_ TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE UPV INGENIEROS DE TELECOMUNICACIÓN

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

IGUALADOR DE CANAL PARA ESTÁNDARES DE COMUNICACIONES BASADOS EN OFDM: *IEEE 802.16m* (*WirelessMAN-Advanced*) Y *LTE - Advanced* (4G)

Vicent Molés Cases

Tutora: Gema Piñero Sipán

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2013-14

Valencia, 4 de septiembre de 2014

Resumen

En este Trabajo Final de Grado se presenta el diseño de un sistema de comunicaciones basado en la modulación ***OFDM*** (*Orthogonal Frequency - Division Multiplexing*), que se ha adaptado a 2 estándares de comunicaciones de cuarta generación, como son ***LTE-A*** (*Long-Term Evolution - Advanced)* y ***IEEE 802.16m***. El mencionado sistema ha sido diseñado utilizando el *software LabVIEW 2013* –desarrollado por la compañía *National Instruments*– el cual se basa en el lenguaje de programación gráfica *G* y que está orientado al control de instrumentación. El sistema diseñado incluye un modulador *OFDM*, un demodulador *OFDM*, un módulo de sincronismo temporal para *OFDM* y 5 igualadores de canal basados en el estimador *LS* (*Least Squares)*, si bien éstos utilizan métodos de interpolación distintos. El principal objetivo del proyecto es, una vez diseñado el sistema de comunicaciones, evaluar el funcionamiento de los 5 igualadores de canal diseñados, utilizando *BER (Bit Error Rate)* como parámetro de calidad. Para la evaluación del sistema *OFDM* y de los distintos igualadores de canal se ha hecho uso de 2 dispositivos *USRP2920 (http://sine.ni.com/nips/cds/view/p/lang/es/nid/209948)*, también distribuidos por la compañía *National Instruments,* que están optimizados para funcionar junto a *LabVIEW* y que permiten la transmisión, recepción y procesado de señales de radiofrecuencia.

Resum

En aquest Treball Fi de Grau, es presenta el disseny d'un sistema de comunicacions basat en la modulació ***OFDM*** (*Orthogonal Frequency - Division Multiplexing*), que ha estat adaptat a 2 estàndards de comunicacions de quarta generació, com són ***LTE-A*** (*Long-Term Evolution - Advanced)* y ***IEEE 802.16m***. L'esmentat sistema ha estat dissenyat mitjançant el software *LabVIEW 2013* –desenvolupat per l'empresa *National Instruments*– el qual està basat en el llenguatge de programació gràfica *G* i que està orientat al control d'instrumentació. El sistema dissenyat inclou un modulador *OFDM*, un demodulador *OFDM*, un mòdul de sincronisme temporal per a *OFDM* i 5 igualadors de canal basats en l'estimador *LS* (*Least Squares*)*,* malgrat que aquests utilitzen mètodes d'interpolació distints. El principal objectiu del projecte és, una vegada ja ha estat dissenyat el sistema de comunicacions *OFDM*, l'avaluació del funcionament dels 5 igualadors de canal dissenyats, utilitzant *BER* (*Bit Error Rate)* com a paràmetre de qualitat. Per a l'avaluació del sistema *OFDM* i dels distints igualadors de canal s'han utilitzat 2 dispositius *USRP2920 (http://sine.ni.com/nips/cds/view/p/lang/es/nid/209948)*, també distribuïts per l'empresa *National Instruments*, que estan optimitzats per al seu funcionament junt a *LabVIEW* i que permeten la transmissió, recepció y processat de senyals de radiofreqüència.

Abstract

This Final Project presents the design of a communications systems based on the ***OFDM*** *(Orthogonal Frequency - Division Multiplexing)* modulation. It has been adapted to 2 four-generation communication standards, like ***LTE-A*** (*Long-Term Evolution – Advanced)* and ***IEEE 802.16m***. The previously mentioned system has been designed by means of the software *LabVIEW 2013* –developed by the company *National Instruments*– which is based on the *G* programming language and is oriented to the control of instrumentation. The designed system includes an *OFDM* modulator, an *OFDM* demodulator, an *OFDM* temporal synchronization block and 5 channel equalizers which are based on the *LS* (*Least Squares*) estimator, however they use different interpolation methods. In addition to the development of the communications system, the main objective of this project is to test the 5 channel equalizers designed, using the *Bit Error Rate* (*BER*) as a quality parameter. In order to test the *OFDM* system and the different equalizers, we have used 2 *USRP2920* devices *(http://sine.ni.com/nips/cds/view/p/lang/es/nid/209948)*, which have been designed by *National Instruments* too, and which are optimized to work with *LabVIEW*, allowing the transmission, reception and processing of radio frequency signals.

Índice

Capítulo 1 - Introducción

1.1 - Motivación

El presente proyecto nace como consecuencia de la adquisición por parte del Departamento de Comunicaciones de la *Universitat Politècnica de València* de dos dispositivos *USRP 2920* de la compañía *National Instruments*, los cuales permiten la evaluación de sistemas de comunicaciones diseñados mediante el *software LabVIEW* .

La utilización de *LabVIEW* aporta numerosas ventajas respecto a otras herramientas de diseño. Por una parte, permite realizar simulaciones igual que otros *softwares* como *Matlab*. Por otra, los sistemas diseñados se pueden aplicar en entornos reales gracias a la utilización de los dispositivos *USRP* y se puede evaluar su comportamiento no solo de forma experimental sino también de forma simulada, lo que fomenta la profundización en ciertos aspectos como el sincronismo (aspecto fundamental en un sistema real y que normalmente no se tiene en cuenta en las simulaciones).

Una vez elegido el *software (LabVIEW 2013)* y el *hardware (USRP 2920)* que se iba a utilizar para la realización del proyecto, se planteó qué tipo de sistema de comunicaciones se quería implementar y más concretamente en qué parte del sistema se profundizaría. Llegados a este punto, se optó por la elección de la modulación *OFDM*, una de las más avanzadas y novedosas que existen hoy en día y que proporciona mayores tasas, así como un mejor comportamiento frente a desvanecimientos selectivos respecto a modulaciones de una sola portadora. También se eligieron los estándares que se iban a implementar, éstos debían utilizar *OFDM*. Finalmente fueron elegidos *LTE-Advanced* y *WiMAX - IEEE 802.16m*, dos estándares de cuarta generación que actualmente están siendo implantados por los operadores de comunicaciones (para el caso de *LTE-Advanced*) y por empresas que ofrecen acceso inalámbrico (para el caso de *IEEE 802.16m*).

Por último, se decidió estudiar la igualación de canal, uno de los puntos más importantes de un sistema de comunicaciones, el cual nos permite corregir los errores que se han producido en la recepción. Por ello, se ha realizado el diseño de diversos igualadores para su posterior evaluación.

Con todo lo anterior, se ha conseguido realizar el diseño, simulación y evaluación en un entorno real de un sistema de comunicaciones basado en *OFDM*, aplicado a estándares de cuarta generación y con distintos igualadores de canal, resultando un proyecto muy interesante que está en consonancia con los sistemas que se están implementando en la actualidad.

1.2 - Objetivos

El principal objetivo del proyecto era, en principio, el diseño y la posterior evaluación de distintos igualadores de canal para un sistema *OFDM*. Para realizar dicha evaluación, era necesario disponer de un sistema de comunicaciones *OFDM* completo (modulador y demodulador) diseñado mediante *LabVIEW*, por lo que previamente al diseño y evaluación de los distintos igualadores se tuvo que realizar el diseño de dicho sistema.

El proyecto desarrollado tiene como principales objetivos:

- Diseñar un sistema *OFDM* completo (que incluya modulador y demodulador) utilizando el *software LabVIEW 2013* de la empresa *National Instruments*, el cual está basado en un lenguaje de programación gráfico llamado lenguaje G.

- Adaptar el sistema *OFDM* a los estándares *IEEE 802.16m* y *LTE-Advanced*, estándares de cuarta generación basados en la modulación *OFDM* que poseen una gran similitud entre sí en la capa física.

- Diseñar diversos igualadores de canal, todos ellos basados en la estimación de Mínimos Cuadrados, en inglés *Least-Square* (*LS*), así como en la inserción de símbolos piloto, pero utilizando diversas técnicas de interpolado.

- Evaluar dichos igualadores de canal utilizando un *USRP 2920* como transmisor y otro como receptor, siendo la medida de referencia la Tasa de error de bit , en inglés *Bit Error Rate* (*BER*).

- Evaluar la tasa de transferencia de datos una vez descartados los bits usados en símbolos pilotos, en inglés *throughput*, y determinar que tasa se puede alcanzar mediante los dos estándares anteriores. También se estudiará si la *BER* obtenida es adecuada para las comunicaciones mediante estos sistemas.

1.3 - Contribución

El presente proyecto quiere aportar una serie de aspectos novedosos, con alto interés para un estudiante de Grado en Ingeniería de Tecnologías y Servicios de Telecomunicación, éstos son:

- El diseño de un sistema de comunicaciones que incluya un modulador, un demodulador, un igualador de canal, un módulo de sincronismo y una interfaz gráfica utilizando la herramienta de diseño *LabVIEW 2013*. Con este diseño se pretende obtener una herramienta que sea de utilidad didáctica y que ayude a quien la utilice a entender mejor aspectos relacionados con *OFDM*, con la igualación de canal y con las comunicaciones digitales en general.

- Estudio de la capa física de los estándares de comunicaciones inalámbricas *LTE-Advanced* y *IEEE 802.16m*, profundizando en la estructura de las tramas (con duplexación *FDD*) así como en la distribución tanto temporal como frecuencial de los símbolos piloto.

- La evaluación de dicho sistema mediante la conexión de dos *USRP's* a sendos *PC's* utilizando un cable *Ethernet* de categoría 5e, pudiendo evaluar el sistema en un entorno real y analizando las novedades que esto conlleva respecto a la simulación del sistema, profundizando en algunos aspectos como la sincronización temporal o el tiempo de procesado de los módulos del sistema (aspecto crítico si el diseño no se ha realizado correctamente).

- Evaluación de las prestaciones de los igualadores de canal diseñados y posterior determinación de qué igualador es más apropiado, utilizando como referencia parámetros como el *BER* o el *throughput*. De esta forma, se descubre qué características son deseables para un igualador de canal y la correspondiente distribución de pilotos para su correcto funcionamiento en un determinado entorno.

1.4 - Fases del TFG

En este apartado vamos a presentar la metodología seguida para la realización del presente trabajo, indicando las distintas tareas realizadas, así como un diagrama temporal en el que se muestra cuándo y con qué duración se realizó cada tarea.

Seguidamente se detallan las diferentes tareas realizadas para la elaboración del TFG. Éstas

pueden ser clasificadas en 5 fases: fase de trabajo previo y acondicionamiento, fase de búsqueda bibliográfica, fase de diseño, fase de evaluación de prestaciones y fase de redacción.

Fase de trabajo previo y acondicionamiento:

- Instalación y acondicionamiento del software *LabVIEW 2013* en los PC's en los que se iba a trabajar, así como los distintos *toolkits* y *drivers* necesarios (*NI Modulation Toolkit, NI Spectral Mesurements Toolkit, NI Advanced Signal Processing Toolkit y USRP Drivers v1.3*), todo ello utilizando la licencia disponible en la UPV.

- Familiarización con el software *LabVIEW 2013*, realizando pruebas y ejercicios para aprender y dominar conceptos clave como la manipulación de *array's*, *cluster's*, *queue's,* gráficas, paneles de control, variables globales y locales, *subVI's*, estructuras de programación, cálculo matemático con *LabVIEW*, etc.

- Familiarización con el uso conjunto de *LabVIEW 2013* y el dispositivo *USRP 2920*, diseñando programas sencillos (como sistemas *QPSK*) y probando otros previamente diseñados por *National Instruments* (como por ejemplo *Spectral Measurement.vi*).

Fase de búsqueda bibliográfica:

- Estudio y búsqueda de información sobre sistemas *OFDM, LTE-A y IEEE 802.16m* y su diseño, así como de igualadores de canal e interpolaciones. También búsqueda de información relacionada con el diseño de sistemas de comunicaciones mediante *LabVIEW*.

Fase de diseño:

- Diseño del modulador *OFDM* mediante *LabVIEW 2013.*

- Diseño del demodulador *OFDM* mediante *LabVIEW 2013.*

- Comprobación del funcionamiento del modulador y el demodulador realizando simulaciones.

- Incorporación al modulador y demodulador *OFDM*, previamente diseñados, de los módulos necesarios para la interoperabilidad con el *USRP 2920*.

- Diseño del módulo de sincronización temporal, necesario a la hora de un correcto funcionamiento del sistema *OFDM*.

- Adaptación del sistema *OFDM* diseñado a las características y parámetros de *LTE - Advanced* y *IEEE 802.16 m*.

- Diseño de los distintos igualadores de canal para *OFDM*.

Fase evaluación de prestaciones:

- Evaluación de la *BER* obtenida con los igualadores de canal diseñados, así como estudio del *throughput* obtenido mediante los dos mencionados estándares.

Fase de redacción:

- Redacción de la memoria.

En la tabla 1 se muestra el cronograma que refleja la distribución de las tareas del proyecto desde el mes de febrero hasta el mes de julio de 2014.

	Febrero				***Marzo***				***Abril***				***Mayo***				***Junio***				***Julio***			
Semana	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Búsqueda bibliográfica	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Trabajo previo	x	x	x	x	x																			
Instalación *LabVIEW*	x																							
Aprendizaje *LabVIEW*		x	x	x																				
Pruebas *USRP*					x																			
Diseño						x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x			
Modulador *OFDM*						x	x	x	x															
Demodulador *OFDM*										x	x													
Simulación												x												
Módulo del *USRP*													x											
Módulo sincronización														x	x	x	x							
Adaptación a *802.16m*																		x						
Adaptación a *LTE-A*																			x					
Igualadores de canal																				x	x			
Evaluación prestaciones																						x		
Redacción memoria																							x	x

Tabla 1. Cronograma del trabajo realizado

Capítulo 2 - OFDM

2.1 - Introducción a OFDM

Actualmente podemos dividir las modulaciones digitales en dos grandes grupos. El primero, es el de las modulaciones de una sola portadora, grupo al que pertenecen modulaciones como *FSK*, *PSK*, *QAM*, *CPM*, etc. El segundo grupo es el de las modulaciones multiportadora, las cuales transmiten información de forma simultánea por diversas portadoras, lo cual hace que normalmente se puedan alcanzar *throughputs* más elevados que en las modulaciones de una sola portadora. Dentro este grupo encontramos modulaciones como *OFDM* o *DMT*.

Orthogonal Frequency - Division Multiplexing (*OFDM*) [1] es una modulación perteneciente al grupo de las modulaciones multiportadora. Ésta se basa en la utilización de portadoras ortogonales entre sí, lo que hace que idealmente no sea necesario un espaciado frecuencial entre ellas. Así, no es necesario destinar parte del espectro a bandas de guarda, como ocurre cuando se utiliza el multiplexado *FDM (Frequency - Division Multiplexing)*, tal y como se muestra en la figura 1. Por lo tanto, con *OFDM* normalmente se utiliza un ancho de banda mucho menor que con *FDM*.

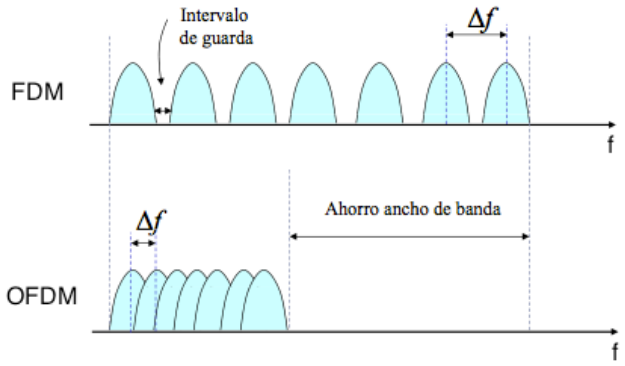


Figura 1. Comparación de *OFDM* vs *FDM*.

Entre las claves que han llevado al éxito de *OFDM* en la actualidad, destacan:

- Para canales con desvanecimiento, presenta mayor *throughput* que las modulaciones de una sola portadora.
- Mayor duración de símbolo transmitido, ya que la duración de un símbolo en *OFDM* es N veces mayor que la duración de un símbolo en una modulación de una sola portadora (siendo N el número de subportadoras utilizadas), por lo que *OFDM* es menos sensible a retardos provocados por el efecto multicamino y puede utilizar como técnica para combatir la Interferencia entre Símbolos (*ISI*) la adición de un prefijo cíclico.
- Mayor facilidad para combatir los desvanecimientos selectivos en frecuencia, debido a la división del ancho de banda total en subbandas, ya que de esta forma convertimos un canal selectivo en frecuencia en un canal no selectivo en frecuencia en cada subbanda.
- Uso de técnicas avanzadas de control de potencia que permiten obtener una gran eficiencia espectral, pudiendo llegar a unos 10 b/(s·Hz).

El éxito en la actualidad de *OFDM* viene dado, además de por las ventajas anteriores, por la facilidad para realizar la separación de la información en diversas subportadoras ortogonales gracias a la utilización de la *Transformada Discreta de Fourier (Discrete-Time Fourier Transform, DFT*), tanto en el modulador como en el demodulador. En el modulador, la *DFT* inversa *(IDFT)* nos permite separar los símbolos, previamente mapeados con alguna modulación de fase (*Phase Shift-Keying*, *PSK*) o amplitud (*Quadrature Amplitude Modulation, QAM*), entre las distintas subportadoras ortogonales en banda base, esto hace que no sea necesario utilizar tantos osciladores como subportadoras en el transmisor, y hace que se pueda utilizar en su lugar, un sólo oscilador. En el demodulador realizamos la *DFT* (*Discrete Fourier*

Transform) para obtener los símbolos. Además, si utilizamos un número de puntos que sea potencia de dos, ambas transformadas se pueden implementar mediante el algoritmo computacionalmente eficiente conocido como *Fast Fourier Transform* (*FFT*), el cual reduce notablemente el tiempo de cómputo. La utilización de la *IFFT* y la *FFT* para realizar este proceso abarata los costes en la fabricación del modulador y del demodulador, y ha contribuido de forma notable al éxito de la *OFDM* [2].

Por todas las ventajas anteriores, *OFDM* es una de las modulaciones más utilizadas en la actualidad, prueba de ello es la cantidad de estándares que utilizan dicha modulación, como por ejemplo *IEEE 802.11b/g/n* (más conocido por sus siglas *WiFi,*), *IEEE 802.16*, *LTE*, *DAB* (*Digital Audio Broadcasting*) o *DVB-T* (*Digital Video Broadcasting-Terrestrial*) entre otros.

2.2 - Esquema básico de un sistema OFDM

El esquema básico de un modulador *OFDM* en banda base es el mostrado en la figura 2. En él, podemos ver los 5 bloques principales del modulador, de los cuales se da una explicación más detallada a continuación:

***1 - Convertidor serie/paralelo**:* realiza la conversión serie/paralelo del *bitstream* de entrada, de tal forma que en cada una de las N salidas en paralelo haya m *bits*, siendo 2^m el número de símbolos de la constelación utilizada.

2 - Mapeador de símbolos: se encarga del mapeo de bits a símbolos utilizando una determinada constelación o modulación lineal. Las constelaciones habitualmente más utilizadas son: *BPSK* (*Binary PSK*), *QPSK* (*Quadrature PSK*), *16-QAM* y *64-QAM*. Normalmente las N entradas son mapeadas utilizando la misma constelación, pero se puede dar el caso de que las entradas se mapeen con distintas constelaciones, sobre todo en canales muy selectivos en frecuencia donde en determinadas subportadoras, en las que hay fuertes desvanecimientos, serán necesarias constelaciones más robustas, mientras que en otras portadoras que no presenten desvanecimientos se pueden utilizar constelaciones más eficientes.

3 - IDFT: realiza la *IDFT* de los símbolos mapeados, lo que equivale a realizar la conversión del dominio frecuencial al dominio temporal, obteniendo los símbolos distribuidos en N subportadoras ortogonales.

4 - Convertidor paralelo/serie: lleva a cabo la conversión paralelo/serie para obtener de forma secuencial y ordenada los fragmentos que forman el símbolo *OFDM*.

5 - Módulo de adición del prefijo cíclico: se encarga de añadir el prefijo cíclico al símbolo *OFDM*. El prefijo cíclico tiene la función de evitar la *ISI*, por ello su longitud debe ser igual o mayor que la longitud del canal radio. Además, para no perder la ortogonalidad entre portadoras, se eligen como prefijo cíclico las CP últimas muestras de cada símbolo *OFDM* y se añaden al principio del mismo. Si en lugar de las CP últimas muestras dejáramos una banda de guarda como prefijo cíclico se perdería la ortogonalidad entre portadoras al producirse retardos entre los distintos caminos.

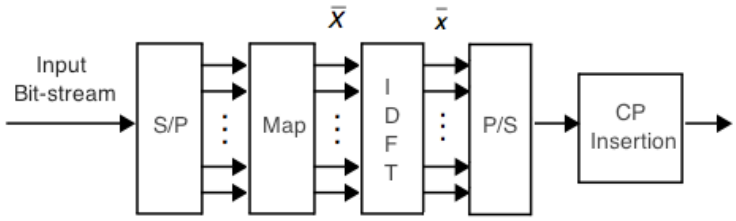


Figura 2. Diagrama de bloques de un modulador *OFDM* en banda base.

De forma casi inmediata, se puede deducir cuál es el proceso que se ha de seguir para la demodulación *OFDM* después de la etapa de sincronización, proceso que se muestra en la figura 3. Así, los seis principales bloques del modelo banda base del demodulador *OFDM* son:

1 - Módulo de supresión del prefijo cíclico: se encarga de quitar el prefijo cíclico que previamente se había insertado en el modulador.

2 - Convertidor serie/paralelo: lleva a cabo la conversión serie/paralelo de las distintas muestras de un símbolo *OFDM* para poder realizar la *DFT* posteriormente.

3 - DFT: se ocupa de transformar el símbolo *OFDM* del dominio temporal al dominio frecuencial.

4 - Igualador de canal: se encarga de estimar el canal radio y compensar su efecto sobre la secuencia de símbolos recibidos, dándonos como resultado una estimación de los símbolos transmitidos.

5 - Demapeador de símbolos: realiza el demapeado de los símbolos utilizando la constelación adecuada.

6 - Convertidor paralelo/serie: se ocupa de la conversión paralelo/serie de los *bits* obtenidos con el demapeador, y nos proporciona el *bitstream* de salida.

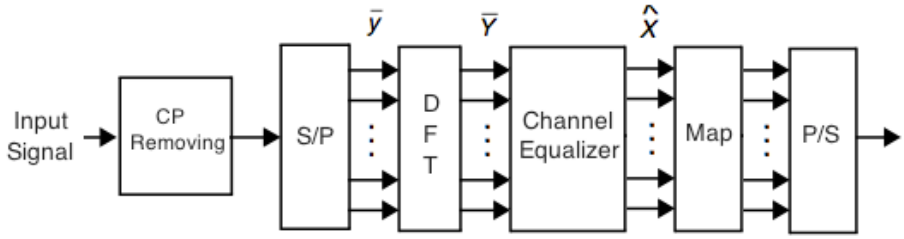


Figura 3. Diagrama de bloques de un demodulador *OFDM* en banda base.

2.3 - Igualación de canal en OFDM

2.3.1 - Introducción a la igualación de canal en OFDM

La igualación de canal es uno de los aspectos fundamentales en *OFDM*, ya que las señales normalmente son reflejadas o sufren algún tipo de dispersión, en inglés *scattering*, llegando al receptor a través de diversos caminos o *paths* (los cuales están desfasados unos de otros). Además, el transmisor, el receptor o los objetos que provocan las reflexiones y el *scattering* pueden estar en movimiento, haciendo que la respuesta del canal varíe rápidamente a lo largo del tiempo. También se ha de tener en cuenta que la respuesta del canal es aleatoria y que tiene relación con el entorno en que tenga lugar la transmisión. Por todo lo anterior, es necesaria alguna clase de corrección para recibir de forma correcta la información. Así, en esta sección se van a estudiar las técnicas disponibles para combatir los anteriores efectos provocados por el canal radio.

Los anteriores efectos provocados por el canal radio pueden ser modelados matemáticamente en *OFDM* según la ecuación 2.1. En ella vemos la respuesta del canal radio, con L caminos diferentes, cada uno de ellos con una atenuación compleja distinta ($\alpha_i(t)$) y con un retardo distinto (τ_i) [3].

$$h(t,\tau) = \sum_{i=0}^{L-1} \alpha_i(t)\,\delta(\tau - \tau_i) \qquad (2.1)$$

La ecuación 2.2 muestra el modelo matemático del símbolo *OFDM* recibido (Y) a la salida del bloque *DFT* en un determinado instante en función de X, H y W. Los dos últimos son vectores

columna de dimensiones $N\ x\ 1$, siendo H la *DFT* del canal $h[n]$ definido en la ecuación 2.1 por el que ha pasado el símbolo *OFDM* (suponiendo que las interferencias y la *ISI* han sido eliminadas) y W la *DFT* del ruido $w[n]$ que ha afectado a la transmisión de este símbolo. El ruido temporal w[n] se caracteriza por ser aditivo, espectralmente blanco y estadísticamente gaussiano, más conocido por sus iniciales inglesas *AWGN* (*Additive White Gaussian Noise*), y dado que la *DFT* es una transformación lineal, los valores del vector W conservan las mismas propiedades. Por último, $diag(X)$ es una matriz diagonal, en cuya diagonal se encuentra el símbolo *OFDM* transmitido, por lo tanto la matriz es de dimensiones $N\ x\ N$.

$$Y = diag(X) \cdot H + W \tag{2.2}$$

Tradicionalmente, en la igualación de canal en sistemas de una sola portadora, se aproximaba la respuesta al impulso del canal radio (a partir de ahora la llamaremos *CIR, Channel Impulse Response*) como un filtro de respuesta al impulso finita, en inglés *Finite Impulse Response* (*FIR*), y la estimación de canal se realizaba en el dominio temporal. En *OFDM*, se puede realizar la estimación de canal de la forma mencionada anteriormente, estimando la *CIR* y pasando al dominio frecuencial mediante la Transformada de Fourier, obteniendo la respuesta frecuencial del canal radio conocida como *Channel Frequency Response* (*CFR*). Sin embargo, una de las grandes ventajas que presenta *OFDM*, es que se puede realizar la estimación directa de la *CFR*. También es necesario tener en cuenta que en *OFDM* se puede evitar la igualación de canal utilizando modulaciones diferenciales como la *DQPSK* (QPSK diferencial), aunque en este caso disminuye la tasa de transmisión y la señal a ruido unos 3 - 4 dB [3].

Podemos clasificar los métodos de igualación de canal en *OFDM* en dos grupos:

- *Estimación de canal ciega:* en este tipo de igualación no se hace uso de símbolos piloto ni secuencias de entrenamiento. La igualación se basa en la características estadísticas de las señales recibidas y por lo tanto se necesita conocer una gran cantidad de información (como por ejemplo la naturaleza del canal radio) para realizar la estimación de forma adecuada. Además, como ha de tener una cierta información del canal, este tipo de igualación no es adecuada para canales muy variantes en el tiempo ya que se tendría que ir renovando la información que se tiene sobre el canal.

- *Estimación de canal no ciega:* en este caso se utilizan para la igualación o bien estimaciones de canal previas (*Decision Directed Channel Estimation*), o bien la inserción en la señal transmitida de una serie de información previamente conocida por el receptor, como símbolos piloto (opción recomendada para canales muy variantes en el tiempo, *fast fading channels*) o secuencias de entrenamiento (opción recomendada para canales poco variantes en el tiempo, *slow fading channels,* en los que se supone que el canal es invariante a lo largo de varios símbolos *OFDM*).

Con el método de igualación usando estimación con pilotos (que es el que se va a utilizar en este proyecto) podemos utilizar, entre otros, los dos siguientes estimadores:

- Estimador *MMSE* (*Minimum Mean Square Error*): hace uso de las estadísticas de segundo orden del canal para minimizar el *Error Cuadrático Medio* (*Minimum Square error, MSE*), obteniendo buenos resultados en general. Como contrapartida, esta técnica tiene un coste computacional muy alto.

- Estimador *LS* (*Least-Squares*): Esta técnica no hace uso de las estadísticas del canal, obteniendo un *MSE* mayor, sobre todo en entornos con baja relación señal-a-ruido, en inglés *Signal-to-noise ratio* (*SNR*). A cambio ofrece un coste computacional muy bajo. El estimador *LS* se puede utilizar como estimador principal o como complemento al *MMSE*, ya que éste necesita ciertos parámetros del canal para la estimación que se tienen que obtener previamente mediante el estimador *LS*.

En este proyecto se decidió utilizar el estimador *LS* debido a las limitaciones a causa de realizar el procesado mediante *LabVIEW* y un PC, ya que en este caso todos los recursos del PC no van destinados al procesado, y probablemente no hubiera sido capaz de soportar la estimación *MMSE*. Por lo tanto, se ha implementado una igualación de canal con estimador *LS* sobre los símbolos piloto y diversos métodos de interpolación.

2.3.2 - Símbolos piloto

La distribución de los símbolos piloto no se puede hacer de forma aleatoria, ya que dependiendo de las características del canal se necesitará una distribución u otra. Por ello, la separación frecuencial de los símbolos piloto tiene que ser menor que el ancho de banda de coherencia del canal y la separación temporal menor que el tiempo de coherencia del canal. Así, para un canal *fast fading channel* selectivo en frecuencia, la separación tanto temporal como frecuencial de los piloto será muy pequeña. En cambio, para un canal *slow fading channel* no selectivo, la separación tanto temporal como frecuencial puede ser grande. La relación entre la separación de los pilotos y el tipo de canal se puede ver en la figura 4. En la figura 5 podemos ver un ejemplo de distribución de los símbolos piloto a lo largo del tiempo y la frecuencia.

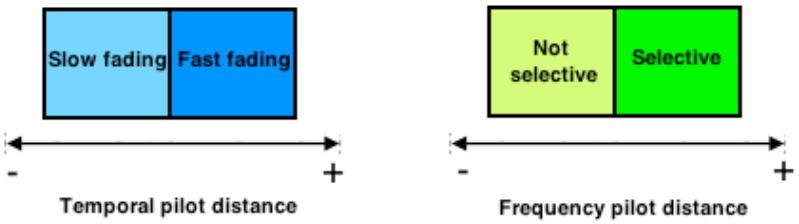


Figura 4. Separación de los símbolos piloto en función del tipo de canal

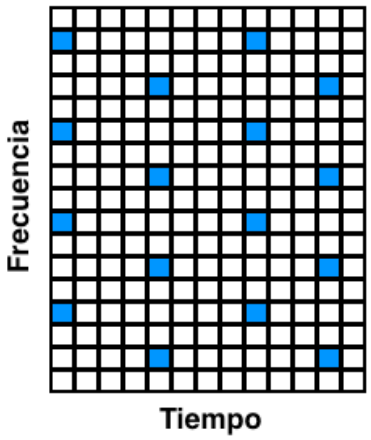


Figura 5. Ejemplo de distribución de pilotos

En cuanto a la constelación utilizada para los símbolo piloto, ésta debe ser de módulo constante, por lo que se utilizarán constelaciones *BPSK* o *QPSK*. También hay que destacar que para que la estimación sea mejor, la potencia de los símbolos piloto ha ser mayor que la del resto de símbolos.

2.3.3 - Estimador LS

El estimador *LS* es el más directo y con menor coste computacional. Está basado en la ecuación 2.2, teniendo en cuenta que se desprecia el término de ruido. Así, la ecuación 2.3 muestra la ecuación que nos da la estimación del canal mediante el estimador *LS*, siendo $Y[k]$ la señal

recibida en una subportadora y $X[k]$ la de referencia insertada en la señal transmitida y que el receptor conoce de antemano [3].

$$\widehat{H}_{LS}[k] = \frac{Y[k]}{X[k]}, \quad k = 0, 1, \dots, N-1 \tag{2.3}$$

En el caso de utilizar símbolos piloto y no una secuencia de entrenamiento, la ecuación 2.3 sólo se calculará para los valores de k en los que haya una subportadora piloto. Por comodidad a la hora de expresar matemáticamente los distintos procesos de interpolación, se utilizará un nuevo índice llamado m, el cual sólo tiene en cuenta las subportadoras que tienen un símbolo piloto (así, el primer símbolo piloto tendrá m = 0, el segundo m = 1, el tercero m = 2, etc), y también se define $\widehat{H}_P$, el cual sólo contiene el valor de la estimación de canal en las subportadoras con símbolos piloto. Así, cuando utilizamos el estimador *LS* con símbolos piloto, la ecuación 2.3 se transforma en:

$$\widehat{H}_P[m] = \frac{Y[m]}{X[m]}, \quad m = 0, 1, \dots, N_P - 1 \tag{2.4}$$

Una vez obtenida la estimación del canal en las posiciones de los símbolos piloto, se procede a la interpolación de la estimación para obtener la respuesta del canal en todo el ancho de banda utilizado en la transmisión. A continuación se presentan distintos métodos de interpolación, en los que es necesario tener en cuenta que la interpolación de la parte real y imaginaria del canal se hace de forma independiente.

2.3.4 - Métodos de interpolación

Los métodos de interpolación nos permiten obtener la estimación del canal en todas las subportadoras a partir del canal obtenido en las portadoras piloto. Existe infinidad de técnicas de interpolación, pero este proyecto se ha centrado en 5 de ellas, que se presentan a continuación.

Interpolación al más cercano:

Es el método de interpolación más sencillo. Consiste en que a las subportadoras en las que no hemos estimado el canal, les asignemos el valor de la estimación de canal de la subportadora piloto más cercana a ella. En la ecuación 2.5 podemos ver de forma matemática el proceso anterior, donde L es la separación entre símbolos piloto (la cual debe ser constante a lo largo de todo el ancho de banda), y l es la distancia desde la subportadora en la que nos encontramos hasta la subportadora piloto inmediatamente anterior. En la ecuación 2.6 se muestra el método de cálculo de l y m, el cual también se utiliza para otras interpolaciones como la lineal o la *Spline Cubic*. Podemos ver un ejemplo de este método de interpolación en la figura 6.

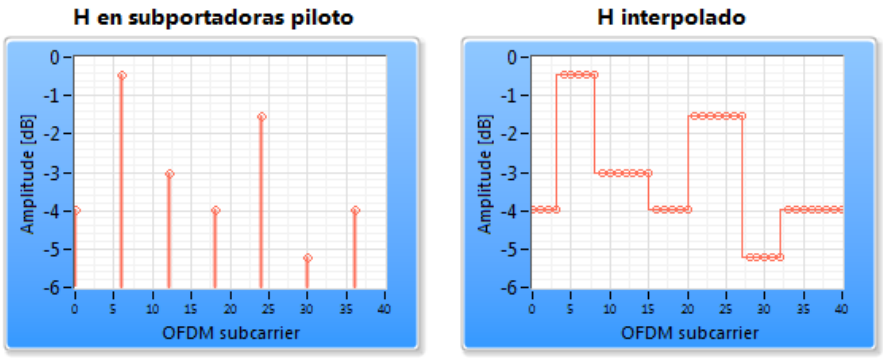


Figura 6. Interpolación al más cercano.

$$\widehat{H}[k] = \widehat{H}[mL + l] = \begin{cases} \widehat{H}_P[m] \ , & 0 < l < L/2 \\ \widehat{H}_P[m+1], & L/2 \leq l < L \end{cases} \tag{2.5}$$

$$l = Residuo(\frac{k}{L}) \quad m = cociente\left(\frac{k}{L}\right) \tag{2.6}$$

Interpolación lineal:

Con este método se calculan los valores del canal en las subportadoras a partir de una recta que une 2 pilotos consecutivos. Este tipo de interpolación sigue el proceso de la ecuación 2.7 para determinar la estimación de $\widehat{H}$ en una determinada subportadora k. En la figura 7 podemos ver un ejemplo de interpolación lineal [4].

$$\widehat{H}[k] = \widehat{H}[mL + l] = \left(\widehat{H}_P[m+1] - \widehat{H}_P[m]\right) \cdot \frac{l}{L} + \widehat{H}_P[m] \tag{2.7}$$

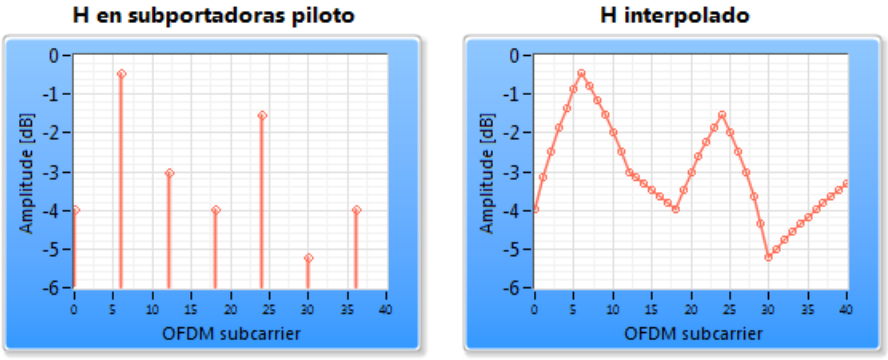


Figura 7. Interpolación lineal.

Interpolación en el dominio temporal:

Este tipo de interpolación se basa en las propiedades de interpolación de la transformada de Fourier. El proceso a seguir es el siguiente [4] :

- Realizamos la *IDFT* de N_P puntos (número de pilotos) de $\widehat{H}_P[m]$, como se indica en la ecuación 2.8.

$$h_p[n] = \sum_{m=0}^{N_P-1} \widehat{H}_P[m] e^{j\frac{2\pi mn}{N_P}} \ , \ n = 0,1,\dots,N_P - 1 \tag{2.8}$$

- Insertamos 0's siguiendo la ecuación 2.9.

$$h[n] = \begin{cases} 0, & n=0 \\ h_p[n], & 1 \leq n \leq \frac{N_P}{2} \\ 0, & \frac{N_P}{2} < n < N_{Útil} - N_P \\ h_p\left[n - N_{Útil} + \frac{3N_P}{2}\right], & N_{Útil} - \frac{N_P}{2} \leq n \leq N_{Útil} - 1 \end{cases} \tag{2.9}$$

- Realizamos la *DFT* de $N_{Útil}$ puntos (total de subbandas que queremos tener interpoladas), como podemos ver en la ecuación 2.10. Finalmente tenemos la respuesta

frecuencial del canal interpolada. En la figura 8 se muestra el resultado obtenido al realizar este tipo de interpolación.

$$H[k] = \sum_{n=0}^{N_{Útil}-1} h[n] e^{-j\frac{2\pi mn}{N_{Útil}}},\ k = 0,1,\dots,N_{Útil} - 1 \qquad (2.10)$$

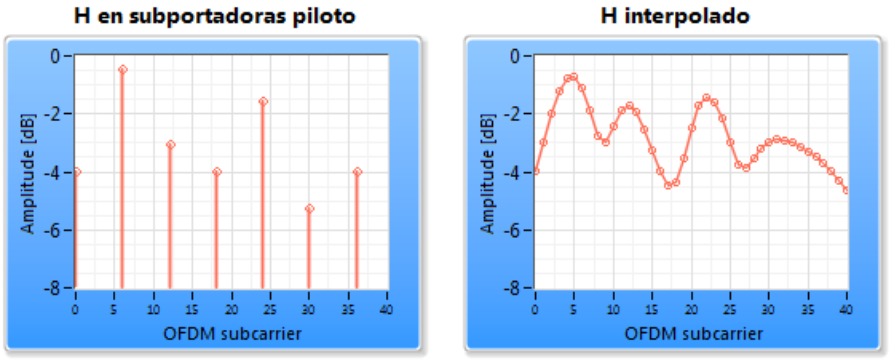


Figura 8. Interpolación temporal.

Interpolación Spline Cubic:

La interpolación *Spline Cubic* nos ofrece un resultado más suavizado gracias a que utiliza un polinomio de tercer orden que tiene la primera y segunda derivadas continuas tanto dentro del intervalo a interpolar como en los puntos conocidos. La ecuación 2.11 representa el proceso de interpolación, siendo los parámetros utilizados en ella los mostrados en las ecuaciones 2.12, 2.13, 2.14 y 2.15 [5]. Asimismo, m, L y l son los índices y parámetros explicados anteriormente en el punto 2.3.2. En la figura 9 se puede observar un ejemplo de interpolación *Spline Cubic*.

$$\widehat{H}[k] = \widehat{H}[mL + l] = A \cdot \widehat{H}_P[m] + B \cdot \widehat{H}_P[m+1] + C \cdot \widehat{H}_P''[m] + D \cdot \widehat{H}_P''[m+1] \qquad (2.11)$$

$$A = \frac{(m+1)L - k}{(m+1)L - mL} \qquad (2.12)$$

$$B = 1 - A \qquad (2.13)$$

$$C = \frac{1}{6}(A^3 - A) \cdot ((m+1)L - mL)^2 \qquad (2.14)$$

$$D = \frac{1}{6}(B^3 - B) \cdot ((m+1)L - mL)^2 \qquad (2.15)$$

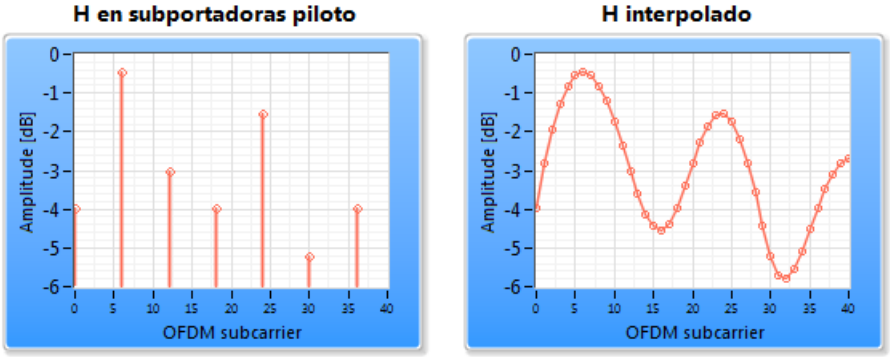


Figura 9. Interpolación *Spline Cubic*.

Interpolación utilizando un filtro paso bajo:

Este tipo de interpolación se basa en la interpolación con 0's del canal estimado en las subportadoras piloto y la posterior aplicación de un filtro paso bajo *FIR* (utilizando la convolución), el cual deja las muestras originales como estaban y realiza la interpolación en aquellas subportadoras que contienen 0's. Por un lado este método, es el que mejor resultados proporciona ya que minimiza el error cuadrático medio, pero por otro, si queremos que el filtro paso bajo se acerque al ideal (con un factor de *roll-off* bajo y una atenuación alta en la banda eliminada), el orden del filtro debe ser muy grande (infinito para el caso ideal) y por lo tanto el proceso de interpolación tiene un coste computacional muy alto. En la figura 10 podemos ver el resultado de una interpolación utilizando dicho método.

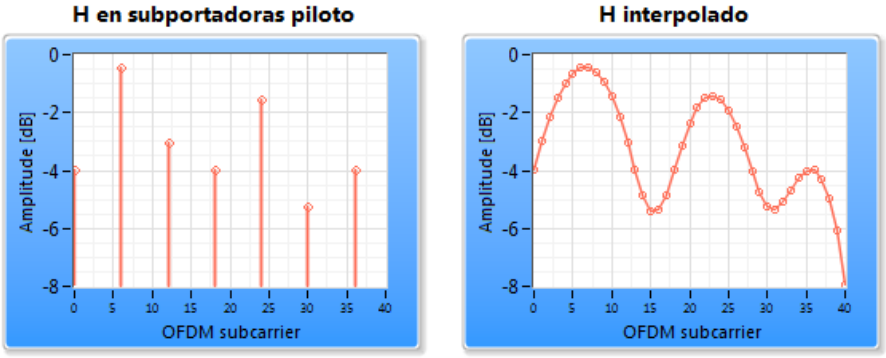


Figura 10. Interpolación con filtro paso bajo.

2.4 - Sincronización en sistemas OFDM

El proceso de sincronización temporal en sistemas de comunicaciones (especialmente en *OFDM*) es uno de los aspectos más importantes del diseño y determina el correcto funcionamiento o no del sistema. Además es un aspecto que normalmente no se contempla en la mayoría de trabajos de simulación a diferencia de una implementación real como es el caso de este proyecto.

En *OFDM* encontramos diversos métodos de sincronización temporal que utilizan diversas herramientas para la sincronización. Así, en un principio se escogió para el diseño el algoritmo de *Van de Beek* [6]. Este algoritmo nos proporciona tanto sincronismo temporal como en frecuencia, y se basa en la redundancia que provoca la inserción del prefijo cíclico, que hace que las muestras del prefijo cíclico y las muestras del final de cada símbolo *OFDM* estén fuertemente correladas. El algoritmo *Van de Beek*, que podemos ver en la figura 11, se ejecuta en ventanas de $2N_{IFFT} + CP$ muestras, y compara grupos de CP muestras (distanciados N_{IFFT} muestras dentro de la ventana). Realiza la autocorrelación y utiliza el estimador *ML* (*Maximum Likelihood*) para determinar la posición donde empieza el símbolo *OFDM*. En el diagrama de bloques de la figura 11, $r(k)$ hace referencia a las muestras recibidas en el receptor *OFDM*, $\hat{\theta}$ el índice de la muestra donde se ha estimado el comienzo del símbolo OFDM y $\hat{\epsilon}$ el error frecuencial estimado.

Después de la evaluación de este algoritmo en el sistema de comunicaciones, se decidió utilizar otro método de sincronización, ya que con *Van de Beek* no se podía determinar el comienzo de uno de cada 2 símbolos *OFDM* debido a que la ventana de observación $2N_{IFFT} + CP$ hacía que en el proceso de determinación del inicio del primer símbolo *OFDM* se perdieran muestras del segundo símbolo *OFDM*. Además, para una correcta sincronización, el mencionado método requería que se aplicara el algoritmo a cada símbolo *OFDM* obtenido, hecho que ralentizaba notablemente el proceso de demodulación.

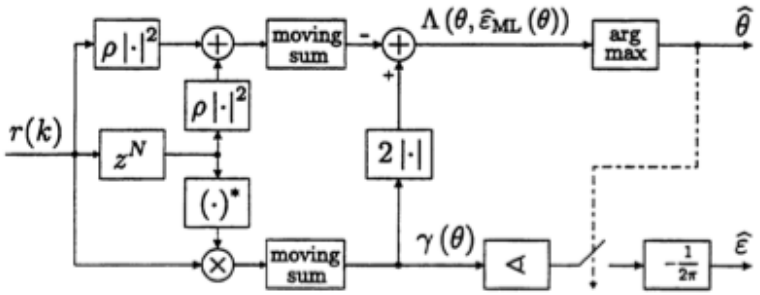


Figura 11. Algoritmo de *Van de Beek* para el sincronismo temporal y en frecuencia.

Finalmente, se optó por un método de sincronización temporal basado en el método de *Schmidl and Cox* [7]. En éste se utiliza un símbolo *OFDM* que actúa como preámbulo de sincronización, el cual tiene 2 mitades iguales en el dominio temporal (ya que tiene buenas propiedades de correlación), lo que se consigue dejando las subportadoras de datos con índice par a 0 en el dominio frecuencial (como se muestra en la figura 12). En el receptor se realiza la correlación cruzada entre un grupo de muestras recibidas en un determinado instante y el preámbulo de sincronización (que es conocido por el receptor), la correlación cruzada utilizada es sin normalizar, y su expresión se muestra en la ecuación 2.16. En ella, $x[k]$ es el grupo de muestras obtenidas por el receptor (siendo su tamaño $N + CP$) e $y[k + i]$ el preámbulo de sincronización desfasado i muestras. Una vez realizada la correlación cruzada se aplica un umbral (obtenido a través de pruebas empíricas) y si algún punto de la correlación obtenida supera dicho umbral, podemos asegurar que en dicho grupo de muestras se encuentra el preámbulo de sincronización y que su inicio se encuentra en la muestra i donde la autocorrelación supera al umbral mínimo. Este método de sincronización temporal funciona correctamente en el sistema *OFDM* y tiene la ventaja de que una vez detectado el preámbulo de sincronización ya no se tiene que volver a ejecutar la sincronización (al contrario de como ocurría en el método de *Van de Beek*). El único inconveniente es que con este método no obtenemos sincronización en frecuencia, pero esta deficiencia la solucionamos posteriormente con la igualación de canal.

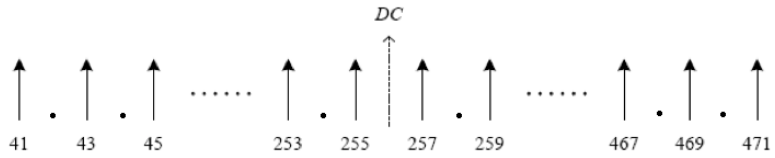


Figura 12. Estructura frecuencial del preámbulo de sincronización.

$$R_{xy}[i] = \sum_{k=0}^{N-1} x^*[k] \cdot y[k + i] \tag{2.16}$$

Capítulo 3 - OFDM en estándares de comunicaciones 4G

3.1 - Sistemas 4G

La modulación *OFDM* es utilizada en una gran variedad de estándares de comunicaciones en la actualidad. El presente trabajo se va a enfocar en dos de los estándares de comunicaciones de cuarta generación que la utilizan: *LTE - Advanced* y *IEEE 802.16m (WirelessMAN-Advanced)*.

La *ITU-R*, mediante el *IMT-Advanced* [8], establece una serie de requisitos que han de cumplir los sistemas de comunicaciones para que sean considerados de cuarta generación, entre otros deben cumplir:

- Utilizar redes *all-IP* (redes de conmutación de paquetes).
- Una tasa de al menos 100 Mbps cuando el usuario está en movimiento y al menos 1 Gbps para usuarios estacionarios.
- Ancho de banda escalable desde 5 MHz hasta 40 MHz.
- Capacidad de interoperar con otros sistemas de acceso radio.
- Equipos universales y que puedan operar en todo el mundo.
- Capacidad de soportar aplicaciones como el *streaming* de video de alta calidad.

Los dos estándares en los que se va centrar este proyecto *LTE - Advanced y IEEE 802.16 m* han sido los primeros en tener la categoría de estándares de cuarta generación y aún se encuentran en proceso de implantación hoy en día. En la figura 13 se puede ver la evolución de los sistemas de comunicaciones inalámbricas hasta llegar a 4G [9].

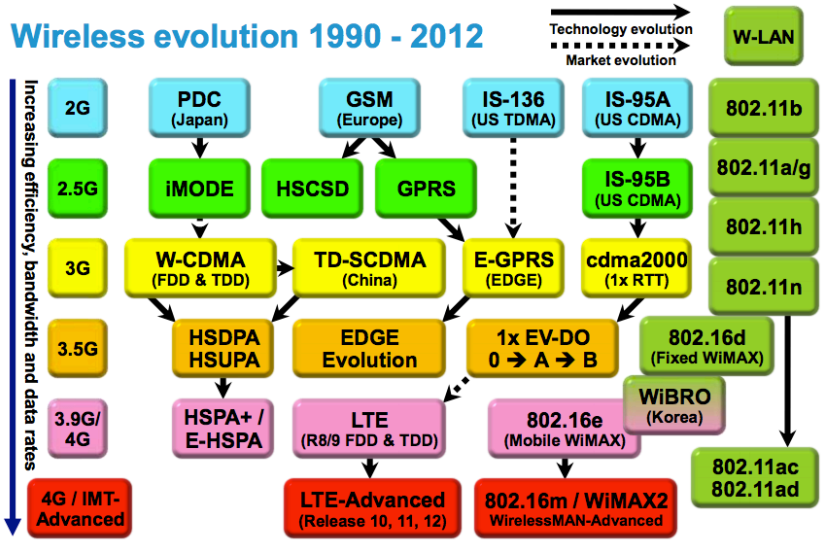


Figura 13. Evolución de los sistemas de comunicaciones inalámbricos.

3.2 - LTE y LTE - Advanced

3.2.1 - Introducción a LTE y LTE-Advanced

3GPP (*3rd Generation Partnership Project*) es la organización encargada de la estandarización de las tecnologías de comunicaciones celulares y, por lo tanto, ha sido quien ha establecido los principios de *LTE - Advanced*, pero ha habido un largo camino hasta llegar a esta tecnología. La tarea de estandarización de *3GPP* empezó en 1999 con el primer estándar basado en A*cceso por División en Código de Banda Ancha* (*Wideband Code Division Multiple Access, WCDMA*) que fue el primer paso en la evolución de 2G a 3G. Actualmente hay hasta 12 *releases*, como

podemos ver en la figura 14, entre los cuales están aquellos que han estandarizado algunas de las tecnologías actuales, como *HSPA+* o *LTE* [10].

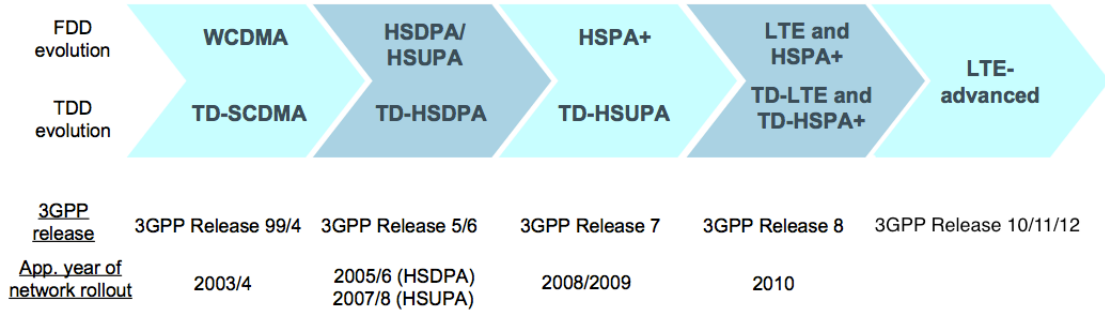


Figura 14. Evolución de los estándares de comunicaciones celulares propuestos por *3GPP* .

El proceso de estandarización de *LTE* fue finalizado en 2008, y las especificaciones de esta tecnología vienen en el *release 8* de *3GPP*. Aunque en un principio se concibió como una tecnología 4G, realmente es más bien 3,9G ya que no llega a cumplir las especificaciones del *IMT-Advanced* para tecnologías 4G, pero en cambio sí que supone un cambio drástico respecto a las tecnologías 3G existentes hasta el momento como HSPA. El principal cambio entre *LTE* y las estandarizaciones anteriores es el cambio en la modulación utilizada, ya que todas las tecnologías 3G anteriores a *LTE* utilizaban *W-CDMA* y no *OFDM* como *LTE*.

Ligado a la modulación utilizada, *LTE* utiliza como técnica de acceso en *downlink OFDMA* (*Orthogonal Frequency-Division Multiple Access*) en la que las distintas subportadoras son distribuidas entre varios usuarios, y en una subportadora sólo encontramos información transmitida por un usuario, y en *uplink SC-FDMA* (*Single-Carrier Frequency Division Multiple Access*) en la que la información de un usuario es distribuida entre varias subportadoras. *SC-FDMA* es un método de acceso que soluciona el excesivo *PAPR* (*Peak-to-average power ratio*) producido en *OFDMA* y que ayuda a la disminución de costes de los terminales de usuario. *SC-FDMA* reduce el *PAPR* ya que aunque un usuario utiliza mayor ancho de banda para transmitir, transmite durante menos tiempo. En la figura 15 se puede ver la comparación entre la distribución de usuarios en *OFDMA* y *SC-FDMA*.

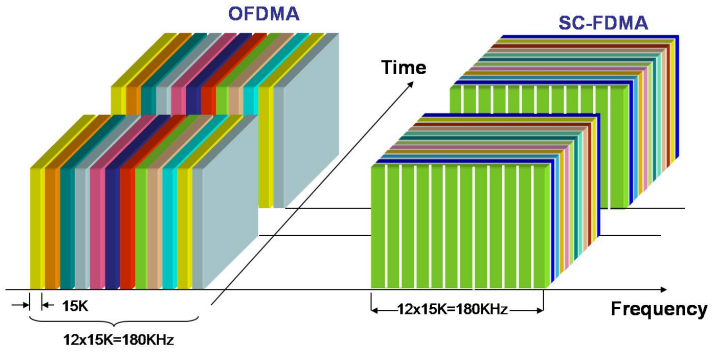


Figura 15. Comparación *OFDMA vs SC-FDMA*.

LTE también introduce cambios en el ancho de banda utilizado por cada usuario, siendo este variable entre 1,4 MHz y 20 MHz, pudiendo llegar a tasas de transmisión muy elevadas cuando se utiliza el ancho de banda máximo. Además, en *LTE* tenemos la posibilidad de utilizar constelaciones como la *16-QAM* o *64-QAM*, con las que se pueden conseguir tasas bastante elevadas.

Otro de los puntos novedosos de la capa física de *LTE* es el uso de técnicas *MIMO* (*Multiple-input and Multiple-output*) para aumentar la velocidad de transmisión. *MIMO* hace referencia a un canal de múltiples entradas (antenas transmisoras) y múltiples salidas (antenas receptoras). En esta estandarización se utilizan sistemas *MIMO* como máximo de 4x4 antenas para el enlace descendente y 2x2 antenas para el enlace ascendente [11].

Uno de los cambios más notables lo encontramos en la arquitectura de red, aunque se sigue manteniendo la conmutación de paquetes se simplifica mucho dicha arquitectura. En sistemas celulares anteriores la arquitectura de red era muy compleja, pues existía un gran número de módulos centralizados diferentes para el control de la red. En *LTE* se ha intentado simplificar el esquema de red distribuyendo las tareas de control entre las estaciones base (llamadas *evolved NodeB*). De esta forma, se ha reducido considerablemente el tiempo de conexión y el tiempo de respuesta ante *handover* , y por lo tanto se ha disminuido la latencia [11].

LTE y *LTE-Advanced* comparten la mayoría de los aspectos de la capa física. Así en *LTE-Advanced* también se utilizan como métodos de acceso *OFDMA* (*downlink*) y *SC-FDMA* (*uplink*), el ancho de banda también es escalable y las constelaciones utilizadas para *OFDM* son *QPSK*, *16-QAM* y *64-QAM*, igual que en *LTE*.

LTE puede alcanzar tasas de 100 Mbps en *downlink* y 50 Mbps en *uplink*. Estas tasas han sido mejoradas con la llegada de *LTE-Advanced* y la introducción de nuevas técnicas como:

- Agregación de portadoras, pudiendo llegar a 100 MHz de ancho de banda utilizando la agregación de 5 pbandas de 20 MHz [12].
- Mejoras en las técnicas *MIMO* utilizadas, con implementaciones de multiplexación espacial 8x8 en el enlace descendente y 4x4 en el enlace ascendente [13].
- *CoMP* o *Coordinated Multi Point Operation*, que consiste en que un usuario es servido por diversas estaciones base, que actúan de forma coordinada, tanto en *downlink* como en *uplink*.

Con todas estas mejoras se quiere llegar a velocidades de hasta 3 Gbps en *downlink* y 1,5 Gbps en *uplink*, y a una eficiencia espectral de hasta 30 b/(s·Hz).

3.2.2 - Estructura de las tramas LTE-Advanced

LTE es una tecnología que soporta tanto duplexación *TDD* (*Time-division Duplexing*) como *FDD (Frequency-division Duplexing)*. La siguiente explicación se va a centrar en la estructura de las tramas *LTE* en el enlace de bajada utilizando *OFDMA* y con duplexación *FDD* (aunque la estructura de las tramas con *TDD* es prácticamente idéntica), ya que, como se explicará posteriormente, ha sido el tipo de tramas que se ha diseñado mediante *LabVIEW*.

Espectralmente, tenemos N subportadoras (igual al número de puntos de la *IFFT*), aunque dejamos N_{guard} bandas de guarda en los extremos del espectro, por lo que las subportadoras que podemos utilizar para la transmisión son $N_{útil} = N - N_{guard}$. El conjunto de portadoras útiles se dividen en *Resource blocks,* éstos son grupos de 12 subportadoras que puede utilizar el usuario para la transmisión. Para el caso de *LTE-A*, la separación entre portadoras es de 15 kHz, por lo que un *RB* (*Resource block)* ocupa 180 kHz. El ancho de banda que puede utilizar un usuario es escalable y puede ir desde 1,4 MHz hasta 20 MHz en incrementos de 180 kHz, o lo que es lo mismo, cuando se quiere aumentar el ancho de banda se aumenta el número de *Resource blocks* utilizados. En la figura 16 podemos ver cómo se estructura frecuencialmente *LTE-A*: con *RB* de 12 portadoras, con la portadora *DC* como banda de guarda y un grupo de bandas de guarda en cada extremo.

Temporalmente, un *RB* tiene la duración de un *slot*, que es la mínima unidad de organización temporal. El número de símbolos *OFDM* en un *slot* depende del tamaño del prefijo cíclico que

se utilice, ya que existe el prefijo cíclico normal ($4{,}7\mu s$) y el prefijo cíclico extendido ($16{,}67\mu s$). Así, para los *slots* en los que se utilice el prefijo cíclico normal (como es el caso de nuestro diseño), un slot incluye 7 símbolos *OFDM*, de los cuales 6 tienen el prefijo cíclico normal y el primero de cada slot tiene un prefijo cíclico ligeramente superior ($5{,}21\mu s$), como podemos ver en la figura 17. Para el caso del prefijo cíclico extendido, un *slot* incluye 6 símbolos *OFDM* [14].

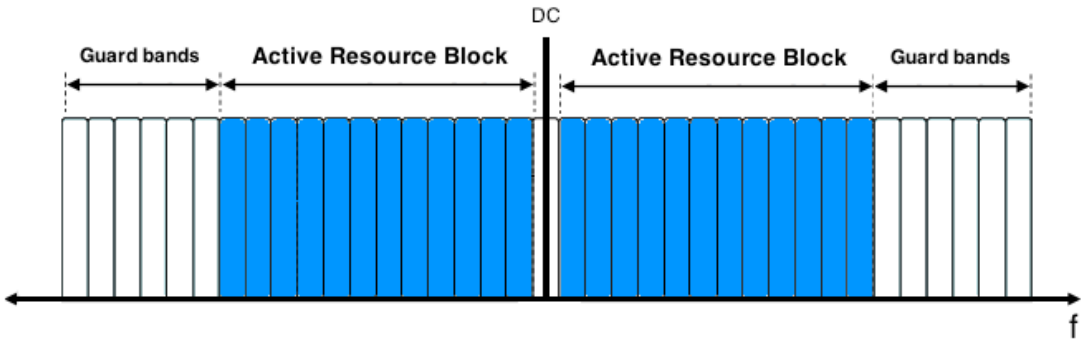


Figura 16. Estructura espectral de *LTE-Advanced*.

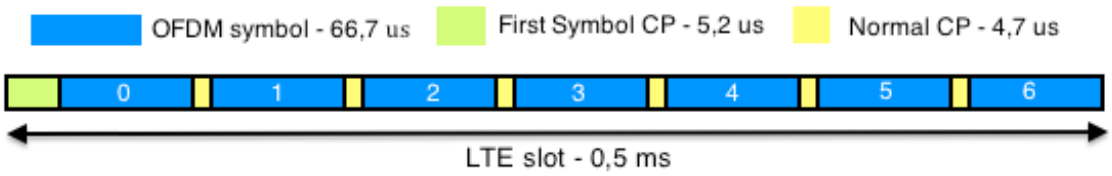


Figura 17. Estructura de un slot *LTE* - Advanced utilizando duplexación *FDD*

Así, un *Resource Block* ocupa 12 subportadoras durante un *slot* y es la unidad a partir de la cual se estructuran y distribuyen los usuarios *OFDMA* durante las transmisiones. Un ejemplo de dicha distribución lo podemos ver en la figura 18.

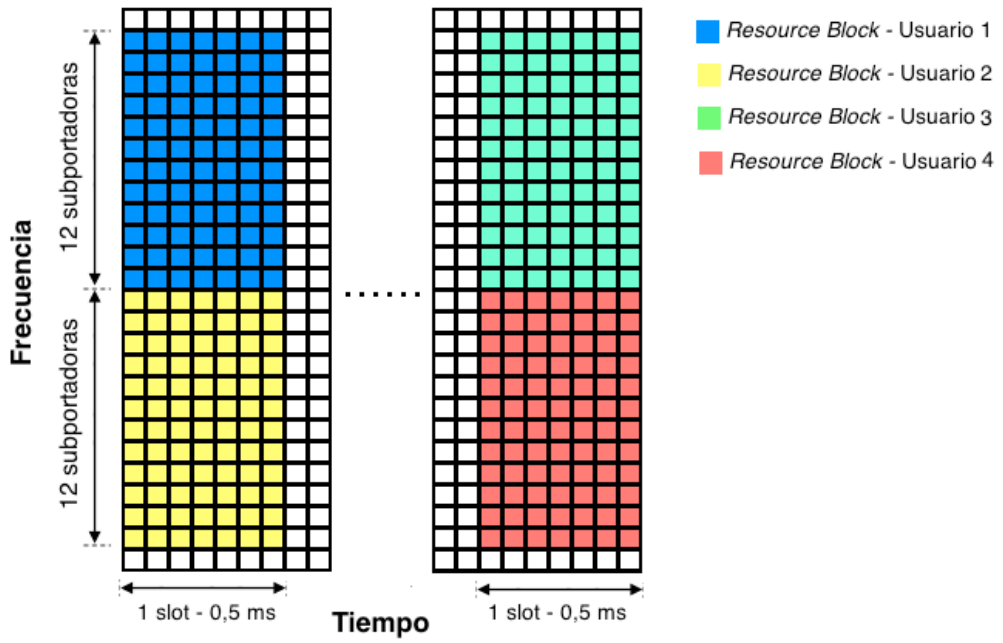


Figura 18. Distribución de los recursos en *OFDMA*.

Finalmente, los *slots* se agrupan en *Radio Frames,* que están compuestos por 20 *slots* como se puede ver en la figura 19. Así, en una transmisión que utilice el prefijo cíclico normal, en un *Radio Frame* tenemos un total de 140 símbolos *OFDM* con sus respectivos prefijos cíclicos y una duración total de 10 ms.

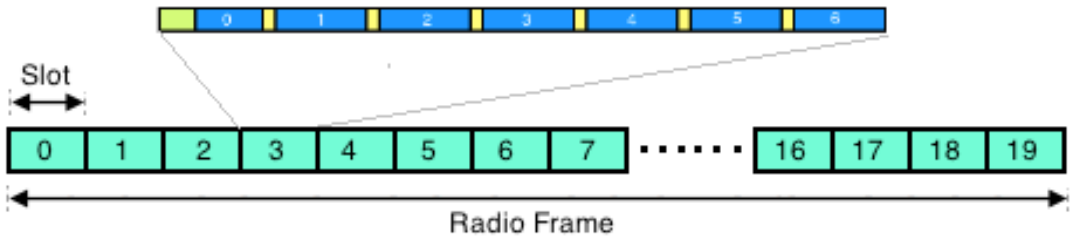


Figura 19. Estructura de una *Radio Frame LTE-Advanced*.

3.2.3 - Distribución de pilotos en LTE-Advanced

En *LTE-Advanced*, la distribución de los símbolos piloto se hace por *Resource Block's*, así en un *RB* tenemos un total de 4 pilotos, distribuidos en 2 instantes temporales (en el primer y quinto símbolo *OFDM* de un *slot*) y con 2 símbolos piloto por instante temporal (situados en la primera y la séptima subportadora del *RB* para el primer símbolo *OFD,* y en la cuarta y décima subportadora para el quinto símbolo *OFDM*). Podemos ver la distribución anteriormente explicada de una forma más clara en la figura 20.

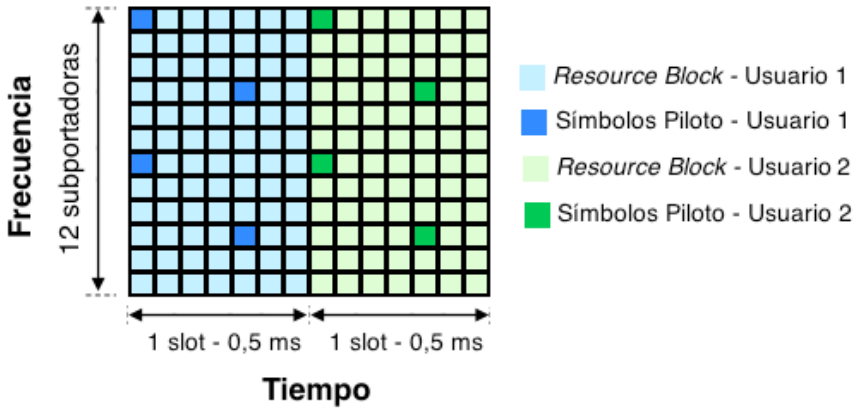


Figura 20. Distribución de pilotos en *LTE-Advanced*.

3.2.4 - Parámetros LTE-Advanced

Como hemos visto anteriormente, el ancho de banda en *LTE-Advanced* es variable y se elegirá un ancho de banda dependiendo de las necesidades. Es importante no confundir el ancho de banda que se utiliza en una transmisión y el ancho de banda que un usuario utiliza de este ancho de banda total o, lo que es lo mismo, no hay que confundir el ancho de banda de la transmisión *OFDM* y el ancho de banda que utiliza un usuario *OFDMA*; por ejemplo, se puede realizar una transmisión *LTE-A* con 20 MHz de ancho de banda y que haya 4 usuarios transmitiendo, cada uno de ellos con 5 MHz. El ancho de banda que utilizan los usuarios puede variar en incrementos de 180 kHz (ancho de banda de un *Resourse Block*).

En *LTE-A*, hay una serie de parámetros que son fijos sea cual sea el ancho de banda de la transmisión, éstos son [14]:

- La separación entre portadoras es siempre $\Delta \boldsymbol{f} = \boldsymbol{15kHz}$. Derivado de la separación entre portadoras obtenemos el valor de la duración de un símbolo *OFDM* que es $\boldsymbol{T_{OFDM}} = \frac{\boldsymbol{1}}{\boldsymbol{\Delta f}} = \boldsymbol{66,66\ \mu s}$
- Como se ha dicho en apartados anteriores el prefijo cíclico normal tiene una duración de **4,7 $\boldsymbol{\mu s}$,** el prefijo cíclico normal del primer símbolo de un *slot* $\boldsymbol{5,21\ \mu s}$ y el prefijo cíclico extendido $\boldsymbol{16,67 \mu s}$**.**

El resto de parámetros dependen del ancho de banda que se quiera utilizar, así se tiene que determinar:

- La proporción entre portadoras útiles ($\%_{Útil}$), bandas de guarda ($\%_{Guarda}$) y el ancho de banda deseado (B_W).

De este modo, si determinamos estos tres parámetros, podemos calcular el resto que serán necesarios para la transmisión. El proceso es el siguiente:

$$f_s = B_W \cdot n \tag{3.1}$$

(Siendo n el factor de muestreo y f_s la frecuencia de muestreo)

$$N_{IFFT} = \frac{fs}{\Delta f} \tag{3.2}$$

(Siendo N_{IFFT} el número de puntos que utilizaremos en la IFFT)

$$N_{Útil} = \%_{Útil} \cdot N_{IFFT} \tag{3.3}$$

(Siendo $N_{Útil}$ el número de subportadoras que contendrán información)

$$N_{Guarda} = \%_{Guarda} \cdot N_{IFFT} \tag{3.4}$$

(Siendo N_{Guarda} el número de subportadoras de guarda)

$$N_{RB} = \frac{N_{Útil}}{12\ subportadoras} \tag{3.5}$$

(Siendo N_{RB} el número de Resource Block)

$$N_{Pilot} = 2 \cdot N_{RB} \tag{3.6}$$

(Siendo N_{Pilot} el número de símbolos piloto por símbolo OFDM)

$$CP = \lceil 4{,}7 \cdot 10^{-6} \cdot f_s \rceil \tag{3.7}$$

(Siendo CP el número de muestras del prefijo cíclico)

$$CP_{Primer\ símbolo} = \lceil 5{,}2 \cdot 10^{-6} \cdot f_s \rceil \tag{3.8}$$

(Siendo $CP_{Primer\ símbolo}$ el número de muestras del prefijo cíclico del primer símbolo OFDM de cada slot)

Finalmente, los parámetros para cada uno de los anchos de banda admitidos en *LTE-Advanced* se presentan en la tabla 2.

B_W [MHz]	1,4	3	5	10	20
Δf [kHz]	15	15	15	15	15
$N_{Útil}$	72	180	300	600	1200
N_{Guarda}	56	76	212	424	848
N_{IFFT}	128	256	512	1024	2048
N_{RB}	6	15	25	50	100
n	1,371	1,28	1,536	1,536	1,536
f_s [muestras/s]	1,92E+06	3,84E+06	7,68E+06	1,54E+07	3,07E+07
Pilotos/(RB y símbolo)	2	2	2	2	2
N_{Pilot}	12	30	50	100	200
CP	9	18	36	72	144
$CP_{Primer\ símbolo}$	10	20	40	80	160

Tabla 2. Parámetros *LTE-Advanced*.

3.3 - WiMAX - IEEE 802.16m

3.3.1 - Introducción a IEEE 802.16m

WiMAX (*Worldwide interoperability for Microwave Access*) es el nombre comercial del estándar de comunicaciones móviles de banda ancha para áreas metropolitanas *IEEE 802.16* estandarizado por el *IEEE 802.16 Working Group on Broadband Wireless Access* y que es dado por el *WiMAX Forum* a todos aquellos equipos que cumplen los requisitos de dicho estándar. Entre las principales funcionalidades de *WiMAX* están proporcionar una alternativa a la red cableada *DSL* (*Digital Subscriber Line*) en la red de acceso y proporcionar conectividad portátil de banda ancha en áreas extensas de ciudades. El estándar ha ido evolucionando desde el comienzo de la tarea de estandarización del *IEEE* en 1999, partiendo de una solución para comunicaciones inalámbricas fijas con visión directa necesaria, utilizando modulaciones de una portadora como *QPSK*, y en la banda de frecuencias de 10 - 66 GHz (*IEEE 802.16 - 2001*), hasta llegar a poder dar servicio a clientes en movimiento, sin visión directa, utilizando *OFDM* y en bandas más bajas como 2500-2690 MHz o 3400-3600 MHz (*IEEE 802.16 m*) [15]. En la figura 21 podemos ver la evolución del estándar IEEE 802.16.

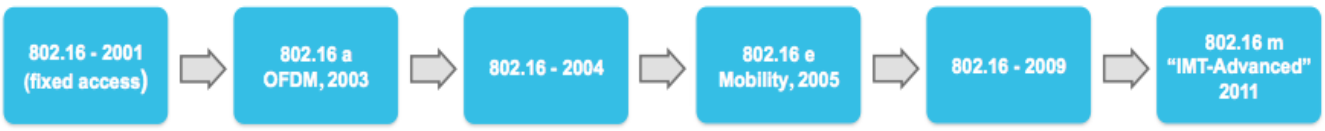


Figura 21. Evolución del estándar *IEEE 802.16*.

La última versión del estándar *IEEE 802.16* es *IEEE 802.16m*, también conocida como *WirelessMAN-Advanced*. Esta versión está diseñada para cumplir con los requisitos del *IMT-Advanced* para tecnologías de 4G (mencionadas en la introducción de este capítulo). *IEEE 802.16m* está concebido para dar conectividad de banda ancha a usuarios tanto estáticos como en movimiento, sin visión directa (*Non-Line of Sight*, *NLOS*), pudiendo dar servicio a un área de

hasta 100 km y velocidades de hasta 350 km/h (teniendo en cuenta que la calidad será menor cuanto mayor sea la distancia a la estación base y mayor sea la velocidad del usuario), con un retardo extremo a extremo de 10 ms y un tiempo de *setup* de 100 ms. Entre las novedades que incluye *IEEE 802.16 m* para cumplir los requisitos del *IMT-Advanced* encontramos:

- Capacidad de agregación de portadoras, pudiendo llegar hasta 100 MHz de ancho de banda con la agregación de hasta 5 bandas de 20 MHz [16].
- Reducción de las bandas de guarda entre portadoras al realizar la agregación de portadoras, pudiendo incluso utilizar las bandas de guarda para la transmisión de información.
- Mejoras en las comunicaciones mediante técnicas *MIMO*.
- Una duración de trama fija de 5 ms, al contrario que en versiones anteriores en las cuales la duración de trama era variable, y que ayuda a la disminución de la latencia en las comunicaciones [16].
- Interoperación con versiones anteriores del estándar como *IEEE 802.16e* y con otras redes inalámbricas de banda ancha como *IEEE 802.11* o *LTE-Advanced*.
- Un número menor de bandas de guarda que otras versiones anteriores del estándar como *IEEE 802.16e*.

Entre las principales características de la capa física de *IEEE 802.16m* podemos destacar [17]:

- Capacidad de duplexación *TDD*, *FDD* y *Hybrid-FDD*.
- *OFDMA* como técnica de acceso tanto en *Uplink* como en *Downlink*.
- Ancho de banda escalable desde 5 MHz hasta 20 MHz.
- Prefijo cíclico ajustable, dependiendo del entorno, pudiendo ser 1/4, 1/8 o 1/16 el tamaño del símbolo *OFDM*.
- Separación entre portadoras de $\Delta f_1 = 7{,}81\ kHz$, $\Delta f_2 = 9{,}77\ kHz$ y $\Delta f_3 = 10{,}94\ kHz$, siendo la última la más utilizada.
- Utiliza bandas asignadas a estándares anteriores como *IEEE 802.16e* (bandas 450 – 470 MHz o 2110 – 2200 MHz) y otras propias como 2500-2690 MHz o 3400-3600 MHz.
- Modulación *OFDM* con constelaciones *QPSK*, *16-QAM* y *64-QAM*.

3.3.2 - Estructura tramas IEEE 802.16m

Como en el caso de *LTE-Advanced*, la siguiente explicación se va a centrar en la estructura de las tramas *IEEE 802.16m* en el enlace de bajada utilizando *OFDMA* y con duplexación *FDD*, ya que, como se explicará posteriormente, han sido el tipo de tramas que se han diseñado mediante *LabVIEW*.

En *IEEE 802.16m*, la unidad mínima en la que se agrupan las distintas subportadoras *OFDM* es la *PRU* (*Physical Resource Unit*), que son grupos de 18 subportadoras y es el número mínimo de subportadoras que puede utilizar un usuario *OFDMA*. De esta forma, el ancho de banda que utiliza un usuario es escalable en grupos de 18 subportadoras (o lo que es lo mismo, cuando quiere aumentar el ancho de banda que utiliza, se aumenta el número de *PRU* utilizados). La distribución anteriormente explicada la podemos observar en la figura 22.

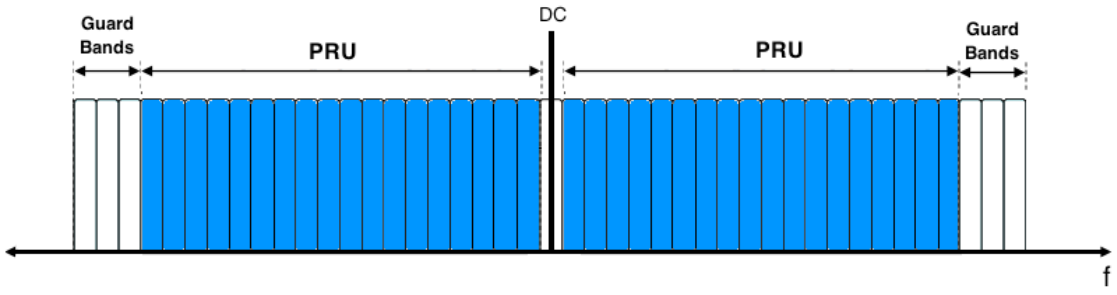


Figura 22. Estructura espectral del estándar *IEEE802.16m*.

Temporalmente, una *PRU* tiene la duración de una *subframe*, por lo que dependiendo del tipo de *subframe* que se utilice tendrá una duración de 6 símbolos *OFDM* (tipo 1), 7 símbolos *OFDM* (tipo 2), 5 símbolos *OFDM* (tipo 3) y 9 símbolos *OFDM* (tipo 4). La elección del tipo de *subframe* dependerá de la longitud del prefijo cíclico utilizado (1/4, 1/8 o 1/16) . En la figura 23 se puede ver una *subframe* del tipo 1 con prefijo cíclico 1/8 [17].

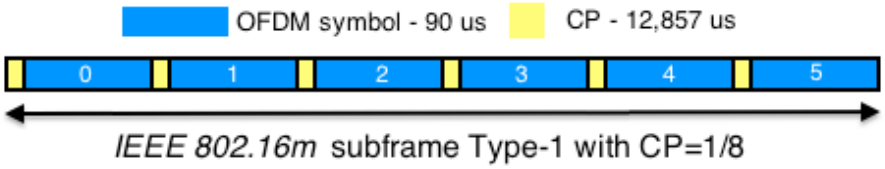


Figura 23. *IEEE 802.16m subframe* tipo-1 con prefijo cíclico 1/8.

Así, una *PRU* espectralmente ocupa 18 subportadoras y temporalmente una *subframe.* En la figura 24 podemos ver la estructura de una *PRU* con *subframe* tipo-1 tanto en el tiempo como en la frecuencia.

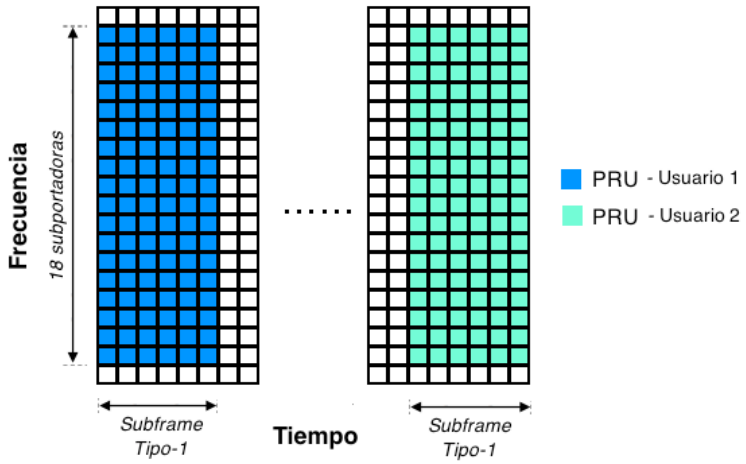


Figura 24. *PRU IEEE 802.16m*.

A su vez, las *subframes* se agrupan en grupos de 8 para formar las *frames* (las cuales tienen una duración fija de 5 ms), que a su vez forman las *superframes* (4 *frames* más una cabecera). Esta jerarquía se muestra en la figura 25.

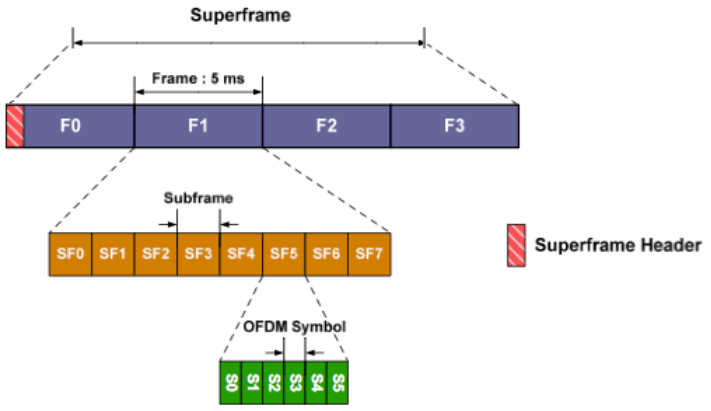


Figura 25. Organización temporal en IEEE 802.16m .

3.3.3 - Distribución de pilotos en IEEE 802.16m

En *IEEE 802.16m* los símbolos piloto se distribuyen por *PRU*, así en una *PRU* hay 1 símbolo piloto por cada 18 subportadoras y símbolo *OFDM*. Por lo tanto, en una *PRU* con *subframes* del tipo 1 tendremos 6 símbolos piloto (uno por cada símbolo *OFDM*) y una *PRU* con *subframes* del tipo 2 contendrá 7 símbolos piloto [17]. La posición frecuencial de los piloto va variando con el tiempo dentro de la *subframe*, en la figura 26 podemos ver una posible distribución de los símbolos piloto dentro de una PRU que utiliza *subframes* tipo-1. Comparando esta distribución con la de la de *LTE-Advanced,* vemos cómo en este caso hay menos pilotos frecuencialmente (1 cada 18 subportadoras en lugar de 2 cada 12 subportadoras como en *LTE-Advanced*), pero al contrario que en *LTE-Advanced* tenemos un símbolo piloto en cada símbolo *OFDM*.

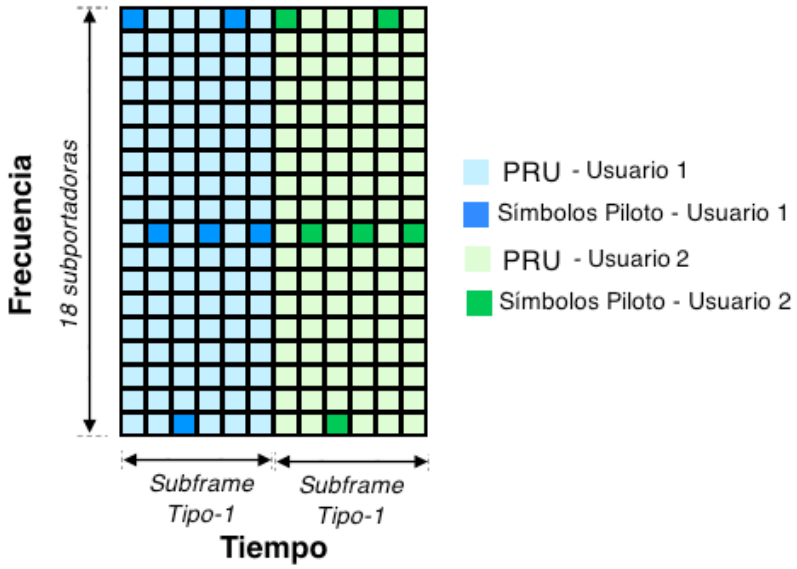


Figura 26. Distribución de los símbolos piloto en *IEEE 802.16m*

3.3.3 - Parámetros IEEE 802.16m

B_W *[MHz]*	**5**	**7**	**8,75**	**10**	**20**
Δf $[kHz]$	10,94	7,81	9,76	10,94	10,94
$N_{Útil}$	432	864	864	864	1728
N_{Guarda}	80	160	160	160	320
N_{IFFT}	128	1024	1024	1024	2048
N_{PRU}	24	48	48	48	96
n	28/25	8/7	8/7	28/25	28/25
f_s *[Muestras/s]*	5,60E+06	8+06	10E+06	1,12E+07	2,24E+07
Pilotos/PRU y símbolo	1	1	1	1	1
N_{Pilot}	24	48	48	48	96
CP	64	128	128	128	256

Tabla 3. Parámetros *IEEE 802.16m* con *subframes* tipo 1 y CP=1/8.

Como hemos mencionado anteriormente, el ancho de banda *OFDM* en *IEEE 802.16m* es variable, pudiendo ir de 5 hasta 20 MHz. Por ello, y al igual que en *LTE-Advanced*, a partir del ancho de banda y la proporción de bandas de guarda podemos obtener el resto de parámetros necesarios a la hora de realizar la transmisión. El proceso de cálculo de dichos parámetros es idéntico al que se muestra en la sección *3.2.3 - Parámetros LTE-Advanced,* por lo que el lector puede revisar el proceso allí mostrado para entender mejor los parámetros de la tabla 3, los cuales son los parámetros de *IEEE 802.16m* con subframes tipo 1 y CP =1/8 (se ha de subrayar que en el caso de *IEEE 802.16m* todos los símbolos tienen la misma longitud de prefijo cíclico, por lo que desaparece el parámetro CP primer símbolo).

Capítulo 4 - Software y hardware utilizado para el diseño

4.1 - LabVIEW

La compañía *National Instruments* describe LabVIEW como: *"una plataforma de programación gráfica que ayuda a ingenieros a escalar desde el diseño hasta pruebas y desde sistemas pequeños hasta grandes sistemas. Ofrece integración sin precedentes con software legado existente, IP y hardware al aprovechar las últimas tecnologías de cómputo. LabVIEW ofrece herramientas para resolver los problemas de hoy en día y la capacidad para la futura innovación, más rápido y de manera más eficiente"* [18].

LabVIEW es una herramienta de programación y diseño basada en el lenguaje de programación gráfico lenguaje G que se utiliza mayoritariamente para tareas de control y automatización de máquinas e instrumentación. En este tipo de lenguaje, cada instrucción corresponde con un bloque gráfico, el cual tiene una serie de *inputs* y *outputs* que se conectan a él (en la figura 27 podemos ver un ejemplo de bloque que calcula el tamaño de un *array*, éste tiene como entrada un *array* y como salida las dimensiones del mismo). Los programas son llamados *VI* (*Virtual Instrument*), y éstos contienen por una parte un panel de control, en el cual se encuentran los controles e indicadores que va a utilizar el *VI*, y por otra el diagrama de bloques, en el cual se encuentran los bloques que van a utilizar la información proporcionada por los controles y que tienen como salida los indicadores. En la figura 28 podemos ver un ejemplo de panel de control y su correspondiente diagrama de bloques.

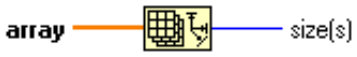


Figura 27. Función *array size* de *LabVIEW*.

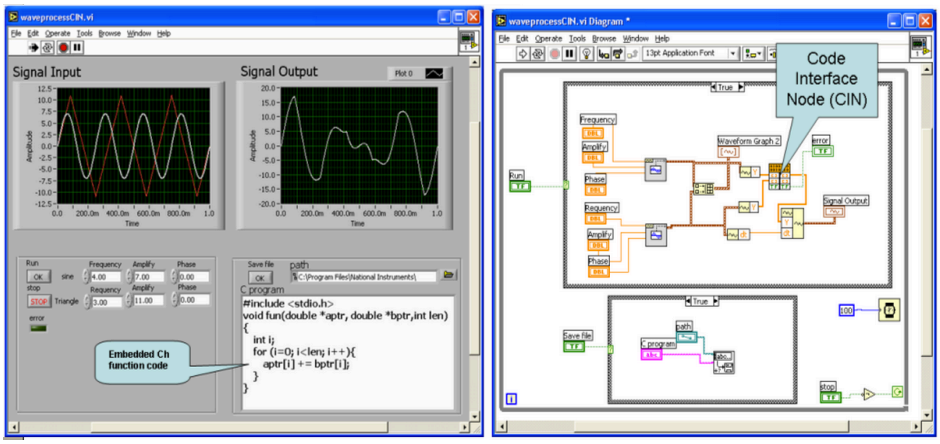


Figura 28. Ejemplo de diagrama de bloques y panel de control en *LabVIEW*.

LabVIEW soporta una gran variedad de tipos de datos, éstos se tienen que separar en 5 grupos según la naturaleza de los mismos. Así, pueden ser *Data String* (cadenas de caracteres), *Numeric Data*, *Boolean Data* (datos con el valor de *True* o *False*), *Dynamic Data* (grupo especial al que pertenecen, por ejemplo el tipo de dato de forma de onda) y *Agrupaciones* (a éste pertenecen *arrays, clusters* y matrices). El grupo que más nos interesa es el *Numeric Data,* dentro del cual encontramos una gran cantidad de subtipos de datos como *DBL* (*Double-precision, floating-point*) con 64 bits para la representación del dato o *U16* (*Word unsigned integer*) con 16 bits para la representación del dato y sólo admitiendo la representación de

números positivos. La principal diferencia entre los distintos subtipos de datos numéricos se encuentra en el número de bits que se utilizan para la representación y en si admiten o no números negativos [19].

En *LabVIEW* (al igual que en otros lenguajes de programación) se utiliza una estructura jerárquica en la que una tarea compleja se divide en una serie de tareas menos complejas. Por lo que aquellos programas que son muy grandes se estructuran en un *VI* principal y diversas *subVI*, esto es, una serie de programas más pequeños que se encuentran dentro del programa principal y que realizan operaciones concretas y más simples (equivalentes a las funciones de *C* o *java*).

LabVIEW tiene una infinidad de funcionalidades, desde módulos para cálculo matemático hasta módulos para crear flujos *UDP* o módulos para la creación de *Reports*. A continuación se van a explicar una serie de funcionalidades de *LabVIEW* que el lector debe conocer antes de interpretar el código del sistema *OFDM*, ya que han sido bastante utilizadas a la hora de programarlo, y a diferencia de otras funcionalidades (como los *FOR LOOP* o los *WHILE LOOP*) no son interpretadas de forma directa fácilmente . Éstas son:

- *Cluster:* Es una colección ordenada de variables de distintos tipos de datos.
- *Local Variable:* Es un tipo de variable que nos permite leer o escribir el valor de un indicador o un control del panel de control de un *VI* de forma local, es decir sólo si tenemos la *Local Variable* dentro del *VI*.
- *Global Variable:* Es un tipo de variable que nos ayuda a pasar información de un *subVI* a otro. Una *Global Variable* es una especie de *subVI*, con la particularidad de que sólo tiene panel de control, por lo tanto sólo almacena información (no realiza procesado) y esta información puede ser leída desde cualquier otro programa.
- *Queue:* Es una especie de *buffer* con política de servicio *FIFO* (*First-In First-Out*), el cual tiene la particularidad de que si la lectura de información es más lenta que la puesta de información (y por lo tanto se llena la *Queue*) se ralentiza aquella parte del código que hace de fuente de la información.
- *Case structure:* Es un tipo de estructura en la que dependiendo del valor que hay conectado a un terminal, llamado terminal condicional, se ejecuta un determinado proceso u otro. Es una funcionalidad análoga a la función *SWITCH* de *c* o *c++*.
- *Event structure:* Es un tipo de estructura que tiene la funcionalidad de esperar hasta que ocurra un determinado evento (presionado de un botón, cambio de valor de un control, etc) para realizar una determinada acción. Este tipo de estructura nos ayuda a no tener que utilizar *WHILE LOOP's* y *CASE STRUCTURE's* para determinar si ha sucedido un evento.
- *Flat Sequence Structure y Stacked Sequence Structure:* Son dos tipos de estructuras cuya principal finalidad es asegurarse de que una parte del código se ejecuta antes que otra y de esta forma no se produzcan errores a la hora de la ejecución.

Una de las principales ventajas de *LabVIEW* es la facilidad a la hora de programar varios hilos, ya que el mismo programa realiza esta tarea de forma óptima sin que el usuario se tenga que preocupar en la división por hilos.

National Instruments permite añadir funcionalidades a *LabVIEW* mediante los llamados *toolkits,* éstos aportan una serie de bloques para realizar procesos en un ámbito concreto. En este proyecto se han utilizado los siguientes *toolkit* (todos ellos relacionados con las comunicaciones RF y el tratamiento digital de señal) :

- *NI Modulation Toolkit:* conjunto de herramientas que permite realizar de forma directa modulaciones analógicas (*AM, FM, PM*) y digitales (*ASK, FSK, MSK, GMSK, PSK, QPSK, PAM, QAM*) de una forma muy sencilla, así como realizar codificaciones de canal y obtener parámetros como la *BER*.

- *NI Spectral Mesurements Toolkit:* brinda un juego de medidas espectrales flexibles, incluyendo espectro de potencia, pico de potencia y frecuencia, potencia en banda, potencia de canal contiguo y ancho de banda ocupada, así como habilidades de espectrogramas en 3D.

- *NI Advanced Signal Processing Toolkit:* es un paquete de herramientas de *software* y utilidades para el análisis de tiempo y frecuencia, análisis de series de tiempo y ondas corta, que incluye herramientas para el enventanado, filtrado, diseño de filtros, operaciones como la autocorrelación, etc

4.2 - USRP

Un *USRP* o ***Universal Software Radio Peripheral*** es una plataforma *hardware* pensada para su uso en comunicaciones de radiofrecuencia funcionando como transceptor. Suele ser utilizado en laboratorios de investigación y universidades (ya que puede ser de gran ayuda en el proceso de aprendizaje en el ámbito de las comunicaciones digitales así como en el testeo de nuevas modulaciones). Su uso está ligado a la utilización de un PC y de algún *software*, en nuestro caso *LabVIEW* mediante el cual se diseña el sistema de comunicaciones (aunque también se podría utilizar *Simulink* de *MATLAB*).

National Instruments ofrece diversos modelos de *USRP* (como el 2921, el 2930 o el 2932), todos ellos tienen una estructura *hardware* muy similar, las principales diferencias entre ellos radican en las bandas de frecuencia en las que funcionan y si incorporan o no una toma para una antena *GPS* para realizar el sincronismo de forma mucho más exacta. En nuestro caso, el dispositivo utilizado ha sido el *USRP 2920*. Éste presenta las siguientes características [20]:

- Funciona en la banda de frecuencias de 50 MHz a 2,2 GHz (por lo que es válido para sistemas como *FM*, *LTE-Avanced*, *UMTS*, *WiMAX*, etc).
- Frecuencia ajustable en pasos de 1 kHz.
- Rango de ganancias del amplificador de potencia en transmisión de 0 a 31 dB, ajustable en pasos de 1 dB.
- Rango de ganancias del amplificador de potencia en recepción 0 a 31,5 dB, ajustable en pasos de 0,5 dB.
- Figura de ruido de 5 a 7 dB.
- Máxima potencia de salida 17-20 dBm (para la banda 50 MHz a 1,2 GHz) y 15-18 dBm (para la banda de 1,2 a 2,2 GHz).
- Máxima potencia de entrada 0 dBm.
- Desviación máxima en frecuencia del oscilador en transmisión y recepción de 2,5 ppm.
- Una SFDR (*Spurious-Free Dynamic Range*) en el conversor digital analógico de 80 dB.
- Una SFDR en conversor digital analógico de 88 dB.
- No incluye sincronización con ayuda del sistema *GPS*.

En la figura 29 se ha incluido una recreación del plano frontal del *USRP 2920*, en él podemos ver como el dispositivo incluye [20]:

- 2 puertos de radiofrecuencia a los cuales se conectarán las antenas (el primero de ellos *full-duplex* y el segundo *half-duplex*).
- Un puerto Gigabit Ethernet para la conexión con un PC.
- Toma de alimentación de 6 V y 3 A de corriente continua.
- Un puerto para darle al USRP una referencia de tiempos.
- Un puerto de sincronización *pps*.
- Un puerto para la conexión *MIMO* de varios *USRP*.
- 6 *leds* que indican el estado del *USRP* en cada momento.

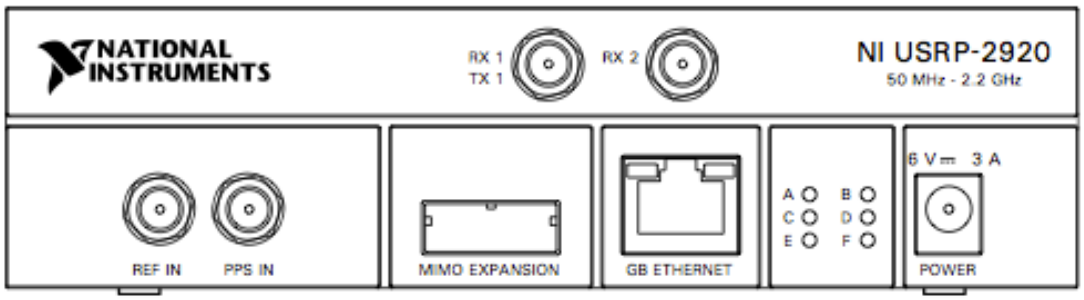


Figura 29. Plano frontal del USRP 2920.

En la figura 30 se puede observar el diagrama de bloques del *USRP 2920*. En el mencionado diagrama podemos observar 2 ramas diferenciadas, la de transmisión y la de recepción. Los principales elementos de cada rama son los siguientes [21]:

Transmisión:

- Módulo de control en transmisión.
- *DUC* (*Digital Up Converter*), módulo para subir en frecuencia la señal de entrada al *USRP*.
- *DAC* (*Digital to Analog Converter*), módulo que se encarga de la conversión de digital a analógico.
- Filtro paso baso bajo de 20 MHz, ya que el máximo ancho de banda que soporta es 20 MHz.
- Multiplicador y mezclador para juntar la componente en fase y en cuadratura.
- Amplificador de potencia variable en transmisión.
- *Switch* para cuando el puerto 1 funcione como *full-duplex.*

Recepción:

- *Switch* para cuando el puerto 1 funcione como *full-duplex.*
- Amplificador de bajo ruido.
- Amplificador de potencia variable en recepción.
- Multiplicadores para la separación de la componente en fase de la de cuadratura.
- Filtro paso baso bajo de 20 MHz.
- *ADC* (*Analog to Digital Converter*), módulo que se encarga del muestreo para la conversión de analógico a digital.
- *DDC* (*Digital Down Converter*), módulo para bajar en frecuencia la señal recibida, previamente filtrada y muestreada
- Módulo de control en recepción.

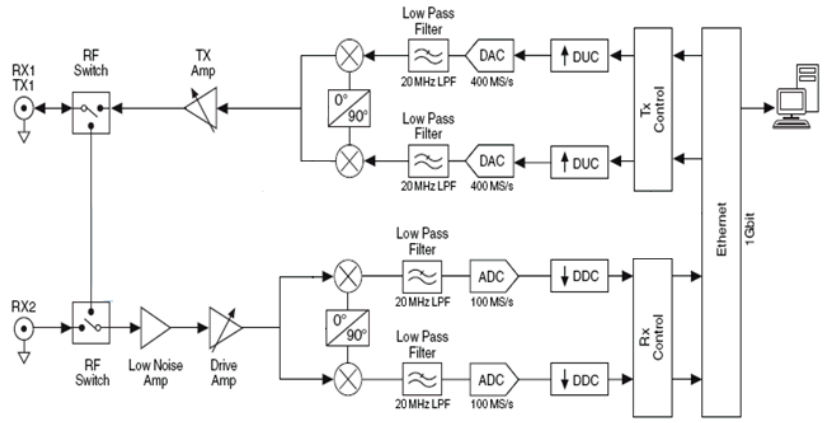


Figura 30. Diagrama de bloques de un *USRP 2920.*

En el anexo se adjunta un plano a escala donde se detallan las dimensiones y propiedades físicas del *USRP 2920*.

4.3 - Conexión del USRP al PC e interconexión con LabVIEW

El *USRP 2920* se conecta a un PC utilizando su puerto *Gigabit Ethernet* y un cable *Ethernet* de categoría 5e (100 MHz de ancho de banda), con el requisito de que la tarjeta de red del PC utilizado tiene que ser *Gigabit Ethernet*. Para el correcto funcionamiento de la conexión la imagen del *firmware* instalada en el *USRP* y el driver instalado en el PC deben ser compatibles, en nuestro caso se ha instalado la última versión (versión 1.3) tanto de la imagen del *firmware* como de los *drivers* del PC.

Para el correcto funcionamiento de la conexión se tienen que configurar los diferentes parámetros de la red de área local formada entre el *USRP* y el *PC*. Así, se tiene que asignar al PC y al *USRP* direcciones *IP* pertenecientes a la misma red de área local. En nuestro caso, se escogió la red 192.168.10.0/24, por lo que para el PC se optó por la dirección *IP* 192.168.10.1/24 y para el *USRP* una dirección *IP* 192.168.10.2/24. Además se tiene que habilitar el sistema *NetBIOS* en el PC, el cual pertenece a la capa de sesión del modelo *ISO* y que permite a aplicaciones instaladas en diferentes host comunicarse a través de una red de área local.

4.4 - Antenas

Las antenas utilizadas junto a los *USRP 2920* han sido dos antenas *VERT 2450*. Éstas son dipolos duales (son adecuadas para la banda de 2,4 - 2,5 GHz y la de 4,9 - 5,9 GHz), omnidireccionales y con 3 dBi de ganancia en las bandas mencionadas. Las características de dichas antenas se pueden encontrar en la página web su fabricante, *Ettus* (https://www.ettus.com /product/details/VERT2450). Viendo las características del *USRP 2920*, no es difícil darse cuenta que su rango de operación llega a 2,2 GHz, por lo que estas antenas no serían las más adecuadas para este dispositivo. Se han tenido que utilizar estas antenas debido a un error en el envío por parte de *National Instruments,* ya que tendrían que haber enviado junto a los dos *USRP 2920* dos antenas *VERT 900*, que son las adecuadas para trabajar junto a ellos. Este error ha provocado que el sistema no se pueda evaluar en escenarios con el transmisor y el receptor separados una distancia grande, ya que las antenas ofrecían muy poca ganancia.

Capítulo 5 - Sistema diseñado

En esta sección se va a describir el diseño realizado mediante *LabVIEW 2013* del sistema *OFDM* aplicado a *LTE-Advanced* y *WiMAX IEEE 802.16m*, estándares que (como se pudo ver en el capítulo 3) tienen una base común en la capa física y por lo tanto, el modulador y demodulador *OFDM* utilizados por ambos estándares serán muy similares y estarán basados en los diagramas de bloques de las figuras 2 y 3 que se vieron en el capítulo 2. Por ello, los dos mencionados estándares se incluyen en el mismo *VI* de *LabVIEW*, tanto en el modulador como en el demodulador, ya que la mayoría del código será común a ambos.

Primero se profundizará en el diseño realizado del modulador y seguidamente se pasará al demodulador (incluyendo una explicación detallada de los distintos igualadores de canal implementados).

5.1 - Modulador OFDM

A continuación se va a presentar el diseño del modulador *OFDM* realizado en este proyecto. El proyecto de *LabVIEW 2013* ha sido nombrado *OFDM_modulator.lvproj* y la *VI* principal de éste es *OFDM_modulator.vi*

Primero, se va a explicar cómo está organizado y estructurado el código de *OFDM_modulator.vi.* Así, el código se basa en la división de las tareas en *subVI*, que realizan tareas específicas. También se hace uso de una variable global que nos ayuda a tener los parámetros del estándar en cualquier *subVI* que lo requiera. Esta variable, *parameters*, se muestra en la figura 31 y sus valores se obtienen del panel frontal de *OFDM_modulator.vi* como se verá a continuación.

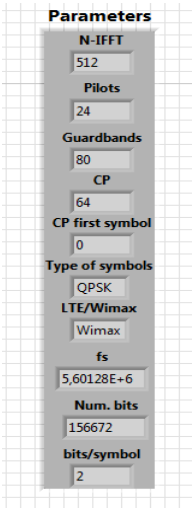


Figura 31. Variable global *parameters*

Asimismo, se ha utilizado una *Stacked Sequence Structure* para establecer un orden en la ejecución de las tareas. En la figura 32 se presenta una vista general del diagrama de bloques de *OFDM_modulator.vi* donde se pueden ver 4 de las 10 *frames* que forman el programa completo. Las 4 primeras (de la 0 a la 3) , están dirigidas al funcionamiento de la interfaz gráfica del modulador, la cuarta contiene los bloques principales del modulador *OFDM*, la quinta incluye módulos para la inserción de un preámbulo de sincronización y de una secuencia de fin, la sexta se encarga de la configuración del dispositivo *USRP 2920* y las 3 últimas están dedicadas a cálculos como el *Bit Error Rate y* al envío del resultado de la modulación *OFDM* al *USRP*

2920. Las *frames* se ejecutan en orden secuencial, y hasta que no ha terminado por completo la ejecución de una no se pasa a la siguiente. De esta forma se evita que partes del programa se ejecuten en un instante temporal no adecuado, lo cual evita errores que llevarían al mal funcionamiento del modulador.

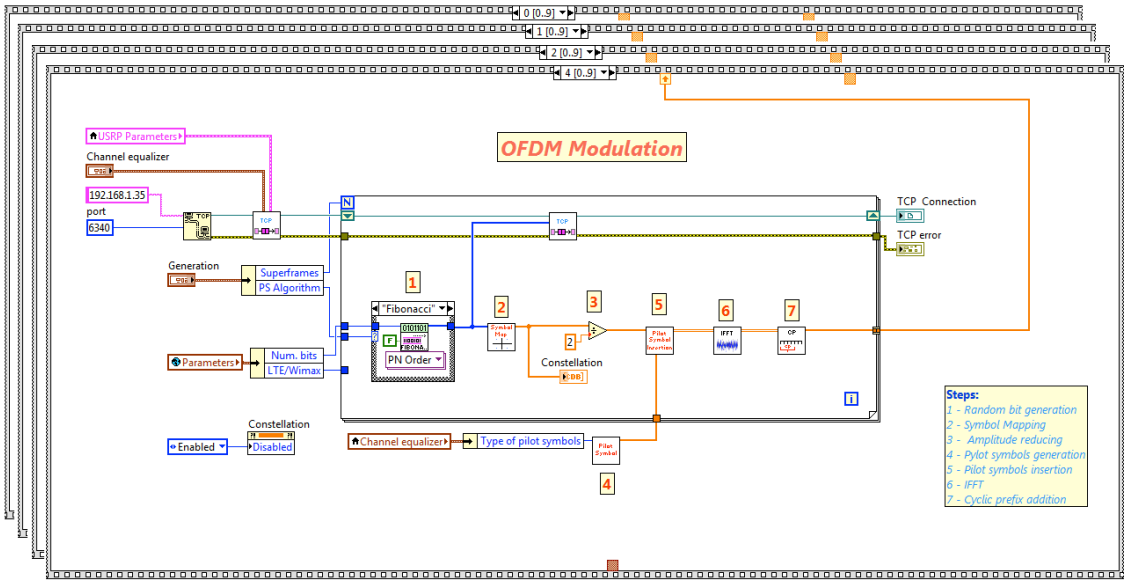


Figura 32. Visión general del diagrama de bloques de *OFDM_modulator.vi*

5.1.1 - Interfaz gráfica del modulador OFDM

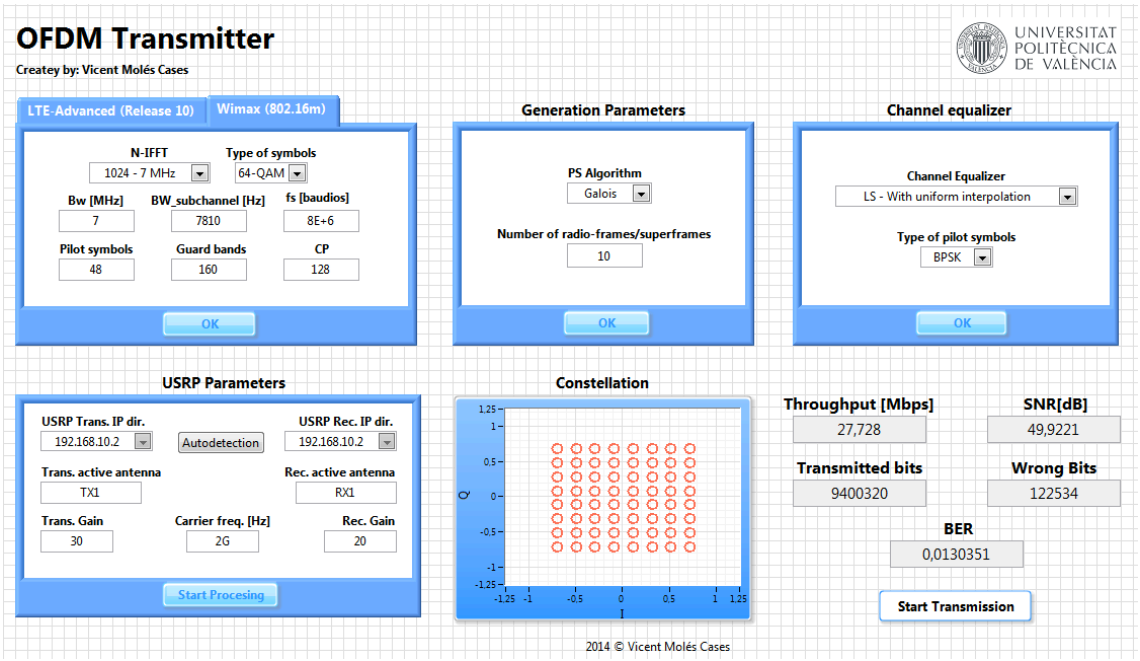


Figura 33. Panel frontal de *OFDM_modulator.vi.*

En la figura 33 podemos ver el panel frontal completo del modulador *OFDM*. En ella se pueden distinguir 5 secciones:

- *Parámetros de los estándares de comunicaciones*: se escoge el estándar, el número de puntos de la *IFFT* y la constelación de símbolos, y a partir de ellos se obtiene el resto de parámetros de la modulación *OFDM*, como veremos más adelante. Una vez elegidos se presiona el botón *OK*.
- *Parámetros de generación de bits*: se escoge el algoritmo de generación *pseudoaleatoria* de bits y el número de *superframes/radio-frames* a transmitir. Al final la selección se presiona *OK*.
- *Parámetros del igualador de canal*: se elige el tipo de igualador a utilizar, el tipo de símbolos pilotos y se presiona el botón *OK*.
- *Configuración de los USRP 2920*: se introducen los parámetros de configuración de los 2 *USRP 2920* que se utilizarán en transmisión y recepción, respectivamente. Al finalizar se presiona el botón *Start Processing* para empezar el proceso de modulación *OFDM*.
- *Resultados*: incluye la constelación de símbolos que se ha obtenido, así como el *Bit Error Rate* de la transmisión, la *SNR* estimada, la *BER*, el número de bits transmitidos y el número de bits erróneos.

La gestión de la interfaz gráfica se lleva a cabo en las *frames 0, 1, 2 y 3* de la *Stacked Sequence Structure*, donde se han utilizado diversas *Event Structure* (como podemos ver en la figura 34) con el objetivo de que el programa espere a que se presionen los botones booleanos *OK*, *Start Processing* y *Start transmission*, para pasar de una fase a la siguiente. También se han utilizado *Property nodes* en modo *Disabled and Grayed Out* para deshabilitar los controles de las fases siguientes mientras se espera el presionado de uno de los botones booleanos. Así, no se pueden seleccionar los parámetros de generación o del igualador de canal si no se han seleccionado primero los parámetros del estándar, como se muestra en la figura 35.

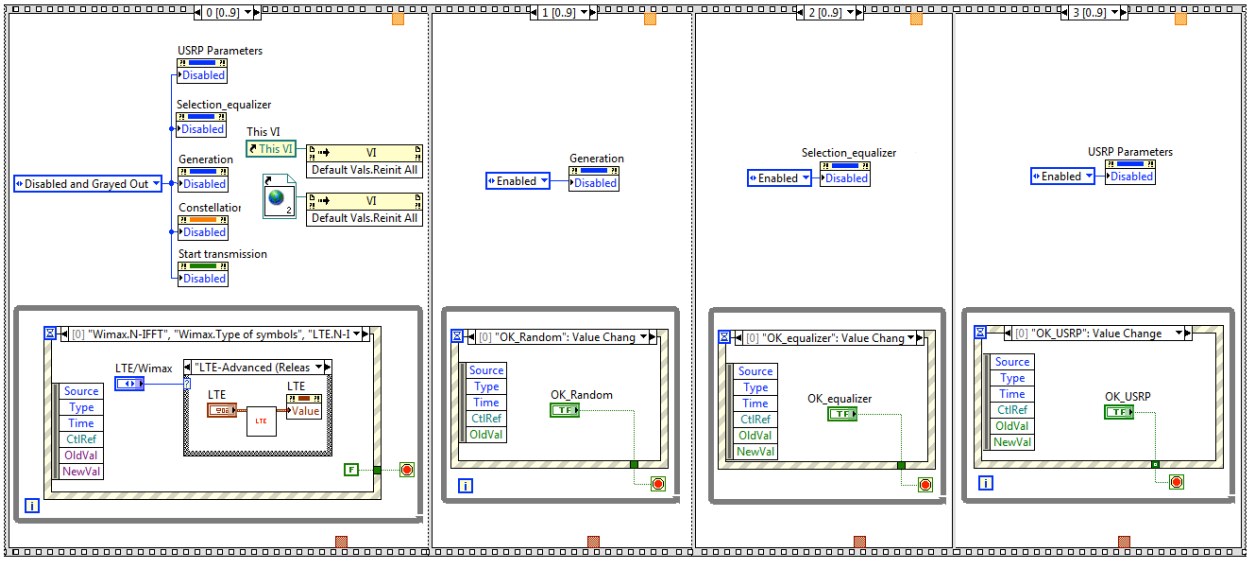


Figura 34. *Frames* 0, 1, 2 y 3 de *OFDM_modulador.vi.*

Cabe mencionar que en la *frame 0*, encontramos las funciones *LTE_parameters_selection.vi* y *Wimax_parameters_selection.vi* (que gracias a la *Stacked Sequence Structure* se ejecutan cada vez que se cambia el valor de *N-IFFT* en el panel frontal), las cuales en función del número de puntos de la *IFFT* que el usuario haya seleccionado se encargan de leer de entre un grupo de variables globales, que contienen los parámetros de las tablas 2 y 3 del capítulo 3. Una vez leídos los parámetros, se ejecuta la función *Number_of_bits.vi* (que calcula el número de bits de información que hay en una *radio-frame/superframe*) y finalmente se actualizan los parámetros en la variable global *parameters*.

Una vez obtenidos todos los parámetros de entrada (tanto del igualador de canal como de configuración del *USRP*), se realiza la modulación de los bits generados, y se espera a que el usuario presione el botón *Start Transmission* para empezar a transmitir las señales moduladas.

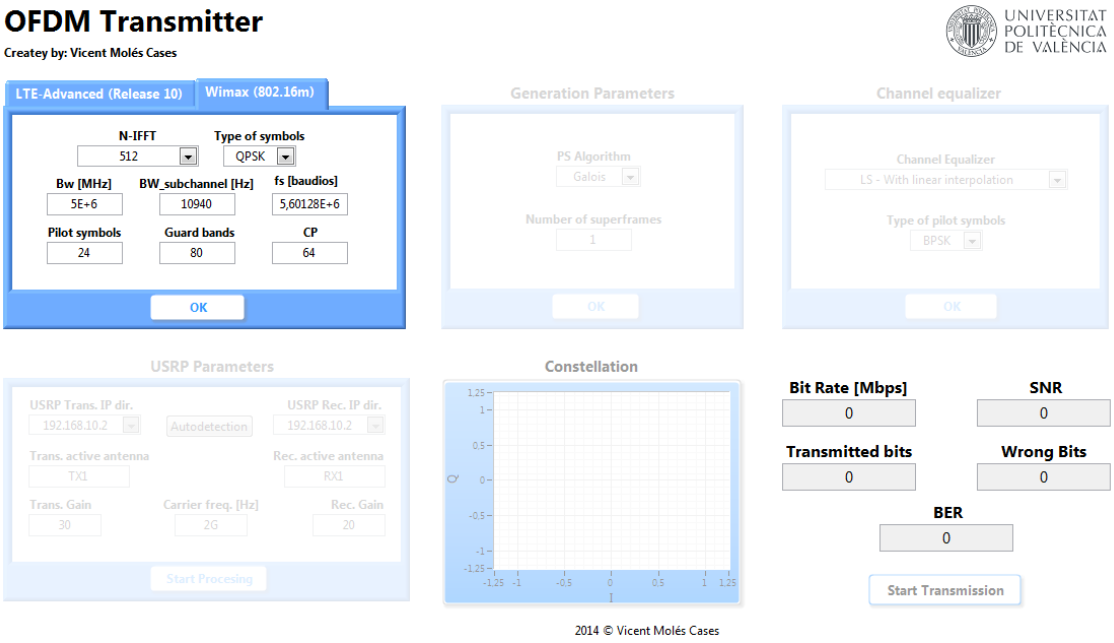


Figura 35. Panel frontal de *OFDM_modulator.vi.*

5.1.2 - Modulación OFDM

Seguidamente se van a explicar las distintas partes del código que se encargan de la modulación *OFDM* (están basadas en el diagrama de bloques de la figura 2 del capítulo 2) . Éstas se encuentran en la *frame 4* como se puede ver en la figura 32. En ella podemos ver cómo el proceso de modulación está basado en 7 procesos, los cuales se repiten dentro de un *FOR LOOP*. Esto es así ya que el modulador diseñado realiza la modulación en bloque, es decir, no realiza la modulación de un sólo símbolo *OFDM* sino de un grupo de símbolos *OFDM* (más concretamente para *LTE-Advanced* en cada iteración se realiza la modulación de una *radio-frame* entera y para el caso de *WiMAX - IEEE 802.16m* se realiza la modulación de una *superframe* entera).

Seguidamente se explican más detalladamente los 7 pasos que se realizan en cada iteración para la modulación de una *radio-frame* o una *superframe*, según convenga.

1 - Generación aleatoria de bits:

Para realizar la generación aleatoria de *bits*, primero hemos de saber cuántos *bits* compondrán la *radio-frame/superframe*. Por ello, la *subVI Number_of_bits.vi* de la figura 36 que se encuentra situada en la *frame 0*, se encarga de calcular el número de *bits* dependiendo de los parámetros que el usuario haya seleccionado. Nótese que en el cálculo del número de *bits* se tienen en cuenta los símbolos piloto que posteriormente se insertarán en la *radio-frame/superframe*.

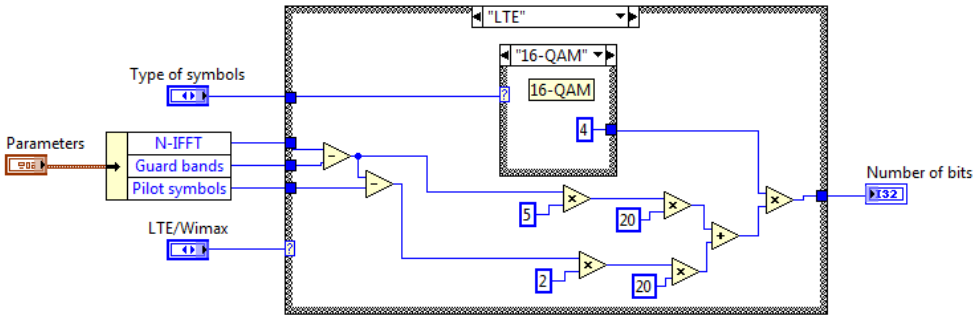


Figura 36. Diagrama de bloques de *Number_of_bits.vi.*

Para la generación aleatoria de los bits que forman la *radio-frame/superframe* se ha utilizado el módulo *MT Generate Bits* (perteneciente al *NI Modulation Toolkit)* el cual podemos ver en la figura 37 [22]. Este módulo permite la generación de secuencias pseudoaleatorias de *bits* utilizando secuencias de *Galois* y *Fibonacci*. Los *inputs* de la función son el total del *bits* a generar, el orden del polinomio generador (o lo que es lo mismo, la longitud del registro de desplazamiento utilizado para la generación de los *bits*), la secuencia con la que se inicia el registro de desplazamiento (*seed in*), el error procedente de otras funciones y una entrada booleana (*Reset*) mediante la cual indicamos si cada vez que se ejecute la función se reinicie el registro de desplazamiento (*TRUE*) o que el registro de desplazamiento se ponga en el estado en el que se había dejado en la última utilización (*FALSE*). En nuestro caso, el número total de *bits* es el número de *bits* de información que hay en una *radio-frame/superframe*, la *PN sequence order* se ha dejado en 9 (valor por defecto), la *seed in* también se ha dejado en su valor por defecto y se ha puesto *reset* con valor *FALSE* (para que las secuencias de *bits* generadas entre iteraciones del bucle FOR del modulador sean distintas). De esta forma, a la salida de esta función obtenemos un *array* de *bits* pseudoaleatorios (*output bit stream*) de tamaño igual al número de *bits* de información de una *radio-frame/superframe* y que varía entre iteraciones.

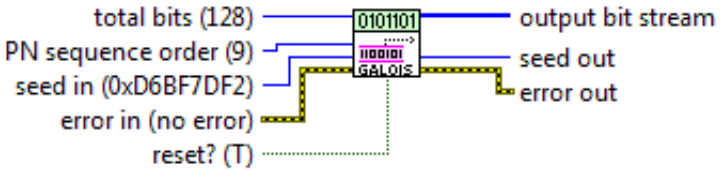


Figura 37. *MT Generate Bits*.

La selección de qué tipo de secuencia va a ser utilizada (*Galois* o *Fibonacci*) se realiza mediante una estructura *CASE*, la cual tiene conectada como condición el control del tipo enumeración *PS Algorithm* perteneciente al panel frontal del modulador *OFDM*.

2 - Mapeado de bits a símbolos:

Una vez generados los bits a modular, se tiene que realizar el mapeado de estos bits a una determinada constelación de símbolos, para el caso de los 2 estándares que estamos implementando estas constelaciones de símbolos son *QPSK*, *16-QAM* y *64-QAM*. Para realizar el mapeado se ha creado una *subVI* llamada *Symbol_mapping.vi* cuyo *block diagram* se muestra en la figura 38. En él, se han utilizado las funciones *MT Generate System Parameters* y *MT Modulate PSK* o *Modulate QAM* (según convenga) para realizar el mapeado. Con la primera función obtenemos la constelación de símbolos que vamos a utilizar para el mapeado y con la segunda realizamos el mapeado de los bits. También vemos cómo se utiliza una *CASE*

STRUCTURE que tiene conectado al terminal condicional el control *Type of symbols* (perteneciente a la variable global *Parameters*) para la selección de la constelación a utilizar.

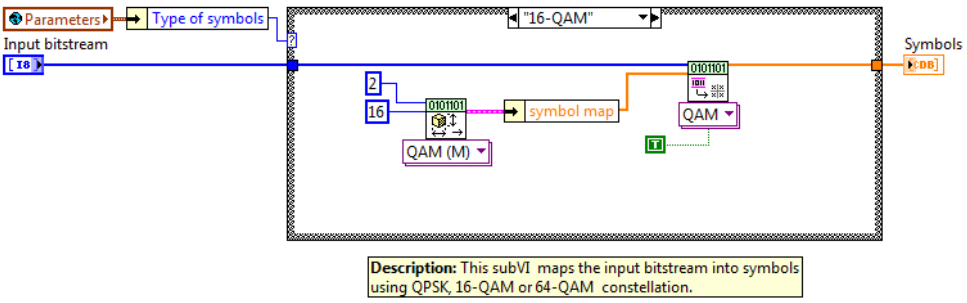


Figura 38. Diagrama de bloques de *Symbol_mapping.vi* .

Así, a la salida de esta *subVI* tendremos una *array* de símbolos *QPSK*, *16-QAM* o *64-QAM* que tendrán el formato *CDB* (*Complex double-precision, floating-point*).

3 - Escalado del módulo de los símbolos:

En este paso se reduce el módulo de los símbolos a la mitad sin alterar la fase (se puede ver en el punto 3 de la figura 32). El objetivo de la reducción del módulo de los símbolos que llevan la información es mejorar la estimación de canal en el receptor, ya que el módulo de los símbolos piloto que posteriormente se insertarán no estará reducido.

4 - Generación de la secuencia de símbolos piloto:

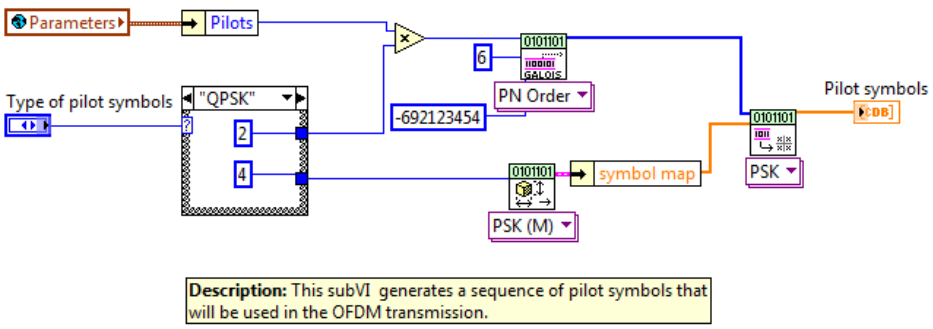


Figura 41. Diagrama de bloques de *Pilot_symbols_generation.vi.*

La generación de la secuencia de símbolos piloto se realiza una vez en cada transmisión, por lo que para cada símbolo *OFDM* tendremos la misma. Así, la generación se realiza fuera del *FOR LOOP* y se realiza mediante la función *Pilot_symbols_generation.vi* que se muestra en la figura 41. En ella se utiliza la función *MT Generate Bits* (con un polinomio de orden 6) para la generación de los bits que posteriormente serán mapeados (de la misma forma que en la función *Symbol_mapping.vi*) utilizando una constelación *BPSK* o una *QPSK*.

5 - Inserción de símbolos piloto:

En este punto del diseño es donde encontramos ciertas diferencias entre *LTE-A* e *IEEE 802.16m,* ya que, aunque la distribución frecuencial de los pilotos es casi idéntica en los dos estándares (sólo cambia el número de subportadoras que hay entre dos pilotos), en la distribución temporal encontramos ciertas diferencias porque en *IEEE 802.16* tenemos símbolos piloto en cada símbolo *OFDM* y en cambio en *LTE-A* sólo tenemos pilotos en el primer y quinto símbolo *OFDM* de un *slot.* Por lo tanto, tendremos que utilizar una estructura *CASE STRUCTURE* para separar el proceso de ambos estándares.

Para la inserción de los pilotos se ha utilizado la *subVI Pilot_symbols_insertion.vi* (la podemos ver en la figura 39 para *LTE-A y en la 40* para *IEEE 802.16m*). En ella se utiliza un *WHILE LOOP* para que en cada iteración podamos extraer un símbolo *OFDM* del *array* de entrada (que contiene tantos símbolos *OFDM* como símbolos de información tiene una *radio-frame/superframe*). Para la extracción del símbolo *OFDM* se utiliza la función *Delete From Array* (que borra los elementos que le son indicados de un cierto *array* y nos los da aislados en un *array* nuevo). Por lo tanto, en cada iteración obtenemos un símbolo *OFDM* y reducimos el *array* de entrada, el cual se va reduciendo hasta tener tamaño 0 (momento en el que se cumple la condición de parada del *WHILE LOOP*).

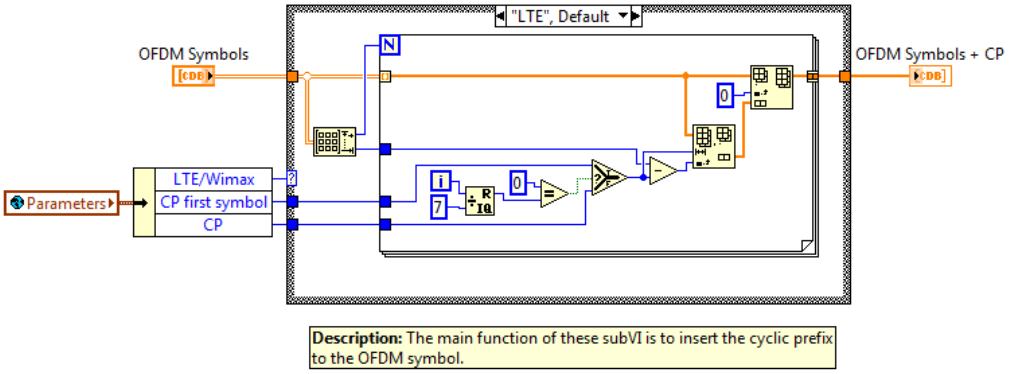


Figura 39. Diagrama de bloques de *Pilot_symbols_insertion.vi* para el caso de *LTE-A*.

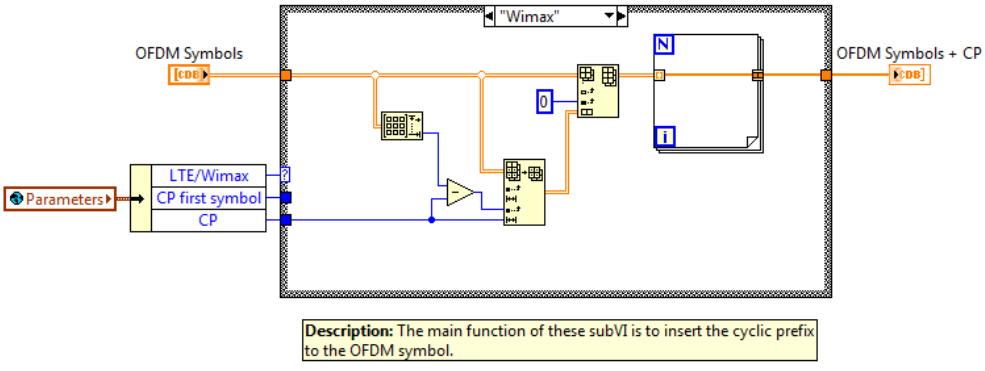


Figura 40. Diagrama de bloques de *Pilot_symbols_insertion.vi* para el caso de *IEEE 802.16m*.

En cada iteración, se realiza la división entera del índice de la iteración con el número de símbolos *OFDM* que contiene un *resource block LTE Advanced* o una *PRU WiMAX,*. De esta forma el residuo de la división nos da la posición en que se encuentra el símbolo *OFDM* actual dentro del *resource block/PRU* y por lo tanto sabemos si le corresponden símbolos piloto. Una vez conocemos la posición, realizamos la inserción de símbolos piloto si procede, esto es, como

se vio en el capítulo 3, en *LTE-Advanced* se insertan 2 símbolos piloto por *resource block* en los símbolos *OFDM* 0 y 4 de un *slot* y en *WiMAX IEEE 802.16* se inserta 1 símbolo piloto por *PRU* en todos los símbolos *OFDM* (siendo las distribuciones frecuenciales las mostradas en las figuras 20 y 26).

Para la inserción de los símbolos piloto se han utilizado las funciones *Decimate 1D Array* (que divide los elementos de un *array* en *x arrays* de misma longitud) e *Interleave 1D Array* (que realiza la operación recíproca).

6 - IFFT:

En este paso, se realiza la conversión de los símbolos del dominio frecuencial al temporal utilizando la *IFFT* y es uno de los puntos clave de *OFDM*, ya que gracias a la IFFT se consigue distribuir la información en *N-IFFT* subportadoras ortogonales frecuencialmente. En la figura 42 se muestra *IFFT.vi* que realiza la conversión entre dominios, para ello, se ha hecho uso de un *FOR LOOP* para recorrer todos los símbolos que forman una *radio-frame/superframe* y símbolo a símbolo aplicar la *subVI reorder.vi* y posteriormente la función *Inverse FFT*, que realiza la *IFFT* de *N-IFFT* puntos y nos da como resultado el símbolo *OFDM* en el dominio temporal.

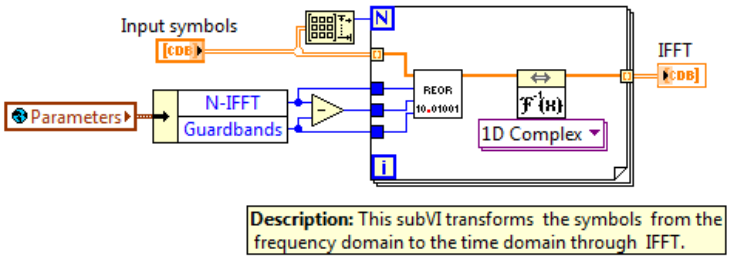


Figura 42. Diagrama de bloques de *IFFT.vi* .

La *subVI IFFT_symbol_structure.vi* (mostrada en la figura 43) realiza la reordenación de los símbolos para que tengan la estructura propia del dominio frecuencial y estén correctamente ordenados para la posterior aplicación de la *IFFT*. Para ello, se tiene que dejar la portadora correspondiente a *DC* con un valor de 0, a continuación tienen que estar la mitad de los símbolos útiles, seguidos de las bandas de guarda que se utilicen en la modulación y finalmente la otra mitad de símbolos útiles. Esta distribución se puede ver de forma más clara en la figura 44.

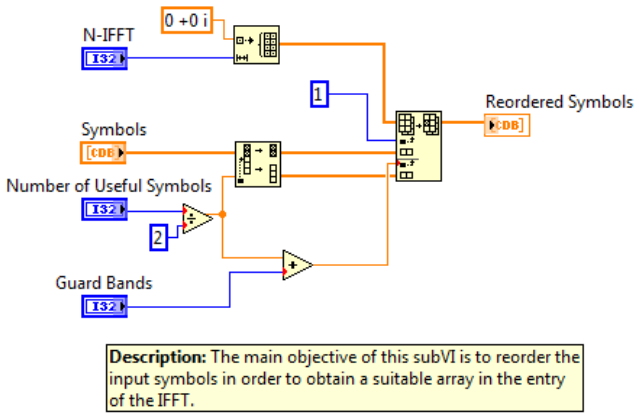


Figura 43. Diagrama de bloques *IFFT_symbol_structure.vi* .

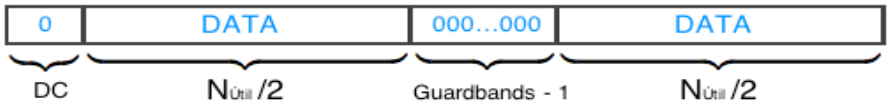


Figura 44. Estructura propia del dominio frecuencial.

7 - Inserción del prefijo cíclico:

El último bloque del modulador *OFDM* es el que se encarga de la inserción del prefijo cíclico y se ha implementado en la *subVI Add_CP.vi*. Este bloque es en el que más difieren los dos estándares, ya que como se vio en el capítulo 3, en *LTE-A* el primer símbolo *OFDM* de cada *slot* tiene un prefijo cíclico de duración mayor al resto de símbolos *OFDM* y en cambio en *IEEE 802.16m* todos los símbolos tienen la misma duración de prefijo cíclico. Por ello, y al igual que en la *subVI Pilot_symbols_insertion.vi*, se ha hecho uso de una *CASE STRUCTURE* para separar el proceso de inserción en ambos estándares.

Para el caso de *LTE-A* se utiliza un *FOR LOOP* para hacer la inserción del prefijo cíclico símbolo a símbolo, como se ve en la figura 45, y de esta forma poder utilizar distintas longitudes de prefijo cíclico para distintos símbolos. También se ha hecho uso de una división entera del índice de iteración entre el número de símbolos *OFDM* por *slot*, para identificar el primer símbolo de cada *slot* y de esta forma añadir el prefijo cíclico adecuado.

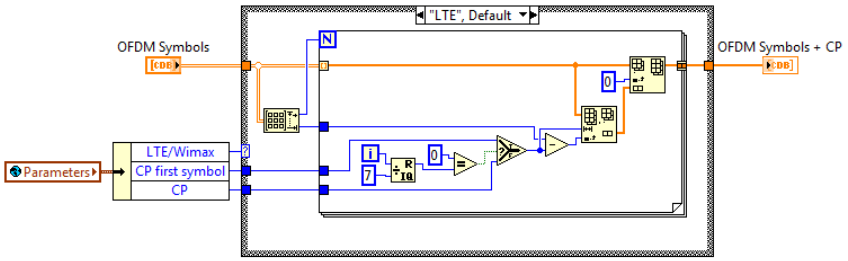


Figura 45. Diagrama de bloques de *Add_CP.vi* para el caso de *LTE-A*.

Para el caso de *IEEE 802.16m* la adición del prefijo cíclico se ha hecho en bloque (usando funciones de *arrays*), es decir se han cogido las últimas muestras de todos los símbolos *OFDM* y se han insertado al principio. Aprovechando que los símbolos están situados en un *array* de dos dimensiones (en el cual cada fila es un símbolo *OFDM*) se ha podido hacer fácilmente el proceso anterior utilizando la funciones *Array Subset* (para coger las últimas muestras de cada símbolo) e *Insert Into Array* (para insertarlas al principio del array). Finalmente se ha utilizado un *FOR LOOP* para tener secuencialmente todos los símbolos *OFDM* en un *array* de una sola fila. El proceso anterior lo podemos ver en la figura 46.

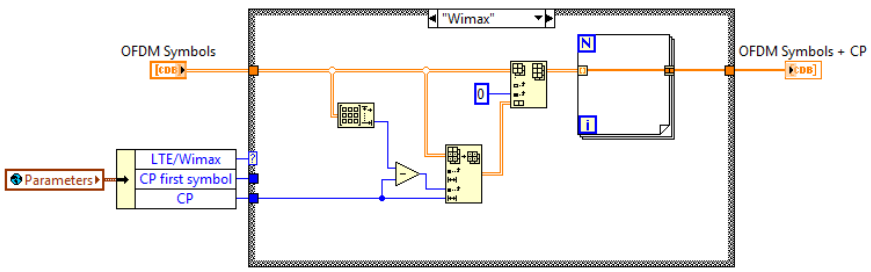


Figura 46. Diagrama de bloques de *Add_CP.vi* para el caso de *IEEE 802.16m*.

5.1.3 - Preámbulo de sincronización y secuencia de fin

Una vez tenemos los *símbolos OFDM* con sus correspondientes prefijos cíclicos, abandonamos la *frame 4* y vamos a la *frame 5*. En ésta, se va a realizar la inserción del preámbulo de sincronización y la secuencia de final de transmisión. Podemos ver una imagen del diagrama de bloques de la *frame 5* en la figura 47.

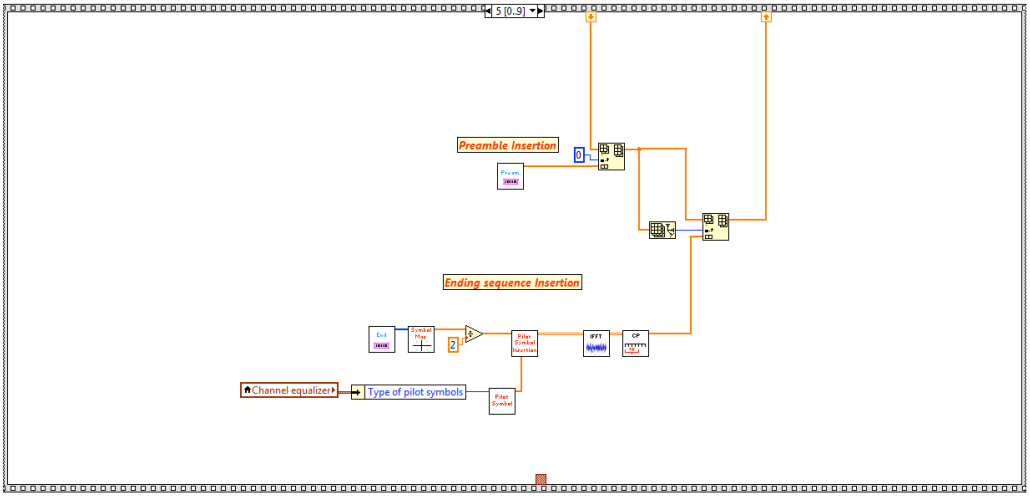


Figura 47. *Frame* 5 de *OFDM_modulator.vi.*

Preámbulo de sincronización:

En la figura 48 podemos ver el *subVI Preamble.vi* el cual tiene la función de generar el preámbulo de sincronización explicado en el capítulo 3. El mencionado preámbulo tendrá un tamaño de 512+64 muestras (ya que incluye prefijo cíclico). De las 512 subportadoras que utiliza el preámbulo, 214 son datos de una secuencia aleatoria, 214 son subportadoras de información cuyo valor se ha dejado a cero (para obtener en el dominio temporal dos mitades iguales del símbolo *OFDM*) y 84 son bandas de guarda. En la mencionada función se generará una secuencia aleatoria de 214 bits (que se distribuyen entre las subportadoras adecuadas), posteriormente se modulan los bits utilizando una constelación *BPSK*, se añaden las bandas de guarda, se aplica la *IFFT* inversa y se añade el prefijo cíclico. Una vez obtenido el preámbulo de sincronización, se añade al principio del *array* en el que se encuentran los símbolos *OFDM* modulados (como se puede ver en la figura 47).

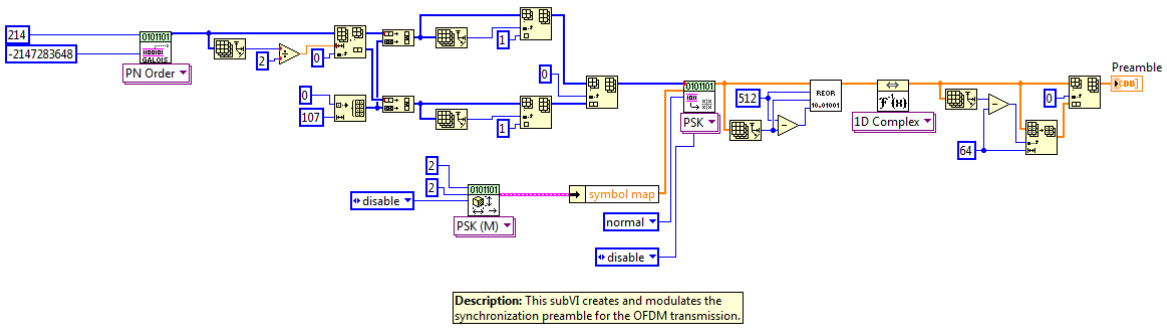


Figura 48. Diagrama de bloques de *Preamble.vi* .

Secuencia de fin:

Por último, la *subVI Ending_sequence.vi* (que podemos ver en la figura 49) se encarga de la generación de una secuencia aleatoria (que se añadirá al final del *array* que contiene los símbolos modulados *OFDM*) utilizando la función *MT Generate Bits* con orden del polinomio generador 5 y semilla de inicio igual a 1. La secuencia generada es conocida por el demodulador y sirve para señalizar el fin de la transmisión.

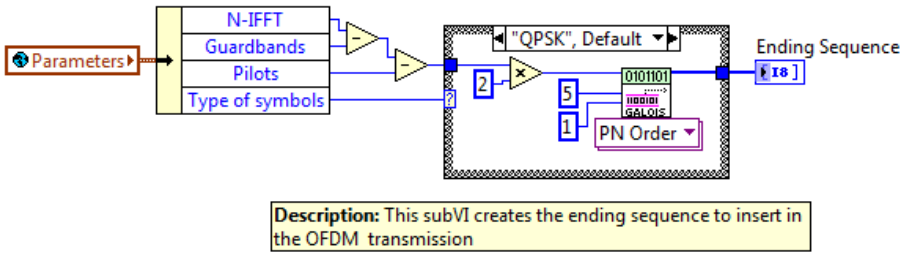


Figura 49. Diagrama de bloques de *Ending_sequence.vi* .

5.1.4 - Conexión TCP/IP

En el diseño del sistema *OFDM* se ha utilizado una conexión *TCP/IP* para que el PC1 que actúa como transmisor *OFDM* y el PC2 que actúa como receptor *OFDM* puedan intercambiar una serie de información necesaria para el correcto funcionamiento del sistema. A través de esta conexión se envía:

- El PC1 envía los valores que almacena la variable global *parameters* y que son esenciales para la correcta configuración del receptor.
- El PC1 envía el tipo de igualador de canal y sus parámetros que serán utilizados en la igualación de canal en el demodulador.
- El PC1 envía información necesaria para que el demodulador *OFDM* pueda configurar correctamente el *USRP 2920* que actúa como receptor.
- El PC1 envía los bits generados en el modulador *OFDM* para que en el demodulador *OFDM* se puedan comparar con los bits recibidos y calcular el *BER*.
- El PC2 envía la *SNR* estimada, así como la *BER* y el número de bits erróneos en la transmisión.

En esta sección se explicará el funcionamiento de las *subVI* relacionadas con la conexión *TCP/IP* que se encuentran en *OFDM_modulator.vi* en el PC1. En el siguiente capítulo se explicarán las *subVI* relacionadas con la conexión *TCP/IP* que se encuentran en *OFDM_demodulator.vi* en el PC2.

Para el correcto funcionamiento de la conexión *TCP/IP*, el PC2 tiene que empezar a escuchar el puerto sobre el que se haya decidido realizar la transmisión (como veremos en la explicación del demodulador *OFDM* más adelante). Seguidamente, el PC 1 inicia la conexión *TCP/IP* mediante el módulo *TCP Open Connection*, al cual se le tiene que indicar la dirección *IP* del PC2 y el puerto que se va a utilizar para la conexión (que debe coincidir con el puerto en el que estaba escuchando el PC2). Una vez establecida la conexión *TCP/IP* se pasa a realizar la transferencia de información. En la figura 50 (que es una parte del código que encontramos en la *frame 4*) podemos ver la función *TCP Open Connection* mencionada anteriormente y también la *subVI TCP_parameters_transmission.vi* cuyo *block diagram* podemos ver en la figura 51.

La *subVI TCP_parameters_transmission.vi* utiliza los distintos módulos de los que dispone *LabVIEW 2013* para el envío a través de conexiones *TCP/IP*. Primero, se utiliza la función *Type cast* para convertir el *cluster* que contiene la información a enviar a *string*, seguidamente se calcula el tamaño de la *string* a enviar con la función *String Length* y se convierte el tamaño

obtenido a *string*. Finalmente se envía primero la *string* correspondiente al tamaño y después la correspondiente al *cluster* mediante la función *TCP Write*. El proceso anterior se repite 3 veces, una para el envío del *cluster* con los parámetros de la modulación, otra para el envío del *cluster* con los parámetros del igualador de canal y otra para el envío del *cluster* con la configuración del *USRP 2920*.

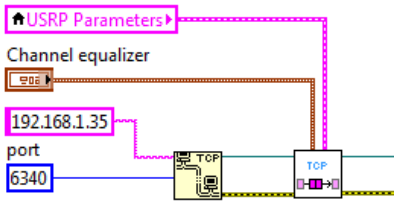


Figura 50. Parte del código que se encarga del establecimiento de la conexión y de la transmisión de parámetros.

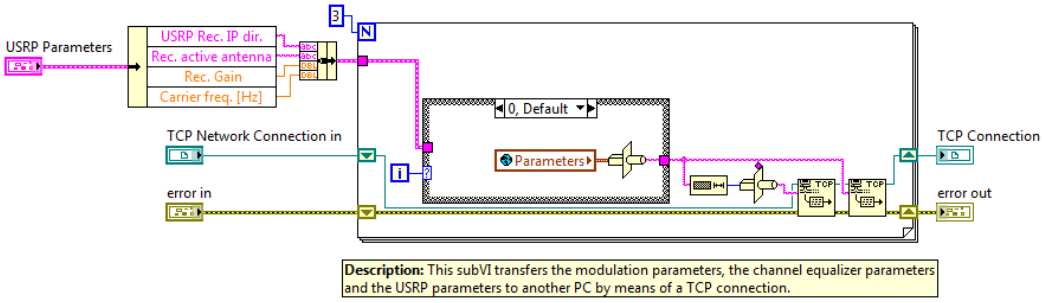


Figura 51. Diagrama de bloques de *TCP_parameters_transmission.vi.*

Para el envío de los bits generados, utilizamos la *subVI TCP_bits_transmission.vi* la cual se ejecuta dentro del *FOR LOOP* de la *frame* 4 que podemos ven en la figura 32. En cada ejecución de la *subVI* se realiza la transmisión de los bits de datos de una *radio-frame/superframe*. El diagrama de bloques del *subVI* lo podemos ver en la figura 52. Es bastante inmediato ver que esta *subVI* tiene el mismo funcionamiento que la anterior, con la única diferencia de que la información enviada es un *array* y no un *cluster* y por lo tanto la conversión mediante la función *Type cast* es de *array* a *string*. Esta *subVI* se ejecutará tantas veces como *radio-frame/superframe* tengamos.

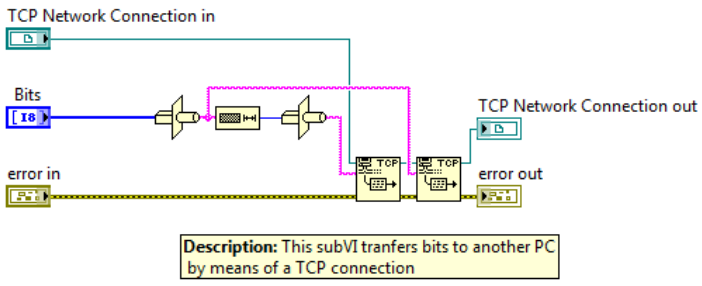


Figura 52. Diagrama de bloques de *TCP_bits_transmission.vi.*

Finalmente, tenemos la *subVI TCP_results_reception.vi* (situada en la *frame 8*) la cual se encarga de la recepción de los resultados obtenidos en el receptor *OFDM* una vez finalizada la recepción *OFDM*. Podemos ver el diagrama de bloques de la función anterior en la figura 53. En ella utilizamos la función *TCP Read*, que se encarga de esperar a que se reciban los 4 bytes que indican el tamaño de la siguiente *string* que se va a recibir, en este caso se espera un tiempo de 900000 ms, tiempo suficiente para que haya terminado la transmisión y recepción *OFDM*. Una vez recibido el tamaño de la *string* se procede a la lectura de la *string* mediante la función

TCP read y se convierte a *cluster* mediante la función *Type cast* y una constante que define la estructura del *cluster*. El *cluster* recibido con esta *subVI* tiene información como la *SNR* estimada o la *BER* calculada por el receptor *OFDM*.

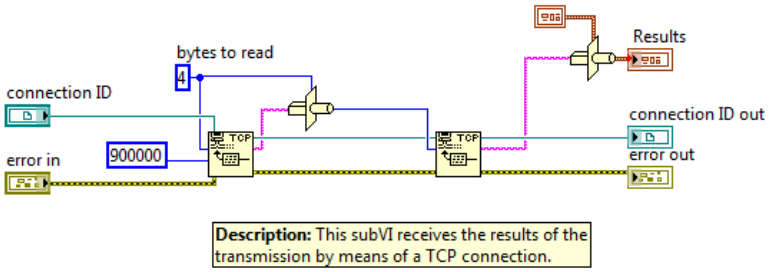


Figura 53. Diagrama de bloques de *TCP_results_reception.vi.*

5.1.5 - Configuración del USRP 2920 en transmisión

En la *frame 6* de *OFDM_modulator.vi* encontramos la *subVI* en la que se realiza la configuración del *USRP 2920* en transmisión. Dicha *subVI* es *USRP_Tx_configuration.vi* y su diagrama de bloques se muestra en la figura 54. En ella se hace uso de un *property node* para la configuración del número de bits con el que se transmiten los símbolos a través del cable *Ethernet* (en nuestro caso hemos elegido el máximo, 16 bits). También se hace uso de la función *Configure Signal*, con la que se especifica la dirección *IP* del dispositivo *USRP 2920*, la frecuencia de operación, la ganancia y la antena del *USRP* que se va a utilizar. Los anteriores parámetros son introducidos por el usuario mediante los controles pertenecientes al panel de control de *OFDM_modulator .vi* que podemos ver en la figura 33. La función *USRP_Tx_configuration.vi* devuelve el *output Coarced Sample Rate*, que es la frecuencia de muestreo más cercana a la introducida por el usuario (de entre todas a las que puede operar el *USRP*).

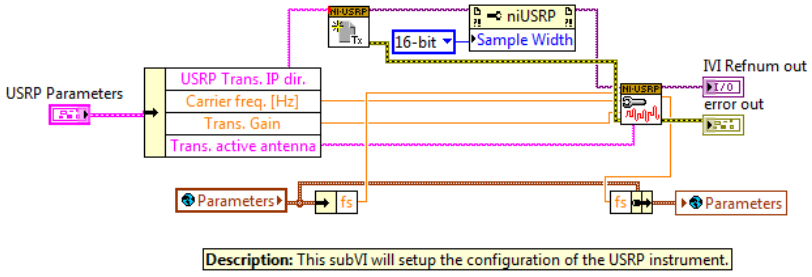


Figura 54. Diagrama de bloques de *USRP_Tx_configuration.vi.*

5.1.6 - Envío de los símbolos al USRP

En la *frame 7* se realiza el envío de los símbolos *OFDM* modulados al *USRP 2920*. Para ello, se utiliza el módulo *niUSRP Write Tx Data* el cual tiene como entrada un *array* con todos los símbolos *OFDM* que se quieren enviar en la transmisión. Una vez se han enviado los símbolos al *USRP* y éstos han sido transmitidos, utilizamos el módulo *niUSRP Close Session* para finalizar la conexión con el *USRP*. El proceso anterior lo podemos ver en la figura 55.

5.1.7 - Cálculo del *Throughtput*

Para el cálculo del *Throughput* tenemos que medir el tiempo que se invierte en la transmisión de los símbolos, por ello utilizamos la función *Get Date/Time In Seconds* que nos proporciona el instante temporal en el que se ha ejecutado la función en segundos (siendo el origen de tiempos el 1 de enero de 1904 a las 12:00 am). Así, ejecutamos la función antes y después de la

transmisión y restando ambos resultados obtenemos el tiempo de transmisión en segundos. Una vez obtenido el tiempo de transmisión, y sabiendo el número de bits de información transmitidos, podemos obtener el *Throughput* dividiendo ambos. El proceso de cálculo del *Throughput* lo podemos ver en la figura 55.

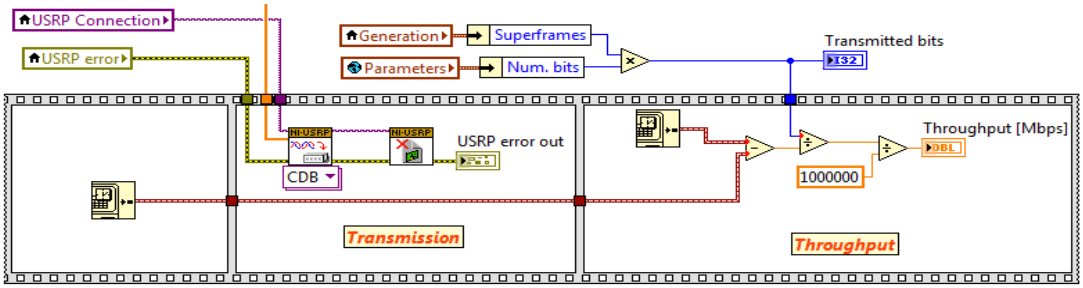


Figura 55. ***Frame 7*** **de** ***OFDM_modulator.vi*****.**

5.2 - Demodulador OFDM

En esta sección se va a presentar el diseño del demodulador *OFDM*, que ha sido implementado en el proyecto *OFDM_demodulador.lvproj*.

Al igual que el modulador *OFDM*, el demodulador esta estructurado en una *VI* principal llamada *OFDM_demodulator.vi* y un conjunto de *subVI* que realizan tareas específicas. Asimismo, se utilizan las variables globales *parameters* y *channel_equalizer* para tener disponibles los parámetros de demodulación y de igualación de canal en cualquier *subVI* que los requiera.

En *OFDM_demodulator.vi* también se ha utilizado una *Stacked Sequence Structure* para el control del orden de ejecución de las diversas tareas, teniendo dicha estructura 4 *frames*. En el *frame 0* se realiza la recepción (a través de una conexión *TCP/IP*) de los parámetros necesarios para la modulación, de los bits enviados y la configuración del *USRP 2920*, en el *frame 1* se implementa el proceso de sincronización temporal del sistema *OFDM*, en el *frame 2* se lleva a cabo el proceso de demodulación *OFDM* y en el *frame 3* se realizan una serie de cálculos para obtener la *SNR* y la *BER*.

5.2.1 - Interfaz gráfica del demodulador OFDM

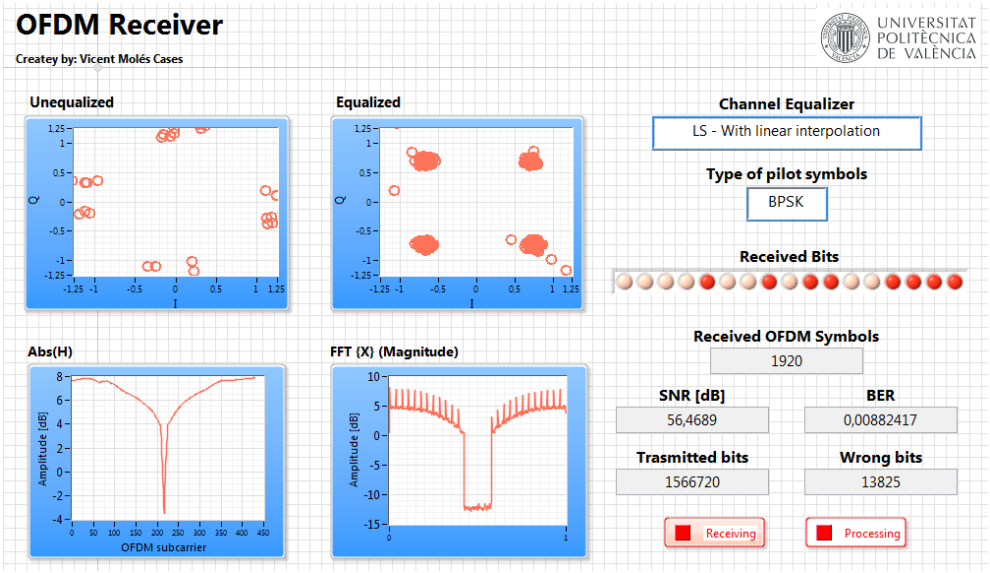


Figura 56. Panel frontal de OFDM_demodulator.vi

En la figura 56 podemos observar el panel de control de *OFDM_demodulator.vi*. En el vemos una gráfica correspondiente con los símbolos recibidos sin ecualizar, una gráfica con los símbolos ecualizados, otra con el espectro en banda base y finalmente una gráfica con la respuesta del canal estimada en las subportadoras que transportan información. Asimismo, hay una serie de indicadores con información relevante como el tipo de igualador de canal utilizado, el tipo de símbolos piloto utilizados, la *SNR* estimada, el número de símbolos *OFDM* recibidos, el número de bits de datos recibidos, el número de bits erróneos y la *BER*.

5.2.2 - Conexión TCP/IP

En el diseño en *LabVIEW* del demodulador *OFDM* se ha hecho uso de una conexión *TCP/IP* para que éste obtenga los parámetros introducidos por el usuario en el modulador así como los bits que posteriormente serán enviados por la interfaz radio, como hemos visto en el punto 5.1.4. Como vimos en el citado punto, el PC1 es el que ejecuta la *VI OFDM_modulator.vi* y el PC2 es el que ejecuta la *VI OFDM_demodulador.vi.*

En *OFDM_demodulador.vi*, la recepción de los parámetros y de los bits se realiza mediante la *subVI TCP_reception.vi* perteneciente a la *frame 0*. En ella, se utiliza la función *TCP Listen* para empezar a escuchar en el puerto que le es indicado hasta que se recibe un establecimiento de conexión *TCP/IP*. Una vez establecida la conexión, la función *TCP Read* se encarga de esperar a que el PC1 envíe los parámetros de la modulación y los bits de referencia, primero leyendo 4 bytes que contienen el tamaño de la información enviada, y luego leyendo la información enviada. En las figuras 57 y 58 podemos ver el diagrama de bloques de la función *TCP_reception.vi* para el caso de la recepción de los parámetros y de los bits de referencia respectivamente, la única diferencia entre ambos casos es el uso que se hace de la función *Type cast* (en el caso de los parámetros se hace la conversión de *string* a *cluster* y en la de los bits de *string* a *array*).

En *OFDM_demodulador.vi* también se utiliza la conexión *TCP/IP* para enviar los resultados obtenidos al PC1. Así, en la *frame 4* encontramos la *subVI TCP_results_transmission.vi*, la cual tiene un funcionamiento casi idéntico a *TCP_parameters_transmission.vi* explicada anteriormente en el punto 5.1.3.

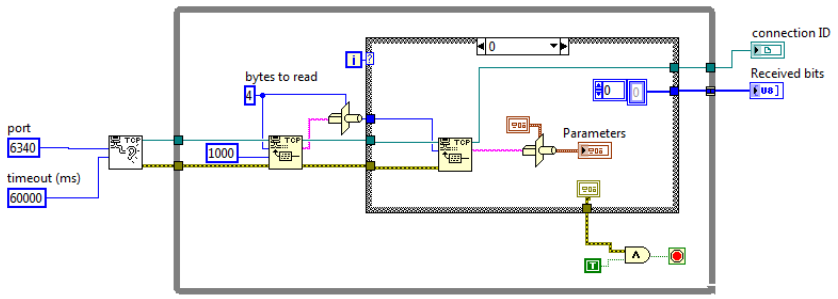


Figura 57. Diagrama de bloques *TCP_reception.vi para el caso de la recepción de los parámetros.*

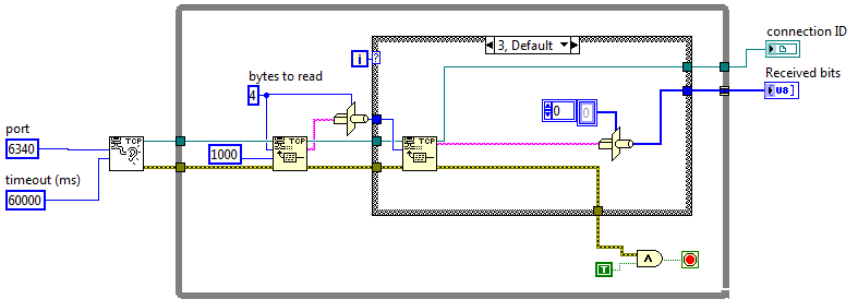


Figura 57. Diagrama de bloques *TCP_reception.vi para el caso de la recepción de los bits*

5.2.3 - Configuración del USRP 2920 en recepción

En la figura 58 podemos ver *la frame 0* del diagrama de bloques de *OFDM_demodulador.vi*. En ella podemos ver la *subVI USRP_Rx_configuration.vi*, función encargada de la configuración del *USRP 2920* en recepción y que utiliza la información recibida mediante la conexión *TCP/IP* para la configuración del *USRP*. Su funcionamiento es idéntico al de *USRP_Tx_configuration.vi* con excepción de que se añade el módulo *niUSRP Initiate* (para indicarle al *USRP* que empiece a recibir muestras), si el lector quiere una explicación más detallada de la *subVI*, se puede dirigir al punto 5.1.5.

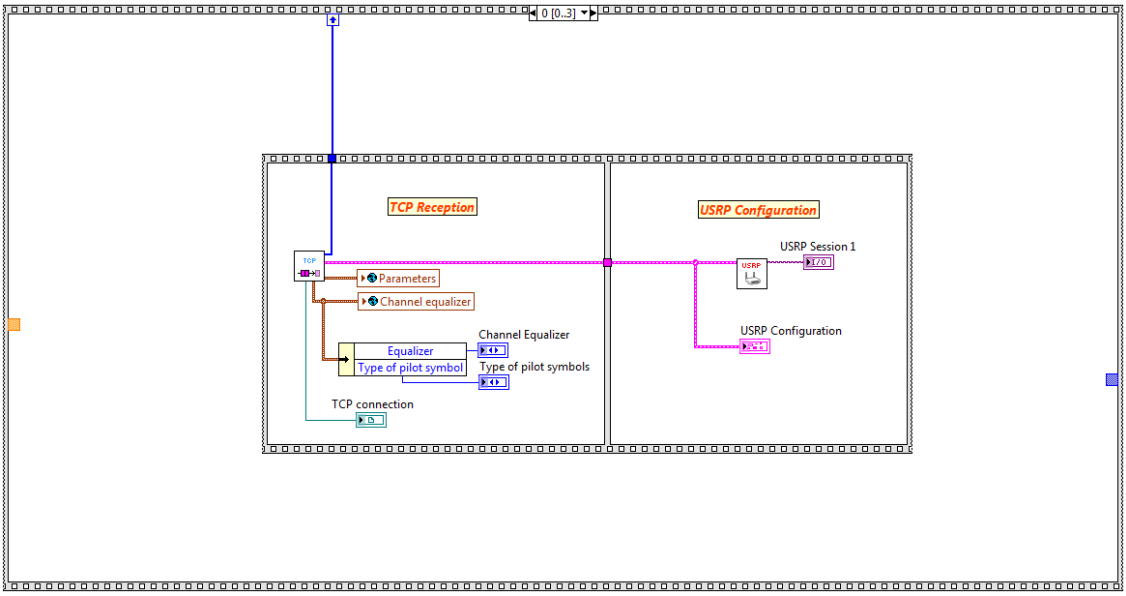


Figura 58. *Frame 0* del diagrama de bloques de *OFDM_demodulador.vi.*

5.2.4 - Sincronización temporal

Antes de explicar el proceso de sincronización, se tiene que entender el modo en que se adquieren las muestras en el *USRP* y cómo las leemos con *LabVIEW*. Así, una vez realizada la configuración en recepción del *USRP*, se ejecuta el módulo *niUSRP Initiate*. A partir de este momento el *USRP* empieza a obtener muestras a una tasa f_s muestras/segundo, y las envía al PC a través del cable Ethernet. En el PC, las muestras son almacenadas en un *buffer* creado por *LabVIEW* en la memoria virtual utilizada por el programa. La *VI* encargada de la lectura de las muestras en *LabVIEW*, utiliza el módulo *niUSRP Fetch Rx Data*, al que se le indica el número de muestras que se quieren leer del *buffer*.

Como hemos visto en el capítulo 2, la sincronización temporal es uno de los puntos más importantes en *OFDM*. En la figura 59 podemos ver el diagrama de bloques de la *subVI synchronization.vi.*, en él se utiliza la función *niUSRP Fetch Rx Data* dentro de un *WHILE LOOP* para la lectura de las muestras del *buffer* hasta que se detecte el preámbulo de sincronización. Las lecturas serán de 576 muestras, ya que el preámbulo de sincronización tiene un tamaño fijo de 512 muestras más un prefijo cíclico de 64 muestras. Una vez leídas las muestras se aplica la *subVI Noise_detection.vi*, que se encarga de calcular el módulo de las muestras recibidas y aplicar un umbral (establecido en 0,01) para determinar si la lectura corresponde con ruido o con señal útil y de esta forma no realizar procesado innecesario en lecturas que corresponden con ruido. Una vez determinado que la lectura actual corresponde con señal útil se realiza la correlación cruzada entre la secuencia de muestras leídas y el preámbulo de sincronización (que el receptor conoce de antemano), y se aplica un umbral al resultado de la correlación cruzada, si algún elemento del *array* resultante de la correlación cruzada supera el umbral podemos decir que en la lectura actual se encuentra el preámbulo de sincronización, y siendo l el índice del array de correlación cruzada donde se ha superado el umbral, podemos

decir que el inicio del preámbulo de sincronización se encuentra en el índice 576 - l de la lectura actual de muestras. Cabe destacar que el umbral que determina si se encuentra el preámbulo de sincronización en la lectura se ha determinado de forma experimental, y que tras un gran número de pruebas se determinó que el umbral adecuado tenía un valor de 1.

Llegados a este punto, tenemos que diferenciar entre 3 casos:

- $l < 576$: en este caso, en la lectura actual se encuentra el inicio del preámbulo de sincronización, pero no el preámbulo entero. Por lo que para tener completamente sincronizado el receptor, tenemos que realizar una lectura de $576 - l$ muestras, para que la próxima vez que se aplique la función *niUSRP Fetch Rx Data* la primera muestra leída corresponda con la primera muestra del primer símbolo *OFDM* transmitido.
- $l = 576$: en este caso el receptor ya está sincronizado y se puede pasar a la lectura de los símbolos *OFDM*.
- $l > 576$: en este caso, en la lectura actual se encuentra el final del preámbulo de sincronización pero no el inicio, y por lo tanto en el final de la lectura también se encontrarán las primeras muestras del primer símbolo *OFDM*. Por ello, en este caso tenemos que realizar una lectura adicional de $N_{IFFT} + CP - 2 \cdot 576 + l$, con la que obtenemos el final del primer símbolo *OFDM*, y que junto a las $2 \cdot 576 - l$ muestras de la primera lectura nos da el primer símbolo *OFDM* y hace que el receptor este completamente sincronizado temporalmente.

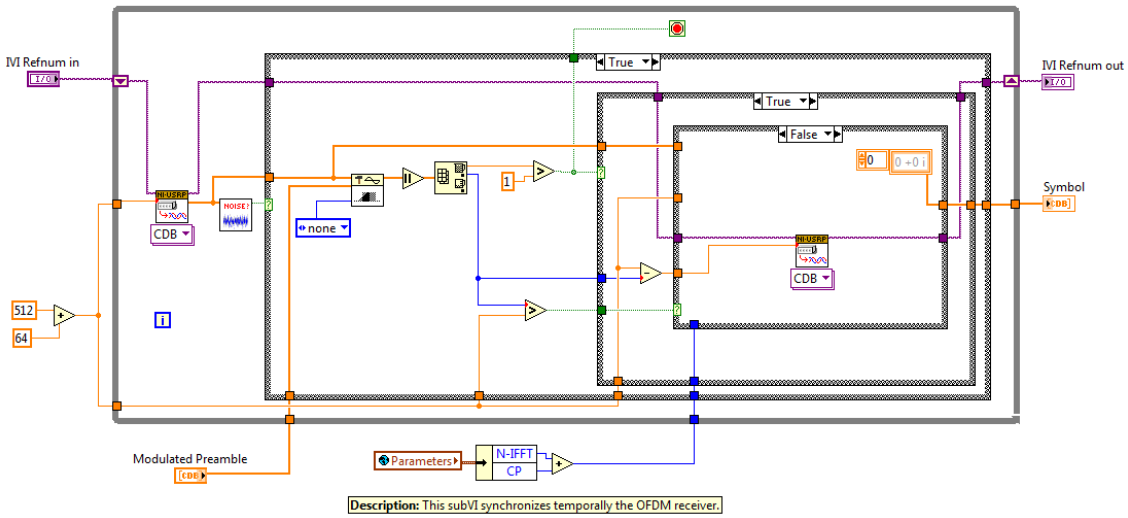


Figura 59. Diagrama de bloques de *Synchronization.vi*.

5.2.5 - Demodulación OFDM

Una vez realizada la sincronización temporal, el siguiente paso es la demodulación *OFDM* de las muestras recibidas, la cual se realiza en la *frame 2* de *OFDM_demodulator.vi* y cuyo diagrama de bloques podemos ver en la figura 60.

Primero vamos a realizar la lectura de muestras del buffer mediante la función *niUSRP Fetch Rx Data*. En cada iteración de lectura vamos a leer tantas muestras como tiene un símbolo *OFDM* transmitido (N-IFFT+CP), obteniendo en cada iteración un símbolo *OFDM* y pudiendo realizar la demodulación símbolo a símbolo. Para el caso de *LTE-A*, tenemos que tener en cuenta en que símbolo *OFDM* nos encontramos dentro de un *slot* para elegir correctamente la longitud del prefijo cíclico, por ello se ha utilizado una *CASE STRUCTURE* para la elección del tamaño de prefijo cíclico dentro del *WHILE LOOP* de lectura.

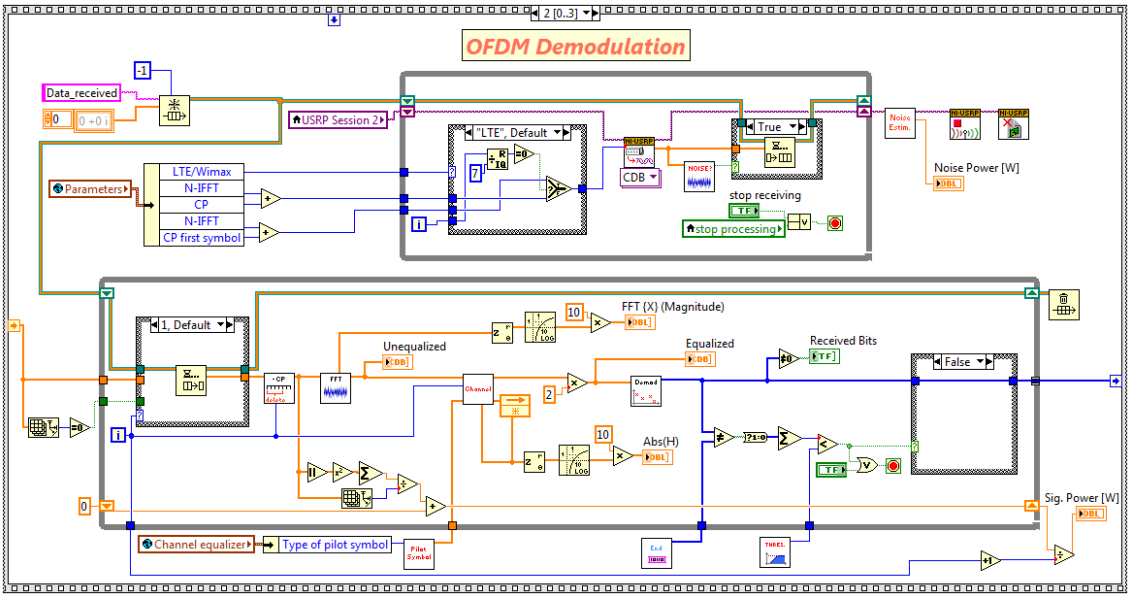


Figura 60. Diagrama de bloques de la *frame* 2 de *OFDM_demodulator.vi*

Debido a que el proceso de demodulación es mucho más costoso computacionalmente que el proceso de lectura de las muestras, se ha decidido separar la lectura y demodulación de las muestras en 2 bucles separados y utilizar una *queue* llamada *Data_received* para la transferencia de las muestras entre ambos bucles. De esta forma evitamos el desbordamiento del *buffer* que almacena las muestras obtenidas por el *USRP* y el correcto funcionamiento del sistema (ya que gracias a la *queue* el bucle de lectura no tiene que esperar a que se realice la demodulación de los símbolos y va leyendo de una forma continua sin que se desborde el *buffer*). La *queue* utilizada tiene un tamaño ilimitado y los elementos que se introducen en ella deben ser *arrays* del tipo *CDB* (*Complex double-precision, floating-point,* mismo tipo de dato que las muestras que obtenemos con la función *niUSRP Fetch_Rx_Data* al leer las muestras del buffer). Para la introducción de las muestras en la cola se ha utilizado la función *enqueue element*.

De forma simultánea a la lectura de muestras del *buffer*, se va realizando la demodulación de los símbolos que se encuentran en la *queue Data_receive.* El bucle que se encarga de la demodulación, lee un símbolo *OFDM* de *Data_received* en cada iteración mediante la función *Dequeue Element* (se tiene que matizar este comportamiento, ya que si en el proceso de sincronización se ha dado el caso en el que se obtiene el primer símbolo *OFDM*, en la primera iteración del bucle de demodulación no se lee de la *queue* y se demodula el símbolo obtenido en el proceso de sincronización).

Una vez hemos obtenido un símbolo *OFDM* se va a proceder a la demodulación del mismo, para ello se van a seguir 4 pasos explicados seguidamente.

1 - Eliminación del prefijo cíclico:

El primer paso en la demodulación *OFDM* es la eliminación del prefijo cíclico. Como hemos visto a lo largo de todo este proyecto, el prefijo cíclico es uno de los puntos donde más se diferencian los dos estándares implementados. Por ello, en la *subVI CP_removing.vi* (encargada de eliminar el prefijo cíclico de los símbolos recibidos) se ha utilizado una *CASE STRUCUTURE* para diferenciar entre *LTE-A* y *IEEE 802.16m*.

En la figura 62 vemos el diagrama de bloques de *CP_removing.vi* para el caso *de IEEE 802.16m*. La eliminación del prefijo cíclico en este caso se hace utilizando la función *Subset Array* que sustrae los *N-IFFT* últimos elementos del *array* que contiene el símbolo *OFDM*.

Para el caso de *LTE-A*, el funcionamiento de *CP_removing.vi* es similar, con la particularidad de que el tamaño del prefijo cíclico eliminado va variando dependiendo de la posición del símbolo *OFDM* dentro del slot temporal, como podemos ver en la figura 53.

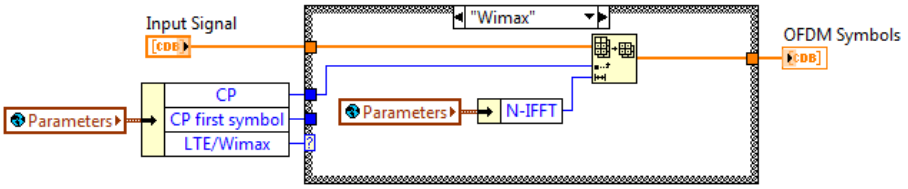


Figura 61. Diagrama de bloques de *CP_removing.vi para el caso de IEEE 802.16m.*

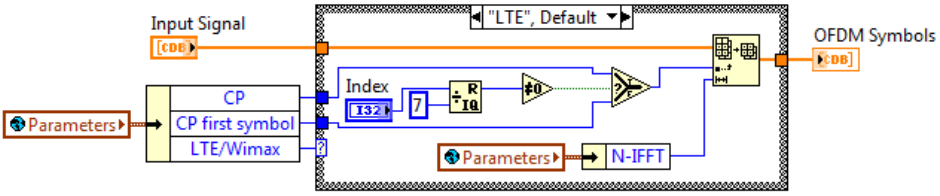


Figura 62. Diagrama de bloques de *CP_removing.vi para el caso de LTE-A.*

2 - FFT:

La *subVI FFT.vi*, cuyo diagrama de bloques podemos ver en la figura 63, se encarga de realizar la *FFT* de *N-IFFT* puntos del símbolo *OFDM* (o lo que es lo mismo, la transformación del dominio temporal al dominio frecuencial). Una vez realizada la *FFT*, se utiliza la *subVI FFT_symbol_structure* para extraer la información de las portadoras de datos (las cuales podemos ver en la figura 44 del punto 5.1.2). El diagrama de bloques de dicha función lo podemos ver en la figura 64.

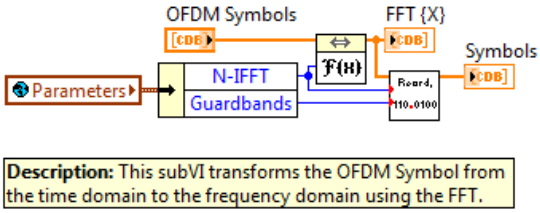


Figura 63. Diagrama de bloques de *FFT.vi* .

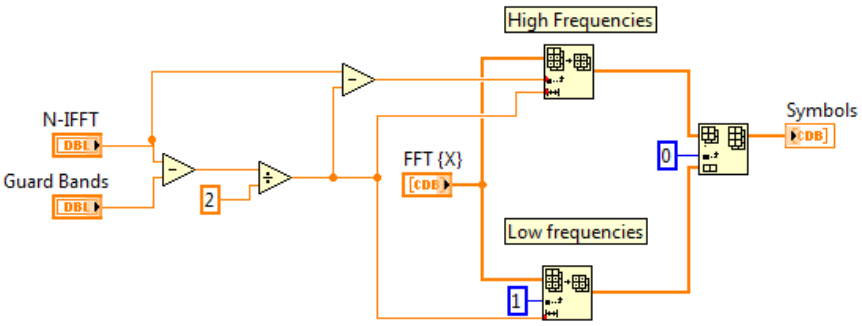


Figura 64. Diagrama de bloques de *FFT_symbol_structure.vi.*

3 - Igualación de canal:

La igualación de canal se realiza mediante la *subVI channel_equalizer.vi*, en las figuras 65 y 66 podemos ver los diagramas de bloques para *IEEE 802.16m* y *LTE-A*, respectivamente. El proceso de igualación de canal en ambos estándares es muy similar, variando tan solo la distribución de las subportadoras piloto.

La igualación de canal empieza con la extracción de la señal recibida en las subportadoras piloto ($Y_p[m]$) utilizando la función *Decimate 1D Array*. Una vez obtenida la señal recibida en las subportadoras piloto, se pasa a la ejecución de las funciones de interpolado (que se explicarán posteriormente) en ella, se divide la señal obtenida entre la señal enviada en las subportadoras piloto ($X_p[m]$) y que es conocida a priori por el receptor (ya que utiliza la misma función *Pilot_symbols_generation.vi* que el modulador *OFDM*), obteniendo el canal en las subportadoras piloto ($\widehat{H}_p[m]$) (siendo el proceso anterior la estimación *LS*). A continuación se realiza la interpolación del canal obtenido en las subportadoras piloto para obtener el canal estimado en todas las portadoras ($\widehat{H}[k]$).

Una vez se tiene la estimación de canal en todas las subportadoras, se vuelve a aplicar el estimador *LS*, el cual realiza la división entre la señal recibida ($Y[k]$) y el canal estimado ($\widehat{H}[k]$) para obtener la estimación de los símbolos enviados ($\hat{X}[k]$). Por último se utiliza otra vez la función *Decimate 1D Array* para quedarnos con los símbolos estimados en las portadoras de datos. La extracción de pilotos se realiza siguiendo la distribución de las figuras 20 y 26 del capítulo 3, y para el caso de *LTE-A* en aquellos símbolos *OFDM* en los que no se inserten símbolos piloto se utiliza el canal estimado en el símbolo *OFDM* anterior.

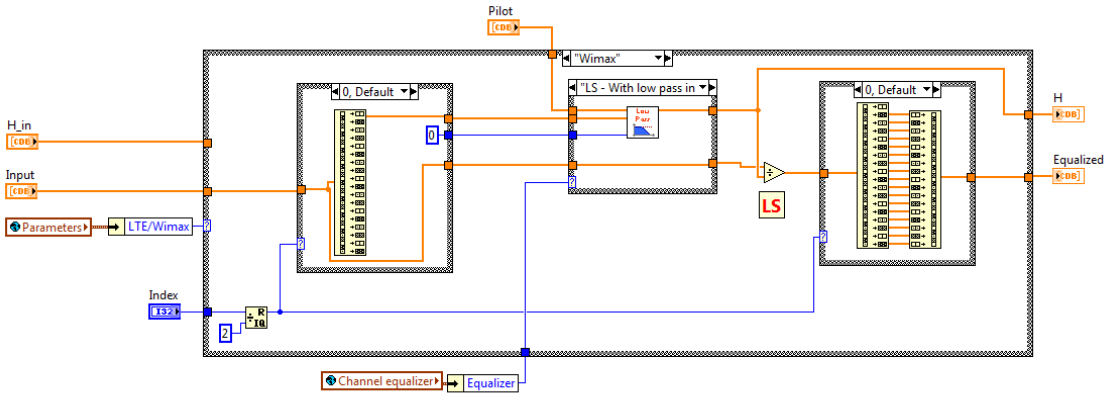


Figura 65. Diagrama de bloques de channel_equalizer.vi para el caso de IEEE 802.16m.

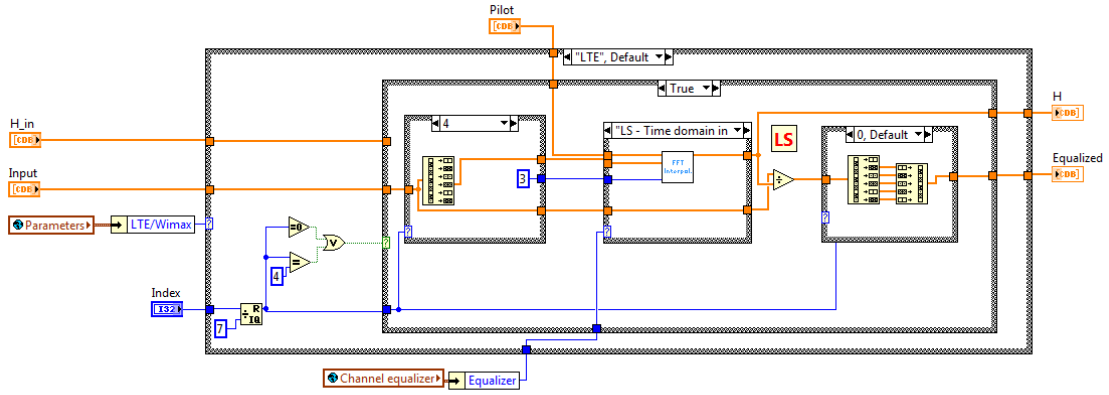


Figura 66. Diagrama de bloques de channel_equalizer.vi para el caso de LTE-A.

A continuación se presentan las *subVI* que implementan los distintos métodos de interpolación del canal y que corresponden con los introducidos en la sección 2.3 (se tiene que tener en cuenta que la interpolación de la parte real e imaginaria se hace por separado):

- *Uniform_interpolation.vi*, el cual realiza la interpolación al valor más cercano. Para ello se ha utilizado la función *Interpolate 1D* con el método *nearest* como se puede ver en la figura 67. A la anterior función le pasamos como entradas el valor del canal en las subportadoras piloto, un *array* con los índices de las posiciones de las subportadoras piloto y un *array* con los índices de todas las subportadoras en las cuales queremos obtener un valor interpolado (los cuales obtenemos con la función *Ramp Pattern*).

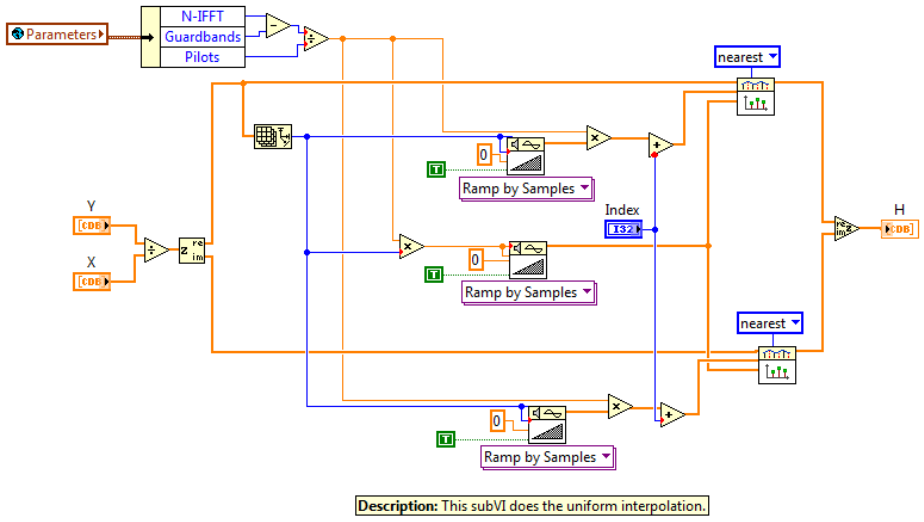


Figura 67. Diagrama de bloques de *Uniform_interpolation.vi* .

- *Linear_interpolation.vi* realiza la interpolación lineal del canal. Su diagrama de bloques lo podemos ver en la figura 68, y tiene un funcionamiento idéntico a *Uniform_interpolation.vi* con la excepción de que el método de interpolación seleccionado es *linear*.

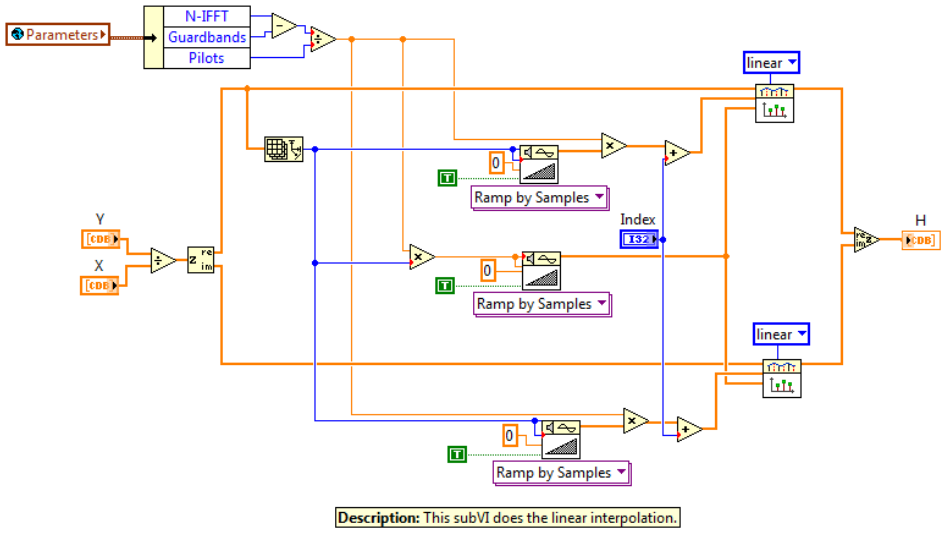


Figura 68. Diagrama de bloques de *Linear_interpolation.vi* .

- *Time_domain_interpolation.vi*, cuyo diagrama de bloques tenemos en la figura 69. En este tipo de interpolación se realiza la *IFFT* de $N_{pilotos}$ puntos del canal en las subportadoras pilotos, seguidamente se aplica la función *IFFT_symbol_structure.vi* al resultado obtenido, y finalmente realizamos la *FFT* de $N_{Útil}$ puntos obteniendo como resultado el canal interpolado. Para el caso en el que el primer símbolo piloto no se encuentra en la primera subportadora frecuencial, realizamos un desplazamiento y a las primeras subportadoras les asignamos el valor de la primera subportadora piloto (como se puede ver en el interior de la *CASE STRUCTURE* de la figura 66).

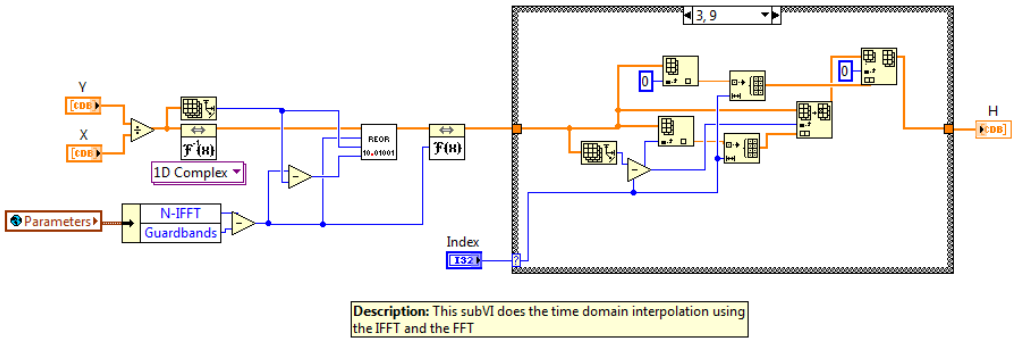


Figura 69. Diagrama de bloques de *Time_domain_interpolation.vi* .

- *Spline_cubic_interpolation.vi* se encarga de la interpolación mediante el método *Spline Cubic* y su implementación se basa en el mismo proceso que *Linear_interpolation.vi* o *Uniform_interpolation.vi,* pero utilizando la función *Spline Interpolation 1D* en lugar de *Interpolate 1D,* como se puede ver en la figura 70.

- *Low_pass_filter_interpolation.vi* utiliza una interpolación con 0's y un filtro *FIR* paso bajo para la interpolación del canal. Para realizar esta interpolación, se ha utilizado un filtro *Multirate* de Nyquist (un filtro *Multirate* se encarga de la conversión de la frecuencia de muestreo, ya sea diezmado o interpolado). El mencionado filtro es *FIR* paso bajo con una *sinc* en el dominio temporal, y que tiene 0's en aquellas muestras múltiples de *M* (factor de interpolación) excepto en la central. El factor de interpolación es igual a la separación entre subportadoras piloto, y por lo tanto para el caso de *IEEE 802.16m* el factor será de *18*, mientras que para *LTE-A* el factor será de 6. Se ha elegido un filtro con un *roll off* (pendiente) en la banda de paso de 0,1 y 60 dB de atenuación en la banda eliminada. Ha sido así, ya que si disminuye el roll off o se aumenta la atenuación en la banda eliminada aumenta el orden del filtro, y por lo tanto aumenta el coste computacional a la hora de su aplicación. Para la implementación del filtro se ha utilizado la función *DFD Nyquist Design*, que tiene como entradas los parámetros explicados anteriormente. Una vez implementado el filtro, se utiliza la función *DFD MRate Filtering* para realizar el filtrado de la señal de entrada mediante el filtro previamente diseñado, obteniendo finalmente el canal interpolado. El proceso explicado anteriormente se puede ver en la figura 71.

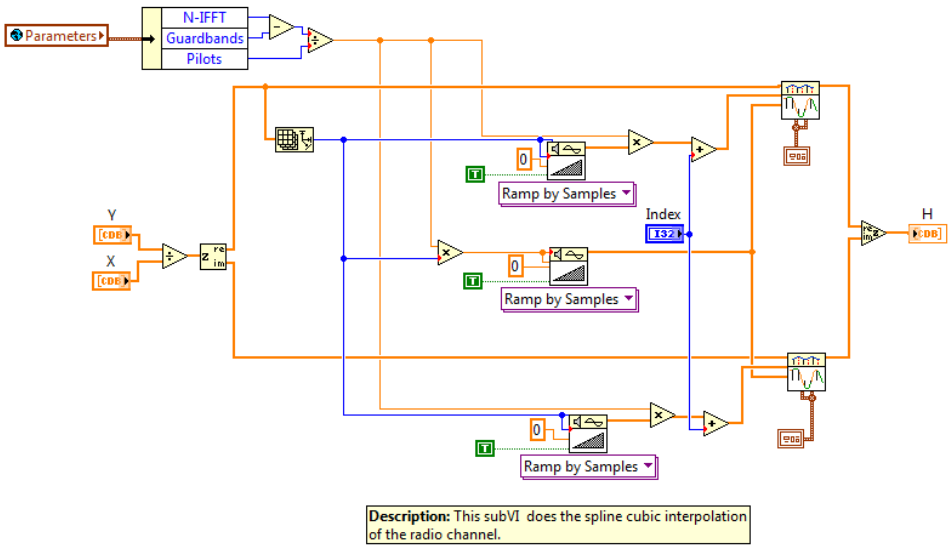


Figura 70. Diagrama de bloques de *Spline_cubic _interpolation.vi*.

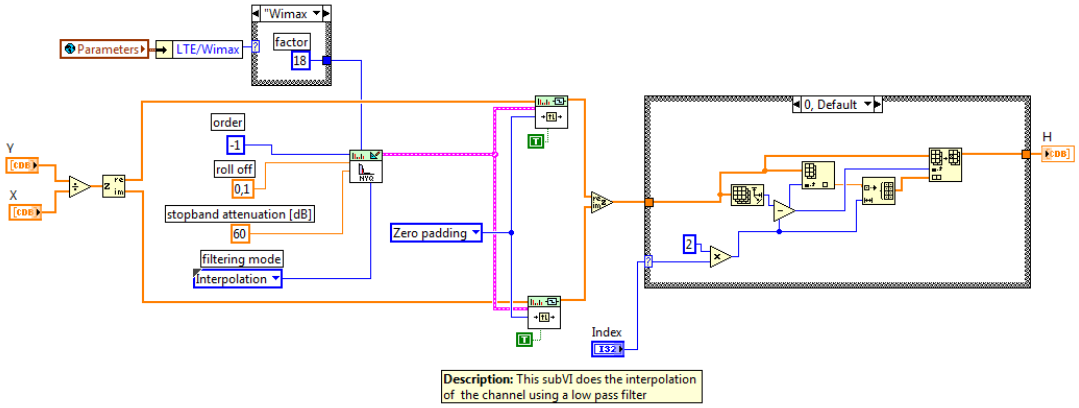


Figura 71. Diagrama de bloques de *Low_pass_filter _interpolation.vi*.

4 - Demapeado de los símbolos:

Una vez realizada la igualación de canal, se procederá a escalar los símbolos obtenidos (se multiplicará por 2 su módulo, ya que en el transmisor su módulo había sido dividido por 2) y posteriormente se aplica la subVI *Demap_symbols.vi* la cual se puede observar en la figura 72. Para realizar el demapeado se han utilizado las funciones *MT Generate System Parameters* y *MT Demodulate PSK* o *MT Demodulate QAM* (según convenga) para realizar el demapeado. Con la primera función se obtiene la constelación de símbolos que servirá para el demapeado y con la segunda se realiza el demapeado de los símbolos a bits.

Por último, en cada iteración se comparan los bits demodulados con la secuencia de fin (creada con la *subVI Ending_sequence.vi* al igual que en el modulador) y se aplica un umbral que determina si se trata o no de la secuencia de fin, y de ser así, se para el proceso de

demodulación. Los umbrales para cada caso han sido determinados de forma experimental buscando aquéllos que daban un mejor de resultado.

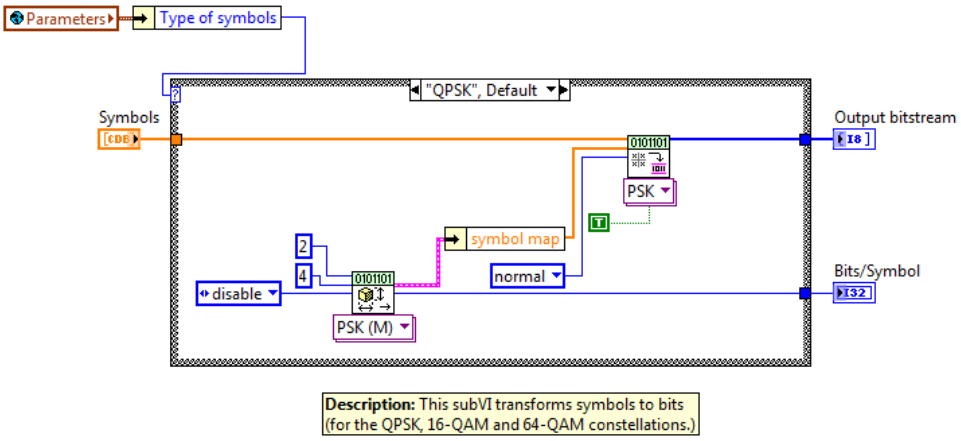


Figura 72. Diagrama de bloques de *Demap_symbols.vi*

5.2.6 - Estimación de la SNR

Para la estimación de la SNR, se ha de estimar el ruido existente en el canal y calcular la potencia con que son recibidos los símbolos *OFDM*. Para la estimación del ruido en el canal se ha utilizado la *subVI Noise_estimation.vi*, cuyo diagrama de bloques aparece en la figura 73. Ésta se ejecuta una vez han sido recibidos todos los símbolos *OFDM*, y consiste en una lectura de muestras en las que sólo haya presente ruido en el canal (la lectura consiste en la obtención del número de muestras que equivalen a 20 símbolos *OFDM*). Una vez obtenidas las muestras de ruido, se hace un promediado de todas las recibidas en cada una de las subportadoras, y se calcula el módulo obteniendo la potencia de ruido presente en el canal. Para el cálculo de la potencia media de los símbolos recibidos se ha promediado el módulo de la potencia obtenida en cada subportadora y en cada símbolo *OFDM* recibido. Finalmente, en la *frame 3* de *OFDM_demodulator.vi*, se ha realizado la división de la potencia media entre el ruido estimado y se ha pasado dicho valor a dB, obteniendo la estimación de la *SNR* en la transmisión (figura 74).

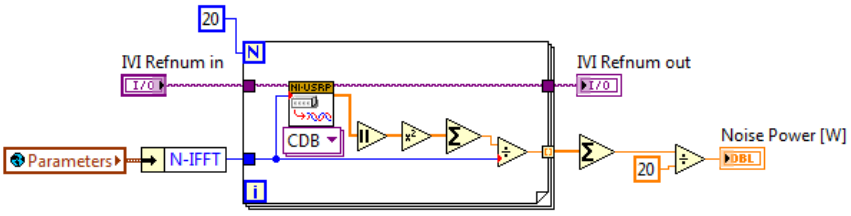


Figura 73. Diagrama de bloques de Noise_estimation.vi.

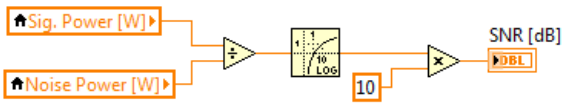


Figura 74. Cálculo de la *SNR* en dB.

5.2.7 - Cálculo del *Bit Error Rate*

El cálculo de la *BER* se ha realizado en la *frame 3* de *OFDM_demodulator.vi;* para dicho cálculo se han comparado los bits recibidos con el demodulador *OFDM* con los recibidos

mediante la conexión *TCP/IP*, obteniendo el número de bits erróneos que ha habido en la recepción *OFDM*. Finalmente se obtiene la *BER* dividiendo el número de bits erróneos entre el total de *bits* transmitidos.

En la figura 75 se puede ver el *frame 3* de *OFDM_demodulator.vi*, donde se detalla tanto el proceso de cálculo de la *BER* como la *SNR,* así como la función *TCP_results_transmission.vi* que se encarga de enviar los resultados obtenidos al PC1.

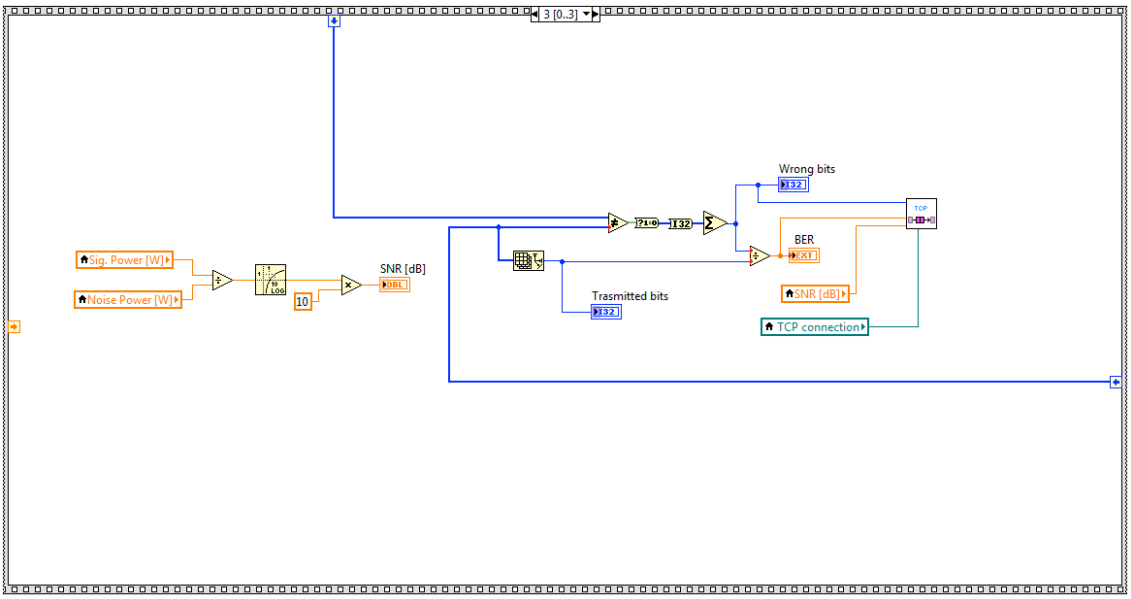


Figura 75. ***Frame 3*** **de** ***OFDM_demodulator.vi***

Capítulo 6 - Evaluación de prestaciones

En esta sección se van a presentar los resultados logrados tras la evaluación del sistema de comunicaciones *OFDM* diseñado. El escenario utilizado elegido ha sido el laboratorio del *GTAC (iTeam)* perteneciente a la *Ciutat Politècnica de la innovació*. La idea inicial era realizar pruebas en distintos entornos, pero debido a una serie de limitaciones no ha sido posible. Así, no se han podido separar una distancia excesivamente grande el transmisor y el receptor (ya que las antenas de las que se disponía no eran adecuadas para el funcionamiento junto a los *USRP 2920*) y no se ha podido evaluar el comportamiento del sistema cuando transmisor o receptor están en movimiento (debido a que los *USRP 2920* necesitan ser alimentados a través de la red eléctrica y no es posible su desplazamiento). Por lo tanto, la evaluación principal del sistema se ha realizado en un escenario como el que se muestra en la figura 76, en el cual transmisor y receptor están en visión directa y separados una distancia de 3 metros.

Figura 76. Escenario en el que se ha realizado la evaluación de prestaciones.

Para realizar las pruebas, se optó por la banda de 2 GHz (banda en la que los amplificadores de potencia del *USRP* tiene un comportamiento bueno y estable, y que es cercana a la banda de operación de las antenas). También, se decidió utilizar una constelación *BPSK* como símbolos piloto, ya que su comportamiento es ligeramente mejor que la de la constelación *QPSK*. Para determinar que método de interpolación funciona mejor, se han realizado las pruebas para un solo ancho de banda y una sola constelación de símbolos para cada uno de los dos estándares, siendo el resultado obtenido en ellos extrapolable al resto de anchos de banda y constelaciones de símbolos. Seguidamente se presentan los resultados obtenidos tanto para *IEEE 802.16m* como para *LTE-Advanced*.

6.1 - Evaluación para IEEE 802.16m

Para este caso, se ha decidido utilizar un ancho de banda de 8,75 MHz, lo que implica una *IFFT* de 1024 puntos y una frecuencia de muestreo de $10 \cdot 10^9\ Muestras/s$, como se puede ver en el panel de control de la figura 77 y como se vio anteriormente en la tabla 3 del capítulo 3. Para la

evaluación, se ha utilizado una constelación de símbolos *QPSK*. Con estos parámetros, se ha conseguido un *throughput* de aproximadamente 10,2 Mbps y se ha estimado una *SNR* de unos 27 dB. En cada transmisión se han transmitido 5 *superframes IEEE 802.16m*, lo que equivale a 960 símbolos *OFDM*. Seguidamente se muestra el resultado obtenido con los 5 métodos de interpolación utilizados.

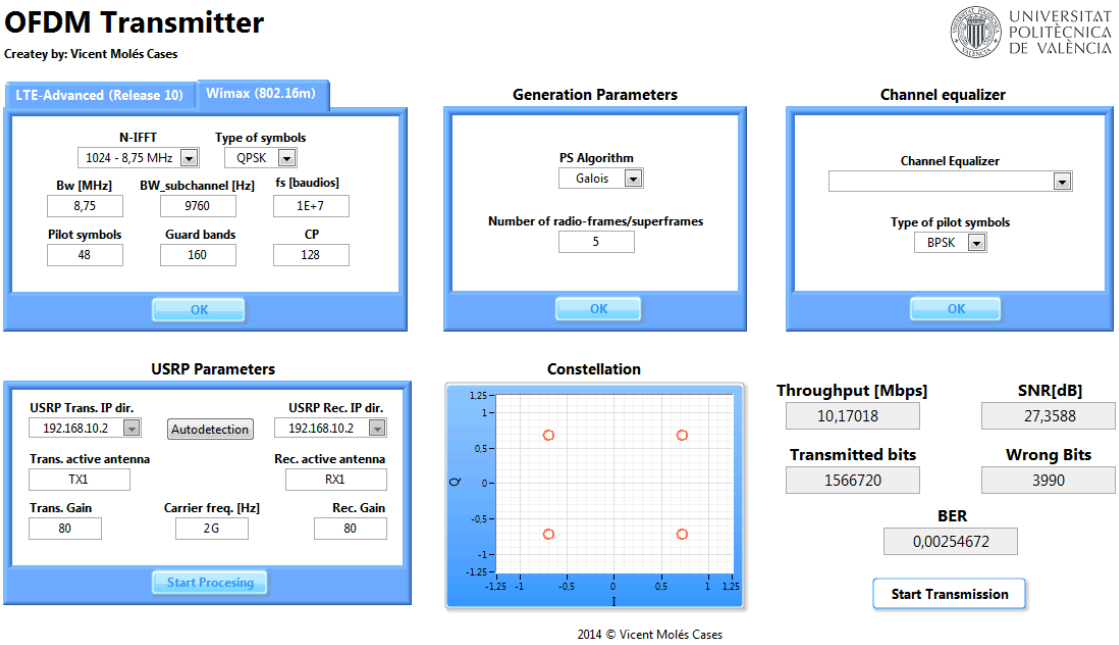


Figura 77. Panel de control con los parámetros de la transmisión *IEEE 802.16m*.

6.1.1 - Interpolación al más cercano

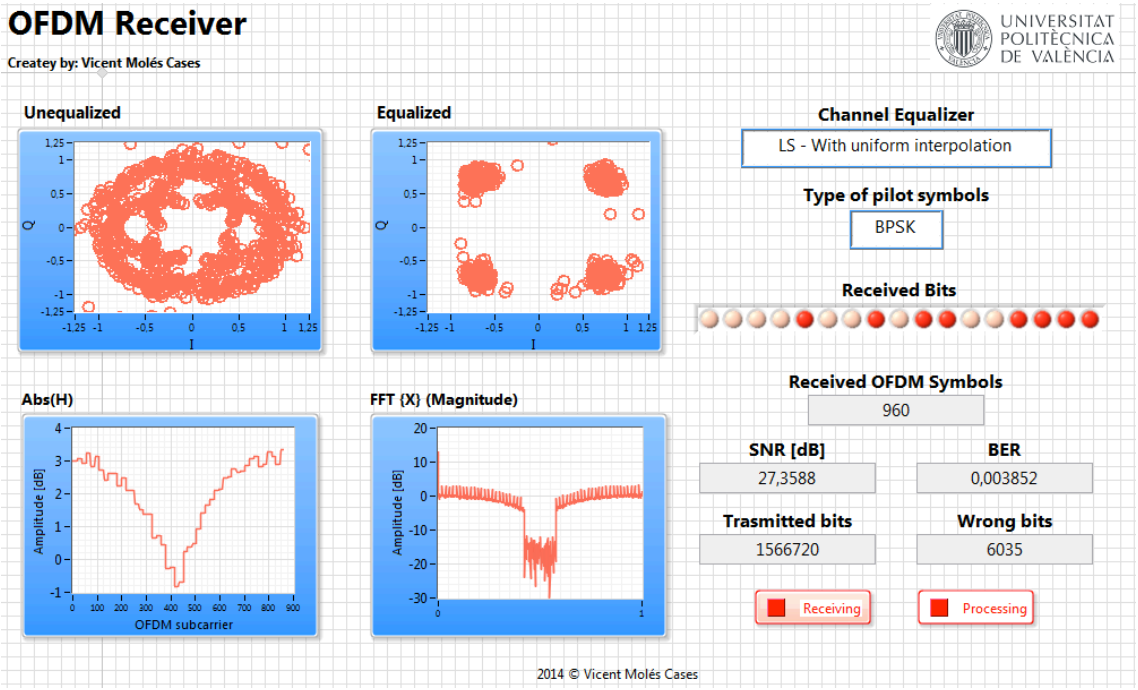


Figura 78. Resultados obtenidos con la interpolación al más cercano.

6.1.2 - Interpolación lineal

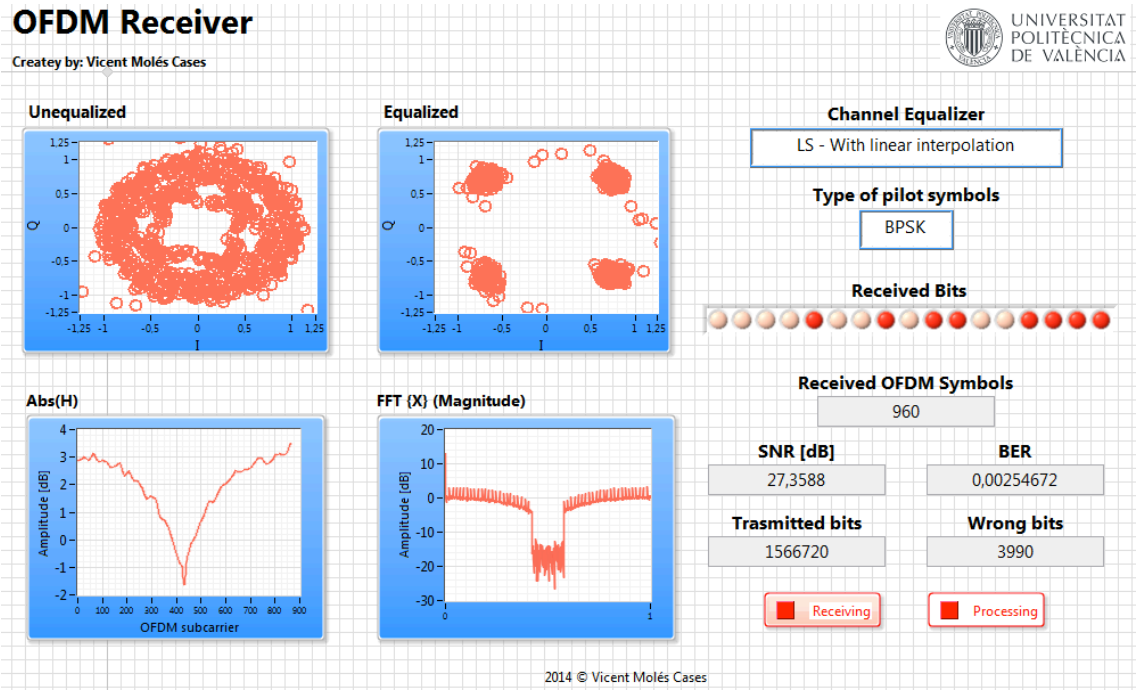


Figura 79. Resultados obtenidos con la interpolación lineal.

6.1.3 - Interpolación Spline Cubic

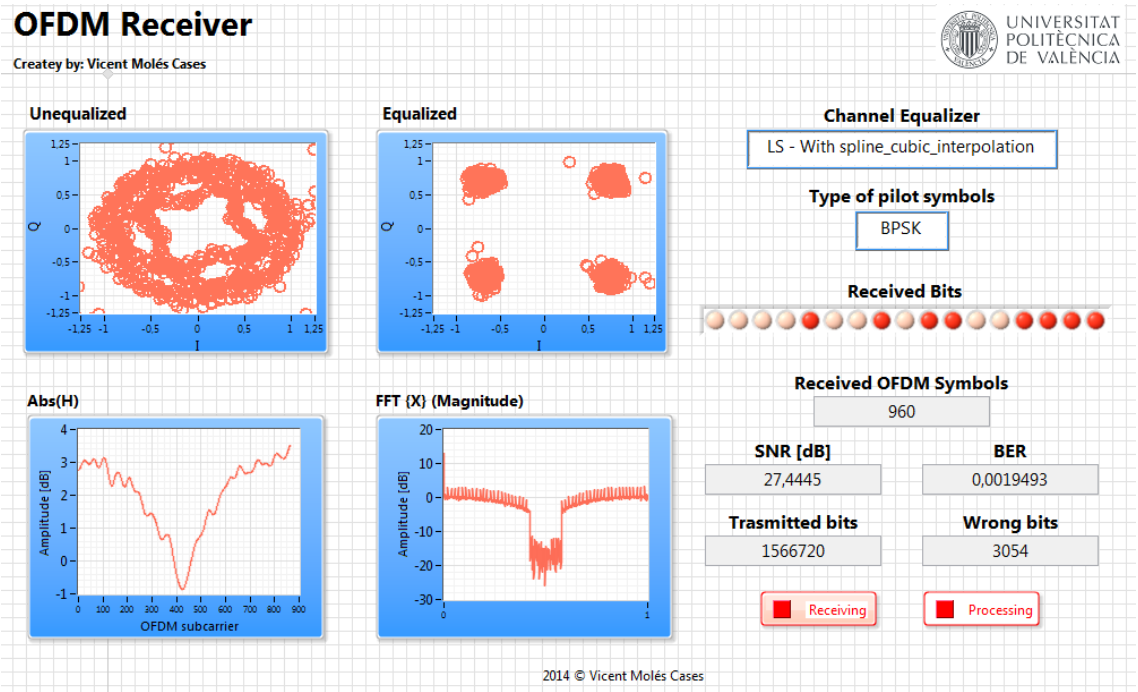


Figura 80. Resultados obtenidos con la *Spline Cubic*.

6.1.4 - Interpolación temporal

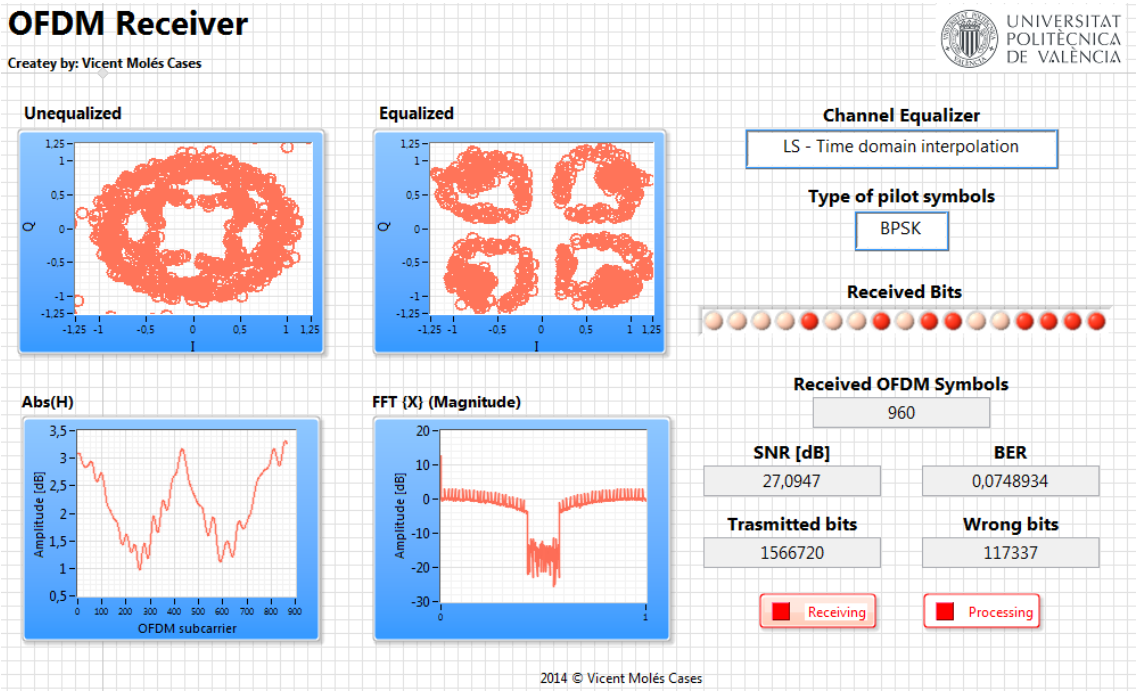


Figura 81. Resultados obtenidos con la interpolación temporal.

6.1.5 - Interpolación con filtro paso bajo

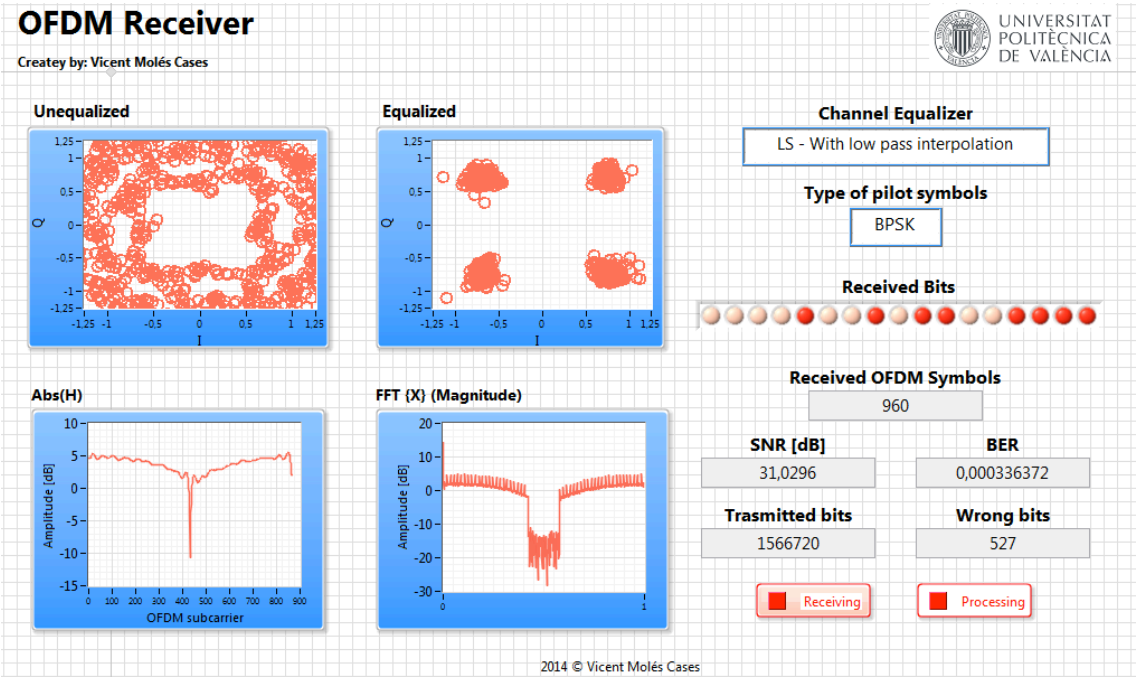


Figura 82. Resultados obtenidos con la interpolación con filtro paso bajo.

6.2 - Evaluación para LTE-Advanced

Para el caso de *LTE-Advanced*, se ha escogido un ancho de banda de 10 MHz, lo que implica una *IFFT* de 1024 puntos y una frecuencia de muestreo de $15{,}4 \cdot 10^9\ Muestras/s$, como se puede ver en el panel de control de la figura 82 y como se vio anteriormente en la tabla 2 del capítulo 3. De la misma forma que en *IEEE 802.16m*, para la evaluación se ha utilizado una constelación de símbolos *QPSK*, consiguiendo un *throughput* de aproximadamente 8,3 Mbps y estimando una *SNR* de unos 29 dB. En cada transmisión se han transmitido 10 *radio-frames LTE-Advanced* (1400 símbolos *OFDM)*.

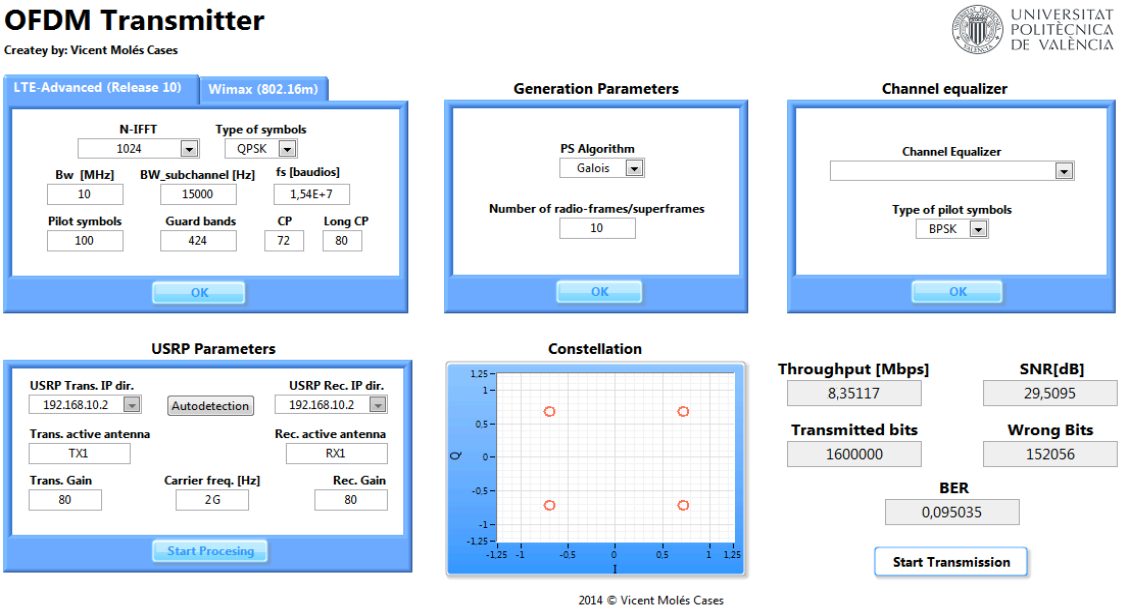


Figura 83. Panel de control con los parámetros de la transmisión *LTE-Advanced*.

6.2.1 - Interpolación al más cercano

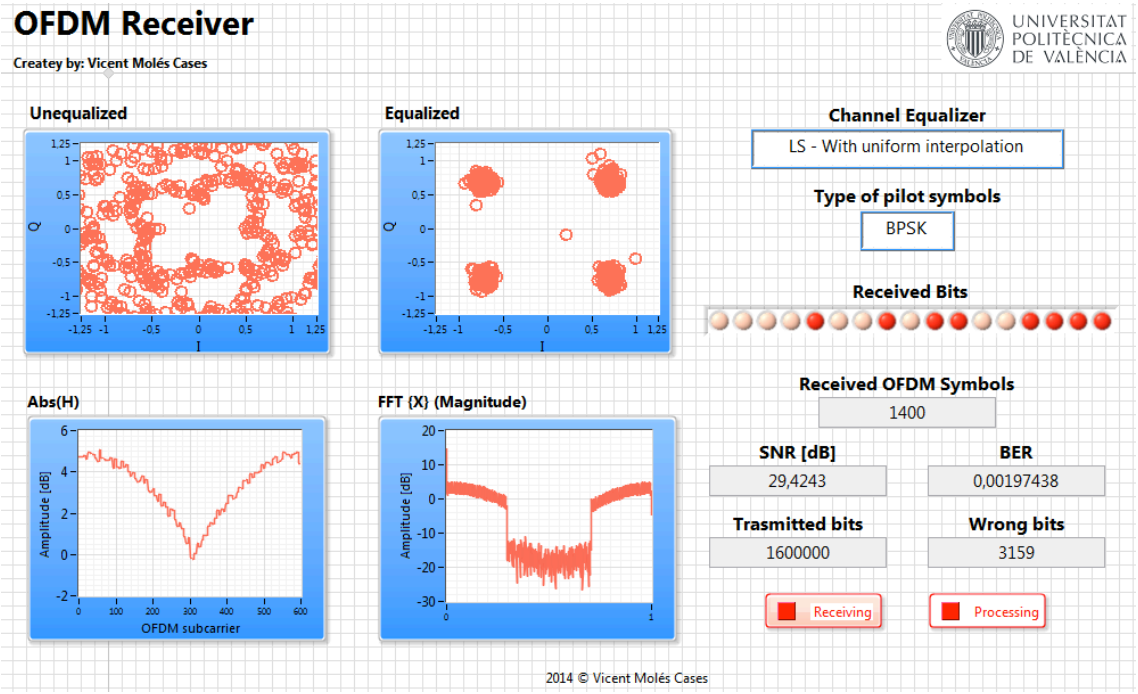


Figura 84. Resultados de la interpolación al más cercano.

6.2.2 - Interpolación lineal

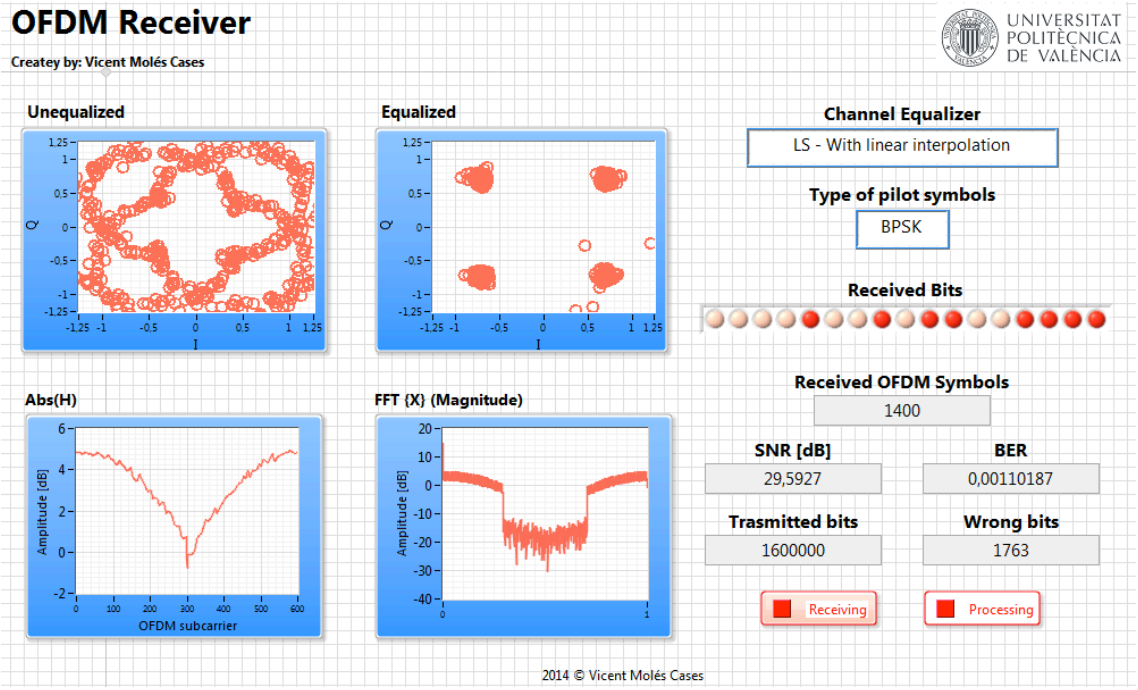


Figura 85. Resultados obtenidos con la interpolación lineal.

6.2.3 - Interpolación Spline Cubic

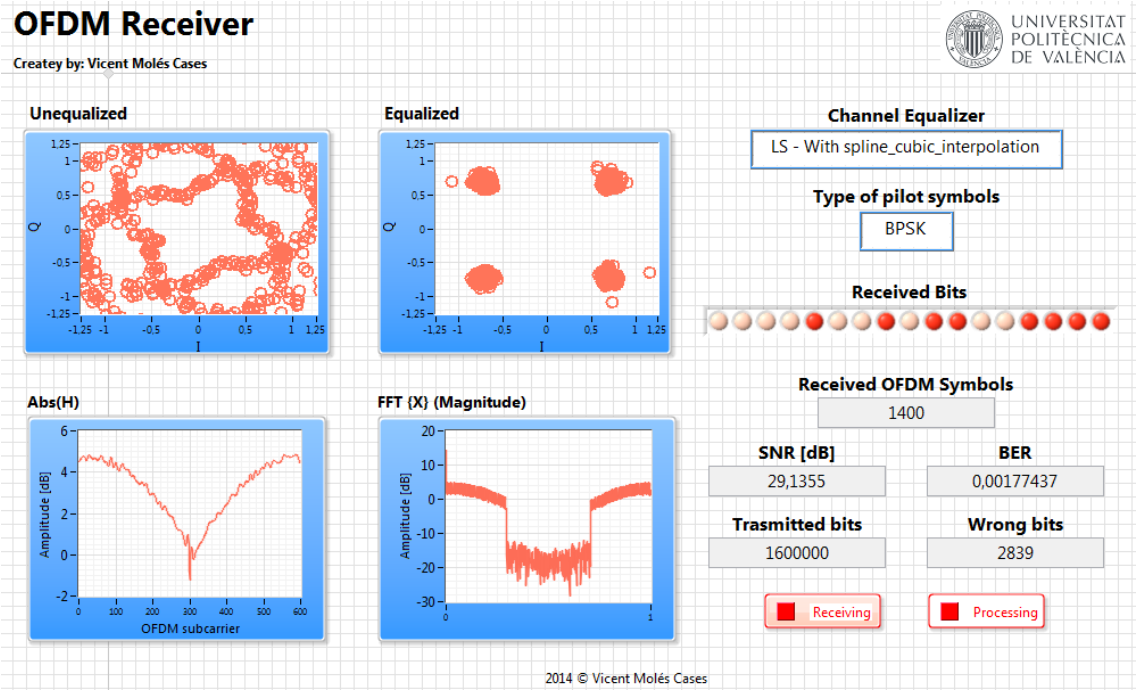


Figura 86. Resultados obtenidos con la interpolación *Spline Cubic*.

6.2.4 - Interpolación temporal

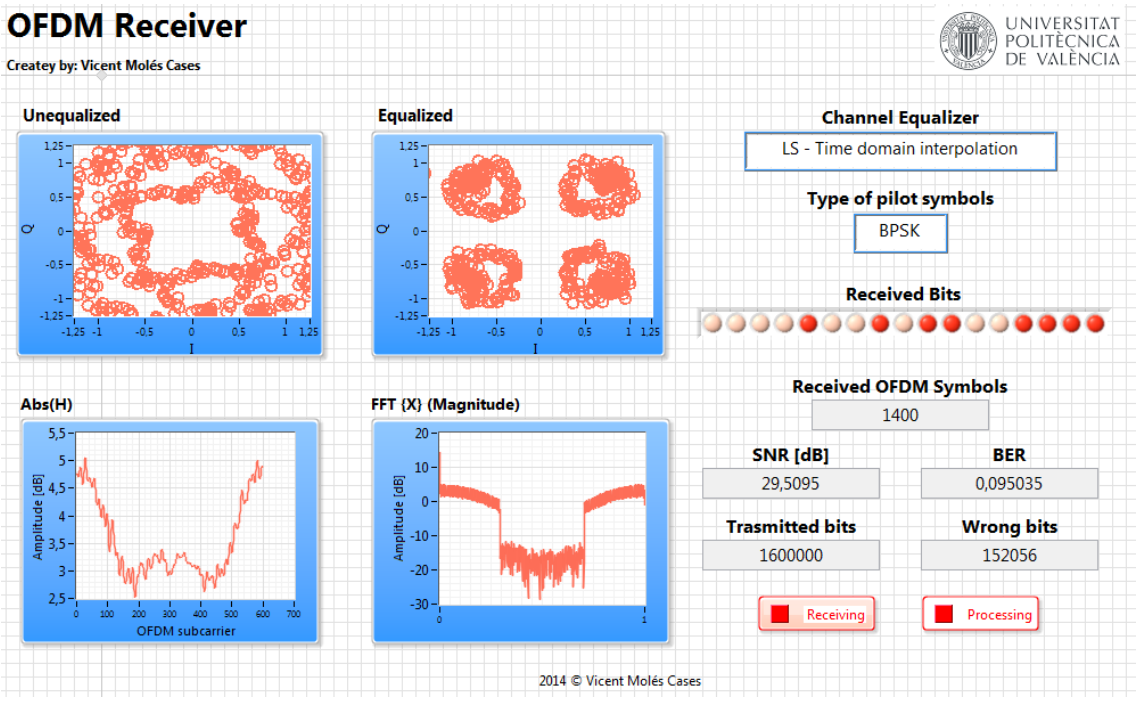


Figura 87. Resultados de la interpolación temporal.

6.2.5 - Interpolación con filtro paso bajo

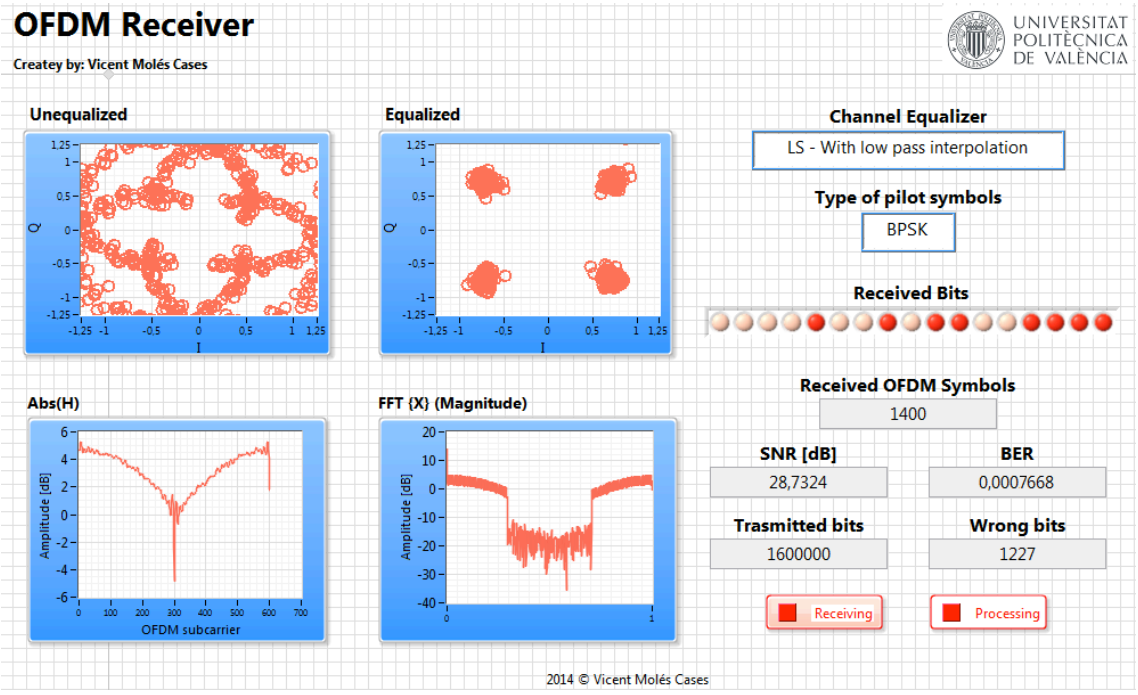


Figura 88. Resultados obtenidos con la interpolación con filtro paso bajo.

6.3 - Análisis de los resultados

En los dos apartados anteriores hemos podido ver los resultados obtenidos con el sistema diseñado para *LTE-Advanced* y *IEEE 802.16m*. En ellos, vemos como el comportamiento de los distintos métodos de interpolación es muy similar para ambos estándares.

Así, el método de interpolado que proporciona un mejor resultado es el interpolado con filtro paso bajo, como ya se predijo en el capítulo 2, ya que en éste se minimiza el error cuadrático medio (aunque tiene como desventaja que su tiempo de cómputo es sustancialmente mayor que el del resto de métodos). Con el mencionado método obtenemos una tasa de error del orden de 10^{-4} (1 bit erróneo de cada 10000). El método que peores resultados nos proporciona es la interpolación temporal, con la que obtenemos una tasa de error del orden de 10^{-2} (1 de cada 100 bits erróneo) y una constelación mucho más dispersa que con el resto. Los otros 3 métodos de interpolación (lineal, *Spline Cubic* y método al más cercano) proporcionan resultados bastante similares, con *BER* del orden de 10^{-3} , siendo ligeramente mejores los métodos *Spline Cubic* y lineal ya que el ofrecen una interpolación más gradual que el método al más cercano. En la tabla 4 se presentan los resultados obtenidos en las pruebas anteriores y que muestran el comportamiento mencionado anteriormente.

	IEEE 802.16m	***LTE-Advanced***
Int. Filtro Paso Bajo	$3{,}36 \cdot 10^{-4}$	$7{,}66 \cdot 10^{-4}$
Int. Spline Cubic	$1{,}94 \cdot 10^{-3}$	$1{,}77 \cdot 10^{-3}$
Int. Lineal	$2{,}54 \cdot 10^{-3}$	$1{,}10 \cdot 10^{-3}$
Int. Al más cercano	$3{,}85 \cdot 10^{-3}$	$1{,}97 \cdot 10^{-3}$
Int. Temporal	$7{,}48 \cdot 10^{-2}$	$9{,}50 \cdot 10^{-2}$

Tabla 4. BER obtenidas en la evaluación del sistema.

En cuanto al *throughput* obtenido en cada estándar, hay que mencionar que aunque que con *IEEE 802.16m* hemos utilizado un ancho de banda menor que en *LTE-Advanced* (8,75 MHz frente a 10 MHz) el *throughput* obtenido es mayor, ya que la densidad de símbolos piloto y el número de bandas de guarda es menor en *IEEE 802.16m*.

El *throughput* máximo alcanzable con el sistema diseñado es de unos 50 Mbps para el caso de *LTE-Advanced* y de 60 Mbps para *IEEE 802.16m*, utilizando en ambos casos un ancho de banda de 20 MHz y una constelación *64-QAM*. Así, si al sistema diseñado se le añadieran técnicas *MIMO* y agregación de portadoras se podrían superar los 300 Mbps (teniendo en cuenta que faltaría añadir la codificación de canal propia de cada uno de los estándares, que reduciría ligeramente el *throughput* que se obtendría).

En los resultados de las 2 secciones anteriores, el módulo de la respuesta estimada del canal tiene (en la mayoría de las subportadoras) un valor mayor que 0 dB. Esto puede parecer extraño en un principio, ya que si el módulo del canal es mayor que 0 dB significa que el canal amplifica la señal en esa determinada subportadora. Se tiene que matizar que este fenómeno ocurre debido a que el *USRP* tiene amplificadores de potencia tanto en transmisión como en recepción, por lo que la amplificación hace que el módulo del canal estimado sea mayor que 0 dB en algunas bandas. Por lo tanto no es que el canal amplifique la señal (todo lo contrario, la atenúa) si no que en la estimación de canal se consideran los amplificadores como parte del canal, resultando un módulo mayor que 0 dB.

Capítulo 7 - Conclusiones y líneas futuras

A lo largo de este documento, se ha podido ver el diseño mediante *LabVIEW* de un sistema *OFDM* aplicado a *LTE-Advanced* y *IEEE 802.16m*, así como de distintos igualadores de canal para *OFDM*. Durante el proceso de diseño y posterior evaluación del sistema, se ha podido llegar a las siguientes conclusiones:

- *LabVIEW* es una herramienta de una utilidad extraordinaria a la hora del diseño de sistemas de comunicaciones, debido a la multitud de herramientas de las que dispone, y que permiten de una forma muy intuitiva diseñar desde sencillos moduladores (como *BPSK* o *QAM*) hasta sistemas de comunicaciones completos (como ha sido el caso de este proyecto). También aporta la ventaja de funcionar junto a los transceptores *USRP*, permitiendo evaluar los sistemas diseñados en un entorno no sólo simulado, sino también real y por lo tanto estudiar los efectos que ello conlleva.

- Controlar el orden de ejecución en *LabVIEW* es una tarea tan importante como sencilla gracias a las estructuras de organización temporal de las que dispone (como las *Stacked Sequence Structure*). Es de suma importancia poder controlar que tareas se ejecutan en cada momento para que cada bloque tenga los inputs adecuados y que no se ejecuten bloques cuyos *inputs* aún no se encuentren disponibles, hecho que provocaría errores en la ejecución.

- La sincronización temporal es un aspecto fundamental en el correcto funcionamiento de un demodulador, en nuestro caso un demodulador *OFDM*. Además, es un aspecto que no se suele tener en cuenta en la docencia, ya que las prácticas se realizan en entornos simulados en los que el alumno no se tiene que preocupar por el proceso de sincronización temporal. Por ello, para el diseño del sistema se probaron diversos métodos de sincronización temporal y se profundizó en el estudio de cada uno de ellos, resultando una modificación del algoritmo *Schmidl and Cox* obtenido a través de diversas pruebas experimentales el que mejor resultado proporcionaba.

- La igualación de canal en *OFDM* es clave, ya que sin proceso de igualación la tasa de error que se obtiene en el demodulador es muy elevada, pudiendo llegar a una *BER* de $0{,}5$. Por ello, el estudio y profundización en esta parte del sistema de comunicaciones es muy interesante, llegando a la conclusión de que el estimador *Least Squares* no es el que tiene un mejor funcionamiento pero ofrece un resultado bueno en relación a su bajo tiempo de computo.

- A su vez, el método de interpolación utilizado también es muy importante para la igualación de canal en *OFDM,* ya que determina si se realiza la ecualización de una forma correcta en las subportadoras en las que no hay símbolos piloto. En este proyecto, se ha llegado a la conclusión de que de entre todos los métodos de interpolación estudiados, el que ofrece un mejor resultado es el método de interpolación que utiliza un filtro *FIR* paso bajo para realizar el proceso, con una *BER* del orden de 10^{-4}. Sin embargo el coste computacional de esta interpolación es muy elevado. Por lo que la interpolación *Spline Cubic* ofrece una alternativa que presenta una *BER* razonablemente baja (del orden de 10^{-3}) y coste computacional bajo, siendo la mejor opción a la hora de realizar transmisiones mediante el sistema *OFDM* diseñado.

Finalmente, el presente proyecto podría ser ampliado en el futuro, partiendo del sistema *OFDM* ya diseñado al cual se le podrían añadir las siguientes mejoras y funcionalidades:

- Modificar el sistema para implementar técnicas *MIMO* y de esta forma poder aumentar el *throughput* y la eficiencia espectral. Para poder implementar dichas técnicas, sería

necesaria la adquisición de más dispositivos *USRP 2920* por parte del departamento de comunicaciones de la UPV.

- Utilizar las funciones de captura de audio de las que dispone *LabVIEW* para capturar audio mediante la tarjeta de sonido del PC y utilizar códecs de audio para poder obtener los bits del audio capturado y realizar la transmisión de dicho audio. También se podría utilizar *codecs* para la transmisión de video *HD*, ya que las tasas que nos ofrece *OFDM* son adecuadas para transmisión de video *HD*.

- Implementar el estimador *MMSE*, pero para ello sería necesario disponer de un *PC* con un muy buen procesador (con mínimo cuatro núcleos y una frecuencia de reloj de mínimo 3 GHz) ya que este estimador requiere una cantidad muy importante de cálculos para realizar la estimación.

- Ofrecer la posibilidad de realizar la agregación de portadoras típica de los estándares de cuarta generación estudiados. Hecho que haría que se pudiera aumentar considerablemente el *throughput* alcanzable.

- Utilizar el *NI LabVIEW Report Generation Toolkit* para la generación de informes que incluyan los parámetros de la transmisión realizada, el *throughput* y la *BER* obtenida, la *SNR* estimada, y diversas capturas de la constelación de símbolos (tanto ecualizada como sin ecualizar), así como gráficas con las estimación del canal realizada en diversos instantes de tiempo.

- Si en un futuro se pudieran conseguir las antenas adecuadas para el funcionamiento de los *USRP 2920,* las *VERT 900* del fabricante *Ettus* por ejemplo, sería interesante evaluar el sistema en entornos sin visión directa y con el transmisor y el receptor separados una distancia mayor de 3 metros.

Capítulo 8 - Bibliografía

[1] - Hara, S; Prasad, R. "Multicarrier Techniques for 4G Mobile Communications", *Artech House, Inc*., capitulos 3 - 4, 2003.

[2] - Oppenheim, V; Schafer, W; Buck, R. "Discrete - Time Signal Processing", *Prentice Hall, Inc*., edición 2ª, capítulos 8 - 9, 1999.

[3] - Ozdemir, K; Arslan, H. "Channel estimation for wireless OFDM systems", *IEEE Communications surveys*, vol. 9, no. 2, pp. 18 - 24, 2007.

[4] - Coleri, S; Ergen, M; Puri, A; Bahai, A. "Channel Estimation Techniques Based on Pilot Arrangement in OFDM Systems", *IEEE Transactions on Broadcasting*, vol. 48, no. 3, pp. 225 - 226, Septiembre 2002

[5] - National Instruments. "Spline Interpolation VI", http://zone.ni.com/reference/enXX/help/371361H-01/gmath/spline_interpolation/. [Online]

[6] - Van de Beek, J; Sandell, M; Ola Börjesson, P. "On Synchronization in OFDM Systems Using the Cyclic Prefix", RVK 96, p. 665, 1996.

[7] - Schmidl, TM. "Robust frequency and timing synchronization for OFDM", *IEEE Transactions on Communications,* vol. 45, Issue 12, Diciembre 1997.

[8] - ITU-R. "Requirements related to technical performance for IMT-Advanced radio interface(s)" , *Report ITU-R M.2134*, 2008.

[9] - Rummey, M. "3GPP LTE Standards Update: Release 11, 12 and beyond", *Agilent technologies*, p.3, Octubre 2008.

[10] - 3GPP. "About 3GPP", http://www.3gpp.org/about-3gpp/about-3gpp. [Online]

[11] - 3GPP. "LTE", http://www.3gpp.org/technologies/keywords-acronyms/98-lte. [Online]

[12] - 3GPP. "Carrier Aggregation explained", http://www.3gpp.org/technologies/keywords-acronyms/101-carrier-aggregation-explained. [Online]

[13] - 3GPP. "LTE-Advanced", http://www.3gpp.org/technologies/keywords-acronyms/97-lte-advanced. [Online]

[14] - Belhouchet, L; Ebdelli, H; "LTE Overview - Design Targets and Multiple Access Technologies", *Session 3, ITU/BDT Arab Regional Workshop on "4G Wireless Systems"*, p. 25 - 57, Enero 2010.

[15] - IEEE 802.16 - Broadband Wireless Access Working Group. http://standards.ieee.org/develop/wg/WG802.16.html. [Online]

[16] - Müller, M. "IEEE 802.16m Technology Introduction, White Paper" *Rohde & Schwarz,* p. 6, Abril 2009.

[17] - Srinivasan, R; Hamiti, S. "IEEE 802.16m System Description Document (SDD)", *IEEE 802.16m-09/0034r4*, pp. 77 - 89, Diciembre 2010.

[18] - National Instruments. "Software de Desarrollo de Sistemas NI LabVIEW" http://www.ni.com/labview/esa/. [Online]

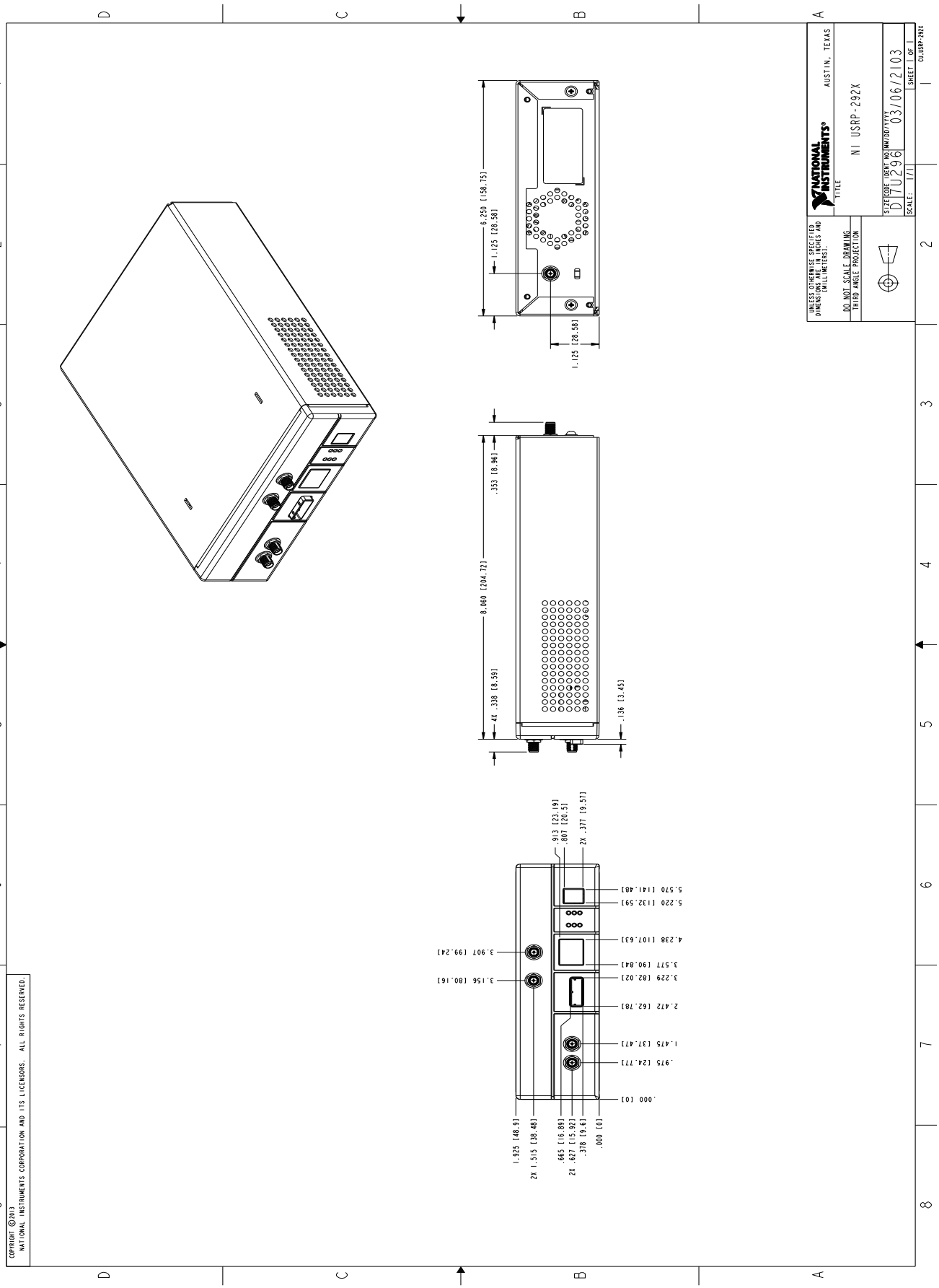
[19] - National Instruments. "Data Structures in NI LabVIEW", http://www.ni.com/getting-started/labview-basics/data-structures. [Online]

[20] - National Instruments. "Device specifications NI USRP^{TM}-2920", http://www.ni.com/pdf/manuals/375839a.pdf , pp. 2 - 4. [Online]

[21] - National Instruments. "NI USRP-292x/293x Datasheet", http://sine.ni.com/ds/app/doc/p/id/ds-355/lang/es. [Online]

[22] - National Instruments. “NI LabVIEW Modulation Toolkit Help”, http://digital.ni.com/manuals.nsf/websearch/8D83AC512A841FB586257B5500202D37, sección MT Generate Bits (poly) VI. [Online]

Anexo


6.250 [158.75]
1.125 [28.581]
1.125 [28.581]
.353 [8.961]
8.060 [204.72]
4X .338 [8.59]
.136 [3.45]
.913 [23.19]
.807 [20.51]
2X .377 [9.57]
5.570 [141.48]
5.220 [132.59]
4.238 [107.63]
3.577 [90.84]
3.229 [82.02]
3.907 [99.24]
3.156 [80.16]
2.472 [62.78]
1.475 [37.47]
.975 [24.77]
.000 [0]
1.925 [48.91]
2X 1.515 [38.48]
.665 [16.89]
2X .627 [15.92]
.378 [9.61]
.000 [0]
NATIONAL INSTRUMENTS
AUSTIN, TEXAS
TITLE
NI USRP-292X
SIZE
CODE IDENT NO.
DWG NO.
D
17U296
SCALE: 1/1
DO NOT SCALE DRAWING
THIRD ANGLE PROJECTION
03/06/2103
SHEET 1 OF 1
COPYRIGHT ©2013
NATIONAL INSTRUMENTS CORPORATION AND ITS LICENSORS. ALL RIGHTS RESERVED.