran siempre en la misma zona de la pantalla y tienen siempre el mismo tamaño.

Atributo	**Descripción de los atributos del elemento "*grupo_senyales*"**
tipo	Es el tipo de señal y marca el tamaño que tienen todas las señales que hay en este grupo. Puede tener los siguientes valores: 1. *CUADRADA*: Señal con tamaño cuadrada. Como máximo puede haber 4 grupos con este tipo dentro de una etapa. 2. *RECTANGULAR*: Señal con tamaño rectangular. Como máximo puede haber 2 grupos con este tipo en una etapa. 3. *CUADRADA_UNICA*: Señal con tamaño cuadrada. Como máximo puede haber 1 grupo con este tipo en una etapa ya que ocupa toda la ventana. 4. *RECTANGULAR_UNICA*: Señal con tamaño rectangular. Como máximo puede haber 1 grupo con este tipo en una etapa ya que ocupa toda la ventana.
defecto	Es el identificador de la señal dentro de este grupo que se ve por defecto cuando el usuario clicke el botón *'Validar'*. Coincide con el atributo *id* de elemento *'senyal'*.
orden_carga	Es un número entero que indica el orden con el que el sistema ejecuta todos los grupos de señales de la etapa.

Tabla 14 Descripción del elemento grupo_senyales

- ***Elemento senyal:***

Cada grupo de señales esta formado uno o más elementos *"senyal"*. Cada elemento *"senyal"* define los parámetros que debe tener una señal dentro de una etapa.

Atributo	**Descripción de los atributos del elemento "*senyal*"**
class	Es el nombre completo de la clase Java que implementa dicha señal. Esta clase debe ser una extensión de la clase: [”sistema.senyales.Senyal”]
id	Es el identificador de la señal, debe ser único entre todas las señales configuradas en la etapa. Debe comenzar siempre por una letra y sólo permite los siguientes caracteres: letras, números, puntos, guiones, subrayados y los dos puntos.
nombre	Es la etiqueta del nombre de la señal que se muestra en el menú *'Señales'* de la etapa. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros *'*.properties'* en el directorio *'recursos'*)
titulo	Es la etiqueta del título de la señal que se muestra encima de la gráfica. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties'* en el directorio *'recursos'*)
etiquetaX	Es la etiqueta del nombre del eje X de la señal. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '*.properties' en el directorio *'recursos'*)
etiquetaY	Es la etiqueta del nombre del eje Y de la señal. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '*.properties' en el directorio *'recursos'*).
num_graficas	Es el número de gráficas que se pintan. Este valor siempre debe ser un entero.
conectada	Indica si los puntos de la gráfica están conectados o no. Los valores que puede tener son: 1. *true*: Los puntos de la gráfica están conectados. 2. *false*: Los puntos de la gráfica no están conectados. Este atributo no es obligatorio y si no se informa tiene el valor "*true*"
tipo_puntos	Indica el tipo de puntos de la gráfica si está conectada. Los valores que puede tener son: 1. "*none*": Sin ningún punto 2. "*points*": Pequeños puntos 3. "*dots*": Puntos más grandes 4. "*various*": Cada gráfica tiene una marca, empieza por un punto del mismo tamaño que "*dots*" y hay diez marcas diferentes, luego se repiten 5. "*pixels*": Puntos de un sólo pixel Este atributo no es obligatorio y si no se informa tiene el valor "*various*".
stems	Indica si los puntos de la gráfica tienen una línea hasta el eje de las X. Los valores que puede tener son: 1. *true*: Los puntos de la gráfica se conectan con el eje X. 2. *false*: Los puntos de la gráfica no se conectan con el eje Y. Este atributo no es obligatorio y si no se informa tiene el valor "*false*".

histograma	Indica si la gráfica es un histograma. Los valores que puede tener son: 1. *true*: La gráfica es un histograma. 2. *false*: La gráfica no es un histograma. Este atributo no es obligatorio y si no se informa tiene el valor "*false*".
num_puntos	Es el número de puntos que tiene la gráfica. Este atributo sólo tiene sentido si se implementa el método de la clase [”sistema.senyales.Senyal”]: *public abstract Double formula(double coordenadaX, int numeroGrafica)* Sino se implementa este método el atributo se ignora. Este valor siempre debe ser un entero.

Tabla 15 Descripción del elemento senyal

6.1.1.2.3.3 Parámetros de salida

Por último, en el fichero de configuración de una etapa hay que definir los parámetros de salida que se van a transmitir para el resto de etapas del sistema, así como los parámetros o resultados numéricos que se van a mostrar al usuario una vez se ha validado la etapa.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE etapa SYSTEM "etapa.dtd">
<etapa>
     <entrada>
     .....
    </entrada>
    <senyales>
    …..
    </senyales>
    <salida>
      <parametro_salida                    id="paramSalida2Ejemplo"
      nombre="paramSalida2.ejemplo.nombre" visible="false"
      cargar="true" guardar="true"/>
      <parametro_salida                    id="paramSalida3Ejemplo"
      nombre="paramSalida3.ejemplo.nombre" visible="true"/>
    </salida>
</etapa>
```

Figura 88 Ejemplo de fichero de configuración de una etapa con parámetros de entrada.

El elemento "*salida*" está formado por ninguno o por más elementos "*parametro_salida*". Cada elemento "*parametro_salida*" define un parámetro de salida que la etapa puede compartir y transmitir con el resto de etapas del sistema.

Atributo	**Descripción de los atributos del elemento "*parametro_salida*"**
id	Es el identificador del parámetro, debe ser único entre todos los parámetros de la etapa. Debe comenzar siempre por una letra y sólo permite los siguientes caracteres: letras, números, puntos, guiones, subrayados y los dos puntos. Mediante este identificador el programador podrá recoger o cargar el valor de un parámetro.
nombre	Es la etiqueta del nombre del parámetro que se muestra en la etapa y el

	menú. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '**.properties'* en el directorio *'recursos'*).
visible	Indica si el valor del parámetro es visible en la etapa o no para el usuario. Los valores que puede tener son: 1. *true*: El parámetro se visualiza. 2. *false*: El parámetro no se visualiza. Este atributo no es obligatorio y si no se informa tiene el valor "*false*".
cargar	Indica si el parámetro se puede cargar desde un fichero a través del menú. Los valores que puede tener son: 1. *true*: Se puede cargar. 2. *false*: No se puede cargar. Este atributo no es obligatorio y si no se informa tiene el valor "*false*".
guardar	Indica si el parámetro se puede guardar a un fichero a través del menú. Los valores que puede tener son: 1. *true*: Se puede guardar. 2. *false*: No se puede guardar. Este atributo no es obligatorio y si no se informa tiene el valor "*false*".

Tabla 16 Descripción del elemento parámetro de salida

6.1.1.2.4 Ayudas

El aplicativo Contenedor tiene un sistema de ayudas a las que el usuario puede dirigirse para aclarar aspectos referentes al uso global del sistema o aspectos particulares de cada etapa. Este sistema de ayudas como todo el Contendedor es configurable por el programador que implemente un sistema concreto y se realiza mediante ficheros de configuración XML que tienen un aspecto parecido al siguiente:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ayudas>
    <ayuda nombre="nivel1.nombre" fichero="nivel1.url">
      <ayuda nombre="nivel2.nombre" fichero="nivel2.url"/>
      <ayuda nombre="nivel3.nombre" fichero="nivel3.url"/>
    </ayuda>
    <ayuda nombre="nivel1.nombre" fichero="nivel1.url"/>
</ayudas>
```

Figura 89 Ejemplo de fichero de configuración de una ayuda

El elemento principal del XML es *"ayudas"* y está formado por uno o más elementos *"ayuda"*. Cada elemento *"ayuda"* a su vez puede estar formado por ninguno o más elementos "*ayuda"*, consiguiendo un efecto de árbol y de anidación entre elementos *"ayuda"*.

Todos estos elementos son multiidioma por lo que las ayudas pueden verse en el idioma en el que se está visualizando el sistema.

Atributo	**Descripción de los atributos del elemento "*ayuda*"**
nombre	Es la etiqueta del nombre de la ayuda que se muestra en el árbol de la página

	de ayuda. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '*.*properties'* en el directorio *'recursos'*).
fichero	Es la etiqueta del fichero que muestra la ayuda. Esta etiqueta debe tener sus equivalencias en los ficheros de cada idioma configurado para el applet (ficheros '*.*properties'* en el directorio *'recursos'* en el que se indicará la página para cada idioma).

Tabla 17 Descripción del elemento ayuda

Los ficheros que se visualizan en las ayudas son ficheros en formato HTML o también en formato XML + XSLT (Extensible Stylesheet Language Transformations). XSLT es un lenguaje basado en XML que transforma un documento XML en otro documento XML o transforma un documento XML en cualquier otro documento estructurado como puede ser HTML. En el caso de las ayudas la transformación debe realizarse para crear un fichero HTML.

6.1.1.3 Tipos de datos

Para que facilitar la gestión de las señales necesarias para el procesado de una etapa, se ha definido un formato de señales y unos ficheros de intercambios de datos con otros sistemas. Este formato se sigue en el caso del simulador en MATLAB también.

6.1.1.3.1 Intercambio de ficheros

Mediante las siguientes especificaciones se determina el formato de los ficheros de señal que el sistema intercambia entre diferentes ejecuciones.

- ***Funcionalidades:***

Se pueden intercambiar ficheros de señal según las condiciones siguientes:

- Una señal guardada mediante una ejecución en Matlab o Java se puede leer desde otra ejecución Matlab o Java, es decir, ambos lenguajes pueden intercambiar ficheros de señal.
- Los ficheros se pueden leer, modificar o preparar desde Excel.

La etapa desde la que se ha guardado la señal debe coincidir con la etapa desde la que se lee. El tipo de señal que se ha guardado debe coincidir con el tipo de señal que se lee. *Ejemplo*: Si se ha guardado la señal modulada en la etapa 3, el tipo de fichero generado se puede leer únicamente en otra ejecución al llegar a la etapa 3 y llamándolo desde el correspondiente menú de la señal modulada. Además la longitud de la señal que se lee debe coincidir con la longitud definida para la correspondiente señal desde la ejecución.

- ***Formato de los ficheros:***

El formato de los ficheros de señal es de tipo texto. En los ficheros de señal se pueden guardar vectores de enteros que sean reales o complejos. Independientemente del tipo de señal que se esté grabando, los datos guardados son números enteros con signo. Cabe comentar que los datos se guardan cuantificados.

- El primer dato del fichero siempre es una variable de escalado a utilizar después de haber leído el fichero o antes de escribir en un fichero.

- El segundo dato es un entero con los siguientes valores:

    - Igual a 1 para el caso de guardar números complejos
    - Igual a 0 si la señal es real y sólo tiene parte real

- A continuación se depositan todos los datos en serie. Si la señal es imaginaria de x-muestras $a+bj$, primero están las x-muestras de parte real (*a's*) y a continuación las x-muestras de parte imaginaria (*b's*).

    - ***Ejemplo de fichero:***

Como ejemplo, se muestra a continuación la lectura de un fichero de datos complejos y los valores numéricos a que han de dar lugar una vez leídos.

```
256
1
256
-128
512
128
-256
-512
128
-256
```

Figura 90 Ejemplo de fichero de intercambio de señales

De este fichero de ejemplo se deduce lo siguiente de los dos primeros datos:

- Factor de escala = *256*
- Indicador de Complejos = 1

De los ocho datos restantes, debido a que la señal se debe interpretar como compleja, se dividen en dos partes: los datos terceros al sexto corresponden a la parte real y los datos séptimos a décimo corresponden a la parte imaginaria. Todos ellos, al leerlos, se deben dividir por el factor de escala. En resumen el vector de señal leído queda del siguiente modo:

$$x[n] = \begin{pmatrix} +1-j \\ -0.5-2j \\ +2+0.5j \\ 0.5-j \end{pmatrix}$$

Figura 91 Señal de ejemplo

- ***Recomendaciones de escritura:***

Cuando desde una aplicación cualquiera, ya sea MATLAB, excel, java, etc..., se tiene un vector y se debe formar el fichero correspondiente para ser leído por LaViCAD, el proceso a seguir debe ser el siguiente.

- Hallar el factor de escala: F a partir del máximo valor absoluto de los datos. (Ver la fórmula 1 donde puede verse la fórmula aplicada)

- Guardar la parte entera de cada dato multiplicado por el factor de escala. De este modo se garantiza que el valor máximo en valor absoluto se guarda con la máxima precisión.

$$F = \text{parte entera}\left[\frac{2^{15}}{\text{abs}\left(x[n]\right)}\right]$$

Figura 92 Factor de escala de ficheros de intercambios

6.1.1.3.2 Señales

Para trabajar con señales dentro del aplicativo Contenedor hay que tener en cuenta como se almacenan dichas señales.

- ***Señales para almacenar:***

Las señales se pueden almacenar en parámetros de entrada o en parámetros de salida de un sistema para su tratamiento y para el intercambio entre etapas o entre otros sistemas de LaViCAD. Se pueden gestionar señales de los siguientes tipos:

- Señales reales
- Señales complejas

Las señales siempre se guardan en un *array* de dos dimensiones del tipo de datos *double*. De esta manera una señal tiene el siguiente formato:

```
double[][] senyal = new double[2][n]
```

Figura 93 Formato de señales en Java

Donde *n* es la longitud de la señal. Además de este *array* para cada señal se tiene que controlar a través del flag *esSenyalReal* si se almacena una señal real o imaginaria.

Tipo señal	**Flag *esSenyalReal***	**Array *double[2][n]***
Señal Real	true	double[0][n] = No se tienen en cuenta double[1][n] = Datos de la señal
Señal Real	false	double[0][n] = Parte real de la señal double[1][n] = Todos los valores a cero

Señal Compleja	true	Este caso no se debe contemplar porque sino se perderá la componente double[0][n] ya que el flag marca que la señal es real.
Señal Compleja	false	double[0][n] = Parte real de la señal double[1][n] = Parte imaginaria de la señal

Tabla 18 Relación entre tipo de señal y flag.

- ***Señales a representar:***

Cuando se quiere representar una señal en una gráfica, el formato de la señal ha de ser siempre el siguiente:

```
      double[][] senyal = new double[2][n]
Donde:
 double[0][n] serán los valores de la componente X de la
señal a representar
 double[1][n] serán los valores de la componente Y de la señal
a representar
```

Figura 94 Formato señal para representación gráfica

6.1.1.4 Javadoc

El aplicativo Contenedor dispone de una documentación de todas sus clases hecha mediante Javadoc, que ha sido de suma utilidad en el desarrollo del applet de Java del sistema DVB-T.

Javadoc es una utilidad de Sun Microsystems (http://www.sun.com) que permite generar documentación en formato HTML de un código fuente Java. Por cada clase Java se genera una página HTML que describe cuales son sus atributos y sus métodos y como deben utilizarse. Para generar esta documentación ha de utilizarse una serie de etiquetas en los comentarios de las clases programadas que después permiten generar estos documentos que hemos utilizado para saber como utilizar cada clase que proporciona el aplicativo Contenedor.

Figura 95 Ejemplo Javadoc Aplicativo Contenedor

6.1.1.5 Firma de applets

El aplicativo Contenedor es un applet, ello implica que por defecto no puede acceder a los recursos del ordenador donde se está ejecutando, ni disco duro, ni impresora ni ningún otro dispositivo. Tampoco puede establecer conexiones con otros ordenadores que no sean el servidor de dónde viene.

El motivo de esta restricción es la seguridad. Ya que si esto fuera posible cualquier persona podría crear un applet que nos destruyera el ordenador o nos robara información que tuviéramos almacenada en nuestro disco duro. Es por este motivo que el navegador donde se ejecuta un applet no permite tener acceso al ordenador del usuario.

Sin embargo, para el aplicativo Contenedor es necesario acceder al disco duro para así poder cargar y guardar ficheros de señales. La forma de conseguir esto es firmar el applet digitalmente.

Cuando un navegador visualiza un applet firmado, le muestra al usuario un aviso indicando que el applet está firmado digitalmente y le proporciona al usuario la posibilidad de confiar en el certificado digital de este applet o no. Si el usuario acepta este mensaje el applet podrá acceder a su ordenador local y si lo rechaza el applet no tendrá permisos de acceso y las funcionalidades que necesiten realizar estos accesos no funcionarán correctamente.

6.1.2 Despliegue, guía para realizar un nuevo sistema

En el presente capítulo se explica como, una vez se ha programado el sistema DVB-T, o cualquier otro sistema, se compila y se crea un applet para ser publicado en un servidor

web para que sea visible desde Internet. Para llegar a este fin se ha de crear una estructura de directorios donde se debe organizar el código fuente, librerías, scripts de creación de ejecutable, etc.

Pasos a realizar para crear un nuevo sistema:

1) Crear la siguiente estructura de directorios para programar el nuevo sistema, todos estos directorios pueden estar dentro de un directorio con el nombre del nuevo sistema:

    - ***classes/conf***: Directorio donde se crearán los ficheros de configuración del sistema
    - ***classes/recursos***: Directorio donde se crearán los ficheros de recursos que contendrán los textos para cada uno de los idiomas configurados en el sistema
    - ***classes/etapas***: Directorio donde se copiarán las clases compiladas (ficheros *.*class*) de cada una de las etapas creadas para el sistema.
    - ***src/etapas***: Directorio donde se crearán las clases del código fuente (ficheros *.*java*) de todas las etapas del sistema.
    - ***lib***: Directorio donde se copiarán las librerías del aplicativo Contendor
    - ***lib/ayudas***: Directorio donde estarán las páginas HTML de las ayudas y las posibles imágenes que puedan tener las ayudas.
    - ***etc***: Directorio donde estará el script para crear el ejecutable y firmar el applet.

    La estructura de directorios utilizada en el desarrollo del applet de Java simulador de la capa física del sistema DVB-T es la que sigue:

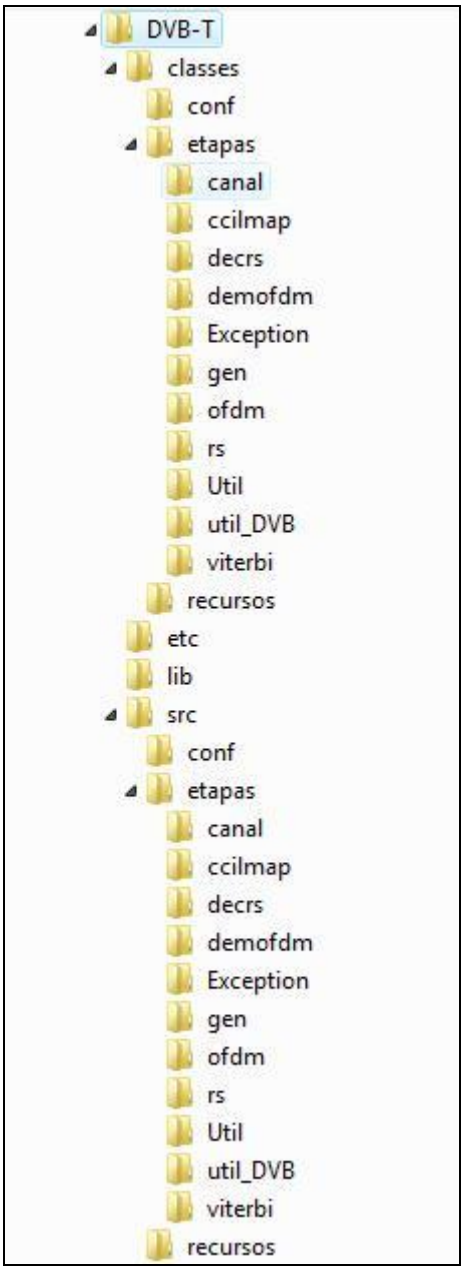


Figura 96 Estructura de directorios para el sistema DVB-T

2) Dentro del directorio '*classes/conf*' se crearán los siguiente archivos:

   - Fichero de configuración del sistema: *'sistema.xml'* tal y como está definido en el apartado 6.1.1.2.2.
   - Ficheros XML's de definición de cada etapa explicados en el apartado 6.1.1.2.3.
   - Fichero de configuración de los idiomas en los que estará configurado el sistema según lo descrito en el apartado 6.1.1.2.1.
   - Fichero de definición de las ayudas del sistema y de las etapas tal y como está definido en el apartado 6.1.1.2.4.

3) Dentro del directorio *'classes/recursos'* se crearán los ficheros que contienen los literales utilizados en la aplicación, debe de haber un fichero por cada uno de los

idiomas definidos en el sistema. En simulador desarrollado en el presente proyecto, se dispone de de tres idiomas, los ficheros son los siguientes:

- 'textos_ca.properties' para el catalán.
- 'textos_es.properties' para el castellano.
- 'textos_en.properties' para el inglés.

4) Dentro del directorio *'src/etapas'* se crearán los ficheros del código fuente de nuestro sistema que implementará el procesado de cada una de las etapas (ficheros **.java*). Se recomienda tener los archivos de cada etapa en un directorio dentro de éste con su nombre para la mejor organización del código y su posterior mantenimiento.

5) Dentro del directorio *'classes'* se copiarán las clases compiladas de las etapas programadas en el punto anterior (ficheros **.class*).

6) En el directorio *'lib'* se deben copiar las librerías del aplicativo Contenedor que son necesarias para la compilación del código fuente del punto 4 y para la ejecución del applet. Las librerías serán las siguientes:

   - *'contenedor.jar'*: Librería del aplicativo Contenedor
   - *'plotmlapplet.jar'*: Librería utilizada para la representación gráfica (Ptolemy II) descrito en el apartado 6.1.1.1.3.
   - *'SciLib.jar'*: Librería gratuita científica de funciones matemáticas utilizada para realizar ciertas funciones de procesado, desarrollada por Ostfold College Lab. [http://www.ia.hiof.no/~kyvh/SciLib/docs/scilib.html].

   Dentro de este directorio también se tendrá una página HTML que será la que habrá que colocar en el servidor web donde se desee publicar nuestro sistema. El contendido de esta página web sería el siguiente:

```
<html>
    <title>Sistema DVB-T</title>
    <body>
    <applet code="sistema.Contenedor"
      archive="contenedor.jar,plotmlapplet.jar,SciLib.jar,sist
ema.jar"
      height="450" width="650" MAYSCRIPT>
    </applet>
    </body>
</html>
```

Figura 97 Página HTML para publicar el sistema DVB-T

7) Dentro del directorio *'etc'* se creará un fichero que permitirá crear el applet de nuestro sistema, este fichero ejecutable tendrá el siguiente nombre '*crearApplet.bat'* y su contenido será el que se muestra a continuación:

```
set PATH=%PATH%;c:\j2sdk1.4.1_07\bin
cd ..\classes
jar cvf ..\lib\sistema.jar conf etapas recursos
cd ..\lib
IF EXIST keystore del keystore
keytool -genkey -alias signFiles -keystore keystore -keypass signFiles -dname
"CN=Lavicad,
OU=COMALAWEB, O=UPC, L=Barcelona, S=Catalunya, C=ES" -storepass keystore
jarsigner -keystore keystore -storepass keystore -keypass signFiles contenedor.jar
signFiles
jarsigner -keystore keystore -storepass keystore -keypass signFiles SciLib.jar
signFiles
jarsigner -keystore keystore -storepass keystore -keypass signFiles plotmlapplet.jar
signFiles
jarsigner -keystore keystore -storepass keystore -keypass signFiles sistema.jar
signFiles
cd ..\etc
```

Figura 98 Script de creación de un sistema

NOTA: En la primera línea del fichero se hace referencia al JDK que se debe tener instalado en el ordenador donde se vaya a programar el sistema, de no ser así, habría que descargarlo de [http://java.sun.com].

Este fichero ejecutable realiza el ejecutable de nuestro sistema y firma con un certificado digital todas las librerías necesarias para su ejecución, tal y como se explica en el apartado 6.1.1.5.

8) Finalmente para crear el sistema y una vez se hayan realizado todos los pasos anteriores, se ejecutará el archivo situado en *'etc/crearApplet.bat'* y se dispondrá, en el directorio *lib*, de todos los archivos necesarios para colocar el sistema en un servidor web.

6.2 Retos en la programación Java

En la implementación Java del simulador DVB-T, se han afrontado los retos intrínsecos al procesado de señal interno de cada etapa, así como a retos propios de la programación Java. En la sección 4.3 de la memoria se ha explicado, etapa a etapa, la explicación detallada de los retos correspondientes al procesado de señal.

Cabe destacar que la realización del applet de Java fue posterior al simulador MATLAB, por lo tanto se ha utilizado la programación del procesado de señal de las etapas del sistema, como apoyo a la programación Java.

La mayor dificultad, aparte del cambio de lenguaje de programación utilizado, ha sido que en Java no se dispone de subrutinas de procesado como el codificador Reed-Solomon o el codificador convolucional ya implementadas en una toolbox del programa como pasaba en MATLAB. También se destacaría que se pierde toda la capacidad que se tenía con MATLAB para el tratamiento de matrices. No obstante, se ha consultado la implementación del sistema OFDM realizado por Carlos y Francisco Vargas en el proyecto LaViCAD para poder emplear subrutinas de procesado como el cálculo de la convolución de dos señales, el cálculo de la *ifft*, el de la *fft* y la autocorrelación de dos señales.

Como en la programación MATLAB se han ido realizando comprobaciones para asegurar el correcto funcionamiento del simulador, se ha utilizado dicha programación para comprobar que el applet de Java funcionara correctamente.

6.2.1 Memoria de la consola de Java

Se trata de un problema que se ha encontrado en la programación, debido a la limitación de memoria de 64 MB que tiene por defecto asignada la consola de Java en cualquier ordenador.

En el transcurso de la programación del applet de Java del simulador del sistema DVB-T se ha encontrado que, dependiendo de la simulación que se realizaba, se mostraba el siguiente mensaje de error en la consola de Java: "*Out of memory error*". Después de monitorizar la memoria usada con "*long a=run.totalMemory ()"*, "*long b= run.freeMemory ()"* y "*long c= run.maxMemory"* se lograron detectar los casos y causas de error.

La causa del error que se detectó, era que la memoria que necesitaba el applet para ejecutar el procesado programado era superior a los 64MB que por defecto tiene asignados la consola de Java. Los casos de error eran cuando se seleccionaban los parámetros de entrada adecuados para obtener las señales de mayor número de muestras.

La primera medida que se ha tomado para solucionar el problema es el empleo del *garbage collector*. Así pues, tras cada rutina de procesado se dejan todos los objetos que ya no se necesitan apuntando a *null.* A continuación se invoca al *garbage collector* y de esta forma se fuerza que los libere de memoria.

La segunda medida, es la optimización de código estudiando el flujo de ejecución del applet. De esta manera se ejecuta el applet de forma eficiente.

Seguidas estas soluciones, es posible ejecutar el sistema completo por debajo de los 64 MB de consumo de memoria en el caso de seleccionar los parámetros más básicos de procesado.

Para poder ejecutar la versión completa el usuario que ejecute dicho applet deberá introducir manualmente la instrucción "*–Xmx170m*" en la columna "Parámetros del entorno de ejecución de Java" del formulario "Versiones de ejecución de Java" dentro del panel de control de Java de su ordenador. Con esta instrucción se permite al applet disponer de hasta 170 MB de memoria del ordenador para su ejecución.

6.3 Diferencias entre JAVA y MATLAB

Las diferencias entre estos dos lenguajes de programación son ampliamente conocidas. Mientras que Java es un lenguaje de programación orientado a objetos, MATLAB es lenguaje de programación de muy alto nivel con un elevado grado de interactividad y flexibilidad en la implementación de algoritmos. Para la ejecución de un programa realizado en Java se requiere una previa compilación, mientras que mediante MATLAB,

no se requiere la etapa de compilación y las etapas de escritura de código y ejecución pueden ser simultáneas (*command driven*).

Para el sistema implementado en Java, se destaca la ventaja de que puede ser ejecutado sin necesidad de licencia y puede ser implementado en forma de applet y colgado en web para ser ejecutado en modo remoto, con la única necesidad de que el usuario sólo ha de disponer de un navegador web con el plugin de java instalado.

También es importante destacar la facilidad de programación que tiene Java a la hora de realizar aplicaciones visuales. Para los sistemas implementados en MATLAB se destaca la ventaja de facilidad de programación, interacción y prestaciones gráficas, especialmente en aplicaciones de procesado de señal, álgebra vectorial y tratamiento estadístico de datos. El inconveniente de las aplicaciones desarrolladas con MATLAB, viene dado por la necesidad de licencias de Mathworks para su ejecución.

En el cambio de programación de MATLAB a Java cabe destacar que:

- Se deben inicializar todas las variables al formato que van a guardar, mientras en MATLAB el programa se lo adjudica automáticamente.

- En Java ya no se puede optimizar código vectorizando los algoritmos, pero en contrapartida es más rápido ejecutando bucles.

- En Java tiene una gran utilidad el depurador o *Debugger*. En MATLAB existen otras alternativas como hacer que el código se imprima en pantalla, o consultar el workspace directamente, que está visible en la ejecución normal del simulador.

- En Java se tiene un problema importante si se desea crear un sistema con señales de gran longitud. Aún y el empleo que se ha hecho del *garbage collector* y la optimización de código llevada a cabo hay etapas que tienen una duración por encima de lo deseado debido a que deben dibujar gráficas de multitud de puntos, o que deben ejecutar algoritmos costosos…en MATLAB no ocurre esto.

En resumen, un applet de Java es una buena solución para poder colgar un simulador de un sistema de comunicaciones en Internet, pero se deben tener en cuenta restricciones de memoria de la consola de Java. Con todo esto se recomienda una implementación más básica del simulador (enviando menos paquetes MPEG-2 o dibujando partes de la señal en lugar de la señal completa, todo esto en el simulador DVB-T) en el caso de Java, para que la ejecución del sistema se realice con fluidez, y no se tenga que recurrir a la solución ya comentada de aumentar la memoria de la consola.

Apéndice A. Lenguaje Java

En el desarrollo, tanto del Contenedor desarrollado por Carlos Vargas y Francisco Vargas en el proyecto LaViCAD [22], como del simulador del sistema DVB-T aquí explicado, se ha utilizado el lenguaje Java. En este apéndice se estudia la información más relevante de dicho lenguaje de cara al desarrollo de la aplicación.

A.1 Características

Esta sección trata sobre las características principales que ofrece Java respecto a cualquier otro lenguaje de programación, para una información más abundante se les remite a la referencia número [37].

Estas características son:

- ✓ Lenguaje simple
- ✓ Orientado a objetos
- ✓ Distribuido
- ✓ Interpretado y compilado a la vez
- ✓ Robusto
- ✓ Seguro
- ✓ Independiente a la arquitectura
- ✓ Portable
- ✓ Multihilo
- ✓ Dinámico
- ✓ Produce aplicaciones autónomas o applets

A.1.1 Orientado a objetos

La orientación a objetos es el paradigma de programación más utilizado actualmente en el desarrollo de aplicaciones. Esto se debe a la existencia de lenguajes como Java, C+ y otros, que aportan otras muchas características que los hacen fáciles de usar, potentes, compatibles con muchas plataformas y, sobre todo, seguros y fáciles de depurar.

El paradigma de programación orientada a objetos establece una asociación entre la información y las sentencias encargadas de gestionar dicha información. Desde un punto de vista de la programación procedimental, la orientación a objetos se puede ver como la asociación entre funciones y procedimientos a los tipos de datos que manejan.

En programación orientada a objetos una *clase* es el conjunto formado por la definición de la estructura de la información y las sentencias que la gestionan. A los subprogramas que gestionan los datos, se les denomina *métodos*. A cada uno de los ítems de información que definen la estructura de la información se les denomina *atributos* o *campos*. En esta forma de organizar el código el módulo mínimo de programación es la *clase*.

Cuando ejecutamos un programa orientado a objetos, a la entidad que tiene los valores de los ítems definidos en una clase se le denomina *objeto*.

Además de esta organización del código, existen primitivas de encapsulamiento, de forma que los atributos existentes en una clase quedan ocultos a los usuarios de esa clase. Para comunicar esa información se utilizan los métodos. Al conjunto de métodos de una clase que pueden ser usados por otras clases se le denomina *interfaz pública*.

Las principales aportaciones de este paradigma al desarrollo de sistemas software son:

- **Abstracción:** característica que consiste en aislar un elemento de su contexto o del resto de elementos que lo acompañan. En programación, el término se refiere al ¿qué hace? más que al ¿cómo lo hace? La abstracción encarada desde el punto de vista de la programación orientada a objetos expresa las características esenciales de un objeto, las cuales distinguen al objeto de los demás. Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real.

- **Encapsulamiento:** característica que consiste en ocultar los detalles de la implementación de un objeto, a la vez que se provee una interfaz pública por medio de sus métodos permitidos. También se define como la propiedad de los objetos de permitir acceso a su estado solamente a través de su interfaz o de relaciones preestablecidas con otros objetos.

  Cada objeto está aislado del exterior, es un módulo natural, y la aplicación entera se reduce a un agregado o rompecabezas de objetos. El aislamiento protege a los datos asociados a un objeto contra su modificación por quien no tenga derecho a acceder a ellos, eliminando efectos secundarios e interacciones.

- **Herencia**: característica que permite construir una clase (conocida como *clase hija*) basándose o teniendo como plantilla otra clase construida previamente (conocida como *clase padre*). Al utilizar una clase padre como base para definir una clase hija, los atributos y métodos definidos en la clase padre están disponibles en la clase hija. Es decir, los objetos de la clase hija tendrán valores para los atributos definidos en la clase hija y también tendrán valores para los atributos definidos en la clase padre. Esta característica permite una mayor reutilización del código, ya que si dos clases de un programa tienen ciertas características en común, se puede construir una clase que sea clase padre de las dos clases albergando los atributos y métodos que sean comunes a ambas.

- **Polimorfismo**: característica de la programación orientada a objetos, muy relacionada con la herencia, que solventa un conjunto de problemas típicos que aparecían en la programación procedimental. En multitud de aplicaciones es necesario gestionar información con distinta estructura de una forma homogénea.

  Utilizando herencia, en la clase padre hay que definir un método que no tiene implementación, llamado *abstracto*. La implementación de este método será establecido en cada una de las clases hijas.

A.1.2 Interpretado y compilado a la vez

Partiendo del hecho de que los lenguajes de programación deben traducirse en algún momento de manera que puedan ser interpretados por el microprocesador de la máquina, el proceso puede llevarse a cabo de dos modos diferentes según el momento: una vez codificado y antes de ejecutar el código, o bien, justo en el instante previo a la ejecución de éste. El primer caso corresponde a los lenguajes compilados, y el segundo a lenguajes interpretados. Java, se podría clasificar en ambos casos. En un primer momento se decidió que los programas desarrollados en Java debían sufrir una compilación como paso necesario previo a su ejecución. Sin embargo, a diferencia de lo sucedido en los compiladores tradicionales, el resultado de la compilación no es un conjunto de instrucciones máquina comprensible por un cierto microprocesador, sino una representación binaria genérica (*bytecodes*), que se ejecuta sobre cualquier máquina a la cual se hayan portado el intérprete y el sistema de ejecución en tiempo real (*runtime*).

A.1.3 Portable

Además de la portabilidad básica aportada por ser de arquitectura independiente, Java implementa otros estándares de portabilidad para facilitar el desarrollo. Por ejemplo, las variables de tipo entero representan siempre números enteros y además, enteros de 32 bits en complemento a2. Además, Java construye sus interfaces de usuario a través de un sistema abstracto de ventanas de forma que éstas puedan ser implantadas en entornos Unix, Pc o Mac.

A.1.4 Orientado a aplicaciones

El lenguaje Java, dispone de muchas bibliotecas orientadas al desarrollo de múltiples aplicaciones, por ejemplo, existen librerías para gestionar bases de datos, para XML, gráficos, etc.

A.2 JDK

Para programar en Java se utiliza el Kit de Desarrollo Java (JDK, Java Development Kit) el cual cuenta con un conjunto de clases que los programadores Java pueden utilizar para escribir aplicaciones y applets. Dicho conjunto se denomina interfaz de programación de aplicaciones (API, Application Programming Interface). También se puede definir la API de Java como un conjunto de clases estándar, formadas por otros conjuntos (subconjuntos) de clases llamadas paquetes (packages) perfectamente caracterizados en cuanto a sus propiedades; desde luego, el usuario puede crear sus propios packages y utilizarlos para reforzar o sustituir algunos de los existentes.

Para el desarrollo de este proyecto, se ha utilizado la versión *j2se-1.4.2*. Para más información visitar referencia [38].

A.3 Applets

Un applet es un programa Java que se ejecuta en un navegador Web, utilizando todos sus recursos como entorno de trabajo. Por ejemplo, la página en la que se carga el applet hace de entorno gráfico en el que se van a dibujar los comandos gráficos del applet.

En el caso del proyecto desarrollado el applet estará almacenado en un servidor Web y será llamado por una página Web. Cuando el usuario entre en esta página Web que incluye el applet, este será descargado al ordenador local del usuario y se ejecutará allí. Para ser más exactos, habría que decir que el applet será ejecutado por el intérprete de Java (la JVM, Java Virtual Machine) que estará cargado en el navegador del usuario.

Resumiendo, los pasos que realiza un applet cuando se ejecuta son:

- El applet, al igual que las páginas HTML, reside en el servidor Web.
- Cuando un usuario quiere acceder a la página, su navegador Web descarga en su ordenador:
  - La página HTML que contiene una llamada al applet
  - El applet que ha sido llamado
- La página y el applet, una vez descargados, se ejecutan en el ordenador cliente. El applet sólo se descargará una vez por simulación y una vez está descargado se ejecutará en local sin tener ningún tipo de interacción con el servidor.

La característica fundamental de los applets es su condición de contenido puro, de esta manera, al tener la capacidad de ejecutar acciones muy complejas, se hacen idóneos para representar contenidos especiales, como lo pueden ser animaciones de imágenes, aplicaciones interactivas, presentar menús y cuadros de diálogo para luego emprender acciones, etc.

Aparte de esta característica, las principales ventajas de utilizar applets son:

- El applet es independiente de la máquina y el sistema operativo.
- Evita problemas de actualización y distribución: si el programador quiere hacer una nueva versión del applet, la deposita en el servidor Web y automáticamente todos los clientes que accedan a partir de ese momento se descargan la nueva versión.
- Debido a las restricciones de seguridad del propio lenguaje, los errores de programación no alteran el sistema operativo del cliente.

Los applets también presentan una serie de restricciones que hay que tener presentes:

- Una restricción de seguridad: un applet no puede escribir o leer del disco duro del cliente. Esta restricción se suaviza en los casos de applets de confianza y para el sistema Contenedor utilizado se soluciona este tema con la firma digital del applet que autentifica al creador y si el usuario acepta esta autentificación el applet podrá interactuar con el sistema de archivos del ordenador local para la carga o descarga de ficheros.
- La descarga del applet consume tiempo, por ello cuando hay que descargar applets voluminosos se recurre a empaquetarlos en archivos JAR. El sistema Contenedor tiene todas sus clases empaquetadas en ficheros JAR’s que minimizan su tiempo de descarga desde el ordenador del usuario.

- Otra restricción al utilizar applets es que el Sistema Operativo del ordenador del usuario debe tener instalado una versión de Java JRE ("Java Runtime Environment"), que corresponde al "Plug-in" utilizado por el navegador para ejecutar aplicaciones Java.

➢ ***Ciclo de vida de un applet***

Cuando el navegador carga un applet por primera vez, éste debe realizar ciertas tareas de inicialización. El navegador llama a un método del ciclo de vida del applet, para que éste realice las tareas correspondientes a cada caso. Los métodos son los siguientes:

- ✓ *init()*: método al que llama el navegador cuando se carga o recarga el applet.
- ✓ *start()*: método al que llama el navegador cuando comienza a ejecutar el applet.
- ✓ *stop()*: método al que llama el navegador cuando se detiene el applet.
- ✓ *destroy()*: método al que llama el navegador justo antes de cerrarse, se libera toda la memoria y recursos ocupados por el applet antes de salir del navegador.

7 Conclusiones

La influencia que ejerce Internet sobre todos los sectores de la sociedad y en la educación en particular, hace de las tecnologías de la información y comunicación (TIC) una herramienta eficaz en la mejora de la docencia.

Centrando la atención en el laboratorio virtual de comunicaciones analógicas y digitales LaViCAD donde se integra el proyecto desarrollado, se piensa en los applets como una buena herramienta docente dentro de las TIC tanto para profesores como para alumnos. Los applets posibilitan la representación de fenómenos relacionados con las comunicaciones y procesado de señal constituyendo un laboratorio de comunicaciones virtual. Se consigue así mostrar el tema a tratar de manera simple, remarcando los aspectos más importantes y facilitando la representación de gráficas y resultados que, mediante la educación tradicional, resultaría complicado o incluso imposible mostrar a los alumnos. Por todo ello se piensa en los applets como una buena herramienta docente que permite al alumno asumir, casi sin esfuerzo, una serie de conceptos claves sobre el tema que exponen.

En la realización el presente proyecto se plantearon unos objetivos:

- ✓ Diseño e implementación MATLAB del simulador de la capa física del sistema DVB-T, bajo las especificaciones del estándar *ETSI EN 300 744*.

- ✓ Completar el *programa plantilla o* contenedor, definido en el sistema completo de modulaciones QAM, para los applets programados en MATLAB.

- ✓ Implementación del sistema DVB-T como applet de Java, a partir de la plataforma del aplicativo *contenedor* desarrollado en el proyecto LaViCAD, para su libre disposición en Internet.

- ✓ Lograr que el sistema pueda considerarse un referente para ser utilizado en la simulación de sistemas de comunicaciones completos a nivel de capa física considerados en estándares de aplicaciones de uso cotidiano.

- ✓ Acercar al usuario el funcionamiento y especificaciones de la televisión digital terrestre (TDT).

Lo primero que se trabajó fue el diseño e implementación MATLAB del simulador. En esta parte del proyecto se definieron, mediante reuniones con ambos co-directores, las bases del sistema DVB-T a desarrollar.

Al principio se hizo una subdivisión del sistema de comunicaciones a implementar (transmisor, canal y receptor) en etapas, decidiendo el procesado de la señal a realizar por cada una de ellas. A su vez, a medida que avanzaba el proyecto, se estableció para cada etapa la interacción del usuario por medio de los parámetros de entrada, y los

resultados con más interés a mostrar. Asimismo se comprobó, mediante pruebas definidas en el estándar o propuestas por los directores del proyecto, el correcto funcionamiento de cada una de las etapas.

En esta fase se ha ceñido el sistema a las especificaciones definidas en el estándar de la capa física DVB-T, salvo en dos aspectos. Por un lado se decidió no implementar el modo de transmisión 8k, ya que la simulación se ralentizaría porque en lugar de utilizar 1705 portadoras por símbolo OFDM, se usarían 6817. Por el otro, también se decidió no contemplar transmisiones jerárquicas al complicar éstas la implementación del sistema y no aportar más interés académico al simulador.

Una vez finalizada esta parte del proyecto se alcanzaron con éxito los objetivos propuestos de implementación MATLAB del simulador y de perfeccionamiento del *programa plantilla* MATLAB. Además, en el apartado 5.1.5.2 se exhiben unas recomendaciones de programación y optimización de código para que puedan servir de ayuda a futuros programadores de sistemas de la plataforma LaViCAD en MATLAB.

Llegados a este punto y antes de pasar a la programación del applet de Java, se investigó acerca de las posibilidades de despliegue de aplicaciones que ofrecía MATLAB. De ellas, dos fueron de interés de cara al proyecto: creación de aplicaciones *standalone* y compilación para Java (*Builder for Java*). La ventaja que aportan ambas posibilidades es que permiten ejecutar programas escritos en MATLAB sin necesidad de una licencia de MATLAB (aunque sí requieren la instalación del software gratuito MATLAB Compiler Runtime, MCR, de The Mathworks).

Como reto paralelo se creó un ejecutable autónomo (aplicación *standalone*) del simulador implementado y también un componente de software Java con el producto *Builder for Java.*

En aquel momento todo el interés del proyecto se centró en el producto *Builder for Java*, por la posibilidad de crear el applet de Java a partir de la aplicación MATLAB. Incluso se asistió a una conferencia virtual de The Mathworks acerca del producto y se contactó con su sede en Madrid para obtener toda la información posible al respecto. Más adelante se encontraron diversas dificultades que provocaron desestimar esta opción.

Al comprobar la inviabilidad, al menos inmediata, de crear el applet de Java con la programación MATLAB, se comenzó el desarrollo del simulador DVB-T utilizando el aplicativo Contenedor programado en el proyecto LaViCAD. El primer paso fue familiarizarse con el entorno de programación (se empleó Eclipse), y el estudiar el aplicativo Contenedor. Posteriormente se procedió al diseño etapa por etapa del applet de Java siguiendo al máximo el estilo del simulador MATLAB y ayudándose del Javadoc generado en el proyecto LaViCAD.

En el desarrollo del applet de Java interfirió el hecho de que Java no dispone de subrutinas de procesado ya implementadas en una toolbox del programa como ocurre en MATLAB. Esto, sumado al cambio de un lenguaje de programación de alto nivel por uno orientado a objetos, hizo que esta parte del proyecto resultara un reto importante del mismo. No obstante, fue de gran ayuda la implementación del sistema OFDM del proyecto LaViCAD, para poder consultar subrutinas de procesado reaprovechables en el

sistema DVB-T, y también el codificador convolucional (y decodificador de Viterbi asociado) programado por Pedro Espinosa Fricke para el departamento de TSC, que ha sido integrado en el simulador DVB-T.

Por otra parte, se experimentaron problemas de limitación de memoria de la consola de Java, tal como se detalla en el apartado 5.2.1. En el caso de tener señales de gran longitud, a pesar del empleo del *garbage collector* y la optimización de código llevada a cabo, hay simulaciones del sistema que provocan que la memoria de la consola necesite más que los 64MB asignados por defecto, puesto que deben dibujar gráficas de muchos puntos, guardar en memoria esas señales de gran tamaño o ejecutar algoritmos costosos. Tras indagar al respecto, se vio que la única solución para que el usuario pueda realizar cualquier tipo de simulación con el applet es que introduzca manualmente en su consola Java la memoria que quiere que se le asigne del sistema (con 170 MB se pueden realizar las diferentes simulaciones posibles). Aún sin hacer la recomendación anterior, se puede ejecutar el applet sin problemas de memoria en el caso de seleccionar los parámetros más básicos de procesado.

Una gran ventaja de esta implementación Java es que supuso la adquisición de unos conocimientos más profundos sobre el procesado de señal realizado en el sistema DVB-T, especialmente por los codificadores Reed-Solomon y convolucional, y sus respectivos decodificadores.

Una vez finalizado el applet de Java simulador de la capa física del sistema DVB-T, se colgó en la web de la plataforma COMWEB (comweb.upc.edu), integrándolo en el curso LaViCAD para su libre uso. En este punto se habían alcanzado con éxito los objetivos propuestos tanto de implementación Java del simulador, como de acercar al usuario el funcionamiento y especificaciones de la TDT, lo cual se logra al colgarlo en una página web donde los estudiantes puedan acceder para analizar dicho sistema y estudiar los conocimientos teóricos que soporta una de las aplicaciones de comunicaciones más populares hoy en día.

Capítulo a parte hay que situar el quinto objetivo propuesto, el cual se propone el uso del sistema como referente en los simuladores de sistemas de comunicaciones completos basados en estándares. Se tiene la seguridad de que servirá de gran ayuda y será un referente para futuros simuladores basados en estándares de aplicaciones con uso cotidiano, pues se trata del sistema más completo a nivel de capa física de los implementados hasta la fecha en el proyecto LaViCAD debido a que posee dos codificadores, 3 entrelazadores, modulación uniforme y no uniforme, canal con multicamino, etc.

Tras la finalización del presente proyecto, se observa un alto grado de satisfacción en los objetivos alcanzados. Comparando el desarrollo del simulador DVB-T en MATLAB y Java, se destaca por parte de MATLAB una gran facilidad en la programación y unas grandes posibilidades de optimización de código vectorizando los algoritmos. En cuanto a Java, resulta una buena solución para poder colgar un simulador de un sistema de comunicaciones en Internet, pero se deben tener en cuenta restricciones de memoria de la consola de Java.

Por último, cabe comentar que el simulador implementado es autocontenido, pero también puede interactuar con un laboratorio instrumental. Mediante la aplicación

desarrollada se pueden experimentar y verificar diferentes simulaciones correspondientes a temáticas propias de los estudios de ingeniería en telecomunicaciones.

7.1 Líneas futuras de actuación

Las futuras líneas de actuación, haciendo uso o complementando el simulador de la capa física DVB-T implementado son las siguientes:

- ***Ayuda del sistema capa física de DVB-T***

La línea futura más inmediata será el escribir todas las páginas de ayuda del simulador desarrollado en los tres idiomas definidos (español, catalán e inglés).

Primero se deberá escribir la ayuda del sistema global, que contendrá información del sistema DVB-T implementado y casos particulares como el formato de los ficheros intercambiables.

Posteriormente se procederá a escribir la página de ayuda de cada una de las 8 etapas. Cada una tendrá información de la teoría de la etapa (procesado de señal), de los parámetros de entrada configurables y de los resultados mostrados.

Para realizar todo este trabajo será de gran ayuda el capítulo 4 de la memoria del presente proyecto, puesto que estudia etapa a etapa el sistema implementado, explicando todos los conceptos que deben escribirse en las ayudas.

- ***Posibles mejoras al sistema DVB-T implementado***

Una posible línea futura sería la implementación de transmisiones jerárquicas, y la incorporación del modo de operación 8k al simulador.

Con esta implementación se completaría el simulador desarrollado, teniendo así contempladas en el mismo todas las especificaciones definidas en el estándar de la capa física DVB-T *ETSI EN 300 744*. Gracias a ello, se podría comprobar la mejora aportada por las transmisiones jerárquicas, y se completaría el sistema permitiendo escoger entre modo de operación 2k ó 8k.

En el caso de una transmisión jerárquica, habrá 2 flujos de transporte, uno de ellos denominado de alta prioridad que tiene baja velocidad y por tanto menor calidad de imagen, modula las portadoras con un esquema de modulación muy robusto frente al ruido (QPSK) mientras que el segundo flujo de transporte, denominado de baja prioridad complementa al anterior en cuanto a velocidad y calidad de imagen y combina su información con el anterior de forma que las portadoras son moduladas finalmente con un esquema más exigente en cuanto a relación señal/ruido. En el caso de que este último utilice 4 bits por cada 2 bits del de alta prioridad, se alcanzaría una constelación total para la señal emitida de 64-QAM. En la zona del área de cobertura donde se reciba la señal con buena relación S/N, la imagen recuperada, de alta calidad, corresponderá a la combinación de los dos flujos (alta y baja prioridad) mientras que en caso contrario la calidad de imagen recibida será peor, correspondiendo sólo al flujo de alta prioridad.

Aparte, se podría estudiar la viabilidad práctica de enviar un mayor número de paquetes en la etapa 1 del simulador. De esta manera se podría llegar a enviar una trama completa de 68 símbolos OFDM, con lo que se podría enviar la información TPS tal como indica el estándar.

De esta manera, también se disminuirían los detalles analizados en el capítulo 4 de la memoria, donde se explicaba que debido a que la señal transmitida es mucho más pequeña que la de una transmisión real, los ceros de relleno añadidos por el entrelazador externo, entre otros aspectos, afectaban a la simulación.

Se destaca que todas estas mejoras es conveniente planteárselas en la versión MATLAB del simulador, puesto que la versión Java es difícilmente ampliable por las limitaciones de memoria anteriormente citadas.

- ***Capa física sistema DVB-H***

Se propone la implementación del simulador de la capa física del sistema DVB-H, debido a que dicho sistema se basa principalmente en la especificación DVB-T para la televisión digital terrestre, añadiendo a la misma una serie de características diseñadas para tener en cuenta la limitada duración de la batería de los pequeños dispositivos portátiles, y el particular entorno en el que los receptores de este tipo deben funcionar.

Los nuevos elementos técnicos añadidos al estándar, y los cuales habría que añadir al simulador implementado, son los siguientes:

- ✓ *Bits TPS (obligatorio):* Señalización de los parámetros DVB-H en los bits TPS para acelerar la fase de descubrimiento de servicio. Además, la señalización del *Cell Identifier* es también obligatoria en DVB-H.
- ✓ *Modo 4K*: Se define el modo 4K para disponer de mayor flexibilidad en el diseño de redes SFN. El modo 4K es útil para ofrecer servicios a muy alta velocidad o para operar en bandas de frecuencia elevadas.
- ✓ *In-depth Interleaving (iDi)*: un entrelezador de símbolo OFDM para los modos 2K y 4K (que utilizan el entrelazador del modo 8K) que mejora la robustez en entornos móviles.
- ✓ *Canal de 5 MHz:* La definición de los parámetros DVB-T en canales de 5 MHz

Así pues, los modos DVB-T pasan de ser dos: el 2K y el 8K a ser 5 modos: 2K, 2KiDi8K, 4K, 4KiDi8K y 8K. Y los parámetros configurables se mantienen.

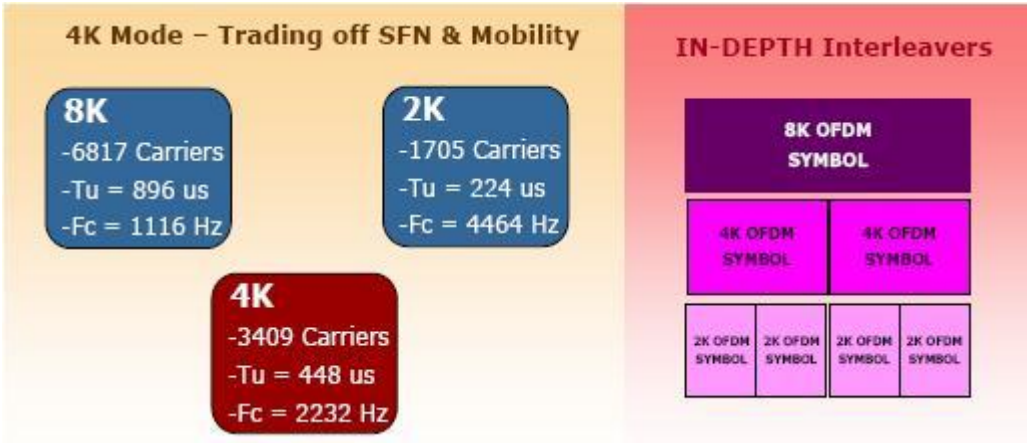


Figura 99 Modos de operación del estándar DVB-H. Figura extraída de [46].

- ***Capa de codificación de fuente MPEG2***

Se tiene planteado el desarrollo de un proyecto final de carrera en el departamento de TSC, titulado "Codificador MPEG para LaViCAD".

Los objetivos de la propuesta de PFC son:

- ✓ Lectura y codificación de imágenes de vídeo para LaViCAD
- ✓ Adaptar un codificador híbrido abierto (*ffmpeg*)
- ✓ Simular las capas de compresión y de sistema de MPEG2 (L_MPEG_PARSER)

Con este proyecto, en un futuro también se integrarán señales que representen imágenes en la emulación del estándar MPEG dentro del sistema completo de simulación del sistema de televisión terrestre (DVB-T).

En esta situación, se tendría la posibilidad de realizar la lectura de una señal de vídeo en la primera etapa del simulador DVB-T, y ver como se recibiría en la ultima etapa.

Por otra parte, enfocando la atención en el laboratorio virtual LaViCAD donde se integra el simulador implementado, se está trabajando en dos temas actualmente:

- ***MATLAB en los applets de Java***

Tras el estudio llevado a cabo durante el desarrollo de este proyecto sobre la posibilidad de crear applets de Java a partir de la aplicación MATLAB con el producto de The Mathworks *Builder for Java*, se concluyó que no era posible crear un applet directamente.

Se está realizando un estudio, llevado a cabo por el estudiante de la ETSETB Pedro Espinosa Fricke para el departamento de TSC, sobre la posibilidad de integrar funciones en MATLAB dentro de los applets de Java, lo cual sería de gran utilidad por la facilidad para programar el procesado de señal en MATLAB.

De momento se conoce que es posible exportar funciones MATLAB al lenguaje Java y se está desarrollando código Java que: sea integrable en el applet LaViCAD y permita el uso de las funciones MATLAB exportadas. El problema radica en que no se pueden crear applets directamente, como ya se ha comentado.

La solución adoptada se esquematiza en la siguiente figura:

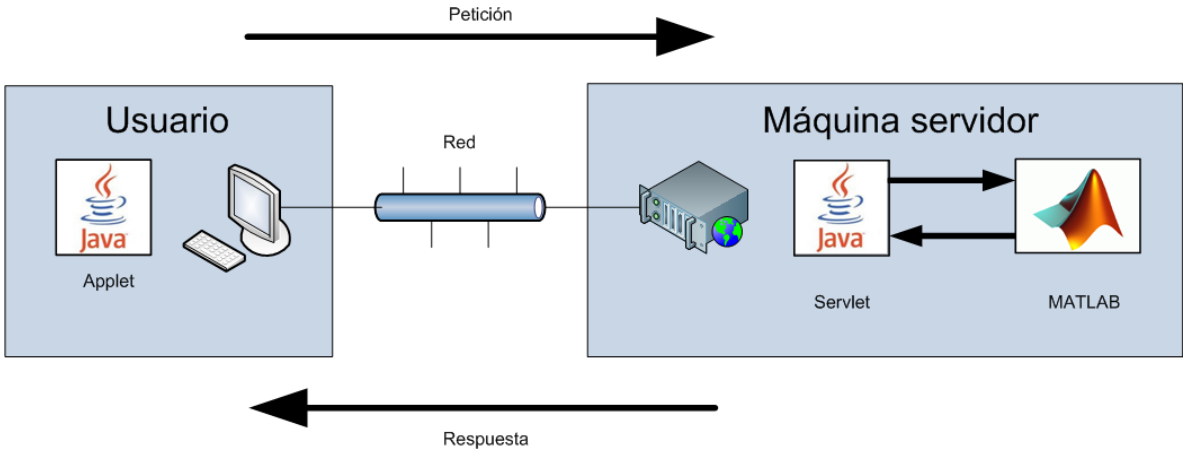


Figura 100 Esquema adoptado Usuario-Máquina servidor

El applet hará peticiones a un servlet alojado en la máquina servidor que ejecutará la función de MATLAB, una vez calculada la respuesta se devuelve al servlet. Las comunicaciones entre applet y servlet se hacen mediante paso de mensajes.

En estos momentos, se ha probado con éxito el procesado íntegro en MATLAB de la primera etapa del applet QAM usando localhost (servlet y applet en la misma máquina), y se prevé la cercana posibilidad de alojar el servlet en un servidor de pruebas, posteriormente alojarlo en el servidor comweb y finalmente desarrollar más etapas.

- ***Comunicación Java-Moodle***

Este estudio se centra en conseguir la comunicación entre los applets desarrollados en la plataforma de e-learning COMWEB y el gestor de cursos Moodle.

Se quiere conseguir que, en las bases de datos de Moodle, se puedan guardar los datos obtenidos por los usuarios mediante los simuladores (de LaViCAD entre otros). De esta forma, los datos obtenidos podrán ser utilizados de manera automática como herramienta de evaluación.

Actualmente se está abordando el diseño de la interfaz de las comunicaciones bidireccionales entre el servidor (Moodle) en el que se dispongan los applets y los usuarios que ejecuten los applets de Java en modo local. Hace falta dotar a los applets

de la posibilidad de comunicarse con el servidor Moodle con la finalidad de que los profesores de las asignaturas de comunicaciones y procesado de señal en general puedan obtener información de retorno de sus propuestas.

El primer objetivo en el que se está trabajando es que se muestre por qué etapas ha pasado el usuario.

Finalmente, se debe decir que existen diferentes experiencias docentes que actualmente utilizan LaViCAD como herramienta complementaria en asignaturas. En el caso del applet simulador de la capa física programado, se propone utilizarlo en la asignatura optativa de Televisión Analógica y Digital como complemento a las clases de laboratorio, junto con el sistema de codificación de vídeo MPEG cuando éste se halle implementado.

Acrónimos

- ACM: Adaptive Coding and Modulation
- API: Application Programming Interface
- ATSC: Advanced Televisión System Comité
- BER: Bit Error Rate
- CENELEC: European Committee for Electrotechnical Standardization
- DTMB: Digital Terrestrial Multimedia Broadcast
- DTTV-SA: Digital Terrestrial Television-System Aspects
- DVB: Digital Video Broadcasting
- DVB-C: Digital Video Broadcasting – Cable
- DVB-CPCM: Digital Video Broadcasting Content Protection & Copy Management
- DVB-CS: Digital Satellite Master Antenna Television Distribution Systems
- DVB-DSNG: Digital Video Broadcasting Digital Satellite News Gathering
- DVB-GSE: Digital Video Broadcasting Generic Stream Encapsulation
- DVB-H: Digital Video Broadcasting Handheld
- DVB-MDS: Digital Video Broadcasting Multipoint video Distribution Systems
- DVB-PCF: Digital Video Broadcasting Portable Content Format
- DVB-S: Digital Video Broadcasting – Satellite
- DVB-SH: Digital Video Broadcasting from Satellite to Handhelds
- DVB-SI: Digital Video Broadcasting Service Information
- DVB-SSU: Digital Video Broadcasting System Software Updates
- DVB-T: Digital Video Broadcasting – Terrestrial

- DVB-TVA: Digital Video Broadcasting TV Anytime
- EBU: European Broadcasting Union
- ELG: European Launching Group
- EPG: Electronic Program Guide
- ESO: European Standards Organisation
- ETSI: European Telecommunications Standards Institute
- FEC: Forward Error Correction
- HDTV: High Definition Television
- ISDB: Integrated Services Digital Broadcasting
- MFN: Multiple Frequency Network
- MHP: Multimedia Home Platform
- MoU: Memorandum of Understanding
- OFDM: Orthogonal Frequency Division Multiplexing
- PES: Packetized Elementary Stream
- PSK: Phase Shift Keying
- QAM: Quadrature Amplitude Modulation
- QPSK: Quadrature Phase Shift keying
- SBTVD-T: Sistema Brasileiro de Televisão Digital
- SDTV: Standard Definition Television
- SER: Symbol Error Rate
- SMATV: Satellite Master Antenna Television
- SFN: Single Frequency Network
- SPI: Serial Peripheral Interface
- SSI: Serial Synchronous Interface
- STB: Set Top Box

- TIC: Tecnologías de la Información y las Comunicaciones
- UHF: Ultra High Frequency

Bibliografía

[1] ETSI EN 300 744 V1.5.1 (2004-06): "Digital Video Broadcasting (DVB); Framing structure, channel coding and modulation for digital terrestrial Television"

[2] AA.VV. *DVB Project.* [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.dvb.org>

[3] AA.VV. *ETSI.* [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.etsi.org >

[4] AA.VV. *ATSC*. [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.atsc.org>

[5]AA.VV. ATSC Forum [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://atscforum.org>

[6] AA.VV. DiBEG [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.dibeg.org>

[7] AA.VV. ARIB [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.arib.or.jp>

[8] ETSI EN 300 468 V1.3.1 (1998-02): "Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems"

[9] ISO/IEC 13818 (Parts 1 to 3): "Information technology - Generic coding of moving pictures and associated audio information".

[10] ETSI TR 101 190 V1.1.1 (1997-12): "Digital Video Broadcasting (DVB); Implementation guidelines for DVB terrestrial services; Transmission aspects"

[11] ETSI TR 101 154 V1.4.1 (2000-07): "Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 Systems, Video and Audio in satellite, cable and terrestrial broadcasting applications".

[12] ETSI TR 101 211 V1.6.1 (2004-05): "Digital Video Broadcasting (DVB); Guidelines on implementation and usage of Service Information (SI)".

[13] AA.VV. *DigiTAG* [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.digitag.org>

[14] Marianov, V.; Oberli, C.; Ríos, M. Ingeniería DICTUC: "Análisis de los estándares de transmisión de televisión digital terrestre y su aplicabilidad al medio nacional". Estudio técnico en la Pontificia Universidad Católica de Chile, Octubre 2006.

[15] Pérez Vega, C. (2007): "Transmisión de señal digital". Departamento de ingeniería de comunicaciones de la Universidad de Cantabria.

[16] AA.VV. *Televisión digital terrestre TDT* [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.televisiondigital.es>. Gobierno de España. Ministerio de industria, turismo y comercio.

[17] AA.VV. *Televisió digital terrestre TDT* [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.tdtcat.net >. Generalitat de Catalunya.

[18] AA.VV. ZonaTDT [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.mundoplus.tv/muchomas/zonatdt>

[19] AA.VV. Asesoría TDT Online [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.aranova.es/asesoria_tdt_online/tdt_local/comunidad_cataluna.html>

[20] HD on DTT. Key issues for broadcasters, regulators and viewers. Publicado por DigiTAG en 2007.

[21] AA.VV. *Com@Web*. [en línea]. [Consulta: 26 septiembre 2007]. Disponible en: <http://comweb.upc.edu>.

[22] Vargas, C; Vargas, F; *Proyecto LaViCAD*. Proyecto final de carrera desarrollado para el depatarmento de Teoría de la Señal y Comunicaciones de la Universidad Politécnica de Cataluña dirigido por Cabrera, M. Barcelona, diciembre 2007.

[23] Informe CRUE (2004): "Las Tecnologías de la Información y las Comunicaciones en el Sistema Universitario Español". Dirigido por Senén Barro Ameneiro (Rector de la Universidad de Santiago de Compostela - USC).

[24] Informe CRUE (diciembre 2007): "Las TIC en el Sistema Universitario Español. UNIVERSITIC 2007." Dirigido Javier Uceda Antolín Rector de la UPM y Presidente de la Sectorial TIC de la CRUE Senén Barro Ameneiro Rector de la USC y Vocal de la CRUE

[25] AA.VV. CRUE [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.crue.org/ >

[26] AA.VV. UNIVERSITIC [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.crue.org/UNIVERSITIC>

[27] AA.VV. Servicios TIC en la UPC [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://upcnet.upc.edu/>

[28] Espinosa, P.; *Algoritmo de Viterbi implementado en Java*. Trabajo propio desarrollado para el departamento de Teoría de la Señal y Comunicaciones de la Universidad Politécnica de Cataluña dirigido por Cabrera, M. Barcelona, enero 2007.

[29] Proakis, J.G. *Communication systems engineering*. Ed. Prentice-Hall

[30] Carlson A. B. *Communication systems*. Ed. McGraw-Hill

[31] Proakis, J.G. *Digital communications*. Ed. McGraw-Hill

[32] Schulze,H; Lüders,C; *Theory and applications of OFDM and CDMA.* Ed. Wiley.

[33] AA.VV. The Mathworks [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://www.mathworks.com/>

[34] Bhargava, V.K.; Fair, I.J.; *Forward Error Correction Coding*

[35] Cabrera, M.; Espinosa, P.; Marcos, C.; *Matlab, Ejecutables y Java*. Proyecto COM@WEB.

[36] AA.VV. Eclipse [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <www.eclipse.org/>

[37] AA.VV. Tutorial Java [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://java.sun.com/docs/books/tutorial>

[38] AA.VV. JavaTM 2 Platform, Standard Edition, v 1.4.2 API Specification [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://java.sun.com/j2se/1.4.2/docs/api/>

[39] Berkeley University (2006, mayo). *Ptolemy Project*. [en línea]. [Consulta: 26 septiembre 2007]. Disponible en <http://ptolemy.eecs.berkeley.edu/java/ptplot/>.

[40] Steele, R; *Mobile radio communications.* Ed. Wiley.

[41] Clarke, C.K.P.; *Reed-Solomon Error Correction.* R&D White Paper.

[42] Blahut, R.; *Theory & practice of error control* codes. Addison-Wesley, 1983

[43] Mariño, J.; Vallverdú, F.; Rodríguez, J.; Moreno, A. (1995) *Tratamiento digital de la señal: una introducción experimental* Edicions UPC. ISBN 84-7653-548-1.

[44] Delgado, A; *Transmisión de señales de tv digital en el estándar terreno dvb-t.* Departamento de Electromagnetismo y Teoría de Circuitos. Universidad Politécnica de Madrid

[45] AA.VV. IES El Cabanyal [en línea]. [Consulta: 15 mayo 2008]. Disponible en: <http://java.sun.com/docs/books/tutorial>. Departamento de electrónica**.**

[46] Documento sobre informe de situación DVB-H. Elaborado por Grupo de Trabajo 7 MHP y otros aspectos técnicos del Foro Técnico de la televisión digital. Foro técnico de la televisión digital

[47] AA.VV. *DVB-T*. [en línea]. [Consulta: 15 mayo 2008]. Disponible en <http://en.wikipedia.org/wiki/DVB-T>.