

APLICACIÓN DE UNA ARQUITECTURA ABIERTA EN EL DISEÑO DE SISTEMAS DE INSTRUMENTACIÓN PARA EL ANÁLISIS DE SEÑALES DE COMUNICACIONES INALÁMBRICAS



DAVID ERNESTO SALGADO SALAZAR

**UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO DE NUEVAS TECNOLOGIAS EN TELECOMUNICACIONES
LÍNEA DE I+D EN GESTIÓN INTEGRADA DE REDES, SERVICIOS Y ARQUITECTURAS DE
TELECOMUNICACIONES
POPAYAN – CAUCA
2011**

**APLICACIÓN DE UNA ARQUITECTURA ABIERTA EN EL DISEÑO DE SISTEMAS DE
INSTRUMENTACIÓN PARA EL ANÁLISIS DE SEÑALES DE COMUNICACIONES INALÁMBRICAS**

DAVID ERNESTO SALGADO SALAZAR

*Trabajo de grado presentado para optar al título
de Ingeniero en Electrónica y Telecomunicaciones*

Directora: Ing. Claudia Milena Hernández Bonilla

**UNIVERSIDAD DEL CAUCA
FACULTAD DE INGENIERIA ELECTRÓNICA Y TELECOMUNICACIONES
DEPARTAMENTO DE TELECOMUNICACIONES
GRUPO DE NUEVAS TECNOLOGIAS EN TELECOMUNICACIONES
LÍNEA DE I+D EN GESTIÓN INTEGRADA DE REDES, SERVICIOS Y ARQUITECTURAS DE
TELECOMUNICACIONES
POPAYAN – CAUCA
2011**

Hoja de Aprobación

Firma del Director del Trabajo de Grado:

Ing. Claudia Hernández

Firma del Jurado:

Ing. Víctor Quintero

Firma del Jurado:

Ing. Edgar Ortiz

ABSTRACT

This document describes how to develop a wireless signal analyzer device using the Software Communications Architecture. It describes the hardware elements used in the system and the software components designed to build the solution. The focus of the software design is the way to take advantage of Software Communications Architecture in the implementation of signal analysis applications. Chapter 1 describes the technologies used to build current signal analysis devices, Chapter 2 describes SDR and how it can be used to build a hardware platform to develop signal analysis systems, Chapter 3 describes a software design based on SCA to analyze wireless communication signals, Chapter 4 describes the algorithms used to analyze OFDM signals and the way they can be implemented following the software design, finally Chapter 5 describes the implementation of one prototype based on USRP and the obtained results.

RESUMEN

Este trabajo describe como desarrollar un dispositivo de instrumentación para el análisis de señales de comunicaciones inalámbricas utilizando la Arquitectura de Software de Comunicaciones. Este trabajo describe los elementos de hardware usados en el sistema y los componentes de software diseñados para construir la solución. El enfoque del diseño del software es la forma en la que se puede aprovechar la Arquitectura de Software de Comunicaciones en la implementación de aplicaciones de análisis de señales. El Capítulo 1 describe las tecnologías utilizadas en los dispositivos actuales para el análisis de señales, el Capítulo 2 describe las tecnologías de SDR y cómo estas pueden usarse en la implementación de una plataforma de hardware para el desarrollo de aplicaciones de análisis de señales, el Capítulo 3 describe el diseño de un software para el análisis de señales de comunicaciones, basado en la arquitectura SCA, el Capítulo 4 describe los algoritmos de procesamiento digital de señales necesarios para el análisis de señales OFDM y como pueden implementarse según el diseño del software planteado, finalmente el Capítulo 5 describe la implementación de un prototipo basado en el hardware USRP y los resultados que se obtuvieron.

*A mi madre por todo su apoyo,
a Tatiana y Catalina por creer inquebrantablemente en mí.*

AGRADECIMIENTOS

Presento mis más sinceros agradecimientos a la ingeniera Cristina Realpe por su interés en mi carrera profesional y por toda la colaboración que me brindó incondicionalmente para hacer de esta meta una realidad. Debo agradecer también a la ingeniera Katherine Quiceno por su amistad y su colaboración desinteresada durante todo el tiempo en el que estuve ausente de la ciudad de Popayán. Finalmente y no menos importante, debo agradecer a la Ingeniera Claudia Hernández por su ayuda como Directora de Proyecto y por compartir de una u otra manera su experiencia y su punto de vista para contribuir con el desarrollo de este trabajo.

TABLA DE CONTENIDO

INTRODUCCIÓN.....	1
1. FUNDAMENTOS DEL ANÁLISIS VECTORIAL DE SEÑALES	5
1.1 SEÑALES.....	5
1.1.1 El dominio del tiempo y el dominio de la frecuencia	5
1.1.2 Relación entre el dominio del tiempo y el dominio de la frecuencia.....	6
1.2 EL ANALIZADOR DE SEÑALES.....	7
1.2.1 El analizador de espectro	7
1.2.2 El analizador vectorial de señales.....	8
1.2.3 Ventajas del analizador vectorial de señales.....	9
1.3 TEORÍA DE OPERACIÓN DEL ANALIZADOR VECTORIAL DE SEÑALES	10
1.3.1 El receptor heterodino	10
1.3.2 Muestreo y digitalización de la señal	11
1.3.3 Detección en cuadratura y diezmado.....	12
1.3.4 Transformada Rápida de Fourier y análisis espectral de la señal.....	13
1.3.5 Las funciones de ventana para la FFT	14
1.3.6 Demodulación y análisis vectorial de la señal	15
2. RADIO DEFINIDO POR SOFTWARE.....	17
2.1 CONCEPTOS DE RADIO DEFINIDO POR SOFTWARE.....	17
2.2 EL USRP COMO PLATAFORMA DE RADIO DEFINIDO POR SOFTWARE	18
2.2.1 La tarjeta madre	19
2.2.2 Las tarjetas hijas	20
2.3 RADIO DEFINIDO POR SOFTWARE EN DISPOSITIVOS DE INSTRUMENTACIÓN PARA ANÁLISIS DE SEÑALES	21
2.3.1 Requerimientos de los sistemas de instrumentación para el análisis de señales.....	21

2.3.2	Beneficios de las tecnologías de Radio Definido por Software en los equipos de instrumentación de RF.....	22
2.3.3	El USRP como plataforma de Radio Definido por Software para el desarrollo de aplicaciones de instrumentación	22
2.3.4	Otras alternativas de plataformas de hardware para el desarrollo de aplicaciones de Radio Definido por Software.....	25
3.	ARQUITECTURA DE SOFTWARE DE COMUNICACIONES	27
3.1	DESCRIPCIÓN DE LA ARQUITECTURA DE SOFTWARE DE COMUNICACIONES	27
3.1.1	El entorno operativo de SCA	27
3.2	OSSIE LA IMPLEMENTACIÓN EMPOTRADA DE CÓDIGO ABIERTO PARA SCA	30
3.2.1	El entorno de desarrollo para aplicaciones de onda.....	30
3.2.2	Ventajas y desventajas de OSSIE.....	32
3.3	SOFTWARE PARA EL ANÁLISIS DE SEÑALES DE COMUNICACIONES BASADO EN SCA	33
3.3.1	Diseño de clases del software para el análisis de señales de comunicaciones	33
3.3.2	Diagramas de secuencia para el software de análisis de señales de comunicaciones	42
3.3.3	Consideraciones acerca del diseño del analizador de señales.....	46
4.	APLICACIÓN SCA PARA EL ANÁLISIS DE SEÑALES OFDM	49
4.1	PRINCIPIOS DE OFDM.....	49
4.1.1	El concepto de transmisión con múltiples subportadoras.....	49
4.1.2	Ortogonalidad de las subportadoras	51
4.1.3	Prefijo cíclico	52
4.2	APLICACIÓN SCA PARA EL ANÁLISIS DE SEÑALES OFDM	53
4.2.1	El Controlador USRP.....	54
4.2.2	Sincronizador OFDM	55
4.2.3	Demodulador OFDM.....	56
4.2.4	Desmapeador de Símbolos OFDM	57
4.2.5	Visor de Espectro	57

4.2.6	Visor en el Dominio del Tiempo	58
4.2.7	Visor de Dispersión.....	59
4.2.8	Detector de errores en recepción	59
4.3	LA INTERFAZ GRÁFICA DE USUARIO DE LA APLICACIÓN DE ANÁLISIS DE SEÑALES	60
5.	VERIFICACIÓN Y COMENTARIOS DE LA IMPLEMENTACIÓN DEL ANALIZADOR DE SEÑALES OFDM.....	63
5.1	VALIDACIÓN DE LOS COMPONENTES SCA DEL PROTOTIPO DE ANÁLISIS DE SEÑALES OFDM.....	63
5.1.1	Características de la señal de prueba utilizada para validar el modelo	64
5.2	VERIFICACIÓN DEL FUNCIONAMIENTO DEL PROTOTIPO DE ANÁLISIS DE SEÑALES OFDM.....	67
5.2.1	Verificación de la conectividad entre el hardware USRP y el Computador Personal	68
5.2.2	Recepción de una señal sinusoidal producida por un generador de señales de laboratorio.....	70
5.2.3	Generación de una señal OFDM de prueba a través del USRP	71
5.2.4	Recepción y análisis de la señal OFDM de prueba a través del USRP	74
5.2.5	Análisis de la señal OFDM de prueba mediante configuración en bucle y simulación del canal inalámbrico.....	76
5.3	DESCRIPCIÓN Y COMENTARIOS ACERCA DE LAS HERRAMIENTAS DE SOFTWARE USADAS.....	78
5.3.1	Entorno integrado de desarrollo Eclipse	78
5.3.2	<i>Plugin OSSIE Eclipse Feature</i>	79
5.3.3	Glade.....	80
5.3.4	Doxygen.....	81
5.3.5	Librerías de software	81
	CONCLUSIONES Y RECOMENDACIONES.....	85
	REFERENCIAS BIBLIOGRÁFICAS	87

LISTA DE FIGURAS

Figura 1. Esquema general de la solución para implementar un dispositivo para el análisis de señales	1
Figura 1.1. Relación entre el dominio del tiempo y el dominio de la frecuencia	6
Figura 1.2. Analizador de espectro superheterodino	7
Figura 1.3. Analizador de espectro basado en la Transformada Rápida de Fourier	8
Figura 1.4. Analizador vectorial de señales.....	9
Figura 1.5. Receptor heterodino	10
Figura 1.6. Señal en el dominio del tiempo, señal de muestreo y señal muestreada	11
Figura 1.7. Diagrama del detector en cuadratura, diezmadore y filtro pasabajos	12
Figura 1.8. Síntesis de una DFT de 8 puntos	14
Figura 1.9. Efecto de la función de ventana sobre la FFT	14
Figura 1.10. Diagrama de bloques de un demodulador de propósito general.....	15
Figura 2.1. Arquitectura ideal del radio definido por software	17
Figura 2.2. Diagrama simplificado del hardware USRP [19]	18
Figura 2.3. Implementación del DDC en el FPGA.....	19
Figura 2.4. Implementación del DUC en el Circuito Integrado AD9862	20
Figura 2.5. Diagrama de la sección de recepción de la tarjeta RFX900	23
Figura 3.1. Organización de las interfaces IDL de SCA [24].....	28
Figura 3.2. Organización de los componentes de un sistema SCA	29
Figura 3.3. Relaciones XML del perfil de dominio.....	30
Figura 3.4. Flujo de diseño en entorno de desarrollo de OSSIE.....	31
Figura 3.5. Arquitectura de un analizador de señales basado en SCA.....	34
Figura 3.6. Diagrama de la clase Controlador	35

Figura 3.7. Diagrama de la clase Vista	36
Figura 3.8. Diagrama de la abstracción de las clases <i>Widget</i> y <i>Ventana</i>	38
Figura 3.9. Diagrama de la clase <i>Widget</i> de Análisis de Señal.....	38
Figura 3.10. Diagrama de la clase Modelo	39
Figura 3.11. Diagrama de la clase Modelo de Aplicación SCA.....	40
Figura 3.12. Diagrama de la clase Administrador SCA.....	41
Figura 3.13. Diagrama simplificado de las clases SCA Domain Manager, Application y Resource	41
Figura 3.14. Diagrama de secuencia del proceso de inicialización.....	43
Figura 3.15. Diagrama de secuencia para el cambio de propiedad de la aplicación SCA	45
Figura 3.16. Diagrama de secuencia para la actualización de trazos	46
Figura 4.1. Concepto de modulación de múltiples subportadoras	50
Figura 4.2. Diagrama de bloques de un sistema de modulación de múltiples subportadoras	50
Figura 4.3. Diagrama de bloques alternativo de sistema de modulación de múltiples subportadoras	51
Figura 4.4. Espectro de múltiples subportadoras ortogonales en frecuencia.....	51
Figura 4.5. Inserción del prefijo cíclico	53
Figura 4.6. Diagrama de bloques de la aplicación SCA para el análisis de señales OFDM	53
Figura 4.7. Diagrama de bloques del sincronizador OFDM	55
Figura 4.8. Diagrama de bloques del demodulador OFDM.....	56
Figura 4.9. Diagrama de bloques del componente visor de espectro.....	57
Figura 4.10. Diagrama de bloques del visor en tiempo.....	58
Figura 4.11. Diagrama de bloques del visor de dispersión.....	59
Figura 4.12. Esquema de un <i>frame</i> con control de errores por CRC.....	59
Figura 4.13. Interfaz gráfica de usuario de la aplicación de análisis de señales OFDM	61
Figura 5.1. Modelo del Modulador y Demodulador OFDM	63

Figura 5.2. Secuencia de símbolos para el modulador OFDM	64
Figura 5.3. Señal en el dominio del tiempo a la salida del modulador OFDM.....	65
Figura 5.4. Espectro de la señal a la salida del modulador OFDM.....	65
Figura 5.5. Señal de sincronización de símbolos OFDM	66
Figura 5.6. Secuencia de símbolos a la salida del demodulador OFDM	66
Figura 5.7. Diagrama de dispersión de la señal a la salida del demodulador OFDM.....	67
Figura 5.8. Señal de sincronización de <i>frames</i>	67
Figura 5.9. Mensajes de salida del <i>Device Manager</i> con la confirmación de la conexión con el USRP	68
Figura 5.10. Configuración de los parámetros de operación del USRP	69
Figura 5.11. Mensajes de confirmación del <i>Device Manager</i> acerca del cambio de parámetros del USRP	69
Figura 5.12. Espectro de una señal sinusoidal de prueba obtenido con el dispositivo de análisis de señales.....	70
Figura 5.13. Trazo <i>Maxhold</i> después de un barrido en frecuencia con señal sinusoidal de amplitud constante.....	71
Figura 5.14. Esquema propuesto para la verificación del prototipo mediante una señal OFDM de prueba.....	72
Figura 5.15. Generación de la señal OFDM de prueba	72
Figura 5.16. Espectro y potencia de la señal OFDM de prueba	73
Figura 5.17. Espectro de la señal OFDM de prueba y de los armónicos generados por el USRP	74
Figura 5.18. Espectro de la señal OFDM de prueba obtenido a través del USRP	75
Figura 5.19. Espectro de la réplica de la señal OFDM de prueba obtenido a través del USRP.....	76
Figura 5.20. Verificación de la aplicación de análisis de señales en configuración de bucle	77
Figura 5.21. Análisis de la señal de prueba mediante la configuración en bucle	78
Figura 5.22. Entorno integrado de desarrollo Eclipse con el plugin OSSIE OWD instalado.....	79
Figura 5.23. Diseñador de interfaces de usuario Glade.....	81

LISTA DE ACRÓNIMOS

AC	<i>Alternating Current</i> [Corriente Alterna]
ADC	<i>Analog to Digital Converter</i> [Conversor Análogo a Digital]
AEP	<i>Application Environment Profile</i> [Perfil de Entorno de Aplicación]
AM	<i>Amplitude Modulation</i> [Modulación de Amplitud]
AWGN	<i>Additive White Gaussian Noise</i> [Ruido Blanco Gaussiano Aditivo]
CDMA	<i>Code Division Multiplex Access</i> [Acceso Múltiple por División de Código]
CF	<i>Core Framework</i> [Framework Principal]
CIC	<i>Cascaded Integrator Comb (Filter)</i> [[Filtro] Integrador en Peine conectado en Cascada]
CORBA	<i>Common Object Request Broker Architecture</i> [Arquitectura Común de Intermediarios en Peticiones a Objetos]
CORDIC	<i>Coordinate Rotation Digital Computer</i> [Calculador por Rotación Digital de Coordenadas]
CP	<i>Cyclic Prefix</i> [Prefijo Cíclico]
cPCI	<i>Compact PCI</i> [PCI Compacto]
CPU	<i>Central Processing Unit</i> [Unidad de Procesamiento Central]
CRC	<i>Cyclic Redundancy Check</i> [Verificación por Redundancia Cíclica]
DAC	<i>Digital to Analog Converter</i> [Conversor Digital a Analógico]

DC	<i>Direct Current</i> [Corriente Directa]
DCD	<i>Device Configuration Descriptor</i> [Descriptor de Configuración de Dispositivo]
DDC	<i>Digital Down Converter</i> [Conversor Digital a Frecuencias Bajas]
DFT	<i>Discrete Fourier Transform</i> [Transformada Discreta de Fourier]
DMD	<i>Domain Manager Configuration Descriptor</i> [Descriptor de Configuración del Administrador de Dominio]
DP	<i>Domain Profile</i> [Perfil de Dominio]
DPD	<i>Device Package Descriptor</i> [Descriptor de Paquete de Dispositivos]
DSP	<i>Digital Signal Processor</i> [Procesador Digital de Señales]
DUC	<i>Digital Up Converter</i> [Conversor Digital a Frecuencias Altas]
DVB-T	<i>Digital Video Broadcasting – Terrestrial</i> [Difusión de Video Digital - Terrestre]
EAP	<i>Environment Application Profile</i> [Perfil de Entorno de Aplicación]
FDMA	<i>Frequency Division Multiple Access</i> [Acceso Múltiple por División de Frecuencias]
FFT	<i>Fast Fourier Transform</i> [Transformada Rápida de Fourier]
FIR	<i>Finite Impulse Response</i> [Respuesta Finita al Impulso]
FM	<i>Frequency Modulation</i> [Modulación de Frecuencia]
FPGA	<i>Field Programmable Gate Array</i> [Arreglo de Compuertas Lógicas Programable]

GNOME	<i>GNU Network Object Model Environment</i> [Entorno de Modelos de Objetos en Red]
GTK	<i>Gimp Tool Kit</i> [Conjunto de Herramientas Gimp]
HBF	<i>Half Bandpass Filter</i> [Filtro de Mitad de Banda]
I2C	<i>Inter Integrated Circuit Bus</i> [Bus de comunicación entre Circuitos Integrados]
I/O	<i>In / Out</i> [Entrada / Salida]
I/Q	<i>In Phase / In Quadrature</i> [En Fase / En Cuadratura]
IDE	<i>Integrated Development Environment</i> [Entorno Integrado de Desarrollo]
IDL	<i>Interface Definition Language</i> [Lenguaje de Definición de Interfaz]
IEEE	<i>Institute of Electrical and Electronic Engineers</i> [Instituto de Ingenieros Eléctricos y Electrónicos]
IF	<i>Intermediate Frequency</i> [Frecuencia Intermedia]
IIF	<i>Infinite Impulse Response</i> [Respuesta Infinita al Impulso]
ISI	<i>Inter Symbol Interference</i> [Interferencia entre símbolos]
JTRS	<i>Joint Tactical Radio System</i> [Programa Conjunto de Sistemas Tácticos de Radio]
LNA	<i>Low Noise Amplifier</i> [Amplificador de Bajo Ruido]
LTE	<i>Long Term Evolution</i> [Evolución a Largo Plazo]
LTI	<i>Linear Time Invariant</i> [Lineal e Invariante en el Tiempo]

NCO	<i>Numerically Controlled Oscillator</i> [Oscilador Controlado Numéricamente]
OE	<i>Operating Environment</i> [Entorno Operativo]
OFDM	<i>Orthogonal Frequency Division Multiplexing</i> [Multiplexación por División de Frecuencias Ortogonales]
OS	<i>Operating System</i> [Sistema Operativo]
OSSIE	<i>Open Source SCA Implementarion Embedded</i> [Implementación Empotrada de Código Abierto para SCA]
OWD	<i>OSSIE Waveform Developer</i> [Desarrollador de Aplicaciones de Onda OSSIE]
PA	<i>Power Amplifier</i> [Amplificador de Potencia]
PCI	<i>Peripheral Component Interconnect</i> [Interconexión de Componentes Periféricos]
PD	<i>Profile Descriptor</i> [Descriptor del Perfil]
PGA	<i>Programmable Gain Amplifier</i> [Amplificador de Ganancia Programable]
PM	<i>Phase Modulation</i> [Modulación de Fase]
POSIX	<i>Portable Operating System Interface</i> [Interfaz para Sistemas Operativos Portables]
PRF	<i>Property File</i> [Archivo de Propiedades]
RBW	<i>Resolution Bandwidth</i> [Ancho de Banda de Resolución]
RF	<i>Radio Frequency</i> [Radio Frecuencia]
SAD	<i>Software Assembly Descriptor</i> [Descriptor de Ensamblado de Software]

SCA	<i>Software Communications Architecture</i> [Arquitectura de Software de Comunicaciones]
SCARI	<i>Software Communications Architecture Reference Implementation</i> [Implementación de Referencia de la Arquitectura de Software de Comunicaciones]
SCD	<i>Software Component Descriptor</i> [Descriptor de Componente de Software]
SDR	<i>Software Defined Radio</i> [Radio Definido por Software]
SMA	<i>Subminiature Version A Connector</i> [Conector Subminiatura Versión A]
SPD	<i>Software Package Descriptor</i> [Descriptor de Paquete de Software]
SPI	<i>Serial Pheripheral Interface</i> [Interfaz Serie para Periféricos]
TDMA	<i>Time Division Multiple Access</i> [Acceso Múltiple por División de Tiempo]
UHF	<i>Ultra High Frequency</i> [Frecuencia Ultra Alta]
UML	<i>Unified Modeling Language</i> [Lenguaje de Modelado Unificado]
UMTS	<i>Universal Mobile Telecommunications System</i> [Sistema Universal de Telecomunicaciones Móviles]
USB	<i>Universal Serial Bus</i> [Bus Serie Universal]
USRP	<i>Universal Software Radio Peripheral</i> [Periférico Universal para Radio Definido por Software]
VEM	<i>Vector Error Magnitude</i> [Magnitud del Vector de Error]
VHDL	<i>Very High Speed Integrated Circuit Hardware Description Language</i> [Lenguaje de Descripción de Hardware para Circuitos Integrados de Muy Alta Velocidad]
VHF	<i>Very High Frequency</i> [Frecuencia Muy Alta]

VXI	<i>VME Extensions for Instrumentation</i> [Extensión de VME para Instrumentación]
WIMAX	<i>Worldwide Interoperability for Microwave Access</i> [Interoperabilidad Mundial para Acceso por Microondas]
XML	<i>Extensible Markup Language</i> [Lenguaje de Marcas Extensible]

INTRODUCCIÓN

El motivo que dio origen a este trabajo consiste en la falta de estandarización de las plataformas de software que se utilizan en los dispositivos de análisis de señales; las únicas alternativas que ofrecen los proveedores en la actualidad son soluciones propietarias diseñadas específicamente para el hardware de los instrumentos que ellos mismos producen, de tal manera que la personalización del software de estos dispositivos se reduce a la capacidad de actualizar el dispositivo con los paquetes que el fabricante distribuye con el fin de agregar capacidades específicas para el análisis de determinadas señales.

Además de la falta de flexibilidad, el costo de los instrumentos de análisis de señales es bastante elevado y se constituye como un factor que reduce la posibilidad de realizar muchos proyectos en los que se requiere un dispositivo para el análisis de señales de comunicaciones y el presupuesto disponible es limitado. En la búsqueda de una solución alternativa de bajo costo se llegó a la conclusión de que existen plataformas de hardware genéricas, diseñadas especialmente para aplicaciones de SDR ("*Software Defined Radio*"), que podrían utilizarse para la implementación de dispositivos de análisis de señales y también, que hace falta construir una plataforma de software que opere sobre la plataforma de hardware y que implemente los algoritmos de procesamiento de señales necesarios, además de controlar todos los aspectos relacionados con la operación del dispositivo incluidos los mecanismos de interacción con el usuario.

Para hacer la implementación del software del dispositivo de análisis de señales de forma económica y eficiente, se requiere utilizar herramientas y librerías de programación de código abierto en la implementación y una arquitectura bien definida en el diseño, de tal manera que se proporcionen los mecanismos para facilitar el reuso y la interoperabilidad de los componentes de software y se brinde la flexibilidad y escalabilidad necesarias para este tipo de aplicaciones.

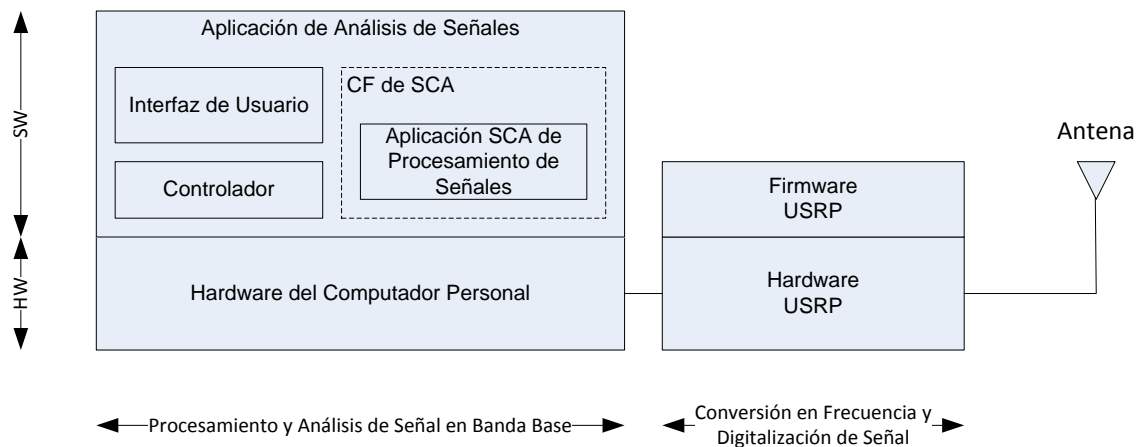


Figura 1. Esquema general de la solución para implementar un dispositivo para el análisis de señales

La Figura 1 muestra el diagrama general de la estructura de la solución que se propone para la implementación del prototipo de un dispositivo para el análisis de señales de comunicaciones inalámbricas. El hardware de la solución está conformado por un computador personal, encargado

de realizar el procesamiento de señales en banda base y el dispositivo de hardware USRP (*“Universal Software Radio Peripheral”*) [1], responsable de la recepción, adecuación, traslación a banda base y digitalización de la señal.

La parte de software está compuesta por diferentes componentes que cumplen funciones específicas: La interfaz de usuario tiene por objeto presentar la información obtenida del procesamiento de la señal y proporcionar los mecanismos de interacción con el usuario. El controlador es el elemento que se encarga de coordinar el funcionamiento de los diferentes componentes del sistema, para proporcionar al usuario un instrumento con las prestaciones y funcionalidades indicadas.

Finalmente la aplicación de procesamiento de señales se ha implementado utilizando la arquitectura SCA (*“Software Communications Architecture”*) [2] e implementa los algoritmos de procesamiento digital necesarios para realizar el análisis de la señal. Pueden coexistir diferentes implementaciones de la aplicación SCA para el análisis de diferentes señales de comunicaciones. El framework SCA proporciona el entorno de operación que define la arquitectura SCA, necesario para la ejecución de las aplicaciones de procesamiento de señales.

Este trabajo se concentra en la parte de la arquitectura del software, exponiendo la conveniencia y utilidad de SCA en el diseño e implementación del software para dispositivos de análisis de señales de comunicaciones inalámbricas. Para demostrar tal utilidad es necesario exponer las ventajas y definir las implicaciones que surgen de la aplicación, en el campo de los dispositivos de instrumentación, de una arquitectura que originalmente fue diseñada para sistemas de comunicaciones [3].

La complejidad de una aplicación de análisis de señales puede incrementarse de acuerdo a la cantidad de parámetros que se quieran obtener de la señal, puesto que este trabajo se enfoca en la arquitectura del software más que en su implementación, se ha desarrollado un prototipo sencillo cuyo software permite visualizar cualitativamente el espectro de una señal OFDM, el diagrama de dispersión de los símbolos y la forma de onda en el dominio del tiempo de la señal en banda base, permitiendo también calcular cuantitativamente algunos parámetros como la potencia de la señal, el BLER (*“Block Error Rate”*) y el EVM (*“Error Vector Magnitude”*).

El contenido de este documento está organizado en seis capítulos que exponen los diferentes aspectos que se abordaron durante el desarrollo del trabajo. Debido a que el trabajo de grado propuesto involucra diversos temas, para conservar la homogeneidad del contenido, se decidió exponer cada tema en capítulos independientes que inician con una breve introducción, luego presentan un resumen de los fundamentos teóricos y posteriormente los diferentes aspectos relacionados con la implementación o integración de los componentes que conforman la solución.

A continuación se hace una breve descripción del contenido de cada uno de los capítulos y se especifica cuál es el objeto de cada uno y su relación con los diferentes elementos que conforman la solución propuesta al inicio de esta sección.

En el Capítulo 1 se hace una descripción del diseño de los dispositivos de análisis de señales disponibles en el mercado, como los analizadores de espectro y los analizadores vectoriales de señales. Se explica en qué se diferencian y cuáles son las ventajas de cada uno estos instrumentos

y también se hace una descripción de los fundamentos teóricos que rigen su operación. El contenido de este capítulo es importante para apreciar cual es el nivel de desarrollo de las soluciones disponibles comercialmente y para identificar cómo los diferentes proveedores han utilizado técnicas de radio definido por software para su implementación.

El Capítulo 1 introduce formalmente al lector en las tecnologías de radio definido por software y cómo estas tecnologías se pueden utilizar en la implementación de las plataformas de hardware para aplicaciones de análisis de señales de comunicaciones. Luego se describe la plataforma de hardware USRP que se ha utilizado para la implementación del prototipo propuesto en el trabajo y cómo se puede utilizar este dispositivo en el desarrollo e implementación de una solución para el análisis de señales. Finalmente en este capítulo se exponen las ventajas y desventajas de utilizar el hardware USRP y se mencionan algunas plataformas de hardware alternativas, con mejores especificaciones que se podrían utilizar para el desarrollo de instrumentos para el análisis de señales.

SCA es una arquitectura diseñada para la implementación de software para aplicaciones de comunicaciones sobre plataformas SDR. En el Capítulo 1 se exponen las características fundamentales de SCA y describe como se ha implementado esta arquitectura en el proyecto OSSIE ("*Open Source SCA Implementation Embedded*") [4]. Además se exponen los beneficios y las implicaciones de utilizar la arquitectura SCA así como las ventajas y desventajas de utilizar OSSIE como plataforma para el diseño e implementación de dispositivos de análisis de señales.

Al final del Capítulo 1 también se describe el diseño del software de la aplicación de análisis de señales que se desarrolló durante este trabajo y que en conjunto con el hardware USRP conforman el prototipo de análisis de señales. En el diseño se definen los mecanismos de interacción entre el controlador, la interfaz de usuario y la aplicación de procesamiento de señales, sin embargo el diseño no especifica una aplicación de análisis de señales específica para una tecnología y asume la existencia de una o más implementaciones de la interfaz de usuario y de la aplicación de procesamiento de señales. Finalmente se describen los beneficios y desventajas de la aproximación del diseño del software propuesto.

El software de la aplicación de análisis de señales se ha diseñado de forma genérica; para demostrar que el diseño se puede ajustar a una aplicación específica, se han implementado los diferentes componentes de la interfaz de usuario que presentan los resultados del análisis de una señal OFDM, los componentes modelo, vista y controlador y también una aplicación de procesamiento de señales de acuerdo con las especificaciones de la arquitectura SCA.

En el Capítulo 1 se exponen los principios de operación de OFDM y el modelo de bloques de la aplicación de procesamiento de señales que se ha implementado para realizar el análisis de señales OFDM. Finalmente se describen los componentes específicos de la interfaz de usuario y de la aplicación SCA de procesamiento de señales y los componentes que implementan las funciones de procesamiento de señal, específicas para el análisis de señales OFDM.

El Capítulo 1 describe el proceso de verificación del prototipo de la aplicación de software para el análisis de señales según los objetivos planteados y presenta los resultados obtenidos. El capítulo también describe el proceso de desarrollo del software, las herramientas que se utilizaron y las

ventajas e inconvenientes derivados de la utilización de cada una de ellas durante el proceso de desarrollo.

Finalmente se presentan las conclusiones obtenidas durante el desarrollo de este trabajo y se hacen las recomendaciones pertinentes para quienes estén interesados en desarrollar proyectos que continúen bajo la misma línea de investigación o desarrollen temas relacionados.

La información específica de la implementación del software para el análisis de señales OFDM, el código fuente, el manual de usuario y las herramientas de software necesarias para el desarrollo de la aplicación se encuentran disponibles en los discos proporcionados como anexos para este trabajo de grado.

1. FUNDAMENTOS DEL ANÁLISIS VECTORIAL DE SEÑALES

Las mediciones de RF (“*Radio Frequency*”) pueden clasificarse en dos categorías diferentes: medición de señales y medición de las propiedades de redes eléctricas. La medición de señales determina las características de una señal determinada como amplitud, potencia, distorsión y mecanismo de modulación entre otras. La medición de parámetros de redes eléctricas determina las características de transferencia de señales en dispositivos o sistemas eléctricos con cualquier número de puertos. Si bien los dos tipos de dispositivos son similares en algunos aspectos, el objeto de este capítulo consiste en realizar una descripción de los dispositivos de medición y análisis de señales, de tal manera que el enfoque en adelante se concentra en este tipo de instrumentos.

1.1 SEÑALES

En el ámbito de las telecomunicaciones, una señal se puede definir como una cantidad eléctrica fluctuante, por ejemplo, voltaje, corriente o intensidad de campo, cuyas variaciones dependen del tipo de modulación y de la información codificada. Una señal posee diferentes características como amplitud, frecuencia, fase, potencia, modulación, ancho de banda y puede clasificarse según diferentes criterios como señal analógica o digital, continua o discreta en el tiempo, periódica o no periódica, determinística o aleatoria.

1.1.1 El dominio del tiempo y el dominio de la frecuencia

La forma más intuitiva de observar una señal corresponde a su representación en el dominio del tiempo en donde se registra la magnitud de la señal respecto al tiempo de la misma forma que lo hace un osciloscopio.

En los sistemas LTI (“*Linear Time Invariant*”) se puede obtener la respuesta del sistema ante una señal arbitraria $x(t)$ a partir de la función de respuesta al impulso $h(t)$. Si se considera la señal de entrada como una sucesión infinita de impulsos desplazados en el tiempo, entonces por el principio de linealidad y superposición, la respuesta $y(t)$ del sistema estará conformada por una sucesión infinita de instancias superpuestas de la señal $h(t)$ desplazadas en el tiempo como se aprecia en la Ecuación 1.1 llamada integral de convolución [5].

$$y(t) = x(t) * h(t) = \int_{-\infty}^{\infty} x(\tau)h(t - \tau)d\tau \quad (1.1)$$

La teoría de Fourier establece que cualquier señal puede generarse a partir de la suma de diferentes componentes sinusoidales con amplitud, frecuencia y fase adecuadas, por lo tanto las señales se pueden representar en el dominio de la frecuencia y a esta representación se le denomina espectro de frecuencia. Este espectro se puede interpretar como la cantidad de energía de cada una de las componentes sinusoidales que conforman la señal [6].

La respuesta estacionaria de un sistema LTI ante una señal sinusoidal es la misma senoide con un cambio en su amplitud y fase, de tal manera que la respuesta $Y(f)$ a cualquier señal arbitraria $X(f)$ puede considerarse como la superposición de las repuestas a cada componente sinusoidal de la señal de entrada. Esta aproximación conduce a la Ecuación 1.2 en donde $H(f)$ corresponde a la representación en el dominio de la frecuencia de la función $h(t)$.

$$Y(f) = X(f) \cdot H(f) \tag{1.2}$$

1.1.2 Relación entre el dominio del tiempo y el dominio de la frecuencia

En la Figura 1.1 se muestra de forma simple la relación entre los dominios del tiempo y la frecuencia. Los dos primeros ejes corresponden a la representación en el dominio del tiempo e indican la amplitud instantánea de una señal respecto al tiempo, el tercer eje corresponde a la frecuencia de cada una de las componentes sinusoidales que conforman la señal y que en conjunto con el eje de amplitud corresponden a la representación de la misma en el dominio de la frecuencia.

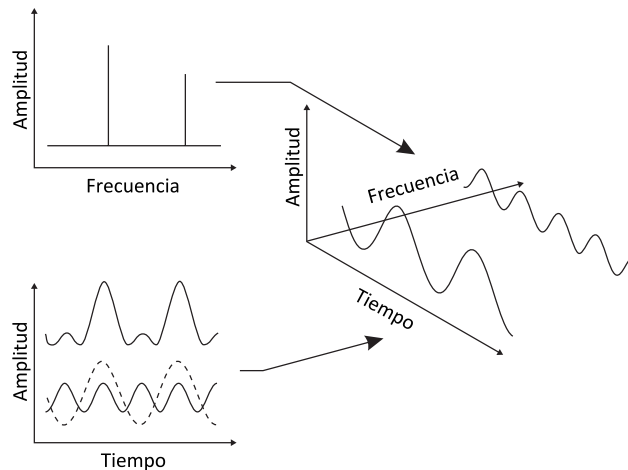


Figura 1.1. Relación entre el dominio del tiempo y el dominio de la frecuencia

Cada uno de los dominios mencionados contiene información completa acerca de la señal y los dos están relacionados unívocamente entre sí por medio de las Ecuaciones 1.3 y 1.4 en donde $x(t)$ es la señal en el dominio del tiempo, $X(f)$ es la señal en el dominio de la frecuencia, $F\{x(t)\}$ representa la transformada de Fourier y $F^{-1}\{X(f)\}$ representa la transformada inversa de Fourier [7].

$$X(f) = F\{x(t)\} = \int_{-\infty}^{\infty} x(t) \cdot e^{-j2\pi ft} dt \tag{1.3}$$

$$x(t) = F^{-1}\{X(f)\} = \int_{-\infty}^{\infty} X(f) \cdot e^{j2\pi ft} df \tag{1.4}$$

Las Series de Fourier y la Transformada Discreta de Fourier corresponden a casos específicos de la transformada de Fourier para señales periódicas y para señales en tiempo discreto respectivamente.

1.2 EL ANALIZADOR DE SEÑALES

Aunque el dominio del tiempo es la forma de representación más simple, el dominio de la frecuencia permite realizar algunos análisis de especial interés en el ámbito de las comunicaciones inalámbricas, por ejemplo la determinación del ancho de banda, análisis de armónicos e intermodulaciones.

En una gran cantidad de mediciones solamente se requiere conocer la amplitud de cada componente espectral sin necesidad de conocer las relaciones de fase presentes entre cada una de ellas y se les denomina análisis escalar de señales. Sin embargo en algunas ocasiones las mediciones requieren que se preserve la información completa acerca de la frecuencia, amplitud y fase de cada una de las componentes sinusoidales que conforman la señal y estas mediciones se denominan análisis vectorial de señales [8].

1.2.1 El analizador de espectro

La Teoría de Fourier define que cualquier señal está compuesta por una o más señales sinusoidales que se pueden descomponer mediante un conjunto de filtros adecuados, los cuales evalúan independientemente la amplitud y fase de cada componente. Existen varias aproximaciones que utilizan este principio para realizar el análisis espectral de señales; a continuación se describen las dos implementaciones más importantes, el analizador de espectro superheterodino y el analizador basado en la transformada rápida de Fourier o FFT [9].

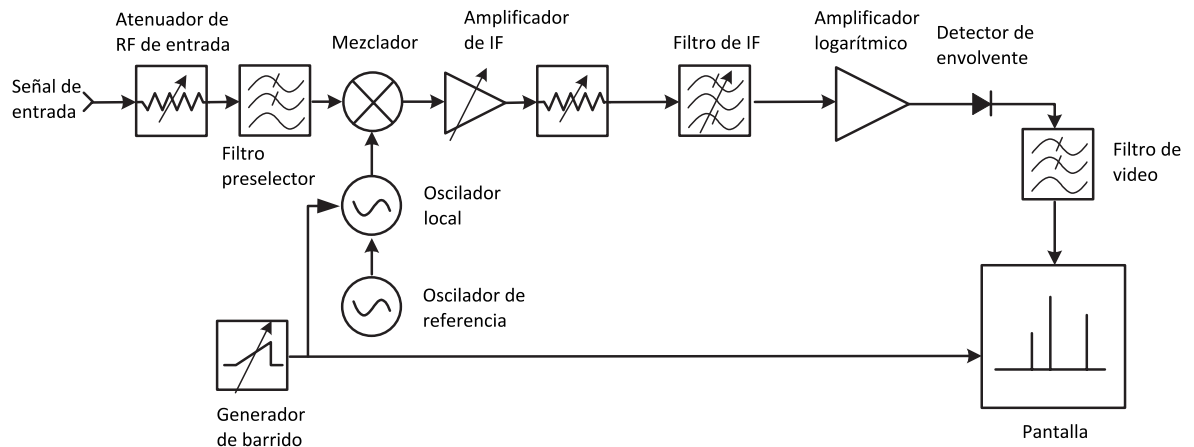


Figura 1.2. Analizador de espectro superheterodino

La implementación más común de un analizador de espectro funciona bajo el principio del receptor superheterodino como el que se observa en la Figura 1.2. Por medio de la variación de la frecuencia del oscilador local y con ayuda del mezclador, se puede barrer el rango completo de

frecuencias de interés convirtiéndolas en una señal de frecuencia intermedia constante que posteriormente es filtrada y su potencia puede ser medida por medio de un detector de envolvente. Puesto que el filtro de frecuencia intermedia determina el RBW (*“Resolution Bandwidth”*), se puede implementar un banco de filtros con diferentes anchos de banda para realizar mediciones con diferentes resoluciones. El amplificador logarítmico comprime la escala de la señal para facilitar la visualización simultánea de señales con gran diferencia de niveles y el filtro de video suaviza la señal que entrega el detector antes de su visualización sobre un tubo de rayos catódicos [11].

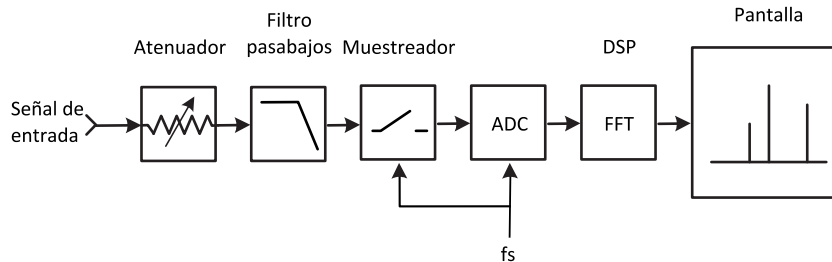


Figura 1.3. Analizador de espectro basado en la Transformada Rápida de Fourier

La Figura 1.3 muestra el diagrama de un analizador de espectro basado en la FFT (*“Fast Fourier Transform”*), el cual utiliza los principios descritos en la Sección 1.1.2 para obtener la representación espectral de una señal. El filtro pasabajos elimina las frecuencias altas de la señal de entrada para prevenir el *aliasing* durante la digitalización de la señal. El DSP (*“Digital Signal Processor”*) recibe una representación digital de la señal de entrada y calcula el espectro de un segmento de esta señal utilizando el algoritmo de la Transformada Rápida de Fourier.

1.2.2 El analizador vectorial de señales

Aunque los analizadores de espectro heterodinos operan en un gran rango de frecuencias, no pueden determinar la fase de cada una de las componentes espectrales de la señal y aunque los analizadores basados en la FFT utilizan tecnologías de procesamiento digital de señales para proporcionar información completa acerca de la magnitud y fase de cada una de las componentes espectrales de la señal, existe un límite en la frecuencia máxima de la señal de entrada debido a la velocidad limitada del ADC (*“Analog to Digital Converter”*) y del DSP. El analizador vectorial de señales combina la tecnología de receptor superheterodino con el hardware de procesamiento digital de señales para lograr mediciones espectrales con la velocidad y la precisión que permiten las técnicas de procesamiento digital de señales y con un rango de operación en frecuencia comparable al de un analizador superheterodino [12].

Como se observa en la Figura 1.4 el analizador vectorial de señales desarrolla una aproximación diferente a la del analizador superheterodino tradicional en donde se ha remplazado la sección de frecuencias intermedias por un hardware digital que incorpora diferentes algoritmos de procesamiento digital de señales para realizar la adecuación, demodulación y análisis de las señales. El algoritmo FFT se utiliza para el análisis espectral de la señal, mientras los algoritmos de adecuación y demodulación son utilizados por las funciones de análisis vectorial y análisis de modulación.

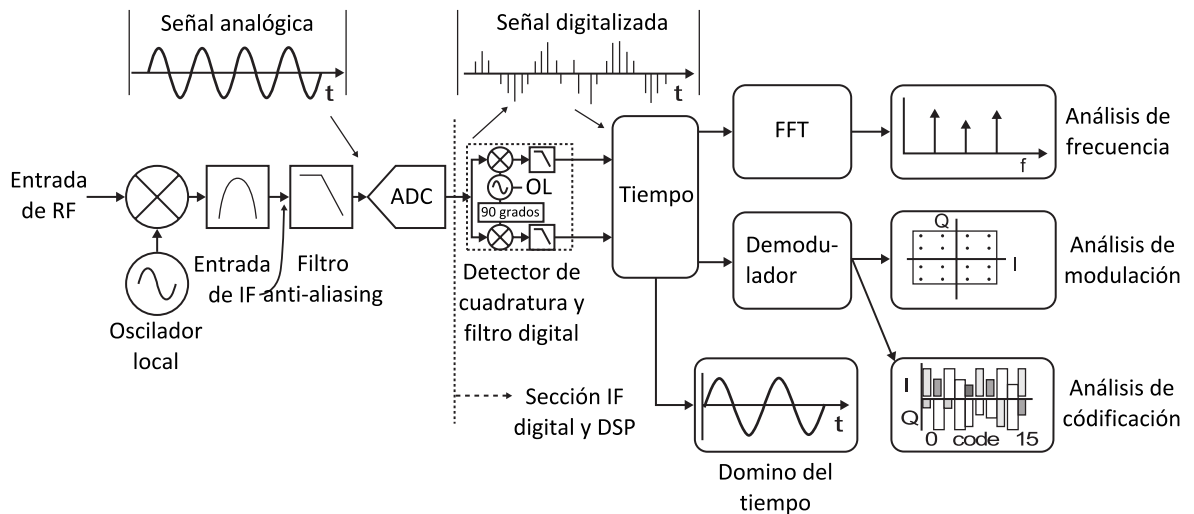


Figura 1.4. Analizador vectorial de señales

Una característica significativa del analizador vectorial de señales es que está diseñado para medir y manipular datos en el dominio de los números complejos, de hecho se denomina analizador vectorial de señales porque tiene la capacidad de medir la componente real e imaginaria de una señal sinusoidal o lo que es equivalente determinar su magnitud y fase. La implementación del analizador vectorial es similar, pero no idéntica a la de un receptor digital de comunicaciones, puesto que se ha diseñado especialmente para determinar una variedad de parámetros diferentes sobre distintos tipos de señales [12].

1.2.3 Ventajas del analizador vectorial de señales

La principal ventaja del analizador vectorial de señales consiste en que su funcionalidad está determinada por las funciones implementadas en el firmware del instrumento y en la capacidad de procesamiento de su hardware, de tal manera que el dispositivo puede ser actualizado o mejorado sin necesidad de cambiar el hardware subyacente. La flexibilidad que otorgan las técnicas de procesamiento digital de señales permite realizar el análisis espectral de una señal y el análisis de modulación. Después de la demodulación se obtiene la información que transportaba la señal y se pueden realizar análisis de los protocolos correspondientes a los niveles de mayor abstracción del sistema de comunicaciones.

En los analizadores heterodinos el tiempo de barrido es proporcional al RBW, debido principalmente a las limitaciones en la respuesta transitoria del filtro analógico de IF (*“Intermediate Frequency”*). El analizador vectorial de señales utiliza filtros digitales para realizar el análisis de la señal, de tal manera que se puede incrementar la resolución espectral sin incurrir en un aumento apreciable de la velocidad de medición que finalmente depende de la capacidad de procesamiento del DSP y de la estabilidad del oscilador local de referencia [12].

La representación digital de una señal es el formato ideal para su almacenamiento y análisis posterior, sea porque no se ha determinado el análisis requerido o porque este requiere de tal capacidad de procesamiento que solamente se puede realizar fuera de línea. Esta característica en

conjunto con la posibilidad de implementar fácilmente algoritmos de gran complejidad permite realizar el análisis de señales dinámicas en el tiempo como señales transitorias, señales en ráfagas o señales con modulación adaptativa.

Aunque el analizador vectorial de señales proporciona beneficios importantes, los analizadores heterodinos que funcionan bajo el principio del barrido de frecuencias todavía cuentan con algunas ventajas como su menor costo de implementación, gran ancho de banda, cobertura en frecuencias altas y rango dinámico superior, entre otras.

1.3 TEORÍA DE OPERACIÓN DEL ANALIZADOR VECTORIAL DE SEÑALES

A continuación se describe detalladamente cada uno de los principios de funcionamiento que se ponen en práctica en el diseño del analizador vectorial de señales, empezando por la descripción del bloque de heterodinación de frecuencias hasta los diferentes algoritmos de procesamiento digital de señales necesarios para realizar el análisis espectral y el análisis de modulación vectorial de la señal.

1.3.1 El receptor heterodino

Los límites en la frecuencia máxima de la señal de entrada en el ADC y en la capacidad de procesamiento del hardware digital imposibilitan el análisis directo de señales en frecuencias muy elevadas. Una solución consiste en desplazar la señal de RF al rango de frecuencias intermedias o banda base en donde pueden ser digitalizadas y procesadas por el hardware de procesamiento digital. Este desplazamiento se realiza por medio de un receptor heterodino como el que se muestra en la Figura 1.5.

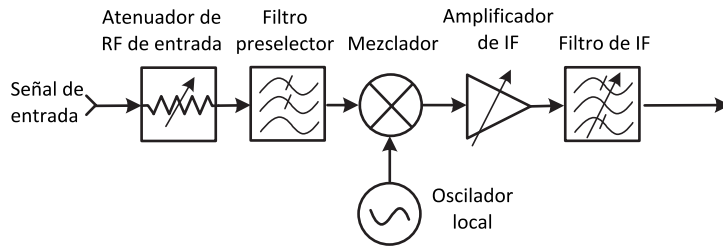


Figura 1.5. Receptor heterodino

Suponiendo que la señal de entrada $v_{RF}(t)$ y la señal del oscilador $v_{LO}(t)$ están determinadas por las funciones de las Ecuaciones 1.5 y 1.6.

$$v_{RF}(t) = A \cos(2\pi f_{RF} t) \quad (1.5)$$

$$v_{LO}(t) = \cos(2\pi f_{LO} t) \quad (1.6)$$

Entonces la señal de salida del mezclador corresponde a la Ecuación 1.7, en donde f_{IF} corresponde a $f_{RF} - f_{LO}$.

$$v_1(t) = \frac{A}{2} [\cos(2\pi(f_{RF} - f_{LO})t) + \cos(2\pi(f_{RF} + f_{LO})t)] \quad (1.7)$$

El filtro de IF se selecciona de tal manera que solamente permite el paso de la primera componente, de tal manera que la señal a la salida está determinada por la Ecuación 1.8.

$$v_{IF}(t) \cong \frac{A}{2} \cos(2\pi(f_{RF} - f_{LO})t) \quad (1.8)$$

1.3.2 Muestreo y digitalización de la señal

El muestreo consiste en multiplicar una señal por la función de muestreo de la Ecuación 1.9 que está formada por una serie de impulsos distribuidos regularmente con periodo T . Suponiendo que $x(t)$ es una señal en el dominio del tiempo, como se representa en la Sección a de la Figura 1.6 y $S(t)$ corresponde a la función de muestreo de la Ecuación 1.1, con $n \geq 0$, como se representa en la Sección b de la Figura 1.6, entonces la señal muestreada se representa en la Sección c de la Figura 1.6 y se describe en la Ecuación 1.10.

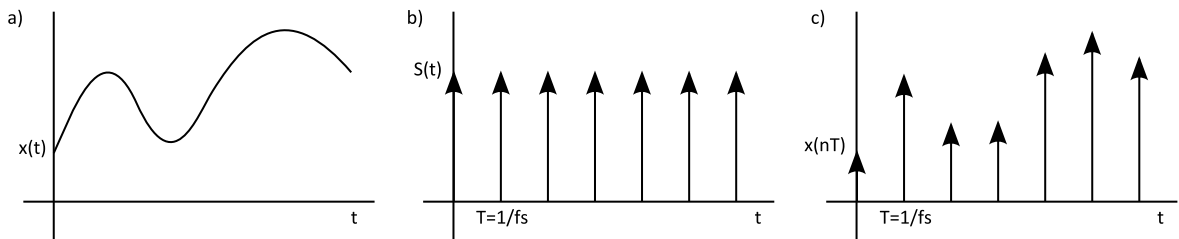


Figura 1.6. Señal en el dominio del tiempo, señal de muestreo y señal muestreada

La función de muestreo y la señal muestreadas se definen por las siguientes ecuaciones [5].

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT) \quad (1.9)$$

$$x(nT) = \sum_{n=-\infty}^{\infty} x(t) \delta(t - nT) \quad (1.10)$$

La señal analógica muestreada se cuantifica mediante un ADC cuya salida consiste de un flujo de números que representan la amplitud de los impulsos de la señal muestreada.

El teorema del muestreo establecido por Nyquist establece que es posible reconstruir una señal continua a partir de sus muestras, siempre y cuando la tasa de muestreo sea superior al doble de la frecuencia máxima de la señal, es decir si se cumple que $f_s > 2f_{MAX}$ [5].

$$x(t) = \sum_{n=-\infty}^{\infty} x(nT)g(t-nT) \quad (1.11)$$

$$g(t) = \frac{\sin(2\pi f_{MAX} t)}{2\pi f_{MAX} t} \quad (1.12)$$

La Ecuación 1.11 muestra la función de reconstrucción definida por el teorema de Nyquist, en donde $T = 1/f_s$ y $g(t)$ es la función de interpolación que se define en la Ecuación 1.12. Si no se cumple con el criterio de Nyquist se presenta el fenómeno de *aliasing* que consiste en que señales con diferentes frecuencias producen la misma señal muestreada haciendo imposible determinar cuál es la frecuencia de la señal que se debe reconstruir.

1.3.3 Detección en cuadratura y diezmado

Después de la digitalización de la señal el procesamiento continúa en forma digital. La primera operación que se realiza sobre la señal en IF es la conversión a banda base por medio de un detector en cuadratura que permite recuperar las componentes en fase y cuadratura o componentes I/Q (*"In Phase/ In Quadrature"*) de la señal en banda base, opcionalmente se puede diezmarse la señal como se muestra en la Figura 1.7.

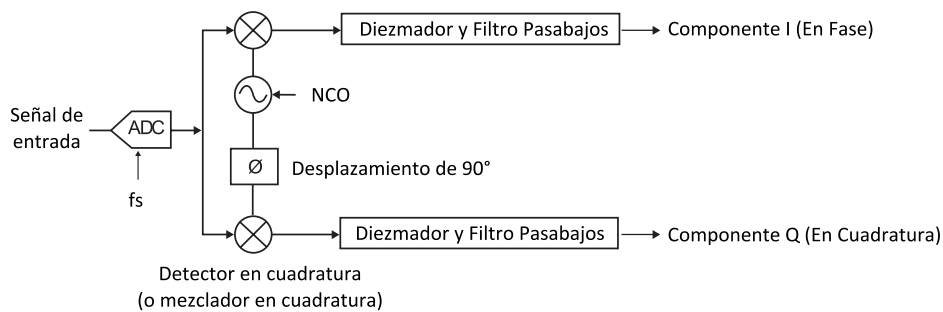


Figura 1.7. Diagrama del detector en cuadratura, diezmador y filtro pasabajos

La conversión de frecuencia intermedia a banda base se realiza en forma similar al receptor heterodino, suponiendo que la señal digitalizada que entrega el ADC es $v_{IF}(n)$, como se define en la Ecuación 1.13.

$$v_{IF}(n) \cong \frac{A}{2} \left[\cos \left(2\pi \frac{f_I}{f_S} n \right) \right] \quad (1.13)$$

En donde la frecuencia intermedia es $f_I = f_{RF} - f_{LO}$, y la frecuencia de muestreo cumple la relación $f_s > 4f_I$ para evitar problemas de *aliasing* en el proceso de conversión a banda base. Si la frecuencia del NCO (*"Numerically Controlled Oscillator"*) es $f = f_I/f_s$, entonces las componentes de salida I y Q del detector en cuadratura, previamente filtradas, corresponden a las Ecuaciones 1.14 y 1.15.

$$v_I(n) \cong A/4 \quad (1.14)$$

$$v_Q(n) \cong 0 \quad (1.15)$$

Que corresponden a los vectores en fase y cuadratura de la señal sinusoidal de la Ecuación 1.13 multiplicados por un factor de $1/2$.

1.3.4 Transformada Rápida de Fourier y análisis espectral de la señal

De la misma forma que el analizador de espectro basado en la FFT, el analizador vectorial de señales también utiliza la Transformada Discreta de Fourier para realizar el análisis espectral de la señal en banda base aplicando la respectiva corrección en frecuencia $\Delta f = f_{RF} = f_I + f_{LO}$ antes de la presentación de los resultados.

La Ecuación 1.17 define la Transformada Discreta de Fourier para una secuencia $x(n)$, si esta secuencia corresponde a las muestras de una señal compleja como se define en la Ecuación 1.16, entonces se requieren $2N^2$ funciones trigonométricas y N^2 multiplicaciones de números complejos para obtener la secuencia $X(k)$ [7].

$$x(n) = I(n) + jQ(n) \quad (1.16)$$

$$X(k) = \sum_{n=0}^{N-1} x(n) \cdot e^{-\frac{j2\pi}{N}nk} \quad (1.17)$$

N es el periodo de la señal y el índice k está limitado por $0 \leq k \leq N-1$. Si N es un número par, mediante el cambio de variable $n = 2m$, la Ecuación 1.17 puede reescribirse como se muestra en la Ecuación 1.18:

$$X(k) = \sum_{m=0}^{N/2-1} x(2m) \cdot e^{-\frac{j2\pi}{N/2}mk} + e^{-\frac{j2\pi}{N}k} \sum_{m=0}^{N/2-1} x(2m+1) \cdot e^{-\frac{j2\pi}{N/2}mk} \quad (1.18)$$

En la Ecuación 1.18 se observa que el primer sumando corresponde a la DFT (“Discrete Fourier Transform”) de una secuencia de muestras con los valores de los elementos de índice par de la señal original más otra DFT de una secuencia con los valores de los elementos de los índices impares multiplicados por un factor que se denomina factor de corrección de fase. Si N es potencia de 2, entonces la Ecuación 1.18 sugiere que se puede continuar descomponiendo la transformada discreta de Fourier de forma iterativa hasta obtener N transformadas simples de un punto que se sintetizan sucesivamente hasta obtener la transformada de la secuencia original [7].

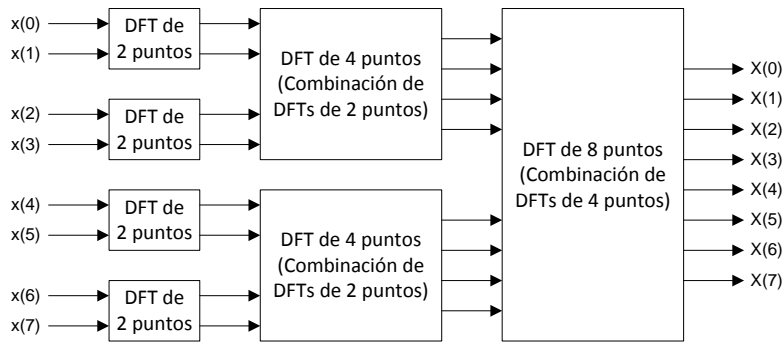


Figura 1.8. Síntesis de una DFT de 8 puntos

La FFT es un algoritmo diseñado para realizar el cálculo eficiente de la DFT utilizando algunas optimizaciones derivadas de las propiedades de la transformada discreta de Fourier expuestas en la Ecuación 1.18, disminuyendo la cantidad de multiplicaciones de números complejos necesarias a $(N/2)\log_2 N$ facilitando así el cálculo de la Transformada Discreta de Fourier.

1.3.5 Las funciones de ventana para la FFT

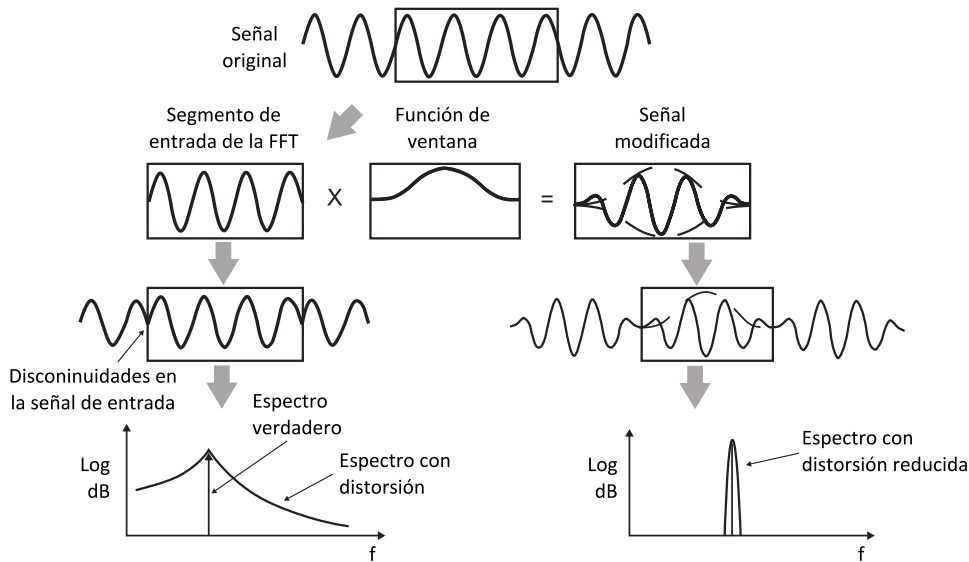


Figura 1.9. Efecto de la función de ventana sobre la FFT

La Transformada Rápida de Fourier de longitud N asume que la señal es periódica, con un periodo de la misma longitud, es decir que $x(n) = x(n + N)$. Cuando esta condición no se cumple el espectro que se obtiene con la FFT no corresponde con el de la señal. Para minimizar este inconveniente se multiplica la señal por una función de ventana que hace que la secuencia de entrada de la FFT se aproxime a cero en los extremos [12]. En la Figura 1.9 se describe este fenómeno y el procedimiento para aplicar la función de ventana sobre una señal.

Existen varias funciones de ventana con características diferentes y diseñadas para diferentes aplicaciones. Entre estas se destacan la función de Hanning, útil para reducir la distorsión en frecuencia, la función flat top, útil para reducir la distorsión en amplitud, y la función exponencial, útil para el análisis espectral de señales transitorias.

1.3.6 Demodulación y análisis vectorial de la señal

El analizador vectorial de señales puede considerarse como un receptor de propósito general, gracias a que el analizador vectorial de señales utiliza muchos elementos y técnicas similares a las que se utilizan para la implementación de receptores digitales, sin embargo el analizador vectorial se ha diseñado para conseguir una precisión significativa en la medición de los parámetros de modulación de la señal.

La Figura 1.10 muestra el diagrama de bloques simplificado de un demodulador digital de propósito general utilizado en un analizador vectorial de señales. Mediante el suministro de los parámetros de configuración adecuados, el demodulador es capaz de decodificar una gran variedad de formatos de modulación. Los óvalos representan parámetros de funcionamiento que el usuario debe proporcionar y los rectángulos redondeados representan parámetros que el usuario puede modificar para un ajuste más preciso.

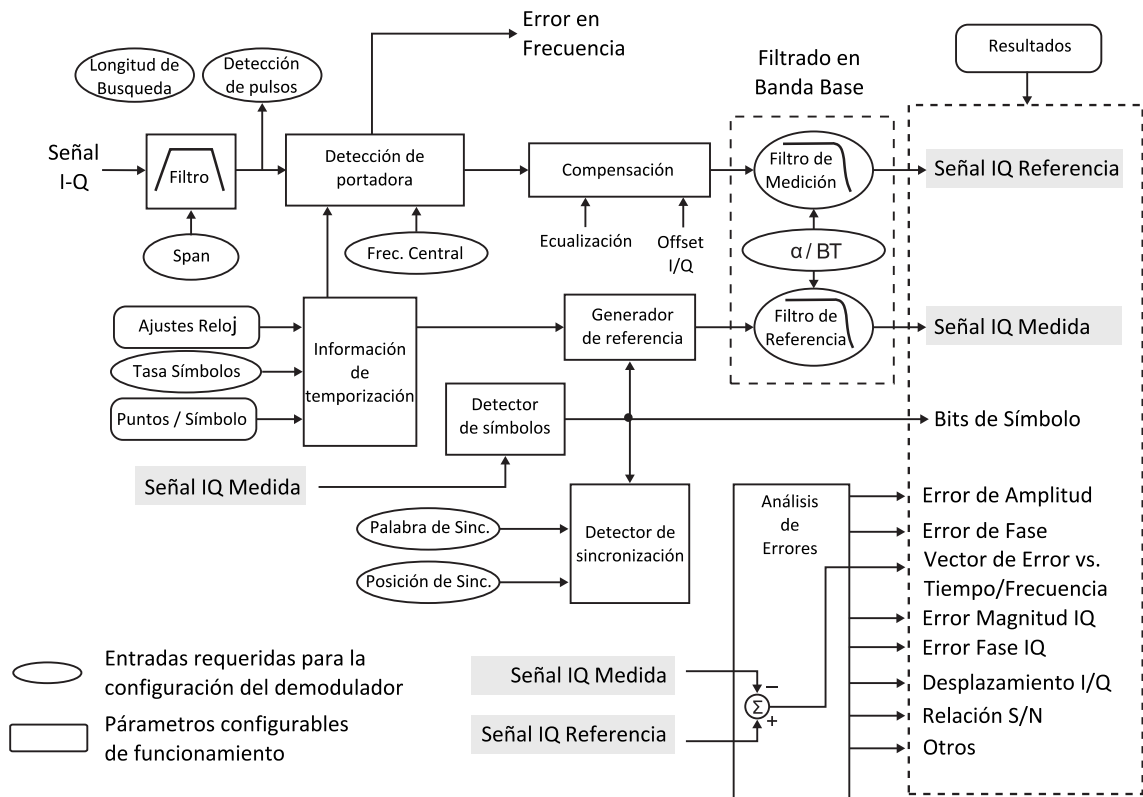


Figura 1.10. Diagrama de bloques de un demodulador de propósito general

La función principal del demodulador de propósito general es obtener los parámetros de señales moduladas en una gran variedad de formatos y comparar dichos parámetros con los de una señal de referencia e identificar y valorar los distintos fenómenos de distorsión presentes sobre la señal medida [12]. El demodulador también tiene como propósito decodificar el flujo de información contenido en la señal con el objetivo de realizar análisis propios de los niveles superiores de abstracción, también llamados análisis de protocolos.

2. RADIO DEFINIDO POR SOFTWARE

SDR (“*Software Defined Radio*”) es un término definido por el Forum SDR y derivado del término *Software Radio* acuñado en 1991 por Joseph Mitola para referirse a los radios con capacidades de reprogramación o reconfiguración [13]. El Forum SDR especifica Radio Definido por Software como la tecnología de transmisión y recepción de radio cuyos principales parámetros de funcionamiento se establecen por medio de software y cuyos aspectos fundamentales de operación pueden reconfigurarse por medio de tal software. En las páginas siguientes se desarrollan los conceptos fundamentales de radio definido por software y también se describe el hardware USRP (“*Universal Software Radio Pheripheral*”) como plataforma para el desarrollo de aplicaciones de SDR.

2.1 CONCEPTOS DE RADIO DEFINIDO POR SOFTWARE

Un radio definido por software es una forma de transmisor/receptor en el que idealmente todos los aspectos de su operación se determinan por medio de software utilizando hardware digital de propósito general como DSPs (“*Digital Signal Processors*”) y FPGAs (“*Field Programmable Field Arrays*”). El paradigma de SDR consiste en remplazar el problema de diseño de hardware de radiofrecuencias por uno más sencillo de diseño de software para procesamiento de señales [14].

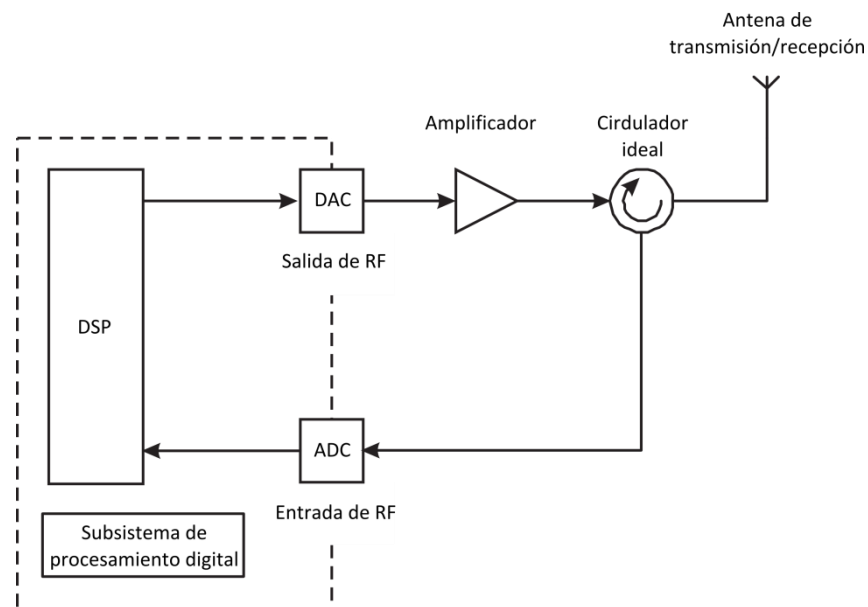


Figura 2.1. Arquitectura ideal del radio definido por software

En la Figura 2.1 se muestra el diagrama simplificado de un radio definido por software ideal. En esencia el concepto de SDR pretende llevar la parte digital de un sistema de radiocomunicaciones lo más cerca posible de la antena. El circulador ideal separa las señales de transmisión y recepción sin importar su frecuencia o ancho de banda y se acopla perfectamente a las impedancias del amplificador y la antena. El amplificador debe asegurar que la señal entregada por el DAC (“*Digital to Analog Converter*”) se transfiere hacia la antena sin distorsión, con una potencia adecuada y sin

emisiones de canal adyacente. De igual modo la antena debe mantener constante su patrón de radiación, su ganancia y su impedancia a lo largo de todo el espectro de radio frecuencias. Los ADC (“Analog to Digital Converter”) y DAC deben proporcionar resoluciones adecuadas para el rango dinámico requerido y deben proporcionar una velocidad suficiente para digitalizar señales incluso en el rango de las microondas. Los filtros ideales tienen una pendiente infinita en su frecuencia de corte y su atenuación fuera de la banda también es infinita para prevenir la distorsión por *aliasing* en el ADC y por rizado en el DAC [15].

La arquitectura que se ha expuesto impone algunas especificaciones imposibles de realizar como el aislamiento perfecto entre las secciones de transmisión y recepción que proporciona el circulator ideal y la velocidad necesaria en los ADC para digitalizar señales directamente en el rango de las radiofrecuencias. En la práctica se realizan diferentes aproximaciones que logran un balance entre costo y complejidad para superar las limitaciones impuestas, el USRP es un ejemplo de la implementación real de un radio definido por software y se describe a continuación.

2.2 EL USRP COMO PLATAFORMA DE RADIO DEFINIDO POR SOFTWARE

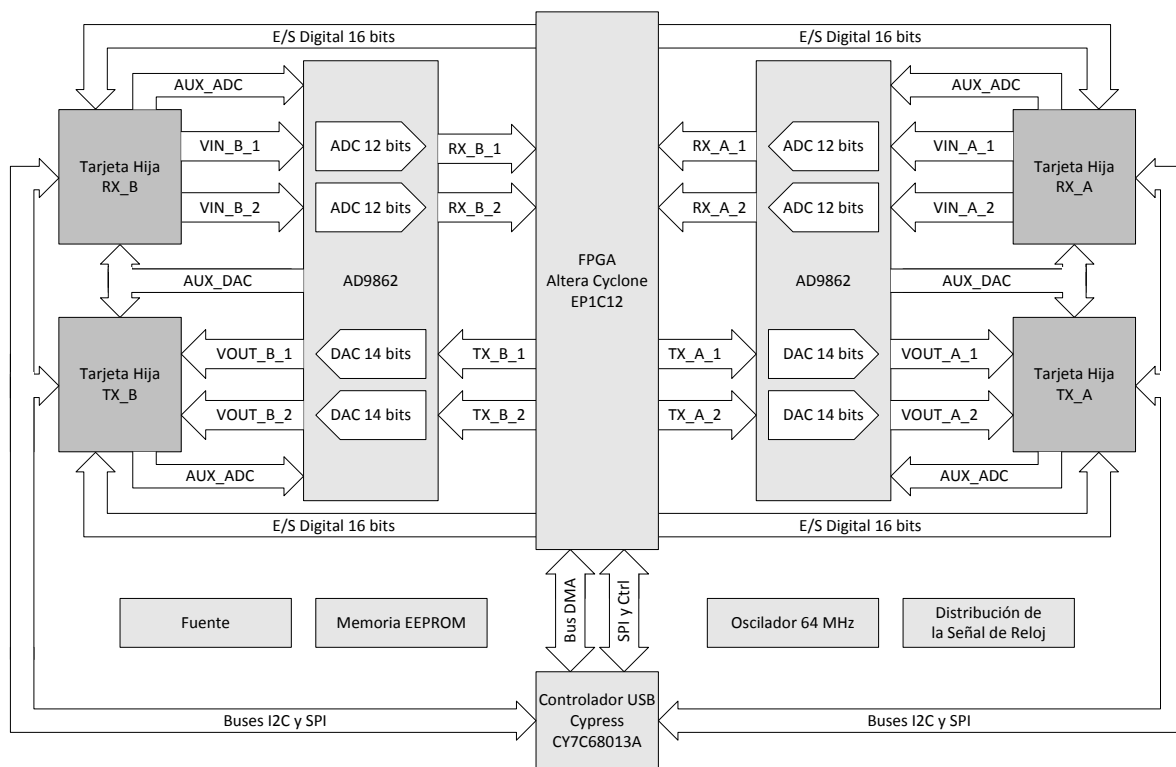


Figura 2.2. Diagrama simplificado del hardware USRP [19]

El USRP es un hardware de bajo costo diseñado para permitir el desarrollo de aplicaciones de SDR utilizando un computador personal. En esencia, el USRP actúa como la sección digital de IF (“Intermediate Frequency”) de un sistema de comunicaciones. La filosofía del diseño divide la implementación de las operaciones de propósito general que requieren una gran capacidad de procesamiento para ser ejecutadas sobre el FPGA de la tarjeta USRP y las operaciones específicas

de la aplicación se implementan sobre la CPU (“*Central Processing Unit*”) del computador anfitrión [19].

Los componentes del USRP se agrupan en dos elementos diferentes: la tarjeta madre y las tarjetas hijas. Las tarjetas hijas contienen diferentes implementaciones de la Interfaz de RF (“*Radio Frequency*”) y la tarjeta madre contiene entre otros elementos la interfaz USB (“*Universal Serial Bus*”), el FPGA, los conversores ADC y DAC de alta velocidad y cuatro puertos para conectar las tarjetas hijas. En la Figura 2.2 se observa el diagrama simplificado del hardware del USRP.

2.2.1 La tarjeta madre

La tarjeta madre contiene cuatro conversores ADC con una frecuencia máxima de muestreo de 64 MHz que permiten digitalizar señales con una frecuencia máxima de 32 MHz, sin embargo en la práctica se elige una frecuencia menor para simplificar la implementación de los filtros *antialiasing*. El rango de los conversores ADC es de 2 Vpp con una impedancia de entrada diferencial de 50 Ω. Antes del ADC se encuentra un amplificador de ganancia programable que se puede fijar por software en un rango de 0 a 20 dB.

En la sección de transmisión hay cuatro conversores DAC que pueden reconstruir señales con una frecuencia máxima de 64 MHz, sin embargo se consideran frecuencias menores para facilitar la implementación del filtro de reconstrucción. El DAC genera un máximo de 1 Vpp diferencial sobre una carga de 50 Ω de impedancia y posee un amplificador de corriente cuya salida puede variar entre 0 y 20 mA.

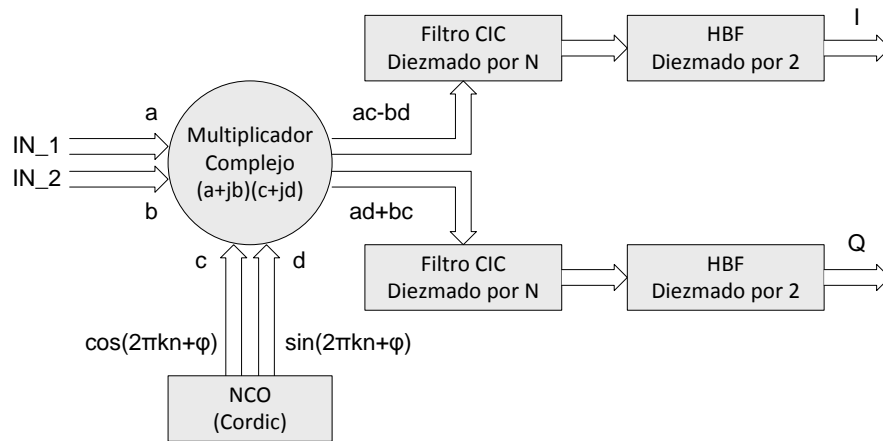


Figura 2.3. Implementación del DDC en el FPGA

El componente más importante de la tarjeta madre es el FPGA, encargado de realizar las operaciones de procesamiento de señales que requieren gran velocidad de procesamiento. En el FPGA se han implementado dos DDCs (“*Digital Down Converter*”) para la ruta de recepción utilizando filtros CIC (“*Cascaded Integrator-Comb*”) de cuatro etapas y un filtro HBF (“*Half Bandwidth Filter*”) para compensar la falta de linealidad característica de los filtros CIC. El objetivo de los DDCs es trasladar la señal de IF a banda base y posteriormente reducir el número de muestras a un valor adecuado para su transmisión sobre la conexión USB con el computador

anfitrión. En la Figura 2.3 se puede observar el diagrama de bloques del DDC implementado en el FPGA.

La conversión en frecuencias de banda base a frecuencias intermedias se realiza en el DUC (“*Digital Up Converter*”) de la sección de transmisión en forma similar al DDC, excepto porque sucede en sentido inverso. El DUC es implementado por el circuito integrado AD9862 dejando al FPGA únicamente la implementación de un filtro CIC interpolador cuyo objetivo es aumentar la velocidad de las muestras que recibe a través de la conexión USB hasta un valor adecuado para el circuito integrado AD9862.

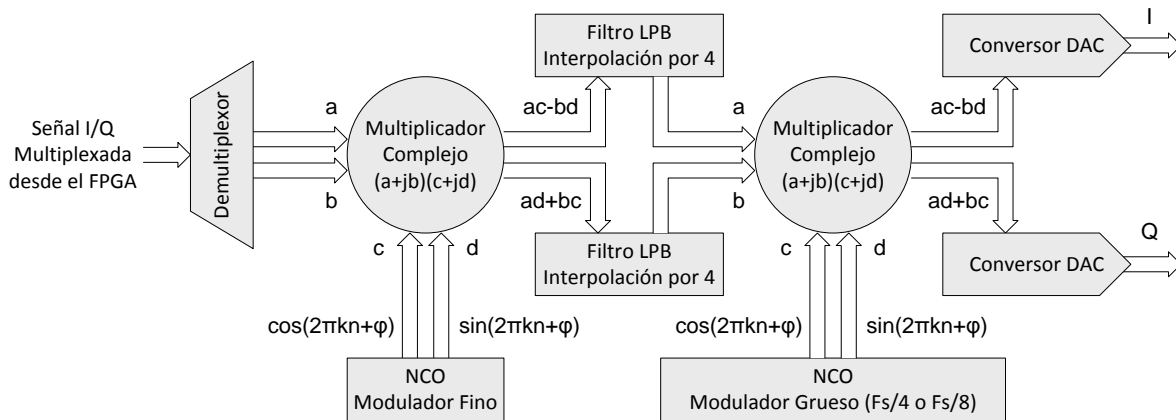


Figura 2.4. Implementación del DUC en el Circuito Integrado AD9862

El USRP controla la conexión USB con el computador anfitrión y puede mantener una conexión con una velocidad de transmisión de datos de 256 Mbps. Puesto que las muestras que se envían por el puerto USB son complejas y tienen una resolución de 16 bits, se pueden transmitir señales digitales con una frecuencia máxima 8 MHz. Dado que la conexión USB funciona en modo *half duplex*, cuando se utiliza el USRP en modo *full* dúplex, la suma de las frecuencias máximas de las señales de recepción y transmisión no deben exceder los 8 MHz para su transmisión simultánea sobre la conexión USB.

2.2.2 Las tarjetas hijas

La tarjeta madre tiene cuatro ranuras de expansión en donde se pueden instalar varias tarjetas hijas. Cada una de las ranuras de expansión tiene acceso a dos convertidores ADC y a dos DAC para permitir la implementación de sistemas con esquemas de modulación y demodulación compleja o en cuadratura, aunque cada convertidor también puede utilizarse como una interfaz independiente para la modulación o demodulación de señales reales.

Referencia	Descripción
Basic Tx/Rx	Los conversores ADC y DAC están acoplados directamente a los conectores de entrada / salida de RF mediante un transformador, sin filtros o amplificadores.
Low Frequency Tx/Rx	Similar a las tarjetas Basic RX/TX, pero los conversores ADC y DAC se acoplan mediante amplificadores diferenciales para extender la respuesta en frecuencia hasta DC y también se han implementado filtros pasabajos para evitar el <i>aliasing</i> .
TVRX	Contiene únicamente un sistema receptor de VHF y UHF completo, basado en un sintonizador de televisión. El rango de frecuencias va desde los 50 hasta los 860 MHz con un ancho de banda de 6 MHz.
DBSRX	Similar a la tarjeta TVRX con un receptor que cubre las frecuencias desde 800 MHz a 2.4 GHz con un ancho de banda variable desde 1 hasta 60 MHz.
RFX	Implementan un sistema de transmisión y recepción <i>full duplex</i> para diferentes bandas como 400, 9000, 1200, 1800 y 2400 MHz. También contienen un interruptor de RF interno para la operación <i>half duplex</i> que permite utilizar una sola antena.

Tabla 2.1. Características de las tarjetas hijas del USRP

2.3 RADIO DEFINIDO POR SOFTWARE EN DISPOSITIVOS DE INSTRUMENTACIÓN PARA ANÁLISIS DE SEÑALES

Los fabricantes de equipos de instrumentación para el análisis de señales de RF deben superar el reto de desarrollar continuamente soluciones para cumplir con las necesidades de sus clientes. El mercado de las telecomunicaciones es aún más desafiante debido a la rápida evolución de las tecnologías y el desarrollo de nuevos estándares. Para mantener este ritmo, los vendedores de equipos de instrumentación han desarrollado nuevas aproximaciones basadas en SDR que permiten reducir el tiempo de desarrollo y adaptarse rápidamente a los nuevos requerimientos [20].

2.3.1 Requerimientos de los sistemas de instrumentación para el análisis de señales

SDR es una aproximación atractiva en aplicaciones que demandan economía y flexibilidad tal como los sistemas de comunicaciones militares, teléfonos móviles y radio bases multiestándar. Estas aplicaciones generalmente comparten las siguientes características básicas:

- Nivel de flexibilidad medio o alto
- Volumen de producción medio o bajo
- Complejidad media o alta

Los equipos de instrumentación de RF también comparten estas características. Se requiere que los equipos de instrumentación sean los suficientemente flexibles para su aplicación en un amplio conjunto de tecnologías diferentes y que brinde además la posibilidad de actualizar el dispositivo y así evitar la necesidad de remplazar los instrumentos de medición cuando surgen nuevos protocolos o tecnologías de comunicaciones.

La complejidad que implica implementar un instrumento de medición compatible con múltiples estándares de comunicaciones y el bajo volumen de producción respecto al volumen de producción de otros dispositivos como teléfonos móviles y radios para enlaces microondas o radio bases hacen de las tecnologías de radio definido por software un aspecto clave para el desarrollo del área de instrumentación para el análisis de señales inalámbricas.

2.3.2 Beneficios de las tecnologías de Radio Definido por Software en los equipos de instrumentación de RF

La aplicación de las tecnologías de radio definido por software en el diseño y fabricación de equipos de instrumentación para el análisis de señales de radiofrecuencias proporciona beneficios para el fabricante de dispositivos y también para el usuario final. La implementación de interfaces de RF de propósito general y la utilización de dispositivos programables facilitan el desarrollo de soluciones con la complejidad y flexibilidad que exigen los sistemas modernos, además estos dispositivos programables permiten que los usuarios actualicen sus instrumentos con una simple actualización del firmware¹ del dispositivo.

El desarrollo del software que implementa las funciones de procesamiento de señales puede apoyarse con las herramientas y metodologías tradicionales que aportan mecanismos de reutilización y estandarización de código, obteniendo así un aumento en la eficiencia del proceso de desarrollo, disminuyendo el tiempo de implementación de nuevos productos y facilitando el soporte para los productos existentes.

La utilización de dispositivos digitales como DSPs y FPGAs permite mejorar significativamente el desempeño de los equipos de instrumentación. Además del área de las comunicaciones inalámbricas, estos dispositivos programables tienen un amplio rango de aplicaciones que habilitan su producción en economía de escala disminuyendo los costos de producción por unidad y por tanto el costo de producción de los instrumentos de análisis de señales.

2.3.3 El USRP como plataforma de Radio Definido por Software para el desarrollo de aplicaciones de instrumentación

Muchos de los parámetros relevantes de un instrumento basado en SDR dependen de la implementación de la interfaz de RF. Aunque la mayoría de los parámetros de diseño del USRP aún no se han documentado, a continuación se realiza un análisis de las características del USRP en conjunto con la interfaz de radio RFX900 y después se enumeran sus implicaciones en el desarrollo de aplicaciones de instrumentación para el análisis de señales inalámbricas.

La Figura 2.5 muestra un diagrama simplificado de la sección de recepción de la tarjeta RFX900, obtenido a partir de su diagrama de interconexiones eléctricas.

¹ Conjunto de instrucciones de programa para propósitos específicos grabando en memoria no volátil que controlan la operación de un dispositivo específico de hardware [21].

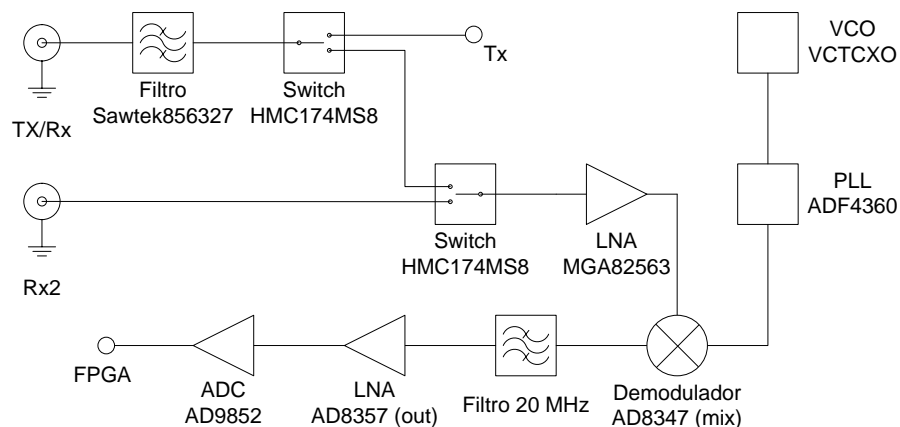


Figura 2.5. Diagrama de la sección de recepción de la tarjeta RFX900

Frecuencia: La mayoría de los analizadores de señales disponibles en el mercado operan en un rango de frecuencias desde los 9 KHz hasta los 3 GHz en el cual se ubican las bandas de mayor interés para la industria de las comunicaciones inalámbricas. Si bien no existe una implementación de la interfaz de RF para el USRP que cubra todo el rango de frecuencias de interés, si existe una implementación para cada una de las bandas más importantes como se describió en la Sección 2.2.2.

Ancho de Banda: En los analizadores de señales el ancho de banda de la señal es importante puesto que existe una relación directa entre la velocidad de los conversores A/D, la capacidad de procesamiento del instrumento y el ancho de banda de la señal que se desea analizar. El USRP tiene dos conversores DAC por canal que, en conjunto con un detector en cuadratura, pueden digitalizar señales con una frecuencia máxima de 64 MHz, sin embargo la conexión USB solamente permite la transmisión de una señal de 8 MHz teniendo en cuenta la capacidad de transmisión del bus de datos y el tamaño de cada muestra de la señal.

Sensibilidad: Definimos la sensibilidad como el mínimo nivel de señal que puede detectar el USRP utilizando el rango completo del conversor ADC. Este parámetro está determinado por el máximo voltaje de entrada del ADC y la ganancia de la interfaz de RF. La potencia máxima a la entrada del ADC es de 40 mW o 16 dBm. La ganancia de la etapa de entrada está determinada por las pérdidas o ganancia de los elementos que la conforman. En la Tabla 2.2 se presenta el resumen de elementos utilizados en la sección de recepción de la tarjeta RFX9000.

Elemento	Ganancia (dB)	Factor de Ganancia	Figura de Ruido (dB)	Factor de Ruido
Filtro SAWTEK 856327	-2.3	5.888E-01	2.3	1.698E+00
Switch HMC174MS8	-0.4	9.120E-01	0.4	1.096E+00
LNA MGA82563	14.0	2.512E+01	2.2	1.660E+00
Demodulador AD8447	69.5	8.913E+06	10.2	1.047E+01
Filtro 20 MHz	-6.0	2.512E-01	6.0	3.981E+00
Conversor A/D AD9852	20.0	1.000E+02	35.9	3.890E+03

Tabla 2.2. Parámetros de los elementos de la sección de recepción de la tarjeta RFX900

Con los valores de la Tabla 2.2 se puede calcular la ganancia total de la sección de recepción que para el caso del conector Rx2 es de 97.1 dB, por lo tanto para utilizar el rango completo del ADC se requiere una señal de entrada con una potencia mínima de -81.1 dBm, sin embargo se pueden detectar señales de menor amplitud con la desventaja de disminuir el rango dinámico y la relación señal a ruido del ADC como se puede observar en las Ecuaciones 2.1 y 2.2 en donde n corresponde al número de bits del ADC utilizados, A corresponde a la amplitud de la señal y A_{\max} corresponde a la amplitud máxima del ADC [22].

$$DR = 6.021dB \cdot n \quad (2.1)$$

$$SNR = 6.021dB \cdot n + 1.763dB - 20 \log \left(\frac{A_{\max}}{A} \right) dB \quad (2.2)$$

Figura de Ruido: La figura de ruido del primer amplificador tiene un efecto significativo sobre la figura de ruido total de la sección de recepción. En la Ecuación 2.3 se describe como se debe calcular el factor de ruido total de una serie de elementos conectados en cascada, en donde las f_n corresponden al factor de ruido y las g_n corresponden a la ganancia de cada elemento en cascada. Con esta ecuación, en conjunto con los valores recopilados en la Tabla 2.2, se puede verificar que la figura de ruido de la sección de recepción para el conector Rx2 es de 3.5 dB, valor similar al de la figura de ruido del primer amplificador.

$$f = f_1 + \frac{f_2 - 1}{g_1} + \frac{f_3 - 1}{g_1 g_2} + \dots + \frac{f_n - 1}{g_1 g_2 \dots g_{n-1}} \quad (2.3)$$

Rango Dinámico: El conversor ADC tiene una resolución de 12 bits que otorga un rango dinámico de 72,25 dB, sin embargo el ruido introducido por las etapas anteriores y por el mismo proceso de digitalización de la señal, disminuye el ENOB (*Effective Number of Bits*) disminuyendo también el rango dinámico de toda la sección de recepción a un valor de 66,99 dB, como se puede observar en las Ecuaciones 2.4 y 2.5, en donde DR y NF corresponden al rango dinámico teórico y figura de ruido respectivamente.

$$ENOB = \frac{DR - NF - 1.763dB}{6.021dB} \quad (2.4)$$

$$DR_{Total} = ENOB \cdot 6.021dB = DR - NF - 1.763dB \quad (2.5)$$

Capacidad de Procesamiento: A partir de la digitalización de la señal, el procesamiento sobre esta se hace mediante algoritmos de procesamiento digital de señales. Dependiendo de la complejidad y la carga computacional que exigen estos algoritmos, es importante asegurar que el hardware digital cuente con la capacidad de procesamiento suficiente para ejecutar los algoritmos de procesamiento. El USRP cuenta con un FPGA de alto desempeño que realiza las tareas genéricas que exigen gran velocidad de procesamiento y en donde están implementados los filtros interpoladores que se aplican a la señal de entrada para disminuir su frecuencia máxima y su velocidad de muestras. Los algoritmos específicos de la aplicación se implementan sobre el

procesador general del computador anfitrión cuyo desempeño es limitado pero es útil en aplicaciones de baja o mediana complejidad.

Los parámetros expuestos anteriormente se han obtenido a partir de un estudio teórico y no se han considerado las pérdidas de los conductores ni fenómenos complejos que distorsionan la señal. Existen algunos parámetros relacionados con las no linealidades de los componentes como la distorsión armónica, los puntos de intersección de intermodulaciones, etc. que no han sido documentados por el fabricante del USRP y tampoco se han obtenido en este análisis puesto que se requerirían procedimientos complejos que desvían la dirección del trabajo de grado de los objetivos principales.

Si se comparan las prestaciones de los analizadores de señales disponibles comercialmente con las características del USRP, se puede verificar que las capacidades del USRP son limitadas en aspectos como el rango de frecuencias de operación, rango dinámico, linealidad y estabilidad de los componentes y capacidad de procesamiento. Sin embargo se constituye como una alternativa económica para diseñar prototipos con requerimientos de complejidad moderados que facilitan la evaluación preliminar de nuevas propuestas tecnológicas con fines investigativos o educativos.

2.3.4 Otras alternativas de plataformas de hardware para el desarrollo de aplicaciones de Radio Definido por Software

Si bien, las limitaciones en las especificaciones técnicas del USRP lo hacen inadecuado para su aplicación en soluciones con requerimientos comerciales o profesionales, existen otras alternativas que podrían ser viables para el desarrollo de estas aplicaciones, en especial las de análisis de señales de comunicaciones basadas en tecnologías de radio definido por software.

Las arquitecturas de estas plataformas alternativas de radio definido por software pueden dividirse en dos grupos: Empresas como Pentek Inc, IHS Jane's, National Instruments, entre otras, proporcionan plataformas de hardware en diferentes formatos de fábrica como PCI (*“Peripheral Component Interconnect”*), cPIC (*“Compact Peripheral Component Interconnect”*) y VXI (*“VME Extensions for Instrumentation”*) que permiten al usuario configurar diferentes soluciones según sus requerimientos específicos. En el otro grupo se encuentran empresas como Rohde & Schwarz, Agilent y Tektronix que ofrecen sistemas de instrumentación y entre estos, algunos dispositivos de análisis de señales que tienen la posibilidad de exportar digitalmente la señal analizada en banda base o en frecuencias intermedias, de tal manera que el dispositivo puede actuar como una simple interfaz de radiofrecuencias y el análisis se puede realizar en un computador convencional.

Una solución para el análisis de señales de comunicaciones inalámbricas bajo cualquiera de las dos aproximaciones anteriores tiene un costo bastante mayor al del USRP, sin embargo las plataformas basadas PCI, cPCI o VXI permiten configurar soluciones personalizadas con una gran capacidad de procesamiento a un costo significativamente menor que utilizar un analizador de señales disponible comercialmente.

3. ARQUITECTURA DE SOFTWARE DE COMUNICACIONES

La arquitectura de software describe la organización de un sistema más allá de algoritmos y estructuras de datos, define los diferentes elementos que lo componen, sus propiedades, sus funciones y responsabilidades, la relación entre componentes y la relación de cada componente con el entorno. El uso de arquitecturas de software promueve el desarrollo de soluciones de forma organizada y eficiente, proporciona un marco de abstracción común, en forma de modelos que pueden transferirse fácilmente de un sistema a otro con requerimientos similares, y habilita el reuso del conocimiento a gran escala.

Este capítulo describe los beneficios de utilizar arquitecturas en el desarrollo de aplicaciones SDR (*“Software Defined Radio”*) y describe cómo se puede utilizar SCA (*“Software Communications Architecture”*) para el diseño de dispositivos de instrumentación para el análisis de señales de comunicaciones inalámbricas.

3.1 DESCRIPCIÓN DE LA ARQUITECTURA DE SOFTWARE DE COMUNICACIONES

SCA es una arquitectura abierta desarrollada por el Departamento de Defensa de los Estados Unidos para estandarizar el desarrollo de aplicaciones SDR, facilitar la operación entre diferentes sistemas de radio y reducir el tiempo y costo del proceso de desarrollo. Aunque la arquitectura fue desarrollada para aplicaciones militares, SCA introduce principios y características que se pueden aplicar en cualquier otro tipo de aplicaciones basadas en procesamiento digital de señales [24].

SCA facilita el reuso de componentes introduciendo un nivel de abstracción entre el hardware y el software de un sistema de SDR e introduce un entorno operativo común para todas las aplicaciones de comunicaciones independientemente de la plataforma que las soporta. Para lograr este objetivo SCA se soporta en estándares comerciales y en principios y patrones de diseño utilizados en la ingeniería de software.

3.1.1 El entorno operativo de SCA

SCA define un entorno operativo, llamado OE (*“Operativig Enironment”*), para su utilización en los radios basados en tecnologías de SDR. También especifica los servicios e interfaces que el OE ofrece para las aplicaciones. Estas interfaces se definen mediante el lenguaje IDL (*“Interface Definition Language”*) de CORBA (*“Common Object Request Broker Architecture”*) y sus representaciones gráficas se realizan utilizando UML (*“Unified Modeling Language”*).

El OE está conformado por un CF (*“Core Framework”*), un sistema operativo basado en POSIX (*“Portable Operating System Interface”*) y CORBA como middleware². El sistema operativo sobre el

² Software de conectividad que ofrece un conjunto de servicios que hacen posible el funcionamiento de aplicaciones distribuidas sobre plataformas heterogéneas [25].

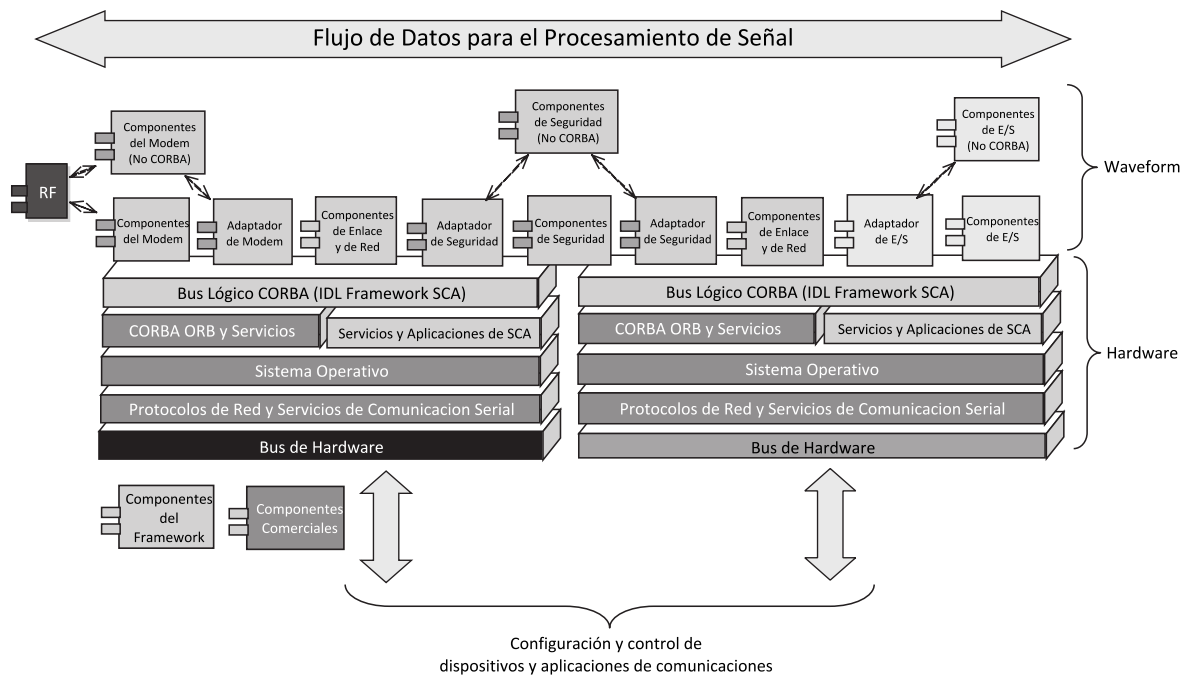


Figura 3.2. Organización de los componentes de un sistema SCA

Conceptualmente, SCA se divide en tres partes, la aplicación de onda, el CF y el DP (*“Domain Profile”*). La aplicación de onda está conformada por todos los componentes que implementan las operaciones de procesamiento de señales necesarias por la aplicación de comunicaciones; el CF incluye todo el software necesario para la administración del sistema de radio y las diferentes aplicaciones de onda.

El CF utiliza el DP para describir los diferentes objetos que conforman el sistema. El DP es un conjunto de archivos XML (*“Extended Markup Language”*) que describen la identidad, capacidades, propiedades, dependencias y ubicación de los elementos de hardware y software que conforman el sistema. Las características de los objetos de software se encuentran en el SPD (*“Software Package Descriptor”*), en el SCD (*“Software Component Descriptor”*) y en el SAD (*“Software Assembly Descriptor”*). Las características de los dispositivos de hardware se almacenan en el DPD (*“Device Package Descriptor”*) y en el DCD (*“Device Configuration Descriptor”*). Los archivos PRF (*“Properties Files”*) contienen información acerca de las propiedades de los elementos de hardware o software, el PD (*“Profile Descriptor”*) contiene información de los DCD, SPD y SAD que conforman el sistema. Finalmente el DMD (*“Domain Manager Descriptor”*) contiene información acerca del administrador de dominio. En la Figura 3.3 se muestran las diferentes relaciones entre los archivos XML que conforman el perfil del dominio.

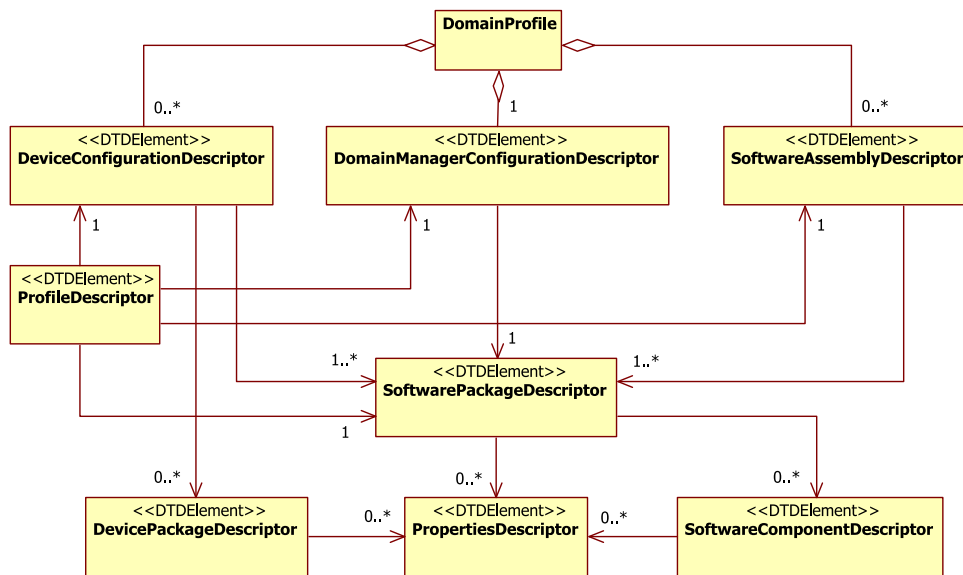


Figura 3.3. Relaciones XML del perfil de dominio

3.2 OSSIE LA IMPLEMENTACIÓN EMPOTRADA DE CÓDIGO ABIERTO PARA SCA

OSSIE (*“Open Source SCA Implementation - Embedded”*) es una iniciativa del grupo de investigación de radios móviles y portables del Instituto Politécnico y Universidad Estatal de Virginia para desarrollar una implementación de código abierto de SCA [4]. OSSIE está escrito en C++ y usa la implementación de CORBA omniORB y el analizador de XML Xerces, ambos disponibles abiertamente. OSSIE fue desarrollado sobre el sistema operativo Linux, inicialmente para la distribución Fedora y ahora para la distribución Ubuntu.

Aunque OSSIE todavía no es una implementación de SCA certificada por el JTRS (*“Joint Tactical Radio System”*), aún proporciona un marco de trabajo adecuado para la investigación y desarrollo de prototipos gracias a su libre disponibilidad y sencillez puesto que implementa los elementos mínimos requeridos por la arquitectura SCA para ser funcional.

La implementación de OSSIE está organizada en tres partes: el analizador de XML, el CF y las aplicaciones de ejemplo. EL analizador contiene las clases que se utilizan para validar y acceder a la información con la descripción de la aplicación de onda que contienen los archivos XML. El CF implementa las interfaces y clases que soportan la funcionalidad del OE de SCA y las aplicaciones de ejemplo consisten en aplicaciones simples que demuestran la funcionalidad del CF.

3.2.1 El entorno de desarrollo para aplicaciones de onda

OSSIE incluye un conjunto de herramientas de software libre, para el desarrollo, depuración y ejecución de componentes y aplicaciones de onda. OWD (*“Ossie Waveform Developer”*) es el

entorno de desarrollo de aplicaciones de onda de OSSIE y consiste en un complemento para el IDE (*"Integrated Development Environment"*) Eclipse³ que proporciona las herramientas necesarias para el desarrollo de componentes y aplicaciones de onda. La aplicación ALF consiste en una herramienta para la visualización y depuración de componentes y aplicaciones de onda. La herramienta OSSIE WaveDash (*"OSSIE Waveform Dashboard"*) proporciona una interfaz personalizable que permite instalar, administrar y ejecutar las diferentes aplicaciones de onda así como configurar las propiedades de sus componentes [27].

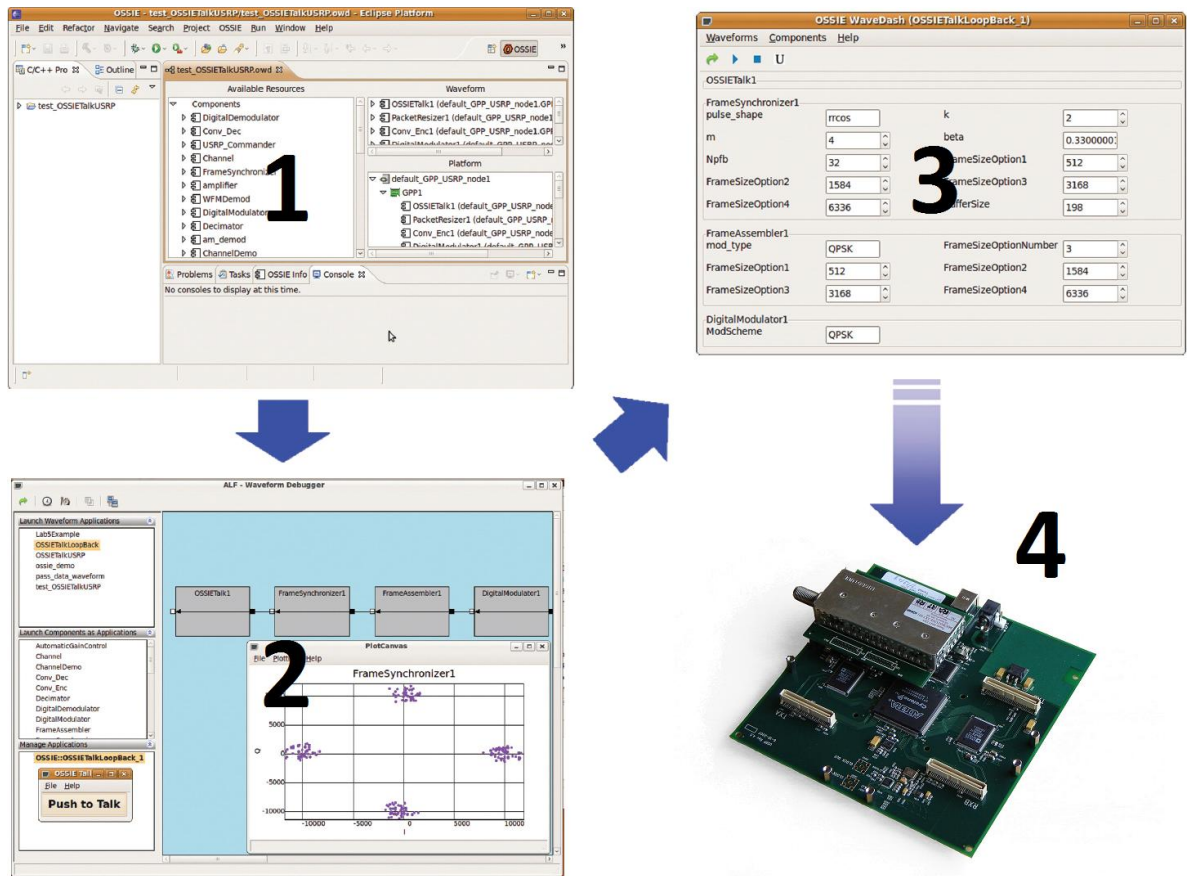


Figura 3.4. Flujo de diseño en entorno de desarrollo de OSSIE

La Figura 3.4 muestra el flujo de diseño para el entorno de desarrollo de aplicaciones de onda de OSSIE. El primer paso consiste en el desarrollo del software de la aplicación utilizando las herramientas basadas en Eclipse para el desarrollo de componentes y aplicaciones de onda. El segundo paso consiste en depurar la aplicación y sus componentes utilizando la herramienta ALF. El tercer paso consiste en la configuración y despliegue de la aplicación de onda utilizando la herramienta OSSIE WaveDash que finalmente la ejecuta sobre una plataforma de SDR basada en el USRP.

³ Entorno de desarrollo integrado de código abierto basado en Java.

3.2.2 Ventajas y desventajas de OSSIE

Existen diferentes herramientas para el desarrollo de aplicaciones de SDR, algunas de código propietario y otras de código abierto. Algunas herramientas de código abierto se rigen según su propia arquitectura como el proyecto GNU Radio [28] y otros como SCARI Open ("*SCA Reference Implementation - Open*") [29] y OSSIE se rigen por una arquitectura estándar como lo es SCA.

Durante la concepción de este trabajo de grado se determinó que se trabajaría con la arquitectura estándar SCA y entre sus diferentes implementaciones se examinaron la de código abierto SCARI Open desarrollada por el Centro de Investigación en Comunicaciones de Canadá y OSSIE desarrollado por el Laboratorio de Investigación en Comunicaciones Inalámbricas del Instituto Politécnico de Virginia, concluyendo que se utilizaría la segunda por las razones que se exponen a continuación.

- Tanto SCARI Open como OSSIE son implementaciones de código abierto que se pueden modificar libremente, lo que facilita la corrección de errores y la adaptación a las necesidades particulares del proyecto.
- Aunque SCARI Open fue la primera implementación de SCA, actualmente su desarrollo se encuentra suspendido mientras OSSIE se encuentra en permanente desarrollo desde su inicio, lo que le ha permitido alcanzar un mayor grado de madurez.
- El desempeño de OSSIE es superior al de SCARI puesto que el primero se ha implementado en C++ mientras el segundo se ha implementado en Java lo que implica que en el segundo se requiere compilar el bytecode de Java antes de su ejecución sobre el procesador del equipo anfitrión [33].
- El soporte de Java en dispositivos empujados es limitado mientras las implementaciones en C++ se pueden migrar fácilmente de una plataforma a otra mediante la recompilación del código fuente, lo que hace a OSSIE adecuado para el desarrollo de prototipos de SDR que se quieren implementar posteriormente sobre plataformas empujadas.
- Mientras SCARI Open se ha concentrado únicamente en la implementación de SCA, el proyecto OSSIE ha implementado controladores para diferentes dispositivos de hardware y ha desarrollado una serie de complementos para el entorno de desarrollo que permiten lograr mayor eficiencia en el desarrollo de aplicaciones de SDR.

Sin embargo existen dos desventajas que se deben considerar cuando se plantea utilizar OSSIE como el framework SCA para el desarrollo de aplicaciones de SDR.

- OSSIE es un proyecto de código abierto y de libre utilización, por lo que no existe un compromiso de soporte para el desarrollo de aplicaciones y la información relacionada suele ser escasa e incompleta.
- OSSIE aún no ha alcanzado una versión estable y aunque todavía no está oficialmente conforme con la especificación SCA; este es un objetivo que se ha planteado formalmente y espera cumplirse en el futuro.

3.3 SOFTWARE PARA EL ANÁLISIS DE SEÑALES DE COMUNICACIONES BASADO EN SCA

En el Capítulo 1 se describieron los beneficios de utilizar hardware digital en la implementación de dispositivos de instrumentación para el análisis de señales. A continuación se describen los beneficios de utilizar una arquitectura de abierta y estándar para el desarrollo del software de estos dispositivos de instrumentación.

SCA ha sido diseñado para el desarrollo de aplicaciones de comunicaciones basadas en tecnologías de SDR, en donde la interacción con el usuario es mínima. Aunque la interfaz de usuario en un sistema para el análisis de señales es algo compleja y no se ha definido en el ámbito de SCA, los beneficios de utilizar esta arquitectura superan los inconvenientes, puesto que excepto la necesidad de una interfaz de usuario compleja, los algoritmos de procesamiento de señales y sus técnicas de implementación son similares tanto en una aplicación de comunicaciones como en una aplicación de análisis de señales de comunicaciones.

Gracias a la posición que ha alcanzado SCA y a su grado de estandarización nacional dado por el Departamento de Defensa de los Estados Unidos, existen varias implementaciones disponibles en el mercado, tanto implementaciones del CF sobre diferentes plataformas de hardware como componentes de procesamiento de señales compatibles con SCA. Esto es una ventaja en el desarrollo de aplicaciones de análisis de señales puesto que permite reutilizar el código de diferentes componentes que se usan en el desarrollo de aplicaciones de comunicaciones y también permite desarrollar nuevos componentes y reutilizarlos en diferentes plataformas compatibles con SCA. También existen varios entornos de desarrollo para SCA que permiten simplificar el proceso desarrollo de la parte de procesamiento y análisis de señales y concentrarse en los otros aspectos como la interfaz de usuario.

3.3.1 Diseño de clases del software para el análisis de señales de comunicaciones

En el Capítulo 1 se ha descrito una plataforma de hardware que podría utilizarse para la implementación de un dispositivo para el análisis de señales de comunicaciones. A continuación se describe la arquitectura basada en SCA, propuesta para el software de tal dispositivo de análisis de señales.

En la Figura 3.5 se muestra el diagrama de clases del software de análisis de señales. El diseño se ha realizado siguiendo el patrón Modelo-Vista-Controlador [34] para desacoplar la dependencia que existe entre los elementos de la interfaz de usuario y el modelo de la aplicación. La aplicación puede considerarse como si estuviera compuesta por cuatro segmentos principales que interactúan entre sí para proporcionar todas las funcionalidades que se requieren en un dispositivo para el análisis de señales de comunicaciones. Estos segmentos son la aplicación SCA, el controlador de aplicación, el modelo de aplicación y la interfaz de usuario.

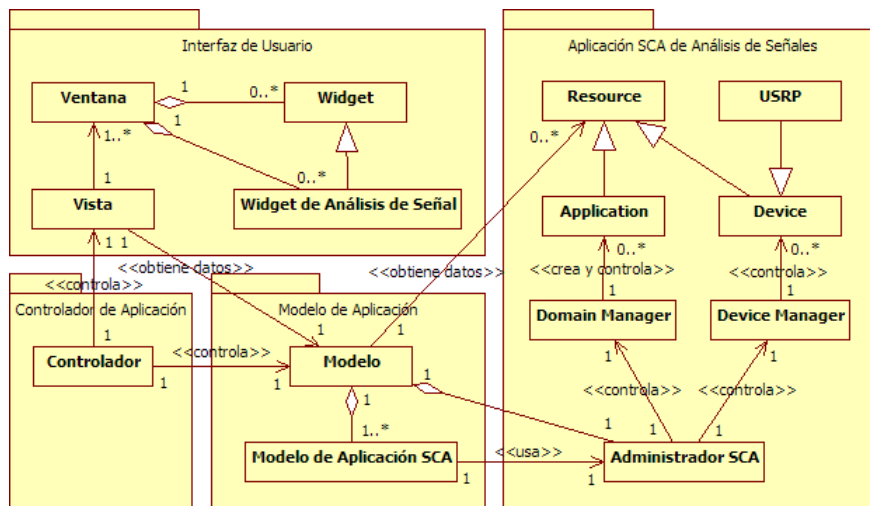


Figura 3.5. Arquitectura de un analizador de señales basado en SCA

La aplicación SCA es una aplicación de procesamiento digital de señales, diseñada según la arquitectura SCA, que tiene por objeto realizar todo el procesamiento necesario para medir los diferentes parámetros de la señal de entrada. De este segmento también depende el control de todos los dispositivos necesarios para la captura y procesamiento de la señal como la interfaz de RF (*“Radio Frequency”*), DSPs (*“Digital Signal Processors”*) y FPGAs (*“Field Programmable Gate Arrays”*). Pueden existir diferentes implementaciones según los requerimientos del análisis que se desea realizar; por ejemplo puede existir una aplicación para el análisis de señales OFDM (*“Orthogonal Frequency Division Multiplexing”*) y otra implementación diferente para el análisis de señales CDMA (*“Code Division Multiple Access”*). La arquitectura de este segmento está completamente definida por la especificación de SCA y su implementación se describe con mayor detalle en el Capítulo 1.

El modelo de aplicación consiste en una abstracción de la implementación del CF de SCA así como de las aplicaciones SCA para el análisis de señales. Su objetivo principal es proporcionar las interfaces necesarias para interactuar con los diferentes elementos del OE de SCA como el *Domain Manager*, el *Device Manager* y las aplicaciones SCA con sus componentes. El modelo de aplicación tiene una función muy importante que consiste en obtener los resultados del análisis de la señal que se realiza en los diferentes dispositivos y componentes de SCA para visualizarlos en la interfaz de usuario.

La interfaz de usuario comprende todos los elementos necesarios para presentar en forma gráfica la información obtenida a partir del análisis de la señal de entrada y proporcionar los mecanismos de interacción con el usuario. La interfaz de usuario está compuesta por un conjunto de ventanas y *widgets* que son administradas por una clase llamada Vista que también atiende cualquier evento que se pueda producir en uno de estos elementos. La interfaz de usuario puede adoptar diferentes formas de visualización, llamadas vistas de usuario, dependiendo de qué tipo de información se vaya a presentar o cual es la aplicación de análisis de señales que se está utilizando. Además de la información resultante del análisis de la señal, la interfaz de usuario expone un conjunto adicional de *widgets* que permiten seleccionar diferentes implementaciones de la aplicación SCA de análisis

de señales y configurar los diferentes parámetros de visualización como de operación de estas aplicaciones.

Finalmente el segmento de control de aplicación está compuesto por un solo elemento llamado Controlador, el cual define por completo el comportamiento de la toda la aplicación. Aunque la Vista es quien administra los elementos de la interfaz gráfica y quien recibe las ordenes del usuario, es el controlador el que define cual es la vista que se debe presentar al usuario y el que coordina todos los procedimientos cuando el usuario invoca una operación para cambiar la aplicación SCA de análisis de señales o para cambiar uno de los parámetros de funcionamiento de la aplicación.

A continuación se hace una descripción de cada uno de las clases que conforman el diseño de la aplicación expuesta en la Figura 3.5.

3.3.1.1 Controlador

El Controlador define por completo el comportamiento de la aplicación y actúa como intermediario entre la vista y el modelo para coordinar las operaciones que el usuario invoca a través de la interfaz de usuario. También determina cuando se debe iniciar o detener la aplicación SCA y cuál es la interfaz de usuario indicada para cada uno de los diferentes análisis que se pueden hacer sobre la señal.

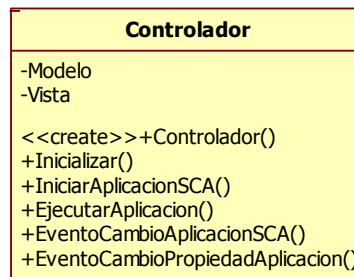


Figura 3.6. Diagrama de la clase Controlador

La clase Controlador mantiene una referencia del Modelo y de la Vista de la aplicación y proporciona los siguientes métodos:

- **Controlador:** Crea una nueva instancia de la clase, este constructor recibe las referencias a las instancias de las clases Modelo y Vista para las que actuará como controlador.
- **Inicializar:** Inicializa todos los recursos y librerías de la aplicación y suscribe el controlador a los eventos de la interfaz de usuario, este procedimiento asegura que todos los elementos se encuentran en un estado conocido antes de iniciar la aplicación SCA para el análisis de señales.
- **InicializarAplicacionSCA:** Inicializa la aplicación SCA para el análisis de señales. Esta función permite elegir entre diferentes implementaciones de la aplicación de procesamiento de señal y detiene cualquier instancia que se encuentre en ejecución para crear e inicializar la implementación seleccionada.

- **EjecutarAplicacion:** Las mayoría de las aplicaciones gráficas requieren que el hilo de ejecución principal atienda los diferentes eventos que producen los elementos de la aplicación, esta función transfiere el control al objeto Vista, el cual tiene acceso a la librería gráfica que implementa el bucle de despacho de eventos
- **EventoCambioAplicacionSCA y EventoCambioPropiedadAplicacion:** Controladores de los eventos que se producen en la interfaz de usuario cuando el usuario desea cambiar la implementación o cambiar un parámetro de funcionamiento de la aplicación de análisis de señales. Una vez el Controlador es notificado acerca de un evento este coordina los procedimientos necesarios para realizar el cambio de la aplicación de análisis de señales o el cambio en la propiedad de la aplicación en funcionamiento.

3.3.1.2 Vista

La Vista tiene la función de administrar todos los elementos que conforman la interfaz de usuario, entre estos elementos se encuentran las ventanas y sus respectivos *widgets*. Comunica al Controlador acerca de la invocación de cualquier operación por parte del usuario y obtiene la información que se debe visualizar accediendo al modelo de la aplicación.



Figura 3.7. Diagrama de la clase Vista

La Vista mantiene una referencia a la colección de ventanas que administra, así como una referencia a la instancia del modelo de la aplicación. A continuación se describen brevemente cada uno de los métodos que se implementan:

- **Vista:** Crea una nueva instancia de la clase Vista, a su vez crea todas las ventanas y *widgets* necesarios para la visualización de la información. Este constructor recibe una referencia a la instancia del Modelo de la aplicación para obtener los datos que se deben visualizar en las diferentes ventanas y *widgets*.
- **EjecutarAplicación y EjecutarBucleDespachoEventos:** EjecutarAplicación es un método público que expone al controlador la función de la librería de gestión de ventanas para invocar el bucle de notificación de eventos, necesario para la ejecución de una aplicación en un entorno gráfico. Este método invoca al segundo método EjecutarBucleDespachoEventos, que

implementa los procedimientos necesarios de acuerdo a la librería específica de gestión de ventanas que se utiliza en la aplicación.

- **SeleccionarVistaAplicacionSCA:** Cuando se han implementado diferentes aplicaciones de análisis de señales para diferentes propósitos, es necesario visualizar una interfaz de usuario adecuada para tal aplicación de análisis de señales. Esta función permite al Controlador solicitar la presentación de la interfaz de usuario adecuada, con sus respectivos *widgets* y ventanas, de acuerdo a la aplicación de análisis de señales en ejecución.
- **EventoCambioPropiedad:** Controlador de los eventos de cambio de propiedad de la colección de ventanas y *widgets* de la interfaz de usuario. Algunos de estos eventos representan cambios en las propiedades de la misma interfaz, en cuyo caso el procedimiento a seguir es totalmente controlado por la misma instancia de la clase Vista.
- **EventoTrazoActualizado:** En general, la información obtenida del análisis de la señal se presenta en forma de diferentes gráficos, llamados generalmente trazos. La vista obtiene los trazos desde la instancia del Modelo de la aplicación, pero debido a que estos trazos cambian permanentemente es necesario utilizar un esquema de notificación de eventos en la clase Modelo para el cual esté método actúa como controlador.
- **SuscribirEventoCambioPropiedadAplicacion y NotificarEventoCambioPropiedadAplicacion:** Estos métodos exponen la funcionalidad del patrón de software “observador” para la notificación de eventos [35], el cual es ampliamente utilizado en el diseño del prototipo propuesto en este trabajo de grado. Cuando un evento en la interfaz de usuario requiere de la invocación de un cambio en las propiedades de la aplicación de análisis de señales, el procedimiento correspondiente debe ser coordinado por el Controlador el cual debe suscribirse al evento a través del método `SuscribirEvento` y es notificado a través de la invocación del método `NotificarEvento`.
- **SuscribirEventoCambioAplicacion y NotificarEventoCambioAplicacion:** De forma similar a los métodos anteriores, estos implementan el patrón de notificación de eventos para la invocación del procedimiento que cambia la implementación de la aplicación de análisis de señales en ejecución.
- **HabilitarEventos y DeshabilitarEventos:** Estas funciones habilitan y deshabilitan los mecanismos de notificación de eventos de la clase. Esta característica es útil en el momento de inicialización de la clase, cuando se desean establecer los valores de las propiedades de los elementos de la interfaz gráfica sin invocar los procedimientos de notificación de cambio de aplicación o cambio en la propiedad de la aplicación.

3.3.1.3 *Widget* y Ventana

La interfaz de usuario se compone por una o más ventanas de acuerdo a la aplicación. Cada ventana es un elemento gráfico que actúa como contenedor de los diferentes *widgets* necesarios para la visualización de la información, de hecho la misma clase Ventana suele ser una clase heredada de la clase *Widget*. Estas clases son implementadas por las diferentes librerías de gestión de ventanas, con diferentes nombres según su implementación. Estas dos clases se han

incluido en el diseño con el propósito de abstraer las funciones mínimas que proporcionan sus equivalentes en las librerías de gestión de ventanas.

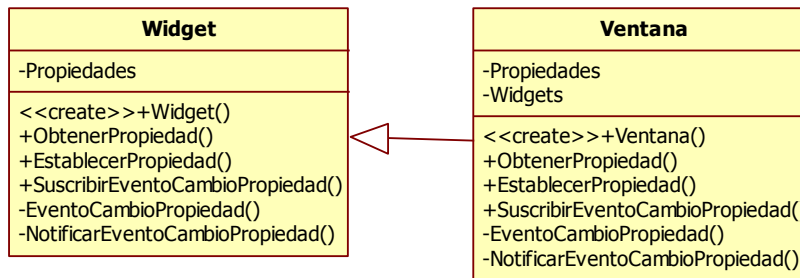


Figura 3.8. Diagrama de la abstracción de las clases *Widget* y *Ventana*

A continuación se describen las funcionalidades que deben proporcionar los elementos de las clases *Widget* y *Ventana* o sus equivalentes según la librería de gestión de ventanas que se vaya a utilizar:

- **Widget y Ventana:** Actúan respectivamente como constructores de instancia para cada una de las clases, estos métodos pueden ser privados si se ha implementado para la construcción de estas clases el patrón de software Fabrica [35].
- **ObtenerPropiedad y EstablecerPropiedad:** Los elementos gráficos de la interfaz de usuario tienen varias propiedades que pueden accederse a través de los métodos *ObtenerPropiedad* y *EstablecerPropiedad* o sus equivalentes.
- **SuscribirEventoCambioPropiedad y NotificarEventoCambioPropiedad:** Las librerías de gestión de ventanas generalmente implementan el patrón Observador para varias o cada una de las propiedades de sus elementos gráficos.

3.3.1.4 Widget de Análisis de Señal

Las librerías de gestión de ventanas implementan muchos de los *widgets* necesarios en una aplicación genérica: botones, cuadros de texto, etiquetas, etc. Sin embargo gran parte de la información obtenida del análisis de las señales debe presentarse gráficamente en forma de trazos. Es necesario implementar uno o varios *widgets* que permitan visualizar cada uno de los trazos de forma adecuada, en forma de espectrograma, diagrama de dispersión, diagrama de apertura, etc.

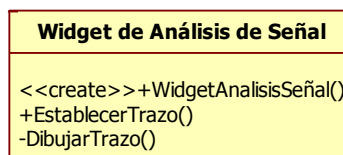


Figura 3.9. Diagrama de la clase *Widget* de Análisis de Señal

Puesto que el elemento *Widget* de Análisis de Señal hereda los atributos y métodos de la clase *Widget*, el elemento heredado debe implementar dos métodos adicionales que se describen a continuación:

- **Establecer Trazo:** Esta función permite entregar al elemento gráfico, toda la información necesaria para la visualización de la información del análisis de una señal en forma de trazos. Aunque la información obtenida del análisis de una señal se presenta comúnmente en forma gráfica, este método también permite entregar la información necesaria a un elemento diseñado para visualizar la información en forma textual o cualquier otra formato que no sea gráfico.
- **DibujarTrazo:** Después de actualizar los datos del trazo que se debe visualizar en el componente gráfico o *widget*, es necesario indicarle a la librería de gestión de ventanas que debe actualizar el contenido del elemento gráfico. Este método implementa el procedimiento específico según la librería utilizada para refrescar el contenido de un *widget* en la pantalla.

3.3.1.5 Modelo

El Modelo contiene la información específica del sistema de análisis de señales. Esta clase abstrae los detalles de implementación del CF de SCA y de las diferentes implementaciones de las aplicaciones de análisis de señal. Esta clase se ayuda de dos elementos, la clase Administrador SCA que abstrae los detalles de implementación del CF y la clase Modelo Aplicación SCA que abstrae los detalles específicos de cada una de las aplicaciones de análisis de señales.

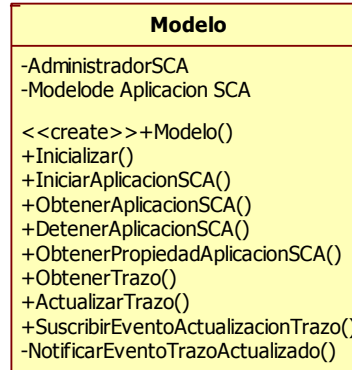


Figura 3.10. Diagrama de la clase Modelo

A continuación se describen los métodos que proporciona la clase Modelo:

- **Modelo:** Crea una nueva instancia de la clase Modelo, a su vez obtiene una instancia del Administrador SCA y crea una instancia del Modelo de Aplicación SCA para cada una de las implementaciones de la aplicación de análisis de señales.
- **Inicializar:** Inicializa todos las librerías y recursos necesarios para el Modelo, además establece sus propiedades en valores conocidos antes de su utilización.

- **IniciarAplicacionSCA y DetenerAplicacionSCA:** El primer método coordina los procedimientos necesarios para crear, instalar y ejecutar una instancia de la aplicación SCA para el análisis de señales en el CF de SCA y el segundo los procedimientos necesarios para detener y desinstalar una aplicación de análisis de señales en el CF.
- **ObtenerAplicacionSCA:** Devuelve una instancia de la clase Modelo Aplicación SCA que representa la aplicación de análisis de señales que se encuentra en ejecución dentro del CF de SCA.
- **ActualizarTrazo y ObtenerTrazo:** Los diferentes componentes de la aplicación SCA, encargados de realizar diferentes análisis sobre la señal de entrada, entregan la información a través al modelo de la aplicación a través de una llamada al método ActualizarTrazo. Los elementos de la clase Vista pueden acceder a la información sobre el trazo actualizado mediante el método ObtenerTrazo.
- **SuscribirEventoActualizacionTrazo y NotificarEventoActualizacionTrazo:** Implementación del patrón de software Observador para notificar cuando un componente SCA ha actualizado la información de uno de los trazos que se deben presentar en la interfaz de usuario.

3.3.1.6 Modelo de Aplicación SCA

El Modelo de Aplicación SCA es una clase virtual que abstrae los requerimientos comunes del modelo de aplicación de cada una de las implementaciones de la aplicación de análisis de señal. Es decir que existe una implementación de Modelo de Aplicación SCA por cada implementación de la aplicación SCA para el análisis de señales.

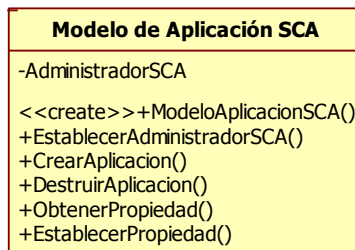


Figura 3.11. Diagrama de la clase Modelo de Aplicación SCA

A continuación se describen las funciones de la clase Modelo de Aplicación SCA:

- **EstablecerAdministradorSCA:** La clase de Modelo de Aplicación SCA abstrae los detalles de implementación de una aplicación SCA en específico, este método permite establecer la instancia del Administrador SCA que requiere esta instancia para controlar la Aplicación SCA dentro del CF de SCA.
- **CrearAplicacion y DestruirAplicacion:** Estos métodos permiten realizar la instalación o desinstalación de una aplicación dentro del CF de SCA.
- **ObtenerPropiedad y DestruirPropiedad:** Estos métodos permiten leer y escribir las diferentes propiedades de cada uno de los componentes que conforman la aplicación SCA.

3.3.1.7 Administrador SCA

Esta clase actúa como intermediario entre el Modelo de la Aplicación SCA y la aplicación real que se encuentra dentro del CF de SCA. Proporciona los mecanismos necesarios para interactuar con los diferentes elementos que existen dentro del CF.

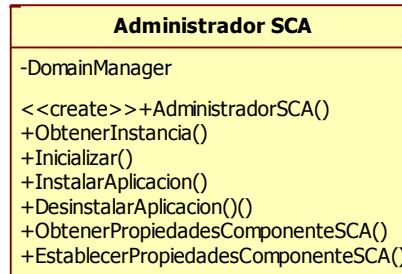


Figura 3.12. Diagrama de la clase Administrador SCA

La clase Administrador SCA proporciona los siguientes métodos:

- **AdministradorSCA y ObtenerInstancia:** La clase Administrador SCA se ha diseñado según el patrón de diseño Instancia Única y por ello la visibilidad del constructor es privada. Para obtener una referencia a esta instancia única se debe invocar el método ObtenerInstancia.
- **Inicializar:** Inicializa los recursos que utiliza esta clase antes de su utilización, en especial las librerías que se requieren para comunicarse con los elementos del CF de SCA.
- **InstalarAplicacion y DesinstalarAplicacion:** Coordinan los procedimientos necesarios para crear e instalar o desinstalar una aplicación SCA directamente con el *Domain Manager* del CF.
- **ObtenerPropiedadesComponenteSCA y EstablecerPropiedadesComponenteSCA:** Proporciona los mecanismos necesarios para leer o modificar cualquier propiedad en cualquiera de los componentes de una aplicación SCA invocando directamente al objeto Resource que forma parte de la aplicación.

3.3.1.8 Domain Manager, Application y Resource

Estas clases hacen parte de la especificación de SCA y su implementación es responsabilidad del CF. Aunque en SCA se definen otros elementos, estos tres son de especial importancia porque proporcionan los métodos necesarios para administrar las diferentes aplicaciones SCA que se implementarán con el propósito de realizar diferentes análisis sobre una variedad de señales.

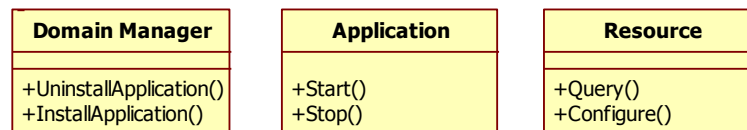


Figura 3.13. Diagrama simplificado de las clases SCA Domain Manager, Application y Resource

SCA define otros métodos que deben implementar los elementos *Domain Manager*, *Application* y *Resource*. A continuación solamente se describen los métodos relevantes para el diseño de esta aplicación.

- ***InstallApplication* y *UninstallApplication***: El *Domain Manager* proporciona estos métodos para ofrecer los mecanismos necesarios para la instalación de cualquier aplicación SCA especificando cuáles son sus archivos descriptores y la desinstalación de cualquier aplicación que se encuentre instalada en el CF por medio del objeto *Application* que la representa.
- ***Start* y *Stop***: Una vez se ha creado una instancia de una aplicación SCA en el *Domain Manager*, se puede invocar su ejecución y su detención por medio de estos dos métodos que se proveen mediante el objeto *Application*.
- ***Query* y *Configure***: Las aplicaciones SCA se componen de uno o varios componentes del tipo *Resource*, cada componente realiza una operación o procedimiento particular que puede ser parametrizado por medio de una o varias propiedades. Los métodos *Query* y *Configure* proporcionan los mecanismos para leer y modificar cualquiera de estos parámetros o propiedades.

3.3.2 Diagramas de secuencia para el software de análisis de señales de comunicaciones

En la sección anterior se presentó el diagrama de clases del software de análisis de señales de comunicaciones. Esta representación proporciona una descripción estática de los elementos de la solución y sus interrelaciones. En esta sección se presentan los diagramas de secuencia para describir claramente la forma en que los diferentes elementos del sistema interactúan entre sí.

Se presentan tres diagramas de secuencia que describen el proceso de inicialización del sistema, el proceso para la selección de una aplicación de análisis de señales, el proceso del cambio de una propiedad en instancia de una aplicación de análisis de señales y finalmente el proceso de actualización de los trazos en la interfaz de usuario a partir de la información proporcionada por los diferentes elementos de la aplicación SCA para el análisis de señales.

3.3.2.1 Secuencia de inicialización

En la Figura 3.14 se muestra el diagrama de secuencia del proceso de inicialización, esta secuencia se inicia desde que el usuario arranca el sistema y ordena la ejecución de la aplicación hasta que esta se encuentra en un estado completamente operacional. Esta secuencia se puede dividir en cuatro procedimientos: creación de instancias, inicialización de recursos, ejecución de la aplicación de análisis de señales y ejecución del bucle de despacho de eventos.

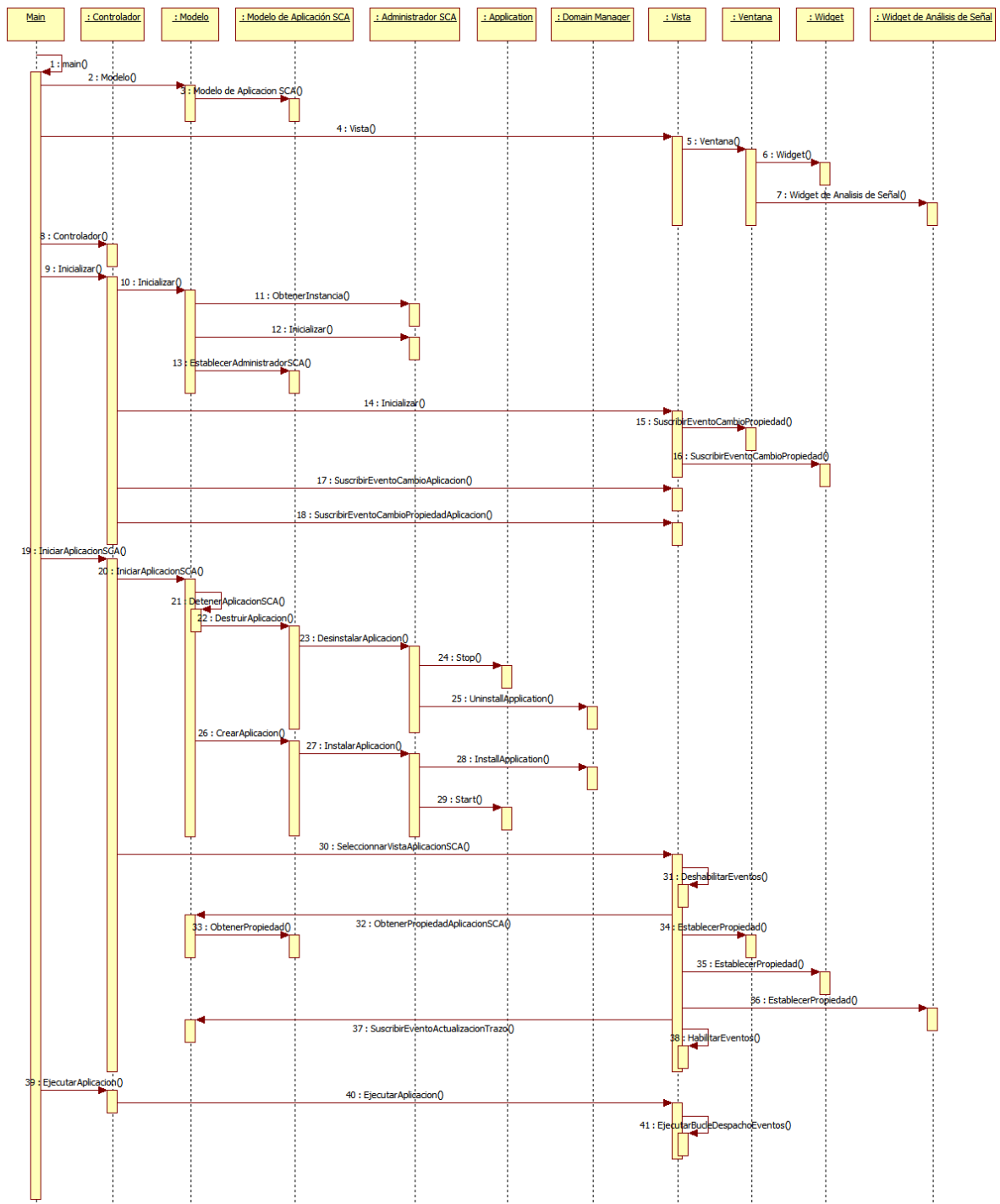


Figura 3.14. Diagrama de secuencia del proceso de inicialización

La secuencia de inicialización comienza con la operación número 1 que corresponde a la invocación del programa por parte del sistema operativo por medio del método estático de entrada *main*.

Una vez se ha invocado la función de entrada, las operaciones del número 2 al 8 crean las instancias del Modelo, la Vista y el Controlador de la aplicación, a su vez estos elementos crean las instancias de los elementos que requieren para su funcionamiento.

Las operaciones del número 9 al 18 corresponden al proceso de inicialización, tanto el Controlador como el Modelo y la Vista implementan el método Inicializar, pero es el Controlador el que controla el orden en el que se inicializan los demás elementos. El primer elemento en ser inicializado a petición del controlador es el Modelo, el cual a su vez inicializa las librerías para comunicarse con CF de SCA y obtiene una instancia del Administrador SCA, después de esto el Modelo crea una instancia del Modelo de Aplicación SCA para cada una de las implementaciones de las aplicaciones SCA para el análisis de señales.

Después de la inicialización del Modelo, se realiza la inicialización de la Vista, la cual se suscribe a los eventos de sus diferentes componentes para recibir notificaciones de las acciones que el usuario realiza a través de los elementos de la interfaz de usuario. Finalmente el Controlador se suscribe a los eventos de la Vista para recibir notificaciones acerca de las órdenes del usuario que la Vista delega al Controlador.

Una vez se ha inicializado el Controlador y los demás elementos, se puede continuar con la ejecución de la Aplicación SCA que realizará el análisis de la señal como se observa en las operaciones del número 19 al 38. La ejecución de la Aplicación SCA es coordinada por el Modelo a petición del Controlador. el Modelo detiene y desinstala cualquier Aplicación SCA en ejecución a través del modelo de aplicación que representa la Aplicación SCA en curso y crea una nueva instancia del Modelo de Aplicación SCA, luego despliega y arranca la aplicación de procesamiento de señales correspondiente en el CF de SCA.

Finalmente las operaciones del número 19 al 41 ilustran el procedimiento para ejecutar el bucle de despacho de eventos. Después de que la Aplicación SCA para el análisis de señales se está ejecutando en el CF de SCA, el Controlador debe transferir el control del *thread* principal a la función que implementa el bucle de despacho de eventos para habilitar la interacción con el usuario hasta que este solicite terminar la ejecución del programa.

3.3.2.2 Secuencia de cambio de aplicación de análisis de señales

El diseño que se ha presentado permite realizar el análisis de diferentes tipos de señales mediante la implementación adecuada de diferentes aplicaciones SCA. El usuario puede invocar el cambio de la aplicación de análisis de señales mediante la interfaz de usuario, cuya invocación es delegada por la Vista al Controlador, de aquí en adelante se repite la secuencia de las operaciones número 20 a 38 del diagrama de secuencia de inicialización que se muestra en la Figura 3.14 de la Sección 3.3.2.1, razón por la cual no se presenta ninguna diagrama para este apartado.

3.3.2.3 Secuencia de cambio de propiedad

Una vez se ha desplegado una aplicación para el análisis de señales, puede ser necesario cambiar cualquiera de los parámetros con los que se está realizando el análisis. La Figura 3.15 muestra el diagrama de secuencia para un cambio en una de las propiedades que establecen los parámetros

de funcionamiento de una aplicación de análisis de señales. A continuación se describe cada una de las operaciones que tienen lugar durante este procedimiento.

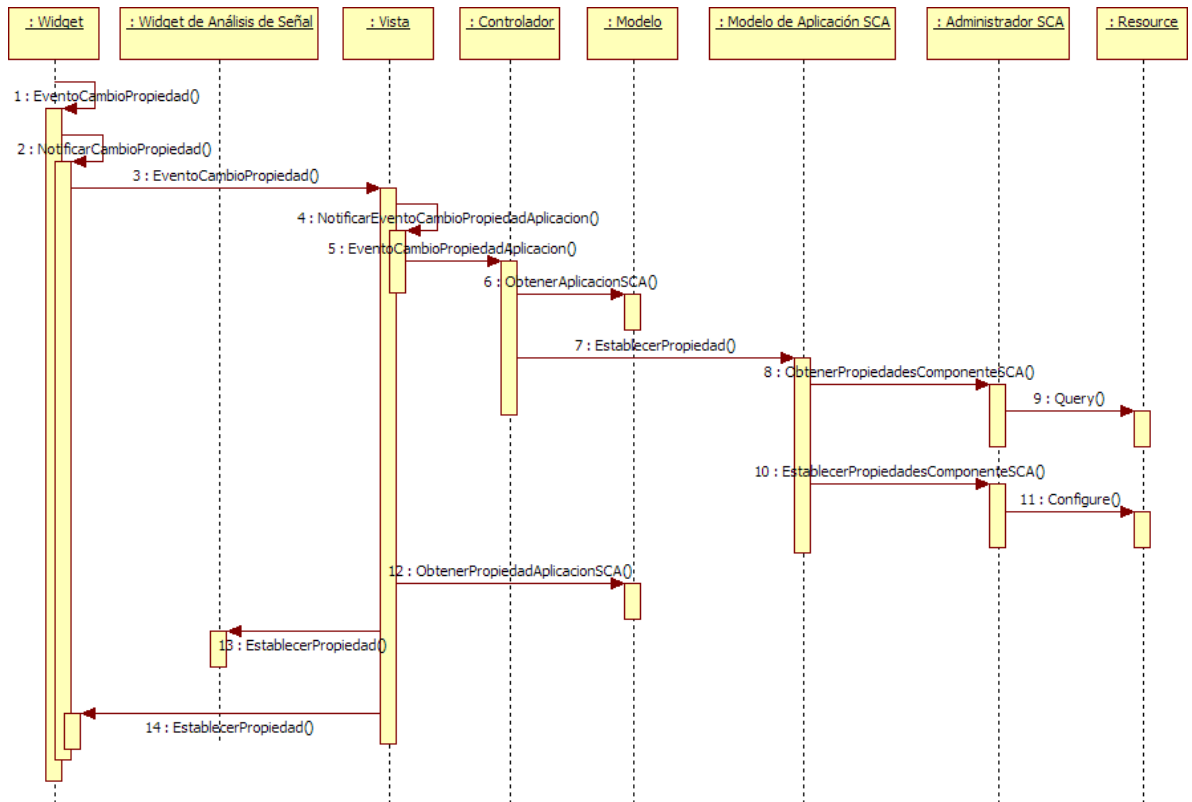


Figura 3.15. Diagrama de secuencia para el cambio de propiedad de la aplicación SCA

Las operaciones del número 1 hasta el número 5 representan el mecanismo de notificación de eventos que utiliza la interfaz de usuario para notificar del cambio en una propiedad de uno de los *widgets* que representan alguno de los parámetros de funcionamiento de la aplicación de análisis de señales. El proceso de suscripción a los eventos se describe en las operaciones 17 y 18 de la Figura 3.14 de la Sección 3.3.2.1.

Una vez el Controlador es notificado de la invocación al procedimiento para cambiar una propiedad de la Aplicación SCA, este solicita la instancia del Modelo de Aplicación SCA en ejecución mediante la operación numero 6; las operaciones del número 7 a 11 ilustran como el Controlador invoca el método para establecer una propiedad de uno de los componentes de la Aplicación SCA y a continuación mediante las operaciones 8 y 9, como el Modelo de la Aplicación SCA invoca al Administrador SCA para que este obtenga una copia de la lista de propiedades del elemento SCA especificado, luego se invoca el establecimiento de las propiedades previamente actualizadas mediante las operaciones 10 y 11.

Finalmente el controlador verifica que el cambio en la propiedad se ha efectuado solicitando el valor de dicha propiedad al Modelo de Aplicación SCA mediante la operación número 12 y actualizando los diferentes elementos de la interfaz de usuario que lo requieran mediante las operaciones 13 y 14.

3.3.2.4 Secuencia de actualización de trazos

Por último se describe la secuencia de actualización de trazos. Esta secuencia se inicia cuando uno de los componentes de la Aplicación SCA actualiza la información que obtiene al realizar el análisis de las señales y transfiere esta información al Modelo mediante la invocación al procedimiento Actualizar Trazo como se indica en las operaciones 1 y 2 que se muestran en la Figura 3.16. A continuación las operaciones 3 y 4 reflejan el mecanismo de notificación de eventos que se ha implementado en la clase Modelo y al cual la instancia de la clase Vista se suscribió en la operación número 37 del diagrama de la Figura 3.14 de la Sección 3.3.2.1.

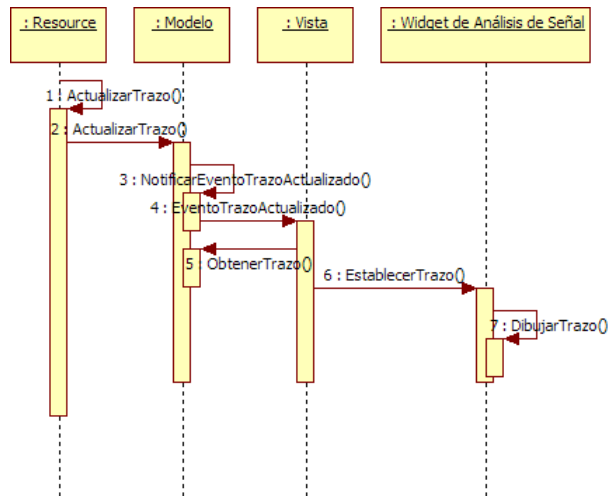


Figura 3.16. Diagrama de secuencia para la actualización de trazos

Finalmente la instancia de la clase Vista, al ser notificada del evento, obtiene la información actualizada desde la instancia del Modelo mediante la operación número 5 e invoca a los diferentes componentes de la interfaz de usuario que presentan la información mediante las operaciones 6 y 7.

La secuencia que se acaba de describir también aplica para la visualización de información obtenida como resultado del análisis de la señal, cuya representación se hace en forma textual o cualquier formato diferente al modo gráfico.

3.3.3 Consideraciones acerca del diseño del analizador de señales

En esta sección se han descrito con detalle todos los elementos de diseño del analizador de señales excepto por la aplicación SCA que realiza el análisis de la señal la cual se ha descrito de forma genérica, dejando para el Capítulo 1 los detalles de implementación de una aplicación para el análisis de señales OFDM. Aunque en este trabajo de grado se decidió que se implementará una aplicación para el análisis de señales OFDM, el diseño permite la implementación de un sistema para realizar el análisis de cualquier tipo de señal siempre y cuando se implementen las aplicaciones SCA adecuadas, sus respectivos modelos y los elementos gráficos que presentarán la información.

Tampoco se profundizó en la interfaz de usuario sino en las funcionalidades que esta debe proporcionar para cualquier aplicación de análisis de señales dejando el diseño e implementación de la interfaz de usuario específica para el análisis de señales OFDM para el Capítulo 1. También se han definido los métodos y procedimientos para relaciones entre objetos con multiplicidad uno a uno, sin embargo no existe mayor dificultad para implementar los procedimientos adicionales cuando la relación de multiplicidad entre los objetos es de varios a uno como se indica en el diagrama de la Figura 3.5.

En este diseño se ha utilizado una aproximación sencilla que consiste en implementar una aplicación que se puede ejecutar sobre el sistema operativo del hardware anfitrión y que a su vez es capaz de interactuar con una aplicación, dentro del CF de SCA, encargada de realizar el análisis de las señales. Un método más eficiente y a la vez un reto más grande consiste en modificar las especificaciones de SCA o proponer las extensiones necesarias para que esta arquitectura brinde la posibilidad de implementar en forma natural, aplicaciones SCA cuyo objetivo sea realizar el análisis de señales para la implementación de dispositivos de instrumentación. Sin embargo la evolución de la especificación SCA es controlada por el Departamento de Defensa de los Estados Unidos y por lo tanto cualquier modificación a la arquitectura SCA debe ser aprobada por un comité designado específicamente para ello.

4. APLICACIÓN SCA PARA EL ANÁLISIS DE SEÑALES OFDM

OFDM (*“Orthogonal Frequency Division Multiplex”*) es un método de multiplexación de datos que utiliza múltiples subportadoras. Esta es una tecnología ampliamente usada en telecomunicaciones inalámbricas gracias a su mayor eficiencia espectral respecto a los sistemas de multiplexación por tiempo o por frecuencia. A continuación se describen las bases teóricas en las que se fundamenta OFDM y posteriormente se describe la implementación de una aplicación SCA (*“Software Communications Architecture”*) para la demodulación de este tipo de señales y que en conjunto con el software descrito en la Sección 3.3 y el hardware descrito en la Sección 2.2 conforman una solución completamente funcional para el análisis de señales OFDM.

4.1 PRINCIPIOS DE OFDM

OFDM se define como una técnica de transmisión en donde se distribuyen los símbolos codificados sobre múltiples subportadoras para reducir la tasa de símbolos sobre cada subportadora y mejorar el desempeño ante fenómenos como la dispersión temporal causada por la propagación multitrayectoria [36].

OFDM se utiliza en diferentes sistemas de radiodifusión digital, entre los más importantes se encuentran el estándar de televisión digital terrestre DVB-T (*“Digital Video Broadcasting - Terrestrial”*), las especificaciones IEEE 802.11a y 802.11g para redes inalámbricas de área local, WiMAX (*“Worldwide Interoperability for Microwave Access”*) y LTE (*“Long Term Evolution”*) el cual se constituye como la evolución de UMTS (*“Universal Mobile Telecommunications System”*).

4.1.1 El concepto de transmisión con múltiples subportadoras

Considerando un esquema de modulación de portadora en fase o en cuadratura como M-QAM (*“M-ary Quadrature Amplitude Modulation”*) o M-PSK (*“M-ary Phase Shift Keying”*) con un tiempo de duración de símbolo T_s y un canal con una dispersión temporal τ_m , debe cumplirse la condición de la Ecuación 4.1 para obtener una recepción libre de ISI (*“Inter-Symbol Interference”*).

$$\frac{\tau_m}{T_s} \approx 0 \quad (4.1)$$

Como consecuencia la tasa de bits para un esquema de modulación determinado, definida por la expresión de la Ecuación 4.2, estará limitada por la dispersión temporal del canal.

$$R_b = \frac{1}{T_s} \log_2(M) \quad (4.2)$$

Esta limitación se puede superar dividiendo el flujo de bits en varios flujos con una tasa de bits menor y transmitiéndolos sobre un conjunto de subportadoras adyacentes. En la Figura 4.1 se ilustra este concepto, en la figura de la sección a se muestra una señal modulada con una sola

portadora, en la cual cada símbolo se transmite en un intervalo de duración T_s y en la sección b se muestra una señal OFDM en donde el ancho de banda BW de la señal original se mantiene, cada subportadora tiene un ancho de banda igual a BW/k y la duración de de cada símbolo se incrementa por el factor k . Sin embargo, el factor k no puede incrementarse indefinidamente debido a que al aumentar el tiempo de símbolo se aumentan la susceptibilidad ante la no coherencia temporal del canal.

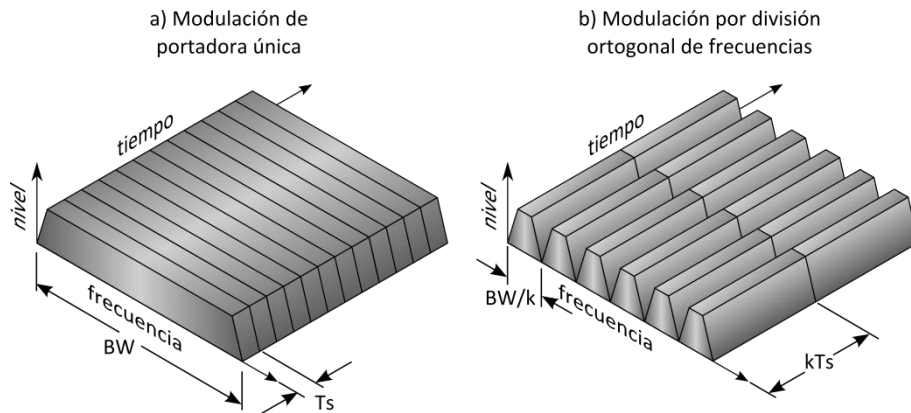


Figura 4.1. Concepto de modulación de múltiples subportadoras

En la Figura 4.2 se muestra la configuración de un sistema de modulación de múltiples subportadoras. Cada subportadora se puede definir como la señal armónica compleja $e^{j2\pi f_k t}$ y cada símbolo de modulación como el valor complejo s_{kl} , en donde k es el índice de la subportadora y l es el índice temporal. Si $g(t)$ es un pulso de transmisión en banda base, entonces los filtros para la formación de pulsos son excitados por el flujo de datos de entrada creando réplicas que modulan las diferentes subportadoras y luego se suman antes de su transmisión.

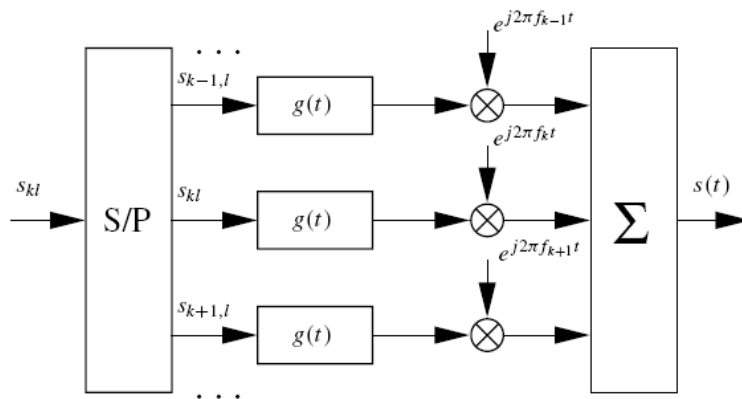


Figura 4.2. Diagrama de bloques de un sistema de modulación de múltiples subportadoras

La señal compleja en banda base es determinada por la expresión de la Ecuación 4.3, en donde T_s es la duración de los símbolos en cada uno de los flujos paralelos.

$$s(t) = \sum_k e^{j2\pi f_k t} \sum_l s_{kl} g(t - lT_s) \quad (4.3)$$

Si se reemplaza la expresión $g_{kl}(t) = e^{j2\pi f_k t} g(t - lT_s)$ en la Ecuación 4.3 se puede obtener la expresión de la Ecuación 4.4.

$$s(t) = \sum_{kl} s_{kl} g_{kl}(t) \quad (4.4)$$

Esta nueva expresión corresponde a una aproximación alternativa del modulador como se muestra en la Figura 4.3.

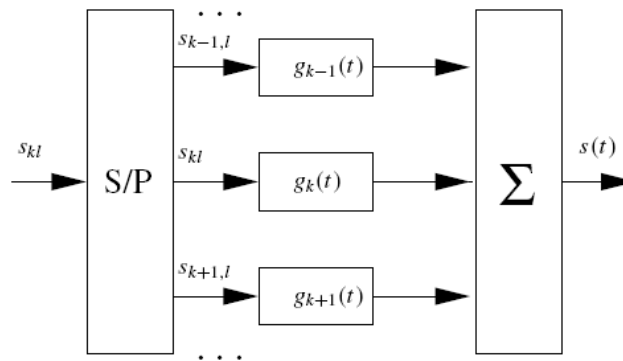


Figura 4.3. Diagrama de bloques alternativo de sistema de modulación de múltiples subportadoras

4.1.2 Ortogonalidad de las subportadoras

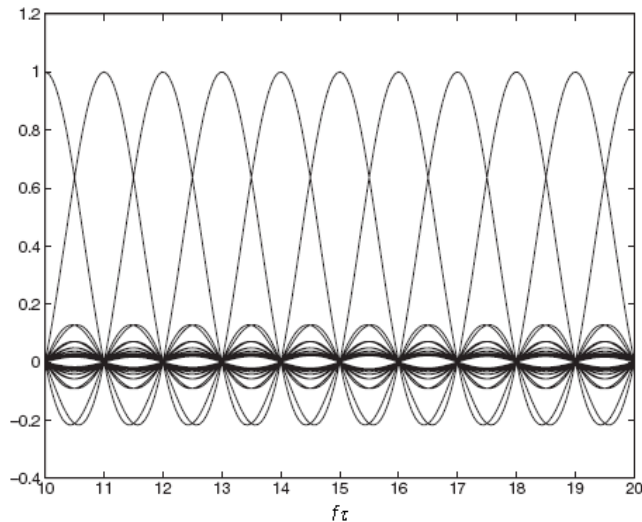


Figura 4.4. Espectro de múltiples subportadoras ortogonales en frecuencia.

Suponiendo que la función $g_{kl}(t)$ corresponde a una señal rectangular con duración T_S , el espectro de la señal OFDM corresponde al de la Figura 4.4 en donde se observa que el espectro de las subportadoras se traslapa.

Para demodular sin interferencia cada uno de los símbolos, se requiere que estos sean ortogonales en el dominio del tiempo y de la frecuencia, para esto se debe cumplir la expresión de la Ecuación 4.5.

$$g_{kl} \cdot g_{k'l'} = 0 \quad \text{si} \quad (k \neq k') \cup (l \neq l') \quad (4.5)$$

En principio las señales de pulsos $g_k(t)$ o $g_l(t)$ son ortogonales si no se traslapan en el tiempo ni en frecuencia respectivamente, sin embargo es imposible lograr que dos señales no se traslapen en tiempo y frecuencia a la vez [36]. El criterio de Nyquist define que si una señal cuyo valor en $t=0$ es diferente de cero y su valor es cero para todos los instantes $t = nT_S$, en donde $n = \pm 1, \pm 2, \dots$ entonces la señal es ortogonal en el dominio del tiempo, es decir que cumple con la expresión de la Ecuación 4.6.

$$g_{ol} \cdot g_{o'l'} = 0 \quad \text{si} \quad l \neq l' \quad (4.6)$$

Mediante el teorema de Parseval se puede demostrar que si una familia de señales cumple en el dominio de la frecuencia lo que establece el criterio de Nyquist para las funciones de pulso en el dominio del tiempo, entonces la familia de funciones $G_k(f)$ es ortogonal entre si y por lo tanto cumple con la expresión de la Ecuación 4.7 [36].

$$G_k \cdot G_{k'} = g_k \cdot g_{k'} = \delta_{kk'} \quad \text{si} \quad k \neq k' \quad (4.7)$$

Definiendo el tiempo de símbolo de tal manera que $T_S = 1/\Delta_f$, en donde Δ_f corresponde a la separación en frecuencia de dos subportadoras adyacentes, se puede lograr que la familia de funciones definida en la Ecuación 4.8 corresponda a un conjunto de funciones ortogonales en el dominio de la frecuencia para cada valor de k y ortogonales en el dominio del tiempo para cada valor de l .

$$g_{kl}(t) = g_k(t - lT_S) \quad (4.8)$$

4.1.3 Prefijo cíclico

Cuando la magnitud de la dispersión temporal del canal τ_m tiene un valor considerable respecto al tiempo de símbolo T_S que se describió en la sección anterior, se experimenta el fenómeno conocido como ISI que produce la pérdida de la ortogonalidad entre los símbolos de cada subportadora. Para mitigar este fenómeno se utiliza una técnica llamada inserción de prefijo

cíclico que consiste en introducir antes del inicio de cada segmento de la señal que representa un símbolo una copia del final del mismo segmento como se muestra en la Figura 4.5 [36].

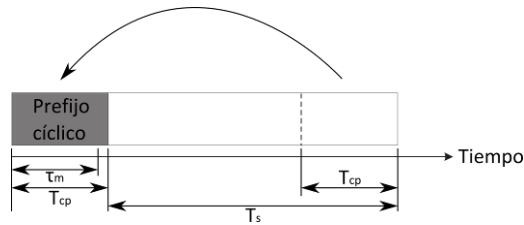


Figura 4.5. Inserción del prefijo cíclico

Para evitar la interferencia entre símbolos, la magnitud de τ_m debe ser inferior a la duración del prefijo cíclico. Esta técnica claramente reduce la capacidad de transmisión del canal que en la práctica se representa como una pérdida de potencia y que produce una disminución de la relación señal a ruido como se observa en la Ecuación 4.9 [37].

$$SNR_{Loss} = 10 \log \left(1 - \frac{T_{cp}}{T_s + T_{cp}} \right) \quad (4.9)$$

4.2 APLICACIÓN SCA PARA EL ANÁLISIS DE SEÑALES OFDM

Según la arquitectura propuesta en el Capítulo 1, se puede integrar una aplicación SCA diseñada para realizar el análisis de cualquier tipo de señal. En esta sección se describe la implementación de una aplicación SCA para el análisis de señales OFDM. Todos los algoritmos que se exponen a continuación han sido validados a través de un modelo en MATLAB que se incluye en el CD anexo que contiene el código fuente del proyecto.

En la Figura 4.6 se muestra el diagrama de bloques de la aplicación para el análisis de señales OFDM. Cada uno de los bloques corresponde a uno de los componentes de la aplicación SCA e implementa diferentes algoritmos de procesamiento de señales que en conjunto proporcionan las funcionalidades necesarias para realizar un análisis completo de la señal de entrada.

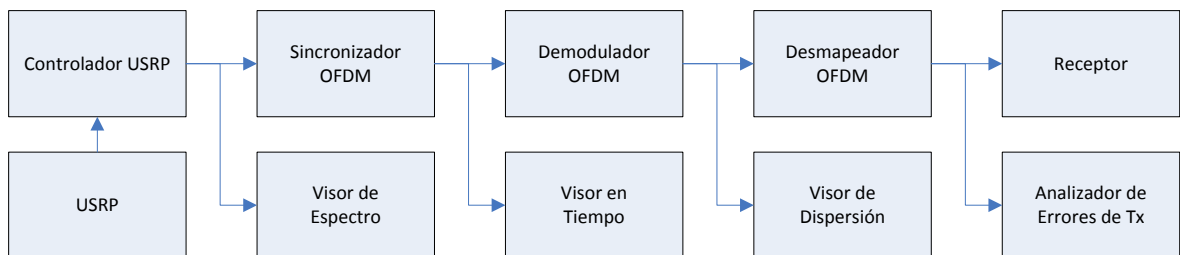


Figura 4.6. Diagrama de bloques de la aplicación SCA para el análisis de señales OFDM

En resumen, la señal de radiofrecuencias es capturada, convertida a banda base y digitalizada por el USRP, el Controlador del USRP entrega la señal a los diferentes bloques que realizan el procesamiento y análisis de la señal. Una copia de la señal de entrada es analizada por el Visor de

Espectro para presentar su máscara espectral. El Sincronizador OFDM tiene por objeto determinar el instante preciso en el que debe comenzar la detección de cada símbolo de la señal de entrada. Una vez se obtiene la información de sincronización, el Visor en Tiempo presenta la señal de entrada en el dominio del tiempo. El Demodulador OFDM realiza la demodulación de la señal mediante la transformada FFT (“*Fast Fourier Transform*”) y entrega una serie de símbolos complejos que son presentados por el Visor de Dispersión. El Desmapeador OFDM convierte los símbolos complejos entregados por el Demodulador OFDM a un flujo binario que contiene la información que requieren los niveles superiores de la aplicación de recepción y también el Analizador de Errores de Recepción para determinar cuál es la tasa de errores en el Receptor.

Existen múltiples implementaciones de sistemas de recepción basados en OFDM de acuerdo a los requerimientos específicos de la aplicación. Para simplificar la implementación del modelo, se asume que el ancho de banda de la señal OFDM es menor al ancho de banda coherente del canal de comunicaciones y por lo tanto que la respuesta en frecuencia del canal es plana para todas las componentes de la señal. Bajo esta suposición se omite el ecualizador adaptativo y no es necesario realizar la inserción de pilotos en las subportadoras. Se asume también que se ha insertado un tiempo de guarda para cada símbolo OFDM mediante la técnica de inserción de prefijo cíclico y como se podrá observar más adelante el Sincronizador OFDM fue diseñado específicamente para obtener la información de temporización a partir del mismo prefijo cíclico.

A continuación se hace una descripción detallada de cada uno de los componentes que conforman los bloques de procesamiento de señal, de los algoritmos que implementan, de los parámetros de funcionamiento y de los puertos de entrada y salida para cada uno de ellos. Para interpretar adecuadamente el funcionamiento de cada uno de estos componentes, se debe tener en cuenta que los nombres de las señales de entrada y salida se asignan de acuerdo a su función dentro del mismo bloque y aunque diferentes bloques tienen puertos con el mismo nombre, no significa que comparten la misma señal.

4.2.1 El Controlador USRP

El Controlador USRP es el componente SCA que se ha implementado en OSSIE para proporcionar una interfaz con el hardware USRP. Este componente permite establecer los parámetros de operación del USRP a través de sus propiedades. A continuación se hace una descripción de cada una de ellas.

rx_freq y tx_freq: Establece la frecuencia de transmisión y de recepción del USRP, estas frecuencias pueden ser diferentes para la operación del dispositivo en modo *full duplex* y su rango de valores depende del rango de operación en frecuencia de la tarjeta de interfaz de RF instalada en el USRP. Las características de las diferentes tarjetas de interfaz de RF para el USRP, incluyendo sus frecuencias de operación, se han enumerado en la Sección 2.2.2.

tx_interp y rx_decim: Permiten establecer la tasa del interpolador en la sección de transmisión y la tasa de diezmado en la sección de recepción, para establecer la velocidad de las muestras de la señal y por lo tanto para establecer su ancho de banda.

rx_size: Indica el número de muestras de la señal de entrada que se almacenan en el buffer de memoria antes de su procesamiento.

rx_gain y rx_gain_max: El primer parámetro establece la ganancia del amplificador programable que se encuentra antes del ADC (“Analog to Digital Converter”), el segundo parámetro establece la ganancia del amplificador en su valor máximo ignorando la ganancia especificada en el primer parámetro.

tx_start y rx_start: Estos dos parámetros habilitan o deshabilitan el funcionamiento de las secciones de transmisión y de recepción de forma independiente.

4.2.2 Sincronizador OFDM

El sincronizador es uno de los elementos más complejos en un sistema de comunicaciones OFDM, el cual tiene como función recuperar la información de sincronismo de la señal de entrada, existen diferentes implementaciones de mecanismos de sincronización que utilizan diversas técnicas. La implementación que se ha realizado para esta aplicación está basada en el método de estimación de máxima verosimilitud para el desplazamiento en tiempo y en frecuencia para sistemas OFDM, propuesto por Jan-Jaap van de Beek en un artículo publicado por el IEEE [39]. La Figura 4.7 muestra el diagrama de bloques del sincronizador en donde $s_i(k)$ corresponde a la señal de entrada y las señales $s_o(k)$ y $\Theta(k)$ corresponden a la señal de salida corregida en frecuencia y a la señal de sincronización respectivamente.

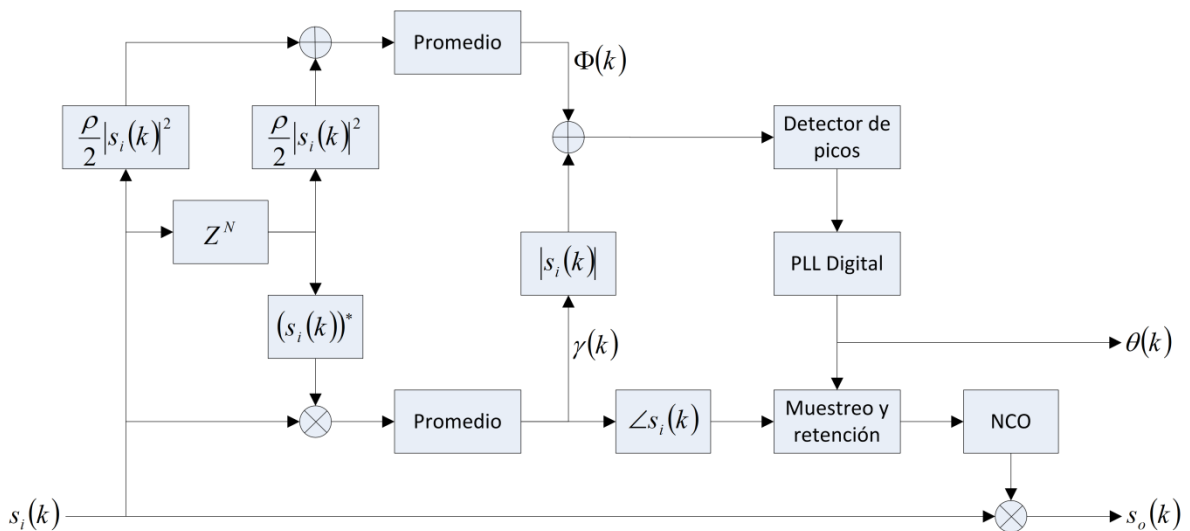


Figura 4.7. Diagrama de bloques del sincronizador OFDM

El componente de sincronización de señales OFDM posee un puerto para la señal de entrada y dos puertos de salida para la señal de corregida en frecuencia y la señal de sincronización. Las propiedades que se han implementado en este componente se describen a continuación.

Tamaño de la FFT: La longitud de cada símbolo OFDM está determinada por la longitud de la FFT más la longitud del prefijo cíclico, este parámetro corresponde al número de muestras de la FFT

que se utilizan para demodular la señal y se utiliza para establecer la longitud del filtro de retraso Z^N .

Longitud del prefijo cíclico: Especifica el número de muestras de cada símbolo que corresponden al prefijo cíclico y se utiliza para establecer la longitud de los filtros de promedio móvil cuya función de transferencia es $(1/M)[Z + Z^1 + \dots + Z^M]$ donde M corresponde a la longitud del filtro.

Relación señal a ruido: La relación señal a ruido se obtiene de comparar la señal recibida con una señal de referencia obtenida mediante la regeneración de la señal a partir del flujo de datos después de la demodulación. Este parámetro se utiliza en este componente para calcular el valor óptimo del factor $\rho = SNR / (SNR + 1)$ para el algoritmo de sincronización de la señal.

4.2.3 Demodulador OFDM

La demodulación de la señal OFDM se hace por medio de la FFT que convierte la secuencia de muestras obtenidas al separar la señal de entrada $s_i(k)$ en segmentos demarcadas por la señal de sincronismo $\Theta_i(k)$, en una serie de números complejos $S_o(k)$ que representan los símbolos con los que se ha modulado cada una de las subportadoras.

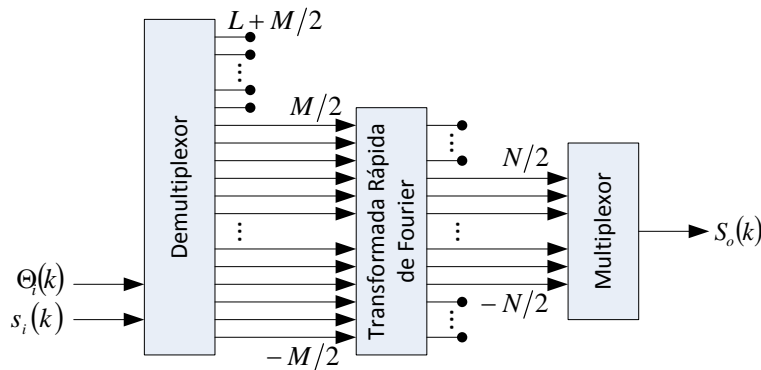


Figura 4.8. Diagrama de bloques del demodulador OFDM

A continuación se describen las dos propiedades implementadas para el demodulador OFDM.

Tamaño de la FFT: Corresponde al parámetro del mismo nombre descrito en el componente Sincronizador OFDM y se utiliza para establecer el tamaño de la FFT que se utiliza para demodular cada uno de los símbolos OFDM que contiene la señal de entrada.

Número de subportadoras: Para demodular una señal OFDM se requiere que el tamaño de la FFT sea igual o mayor al número de subportadoras de la señal. En la práctica se utiliza una FFT con un tamaño superior al número de subportadoras, con el objetivo de simplificar el diseño de los filtros *antialiasing* y de diezmo disminuyendo la pendiente requerida en la frecuencia de corte de estos. El valor N corresponde al número de subportadoras que se ha utilizado en el proceso de modulación y M al tamaño de la FFT utilizada de tal manera que se deben descartar $(M - N)/2$ muestras en cada extremo del arreglo de salida de la FFT, debido a que no contienen información.

4.2.4 Desmapeador de Símbolos OFDM

El Desmapeador de Símbolos OFDM convierte los valores complejos de la señal de salida del Demodulador OFDM y los convierte en una secuencia de bits que corresponde al símbolo más probable de acuerdo con el esquema de modulación que se utilizó en cada una de las subportadoras. También es función de este componente determinar cuáles deben ser los umbrales de los símbolos de acuerdo a la cantidad de atenuación que pueda presentar la señal de entrada. El Desmapeador de Símbolos OFDM implementa una sola propiedad que se describe a continuación.

Esquema de modulación: Esta propiedad corresponde al identificador del esquema de modulación M-ario con el que se moduló las subportadoras de la señal OFDM, para determinar cuál es la constelación de símbolos con los que se debe desmapear los valores de la señal entregados por el Demodulador OFDM.

4.2.5 Visor de Espectro

El visor de espectro hace parte de los componentes SCA que se han diseñado especialmente para interactuar con la interfaz de usuario y visualizar los resultados del análisis espectral. El demultiplexor toma muestras de la señal de entrada $s_i(k)$ que posteriormente son ponderadas a través de una función de ventana. El análisis espectral de las muestras ponderadas se realiza a través de una FFT de N puntos y el arreglo de salida de la transformada es reorganizado para ordenar cada el elemento según la frecuencia que este representa en un rango de $-f_s/2$ hasta $f_s/2$. Las muestras reorganizadas son multiplicadas por un factor de conversión y finalmente los valores resultantes se visualizan en un elemento de la interfaz de usuario.

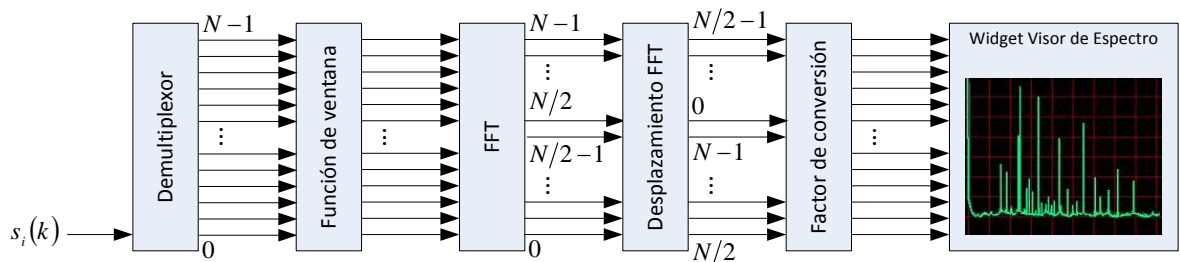


Figura 4.9. Diagrama de bloques del componente visor de espectro

El visor de espectro implementa las propiedades que se describen a continuación.

Tamaño de la FFT: El tamaño de la FFT corresponde al número de muestras utilizadas para realizar el análisis espectral de la señal de entrada. El número de puntos utilizados para realizar la FFT define el tamaño y la resolución en frecuencia del arreglo de salida. Este parámetro no tiene relación con el tamaño de la transformada utilizada durante la modulación de la señal

Función de ventana: Determina es la función de ventana con la cual se ponderan los valores de la secuencia de la señal de entrada antes de realizar la FFT. Existen varias funciones de ventana, cada una con propiedades que proporcionan diferentes ventajas y limitaciones.

Factor de conversión: Este valor describe la relación de equivalencia entre la amplitud de las muestras que han resultado de la FFT y las unidades en las que se presentan los datos de salida. Este valor se puede obtener a partir de la caracterización de cada uno de los elementos de la sección de recepción del hardware utilizado y de los diferentes componentes que realizan el procesamiento de la señal. También se puede obtener a partir de la caracterización de todos los elementos de la sección de transmisión a partir de una señal de prueba con características conocidas

Cuando los elementos de la sección de recepción se consideran lineales, el factor de conversión corresponde a un número real. Si se considera la no linealidad de estos componentes, entonces el factor de corrección debe determinarse mediante una función que considera cada una de las no linealidades de los componentes y que establece el factor de conversión según los parámetros de operación de estos componentes no lineales y la frecuencia y potencia de la señal de entrada.

4.2.6 Visor en el Dominio del Tiempo

Este componente SCA ha sido diseñado para presentar de forma gráfica los datos del análisis de una señal en el dominio del tiempo y su funcionamiento es bastante simple. El demultiplexor toma una secuencia de N muestras de la señal de entrada las cuales son multiplicadas por un factor de conversión y luego se visualiza su amplitud con respecto al tiempo. Si la señal es real, se presenta un solo trazo y si la señal es compleja se presentan dos trazos, uno correspondiente a la componente real y el otro correspondiente a la componente imaginaria.

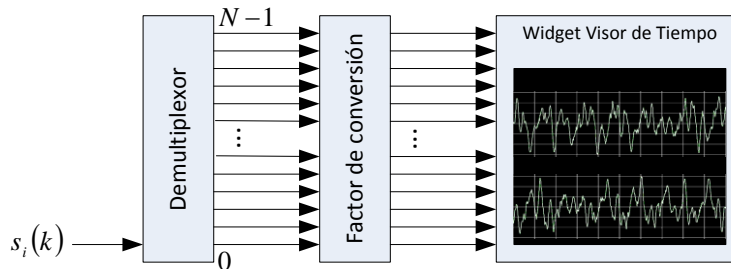


Figura 4.10. Diagrama de bloques del visor en tiempo

El componente visor de tiempo implementa las dos propiedades que se exponen a continuación.

Número de muestras: Esta propiedad establece el número de muestras que se toman de la señal para su visualización en la interfaz de usuario. Este parámetro establece de forma indirecta el rango de tiempo, para la visualización de la señal, mediante la relación $T = N/f_s$ donde T es la duración en tiempo del segmento de la señal de entrada, N el número de muestras y f_s la frecuencia de muestreo de la señal.

Factor de conversión: El factor de conversión corresponde a la relación de equivalencia entre los valores de la señal de entrada y las unidades utilizadas para la representación de la señal en la interfaz de usuario. Este parámetro es similar a la propiedad del mismo nombre utilizada en el componente Visor de Espectro que se describe en la sección 4.2.5.

4.2.7 Visor de Dispersión

Este componente presenta en forma gráfica la ubicación en el plano I/Q de cada uno de los símbolos de la señal OFDM. Para esto, el componente toma N símbolos de la secuencia de obtenida en la salida del demodulador OFDM, los multiplica por un factor de conversión entre la escala de medición y el valor de la muestra, finalmente los dibuja sobre un plano en donde el valor I corresponde a la parte real y Q corresponde a la parte imaginaria del cada uno de los símbolos.

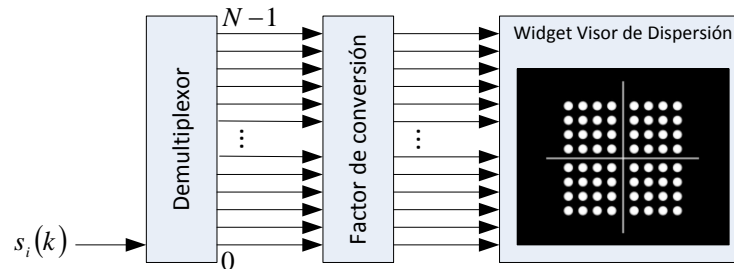


Figura 4.11. Diagrama de bloques del visor de dispersión

Numero de muestras: Este parámetro especifica el número de muestras que se presentan simultáneamente en el diagrama de dispersión o plano IQ.

Factor de conversión: Similarmente a la propiedad del mismo nombre en los componentes Visor de Espectro y Visor en Tiempo, este parámetro relaciona la magnitud de los valores de las muestras de la señal de entrada y las unidades utilizadas en el diagrama de dispersión que se presenta en la interfaz de usuario.

4.2.8 Detector de errores en recepción

El detector de errores es un componente que verifica la integridad de los datos recibidos mediante diferentes esquemas de redundancia y mecanismos de verificación de errores. La implementación de este componente depende totalmente de las especificaciones de la tecnología cuyas señales se van a analizar. En este caso se ha implementado un detector de errores muy simple que funciona con un mecanismo de verificación de redundancia cíclica, sin embargo podría implementarse un mecanismo más complejo como el codificador Reed-Solomon por ejemplo.

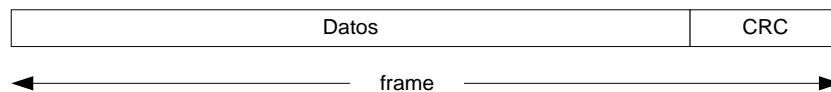


Figura 4.12. Esquema de un *frame* con control de errores por CRC

Para realizar la verificación de los datos, en primer lugar se debe detectar la posición de cada *frame*, a esto se le llama sincronización de *frames*. Para lograr esta sincronización, se toma una secuencia de la misma longitud del *frame* proveniente de la señal de entrada y se calcula el valor del CRC (“*Cyclic Redundancy Check*”), si el resultado es cero, entonces se ha detectado un *frame*, de lo contrario se repite la operación hasta encontrar un nuevo *frame*. Este componente visualiza

en la interfaz de usuario la tasa de *frames* erróneos, que obtiene verificando la distancia que existe entre los *frames* detectados.

A continuación se exponen las dos propiedades que se implementan en el detector de errores en recepción.

Longitud del *frame*: Esta propiedad establece la cantidad de muestras que conforman cada *frame* del flujo de datos contenido en la señal analizada. Esta longitud incluye las muestras de los datos y también las muestras que contienen el resultado del CRC.

Algoritmo de detección de errores: Este parámetro define el algoritmo con el que se ha generado la suma de verificación de redundancia cíclica. El tamaño que ocupa el CRC dentro del *frame* se define de acuerdo al algoritmo seleccionado en esta propiedad.

4.3 LA INTERFAZ GRÁFICA DE USUARIO DE LA APLICACIÓN DE ANÁLISIS DE SEÑALES

En los capítulos y las secciones anteriores se describió la arquitectura y el diseño de un prototipo para el análisis de señales OFDM y se hizo énfasis en los aspectos generales como la estructura de clases y los componentes de la aplicación SCA para el análisis de una señal OFDM. A continuación se describe la interfaz de usuario que se implementó con el prototipo de análisis de señales OFDM

Para el desarrollo del prototipo de análisis de señales OFDM se implementó la interfaz gráfica que se muestra en la Figura 4.13. Al lado izquierdo se localiza el panel de propiedades en donde se establecen los parámetros de funcionamiento de la Aplicación SCA además de algunos parámetros relacionados con la visualización de la información y a la derecha se encuentra el panel de visualización de la señal, compuesto por cuatro *widgets* de los cuales los tres primeros muestran información obtenida del análisis de la señal en modo de gráficos y el cuarto presenta de forma textual los resultados obtenidos a partir del análisis de la señal.

Existen muchos tipos de análisis específicos que se pueden realizar sobre una señal OFDM y existen muchas representaciones visuales de estos, en esta aplicación se han implementado cuatro que muestran la señal en el dominio de la frecuencia, la señal en banda base en el dominio del tiempo, la dispersión que sufren los símbolos transmitidos por un canal no ideal y la tasa de errores en recepción. Con el diseño y la arquitectura que se ha planteado en el desarrollo de este proyecto se pueden implementar nuevas funcionalidades de análisis y nuevas formas de presentar visualmente esta información.

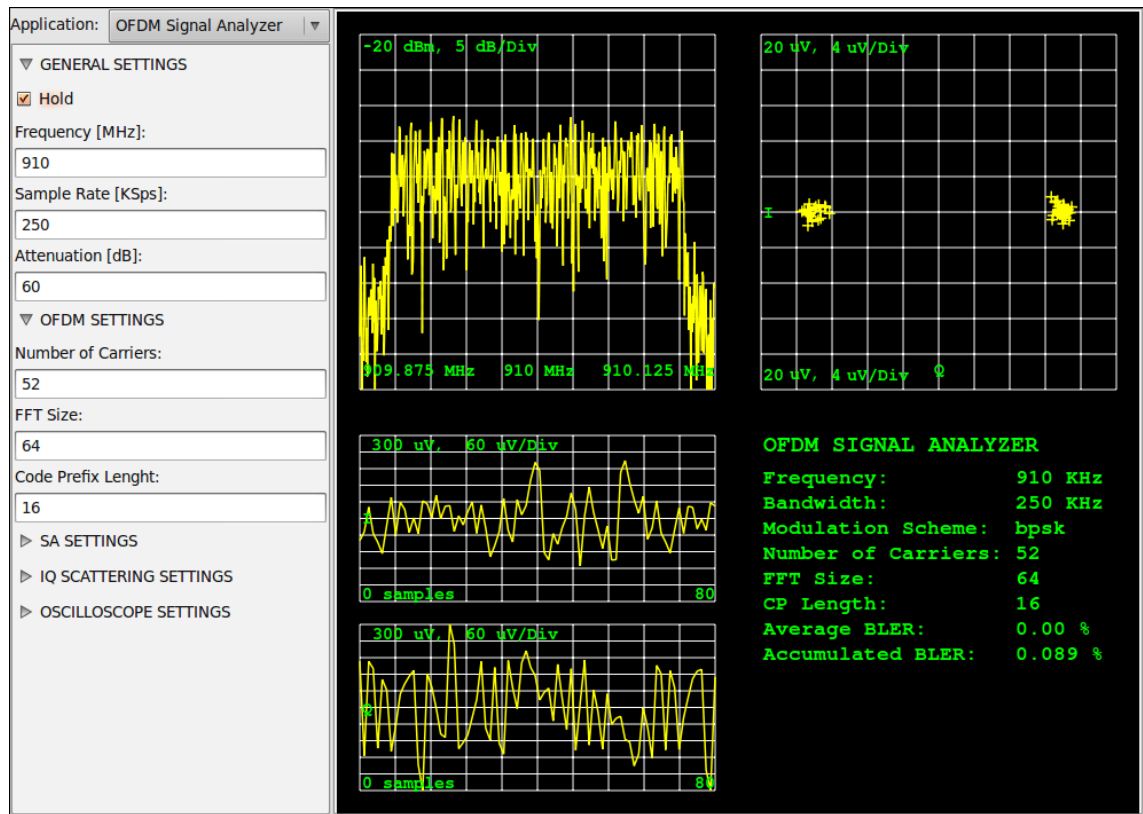


Figura 4.13. Interfaz gráfica de usuario de la aplicación de análisis de señales OFDM

En la implementación de los componentes SCA debe considerarse que estos se implementan como elementos separados que se comunican entre sí mediante CORBA. Estos componentes no comparten el espacio de direcciones con la aplicación que implementa la interfaz de usuario y por lo tanto no pueden interactuar con los elementos de la interfaz gráfica. La solución que se ha adoptado en la implementación del prototipo consiste en utilizar mecanismos de memoria compartida para transferir la información obtenida del análisis de la señal desde el componente SCA hacia la interfaz de usuario. Una solución más adecuada consiste en que la aplicación de la interfaz de usuario cree y registre sus propios componentes SCA en el *Device Manager* de tal manera que los componentes SCA y la interfaz de usuario comparten el mismo espacio de direccionamiento simplificando el desarrollo de aplicaciones, sin embargo esta solución implica modificar el código del *Device Manager* implementado en el CF de OSSIE y esto requiere de un esfuerzo considerable que se deja como una de las posibilidades para continuar con futuros trabajos de grado basados en el presente.

5. VERIFICACIÓN Y COMENTARIOS DE LA IMPLEMENTACIÓN DEL ANALIZADOR DE SEÑALES OFDM

En este capítulo se describen los procedimientos de verificación de la solución y los resultados obtenidos. Se describe también cómo algunas expectativas que se tenían acerca del hardware USRP (*“Universal Software Radio Peripheral”*) no se cumplieron obligando a replantear algunos de los procedimientos de verificación considerados inicialmente. También se exponen algunos aspectos que merecieron especial consideración durante el proceso de desarrollo y merecen ser expuestos como referencia para quienes trabajen sobre lo planteado en este trabajo o en proyectos similares que utilicen las mismas herramientas. Los detalles específicos de la implementación han sido documentados en los diferentes anexos y en el código fuente del software desarrollado.

5.1 VALIDACIÓN DE LOS COMPONENTES SCA DEL PROTOTIPO DE ANÁLISIS DE SEÑALES OFDM

La implementación del prototipo de análisis de señales OFDM (*“Orthogonal Frequency Division Multiplexing”*) se hizo basándose en el modelo de un modulador y demodulador de señales OFDM realizado en MATLAB, cuyo código fuente está incluido en el disco que se proporciona como Anexo. En la Figura 5.1 se observa el diagrama de bloques con los diferentes algoritmos implementados en el modelo.

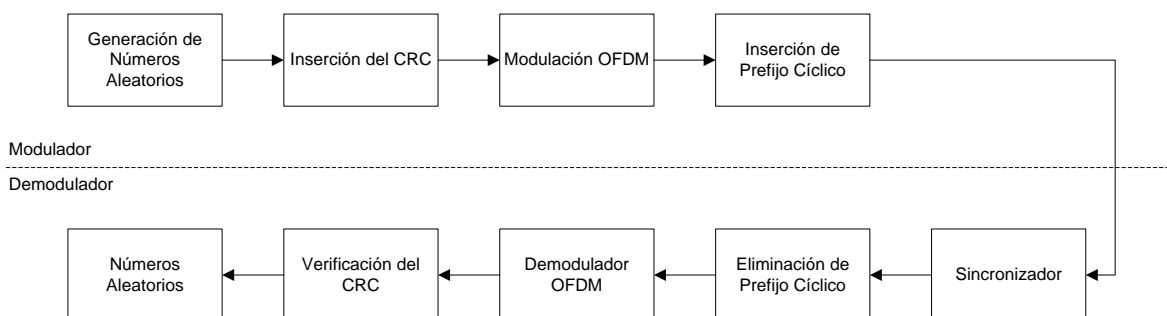


Figura 5.1. Modelo del Modulador y Demodulador OFDM

La verificación del modelo OFDM se realizó comparando la señal de entrada en cada bloque del modulador con la señal de salida de su contraparte en el demodulador. Para la verificación de la funcionalidad de cada uno de los componentes SCA (*“Software Communications Architecture”*) que conforman la aplicación de análisis de señales OFDM, se utilizó una señal digital de prueba, generada con el modelo del modulador, para que cada uno de los componentes de la aplicación SCA dejara en un archivo el registro de la secuencia de valores que recibió a la entrada y la secuencia de valores que obtuvo a la salida.

Al comparar valor por valor, las secuencias registradas por cada componente SCA con las secuencias obtenidas de procesar la misma señal en el modelo de MATLAB, se obtuvieron resultados satisfactorios con diferencias mínimas debidas a la implementación diferente de las funciones matemáticas en MATLAB y en las librerías de C++.

5.1.1 Características de la señal de prueba utilizada para validar el modelo

Para generar la señal digital de prueba se utilizó el modelo del modulador OFDM con los parámetros de operación que se muestran en la Tabla 5.1:

Esquema de modulación de subportadoras	BPSK
Tamaño de la IFFT del modulador OFDM	64
Número de subportadoras OFDM	52
Tamaño del prefijo cíclico	16 bits
Tamaño del <i>frame</i>	52 bits
Algoritmo de verificación de errores	CRC16

Tabla 5.1. Parámetros para la generación de la señal OFDM

En la Figura 5.2 se muestra la secuencia de símbolos BPSK obtenida a partir de la secuencia de bits aleatoria generada en el modulador después de su división en *frames* y la inserción de la suma de verificación. Se puede observar que la parte real de la secuencia corresponde a una secuencia aleatoria de ceros y unos y la parte imaginaria es cero para todos los valores, como corresponde para una señal.

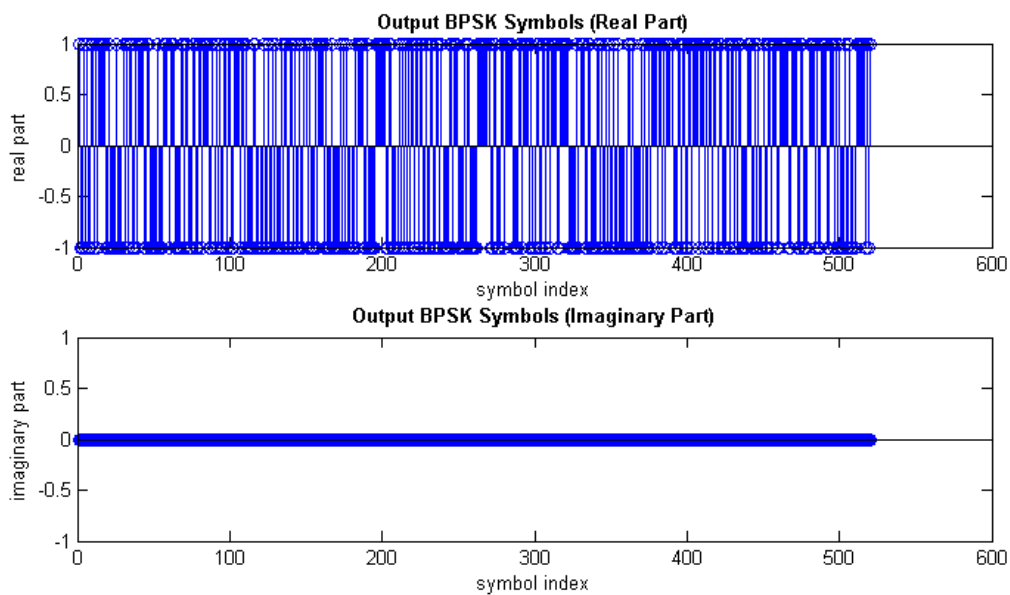


Figura 5.2. Secuencia de símbolos para el modulador OFDM

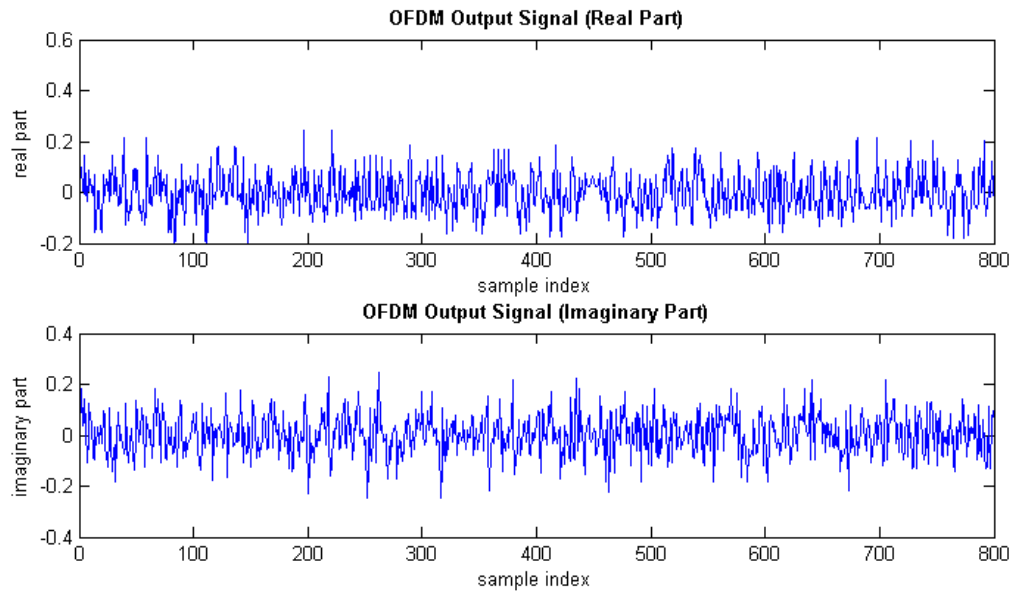


Figura 5.3. Señal en el dominio del tiempo a la salida del modulador OFDM

La Figura 5.3 y la Figura 5.4 muestran la señal a la salida del modulador OFDM en el dominio del tiempo y de la frecuencia respectivamente. En la imagen del espectro de la señal se puede observar que el ancho de banda de la señal corresponde al 81.25 % de la frecuencia de Nyquist y que la portadora central ha sido suprimida con el objetivo de simplificar el diseño del receptor.

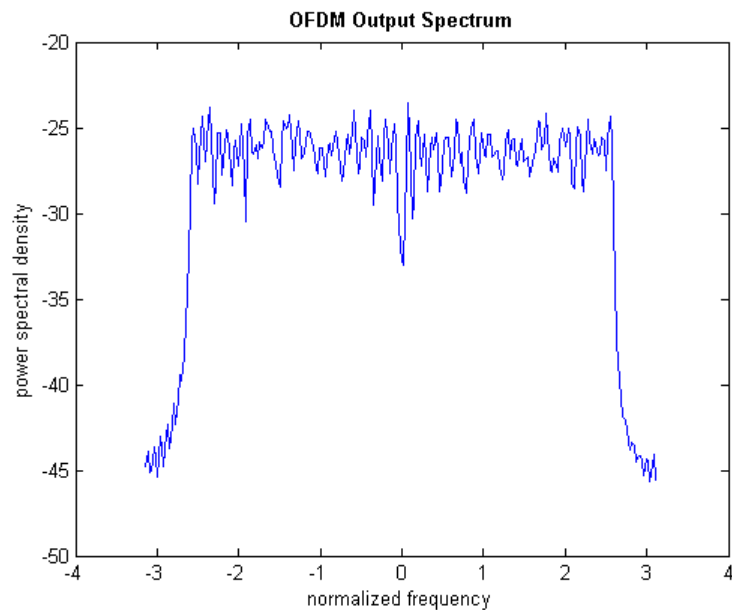


Figura 5.4. Espectro de la señal a la salida del modulador OFDM

La Figura 5.5 muestra la señal de sincronización que se obtiene en el sincronizador OFDM para demodular la señal correctamente, se puede observar que cada pulso está separado por 80 muestras que corresponde al valor del tamaño de la IFFT (“Inverse Fast Fourier Transform”) más el número de muestras del prefijo cíclico.

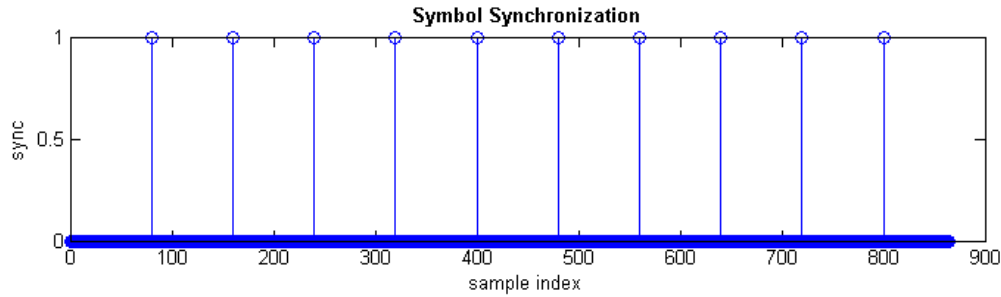


Figura 5.5. Señal de sincronización de símbolos OFDM

En la Figura 5.6 y la Figura 5.7 se muestran la secuencia de símbolos BPSK obtenida a la salida del demodulador OFDM y el diagrama de dispersión de la misma. En las imágenes se puede apreciar que la parte real corresponde a una secuencia de muestras con valor unitario, positivas y negativas y que la parte imaginaria está compuesta de muestras con un valor muy cercano a cero, esto se debe a errores despreciables que se introducen en el cálculo de la IFFT en el modulador OFDM y de la FFT (“Fast Fourier Transform”) en el demodulador.

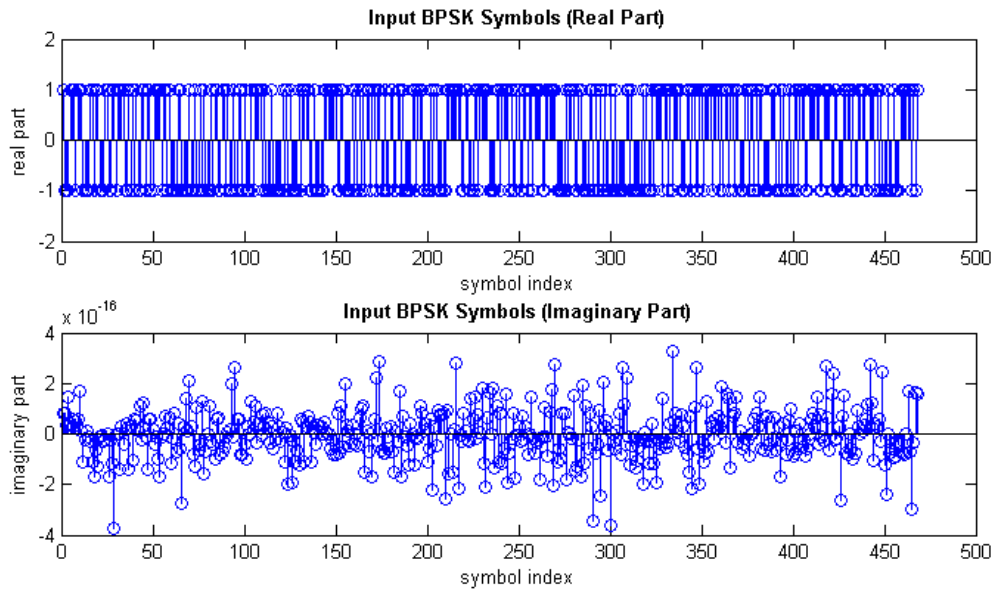


Figura 5.6. Secuencia de símbolos a la salida del demodulador OFDM

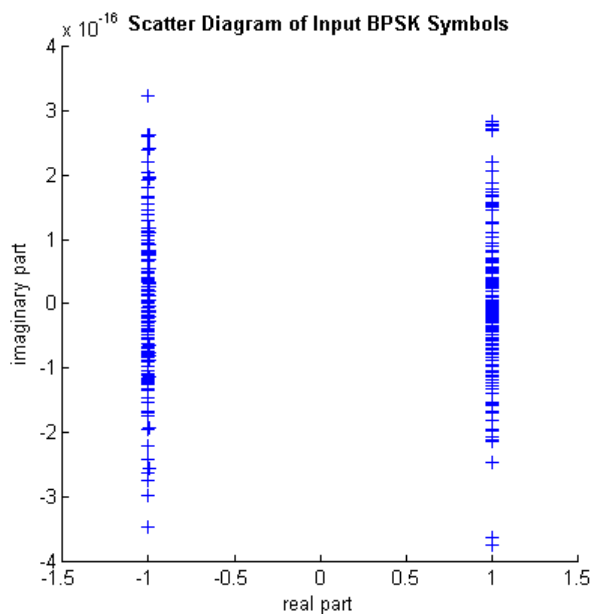


Figura 5.7. Diagrama de dispersión de la señal a la salida del demodulador OFDM

En la Figura 5.8 se observa la señal de sincronización de *frames* que se obtiene después de la verificación del CRC, la longitud entre pulsos es de 52 muestras y corresponde a la misma longitud de *frame* que se especificó en la Tabla 5.1. La tasa de *frames* erróneos se puede calcular verificando cuantos pulsos hacen falta en la señal de sincronización teniendo en cuenta que cada uno de debe estar separado del siguiente por un valor equivalente al tamaño del *frame*.

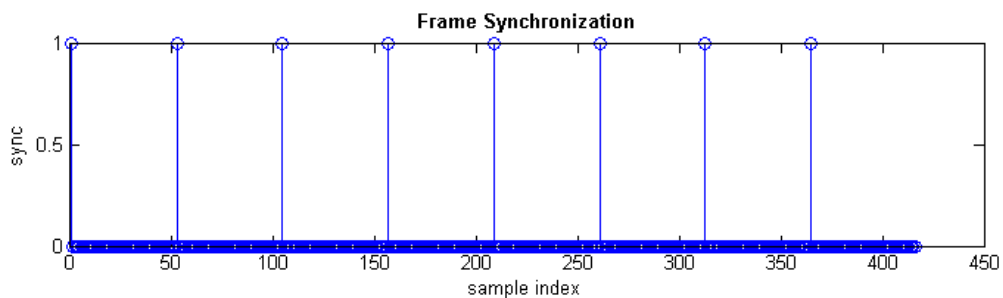


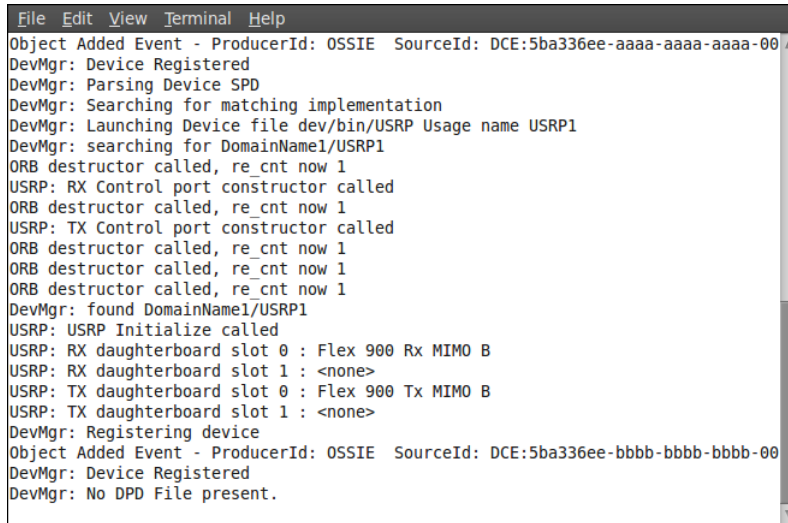
Figura 5.8. Señal de sincronización de *frames*

5.2 VERIFICACIÓN DEL FUNCIONAMIENTO DEL PROTOTIPO DE ANÁLISIS DE SEÑALES OFDM

En los Capítulos 1 y 1 se realizó el diseño del software de la aplicación de análisis de señales utilizando la arquitectura SCA, la implementación en código fuente puede encontrarse en el disco compacto proporcionado en los anexos. Para realizar la verificación del funcionamiento del prototipo de análisis de señales OFDM se realizaron las actividades que se enumeran a continuación.

5.2.1 Verificación de la conectividad entre el hardware USRP y el Computador Personal

El primer paso para comprobar la operación del prototipo consiste en verificar que el software desarrollado interactúa de forma adecuada con el hardware USRP. Para ello se instaló el paquete de software GNU Radio que contiene los controladores necesarios para configurar el hardware del USRP y para enviar y recibir el flujo de muestras digitales que contienen las señales para su transmisión y recepción a través de la interfaz de radio del USRP.



```
File Edit View Terminal Help
Object Added Event - ProducerId: OSSIE SourceId: DCE:5ba336ee-aaaa-aaaa-aaaa-00
DevMgr: Device Registered
DevMgr: Parsing Device SPD
DevMgr: Searching for matching implementation
DevMgr: Launching Device file dev/bin/USRP Usage name USRP1
DevMgr: searching for DomainName1/USRP1
ORB destructor called, re_cnt now 1
USRP: RX Control port constructor called
ORB destructor called, re_cnt now 1
USRP: TX Control port constructor called
ORB destructor called, re_cnt now 1
ORB destructor called, re_cnt now 1
ORB destructor called, re_cnt now 1
DevMgr: found DomainName1/USRP1
USRP: USRP Initialize called
USRP: RX daughterboard slot 0 : Flex 900 Rx MIMO B
USRP: RX daughterboard slot 1 : <none>
USRP: TX daughterboard slot 0 : Flex 900 Tx MIMO B
USRP: TX daughterboard slot 1 : <none>
DevMgr: Registering device
Object Added Event - ProducerId: OSSIE SourceId: DCE:5ba336ee-bbbb-bbbb-bbbb-00
DevMgr: Device Registered
DevMgr: No DPD File present.
```

Figura 5.9. Mensajes de salida del *Device Manager* con la confirmación de la conexión con el USRP

El USRP no interactúa directamente con la aplicación de análisis de señales si no a través del componente Controlador USRP de la implementación del CF (“*Core Framework*”) de SCA implementado en OSSIE (“*Open Source SCA Implementation - Embedded*”). Al conectar el USRP con el computador personal a través del puerto USB (“*Universal Serial Bus*”) y ejecutar el CF de SCA, el *Device Manager* imprime en su salida el mensaje de la Figura 5.9 con el cual se puede verificar que el USRP ha sido detectado e inicializado correctamente.

Luego de verificar la instalación correcta de los controladores fue necesario verificar que se pueden modificar cada uno de los parámetros de funcionamiento desde la interfaz de usuario de la aplicación. Durante la realización de esta actividad se descubrió que el componente Controlador USRP implementado en OSSIE fue diseñado para manejar el USRP con las tarjetas hijas Basic Tx y Basic RX y no con la tarjeta RFX900 que se utilizó en el desarrollo de este trabajo de grado, las características de estas tarjetas se enumeran en la Sección 2.3.3. Este inconveniente no permitió configurar de forma adecuada los conectores de transmisión y recepción de la tarjeta RFX900. La solución consistió en revisar el diseño del hardware de la tarjeta y realizar las correcciones necesarias en el código fuente del componente Controlador USRP de la implementación de OSSIE para adaptarlo a las características de la nueva tarjeta de hardware.

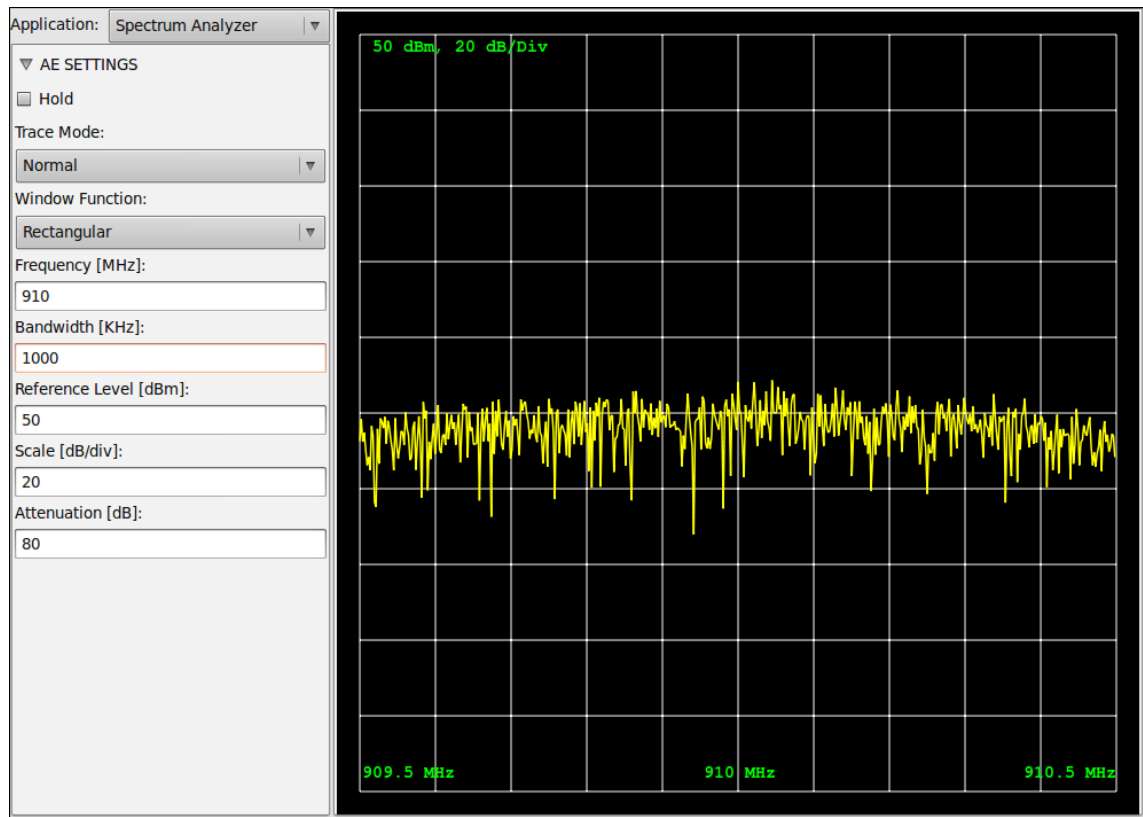


Figura 5.10. Configuración de los parámetros de operación del USRP

En la Figura 5.10 se muestra la interfaz gráfica de la aplicación de análisis de señales en modo de analizador de espectro que permite configurar todos los parámetros de operación del USRP y en la Figura 5.11 se muestran los mensajes de confirmación de la configuración del USRP que proporciona la consola del *Device Manager*.

```

File Edit View Terminal Help
USRP_Commander: Query()
USRP_Commander: Query()
USRP: Setting number of channels to 1
Component - USRP_Commander
  Props length = 9
normal configure call ...
USRP: In RX Control set frequency channel: 0, frequency: 9.1e+08
USRP: USRP RX tune_result: baseband_freq=9.06e+08 dxc_freq=-4e+06 residual_freq=
0 inverted=0
Property id : DCE:3efc3930-2739-40b4-8c02-ecfb1b0da9ee
USRP: In TX Control set frequency channel: 0, frequency: 9.1e+08
USRP: USRP TX tune_result: baseband_freq=9.14e+08 dxc_freq=-4e+06 residual_freq=
0 inverted=0
Property id : DCE:6a2d6952-ca11-4787-afce-87a89b882b7b
Property id : DCE:9ca12c0e-ba65-40cf-9ef3-6e7ac671ab5d
Property id : DCE:92ec2b80-8040-47c7-a1d8-4c9caa4a4ed2
Property id : DCE:93324adf-14f6-4406-ba92-a3650089857f
USRP: set gain to 0
Property id : DCE:99d586b6-7764-4dc7-83fa-72270d0f1e1b
Property id : DCE:2d9c5ee4-a6f3-4ab9-834b-2b5c95818e53
Property id : DCE:0a9b8c8c-f130-4a8f-9ef8-bba023128a4b
Property id : DCE:fd42344f-4d87-465b-9e6f-e1d7ae48afd6
USRP_Commander: Query()

```

Figura 5.11. Mensajes de confirmación del *Device Manager* acerca del cambio de parámetros del USRP

5.2.2 Recepción de una señal sinusoidal producida por un generador de señales de laboratorio.

La forma más sencilla de verificar si el USRP está interactuando correctamente con el software del prototipo de análisis de señales consiste en generar una señal sinusoidal con un generador de laboratorio y verificar que esta señal se puede observar en la interfaz de análisis de espectro de la aplicación.

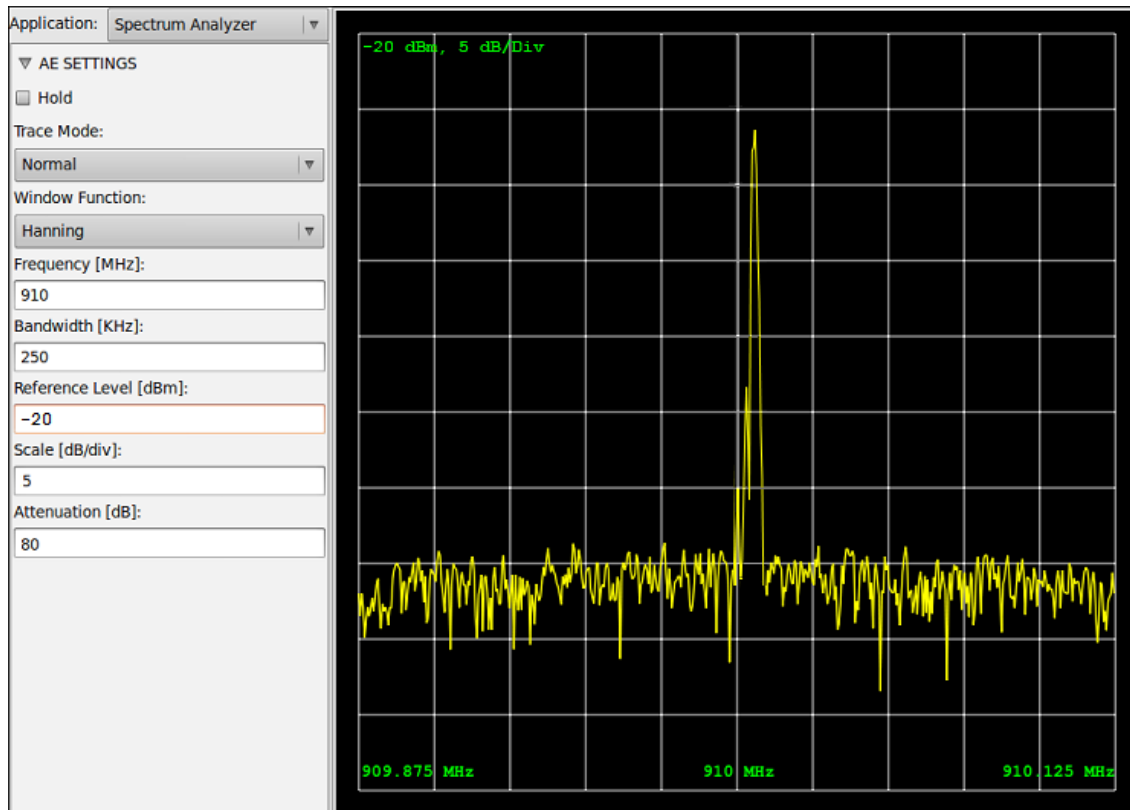


Figura 5.12. Espectro de una señal sinusoidal de prueba obtenido con el dispositivo de análisis de señales

En la Figura 5.12 se observa el trazo obtenido al realizar el análisis espectral de una señal sinusoidal de prueba en 910 MHz y se puede identificar que el USRP tiene un error de frecuencia, con un valor alrededor de los 5 KHz, valor considerable en aplicaciones de comunicaciones inalámbricas en donde algunas aplicaciones tienen canalizaciones con anchos de banda desde los 25 KHz.

Con esta prueba también se pudo observar que el USRP no implementa ningún mecanismo para evitar o detectar la saturación de la etapa analógica de entrada, de tal manera que no existe forma de saber si la ganancia con la que se configura la etapa de entrada del USRP es demasiado alta y produce saturación en el mezclador o en los convertidores ADC (*"Analog to Digital Converters"*). En la imagen de la Figura 5.13 se puede observar la falta de linealidad en frecuencia del USRP, el trazo proporciona una idea de la función de transferencia del hardware del USRP y se obtuvo al realizar

un barrido en frecuencia, con una señal sinusoidal de amplitud constante, y el modo de detección del analizador de señales establecido en *Maxhold*⁴.

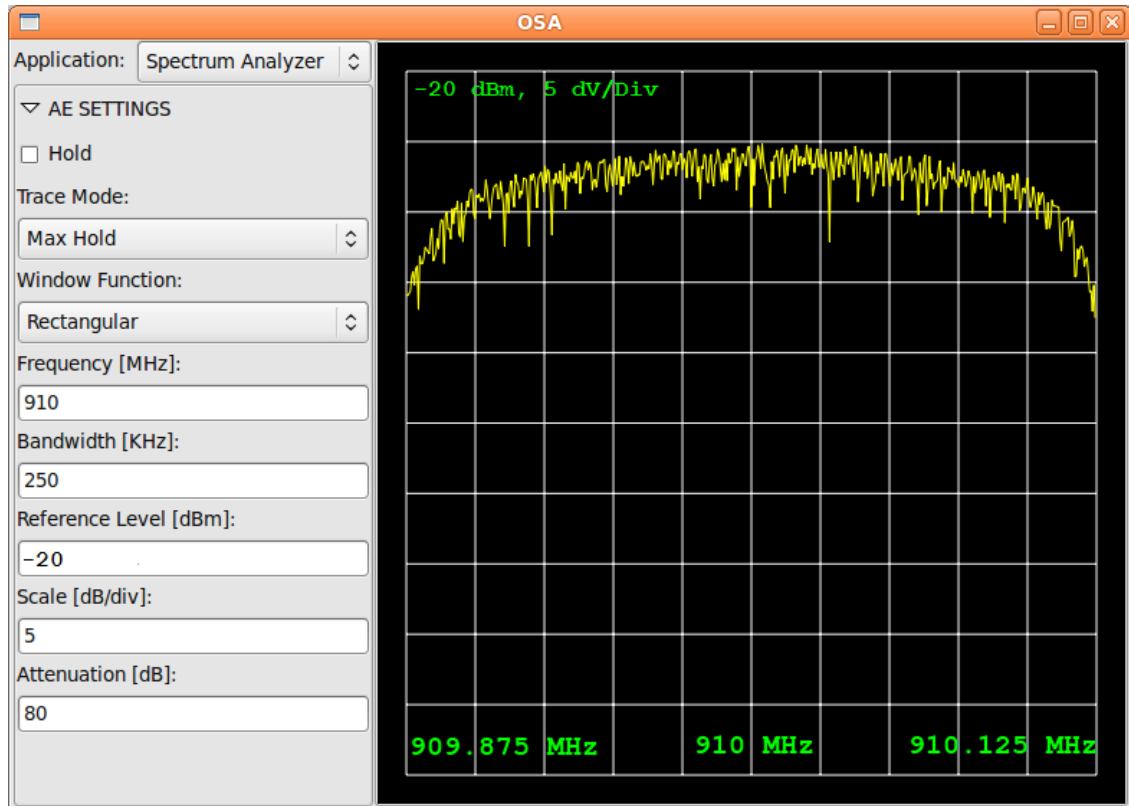


Figura 5.13. Trazo *Maxhold* después de un barrido en frecuencia con señal sinusoidal de amplitud constante

A pesar de las diferentes distorsiones y no linealidades que introduce el USRP, con esta prueba se verificó que el USRP es capaz de recibir señales de radio, convertirlas a banda base, digitalizarlas y entregarlas a la aplicación de análisis de señales para su posterior procesamiento y análisis.

5.2.3 Generación de una señal OFDM de prueba a través del USRP

Una vez se ha verificado que el USRP se integra perfectamente con el CF de SCA implementado en OSSIE y con la aplicación de análisis de señales se debe verificar el funcionamiento de la aplicación de análisis de señales OFDM mediante una señal de prueba generada mediante el mismo USRP. En la Figura 5.14 se muestra el esquema de pruebas propuesto para la validación de la aplicación de análisis de señales OFDM mediante una señal de prueba.

⁴ En los analizadores de señales, la función *Maxhold* retiene el valor máximo de cada componente de frecuencia durante la recolección de varios trazos sucesivos.

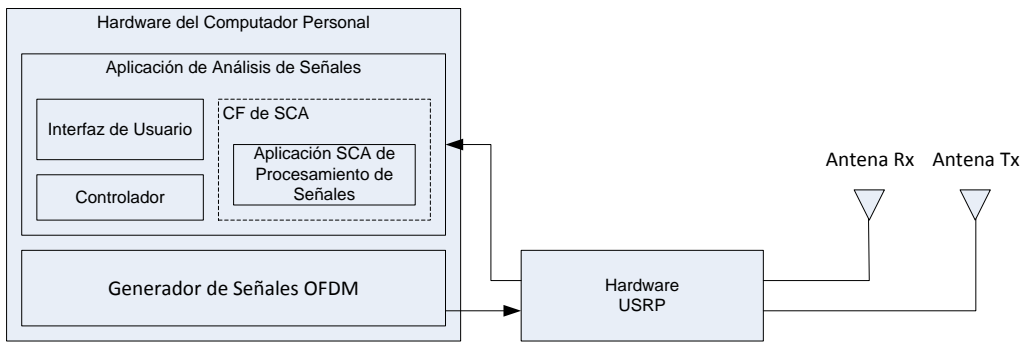


Figura 5.14. Esquema propuesto para la verificación del prototipo mediante una señal OFDM de prueba

La Figura 5.15 describe la forma en que se genera la señal OFDM de prueba. Las muestras digitales que representan la señal de prueba son generadas a partir del modelo de un modulador OFDM implementado en MATLAB y se guardan en un archivo binario. Para transmitir la señal OFDM de prueba mediante el USRP se utilizó un generador de señales que obtiene las muestras de la señal de prueba desde el archivo que se generó con el modelo de MATLAB.

Aunque el proceso de generación de la señal de prueba puede hacerse directamente sin necesidad de leer las muestras de la señal desde un archivo, al hacer esto así se incrementa la carga sobre el computador personal disminuyendo los recursos disponibles y el desempeño de la aplicación de análisis de señales. Por este motivo el proceso de generación de la señal de prueba se dividió en dos partes, primero se generó el archivo de muestras, actividad que requiere gran capacidad de procesamiento y luego estas muestras se leyeron del archivo y se transmitieron por el USRP, proceso que no consume muchos recursos del computador anfitrión dejando toda la capacidad de procesamiento para ejecutar la aplicación de análisis de señales.

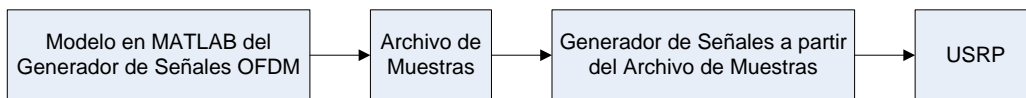


Figura 5.15. Generación de la señal OFDM de prueba

El modelo del generador de señales OFDM y los diferentes archivos de muestras generados con los diferentes parámetros de operación, al igual que la documentación y el código fuente del generador de señales pueden encontrarse en el disco compacto que se proporciona como anexo.

Para realizar la verificación del generador, se transmitió una señal OFDM de prueba con un ancho de banda de 200 KHz, cuyas características se describen en la Sección 5.1.1. Se realizaron algunas mediciones conectando un analizador de espectro directamente a la salida del USRP, obteniendo los resultados que se muestran a continuación.

En la imagen de la Figura 5.16 se puede apreciar el contenido espectral de la señal y la potencia total en un rango de 250 KHz. En el trazo de la imagen se puede observar el mínimo en el centro del trazo, producto de suprimir la subportadora central en la señal OFDM transmitida. Sin embargo la señal generada por el USRP no tiene la máscara espectral característica de una señal OFDM, plana en su parte superior y con pendiente pronunciada en sus extremos. Al verificar por

qué sucede esto se encontró que la etapa de interpolación en el USRP se ha implementado con filtros CIC (“*Cascaded Integrator Comb*”) y que no se implementó el filtro de compensación que requieren los filtros de interpolación, de esto se deriva que la señal a la salida del USRP no tiene el mismo espectro de la señal generada por el modelo en MATLAB.

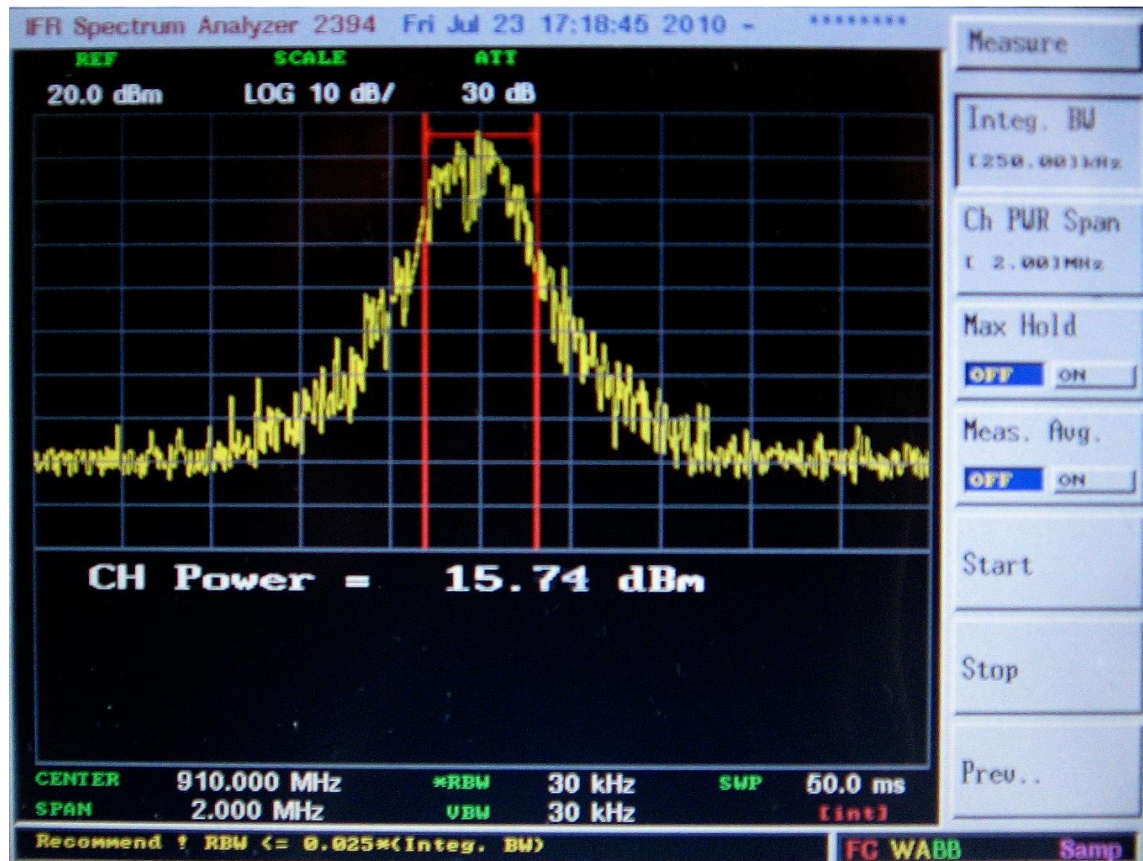


Figura 5.16. Espectro y potencia de la señal OFDM de prueba

El diseño del filtro de compensación depende de los parámetros de los filtros interpoladores que se han implementado en el FPGA (“*Field Programmable Gate Array*”) del USRP y se debe implementar sobre el mismo FPGA. Puesto que no existe documentación acerca de los parámetros de diseño de los filtros de interpolación y que su implementación implica la reprogramación del FPGA mediante código VHDL (“*Very High Speed Integrated Circuit Hardware Description Language*”), se determinó que no es conveniente implementar este filtro de compensación debido a que agrega tareas de complejidad considerable para el desarrollo del , en perjuicio de lograr los demás objetivos planteados.

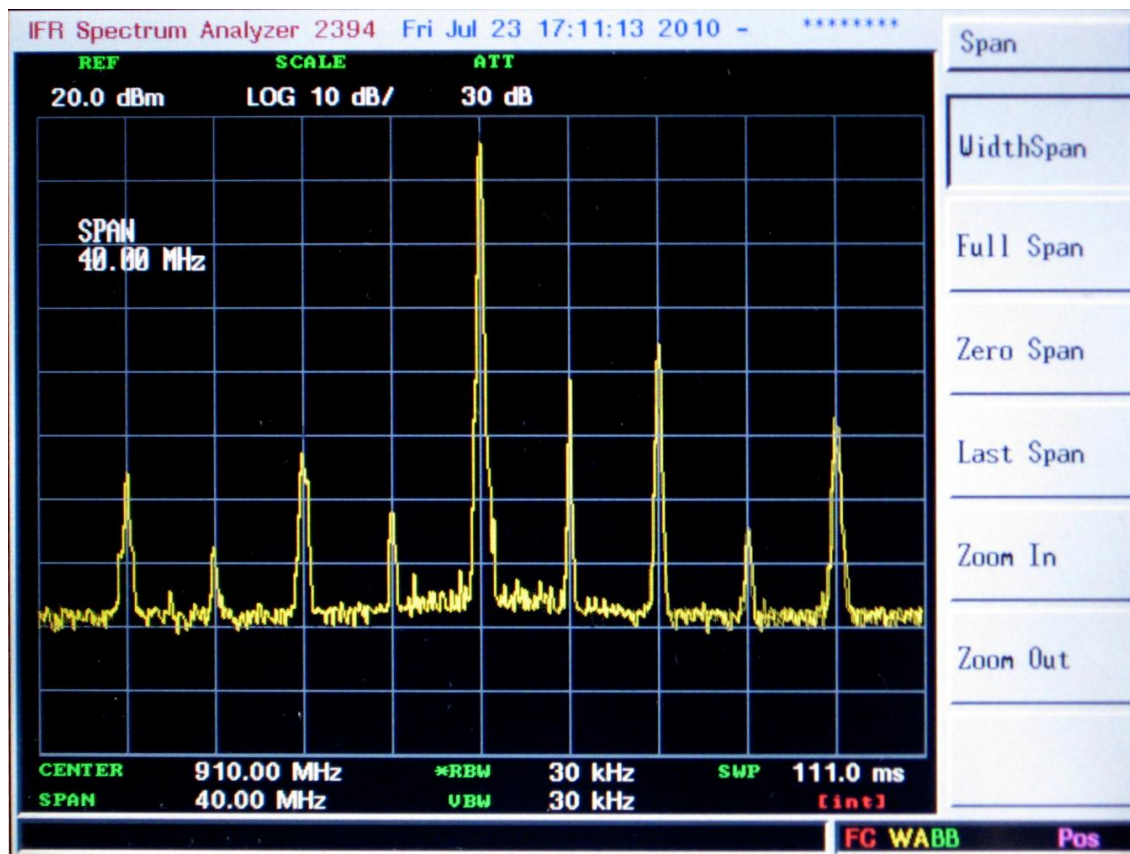


Figura 5.17. Espectro de la señal OFDM de prueba y de los armónicos generados por el USRP

Al ampliar el rango de barrido en frecuencia del analizador de espectro, como se muestra en la Figura 5.17, se pudo determinar que el USRP genera replicas de la señal de prueba, separadas en múltiplos de 4 MHz de la señal original, y también que la diferencia mínima de amplitud entre la señal original y las réplicas es de aproximadamente 30 dB.

5.2.4 Recepción y análisis de la señal OFDM de prueba a través del USRP

La prueba final, planeada inicialmente para la verificación del prototipo del software de análisis de señales, consistía en realizar el análisis de una señal OFDM de prueba generada con el mismo USRP como se describe al inicio de la Sección 5.2.3. Para esto se utilizó el USRP en modo *full duplex* para transmitir y recibir la señal de prueba al mismo tiempo y se utilizaron dos antenas de 850 MHz separadas a una distancia de 2 metros. Al realizar una prueba de análisis espectral de la señal recibida en la aplicación de análisis de señales se obtuvo el resultado que se muestra en la imagen de la Figura 5.18.

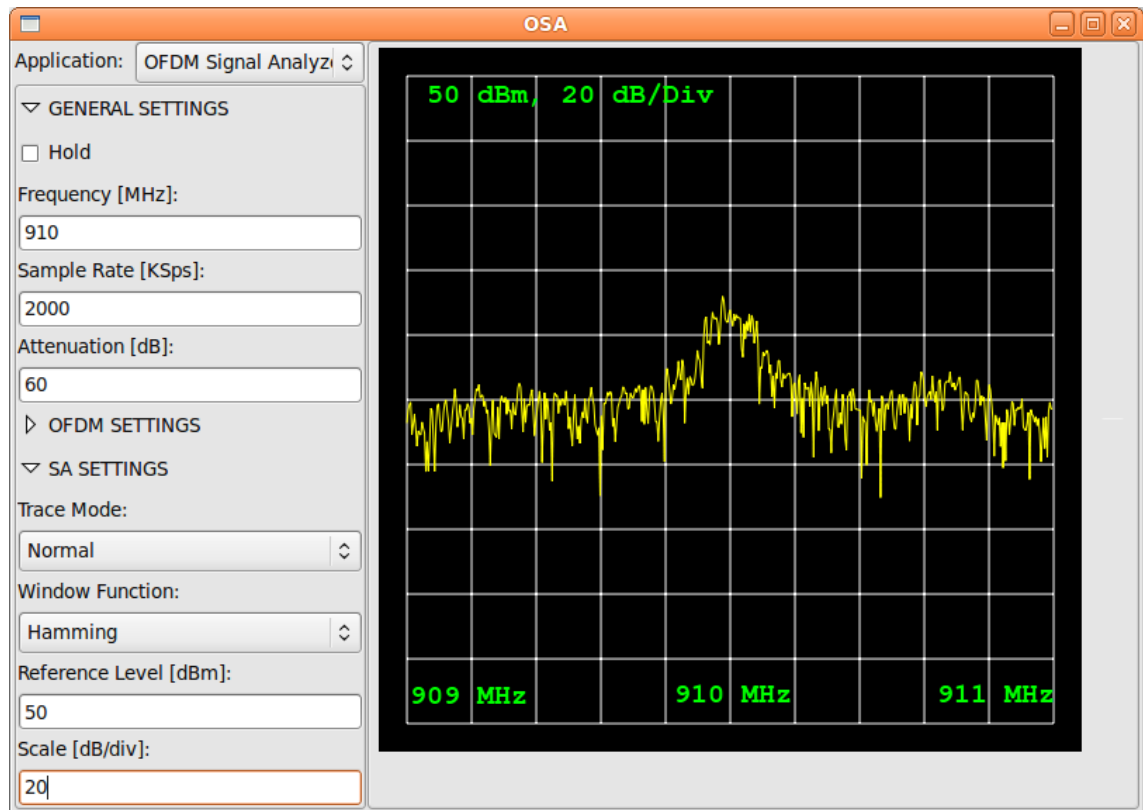


Figura 5.18. Espectro de la señal OFDM de prueba obtenido a través del USRP

Además del inconveniente relacionado con la distorsión en frecuencia introducida por el USRP que se describe en la Sección 5.2.3, el nivel de señal que se recibe respecto al piso de ruido, es bajo para su demodulación y análisis. Mientras la potencia de la señal a la salida del USRP es de 15,74 dBm como se demostró en la sección anterior, la lectura de la potencia de la señal recibida es de aproximadamente -40 dBm. Al verificar el nivel de señal con el que se registran la réplica presente en los 918 MHz se obtuvo el resultado que se muestra en la Figura 5.19.

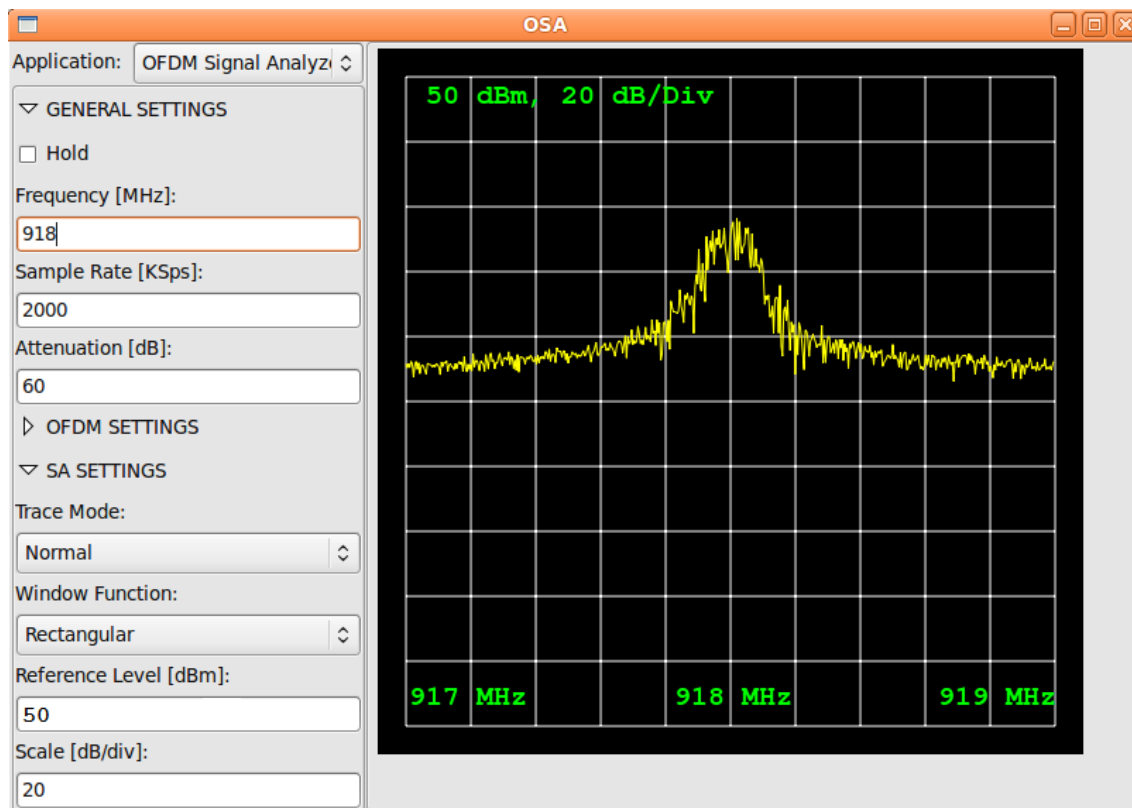


Figura 5.19. Espectro de la réplica de la señal OFDM de prueba obtenido a través del USRP

En las dos imágenes anteriores se observa que el nivel de la réplica de la señal OFDM se obtiene con mayor nivel que la misma señal de prueba, esto probablemente se debe a la falta de aislamiento entre las secciones de transmisión y recepción de la tarjeta RFX900, que produce algún tipo de distorsión que no permite que el USRP pueda operar efectivamente cuando la tarjeta está configurada en modo *full duplex* y la sección de transmisión y de recepción están operando en la misma frecuencia.

5.2.5 Análisis de la señal OFDM de prueba mediante configuración en bucle y simulación del canal inalámbrico

La interoperabilidad entre el software del dispositivo y la plataforma de hardware USRP se demostró con las pruebas que se describen en las secciones 5.2.1 y 5.2.2. El inconveniente que se describe en la Sección 5.2.4 no permite que se pueda recibir la señal de prueba con un nivel adecuado cuando esta se genera utilizando el mismo USRP, esto hizo necesario que se replanteara el método de verificación de la funcionalidad del software de análisis de señales OFDM. En la Figura 5.20 se presenta un método alternativo para verificar la funcionalidad del software de análisis de señales OFDM mediante una configuración en bucle, en donde se genera la señal digital leyendo los valores de las muestras desde el archivo generado con el modelo de MATLAB, luego se le realiza una serie de modificaciones de acuerdo con un modelo de canal AWGN (*“Additive White Gaussian Noise”*) y finalmente se introducen las muestras modificadas a la aplicación de análisis de

señales a través de la misma interfaz en la que recibiría las muestras de una señal recibida con el USRP.

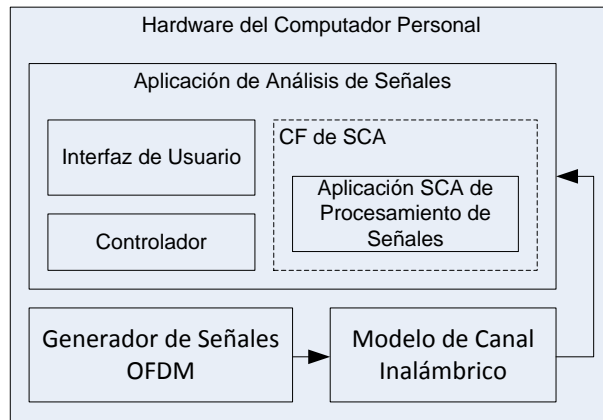


Figura 5.20. Verificación de la aplicación de análisis de señales en configuración de bucle

En la Figura 5.21 se observa la interfaz de usuario de la aplicación de análisis de señales OFDM funcionando en la configuración de bucle descrita anteriormente. En la interfaz de usuario se pueden observar los diferentes trazos obtenidos al realizar el análisis de la señal de prueba mediante la configuración en bucle. El trazo superior izquierdo muestra el contenido espectral de la señal, el trazo inferior izquierdo muestra un segmento de las componentes real e imaginaria de la señal en banda base, presentadas en el dominio del tiempo, el trazo superior derecho muestra el diagrama de dispersión de la señal demodulada y por último, la sección inferior derecha contiene un resumen de los parámetros de operación del analizador de señales y también la tasa promedio de errores de bloque y la tasa acumulada de errores de bloque de la señal demodulada.

Por medio de esta prueba se verificó que cada uno de los trazos es coherente con los parámetros que se utilizaron para generar la señal de prueba y que cada uno de los parámetros de configuración de la aplicación produce los efectos deseados en el proceso de análisis de la señal. Después de realizar estas verificaciones se puede concluir que la implementación del software de análisis de señales OFDM es correcta y que se podría implementar un prototipo completamente funcional utilizando una plataforma de hardware que no tenga las limitaciones del USRP que se han descrito en este capítulo.

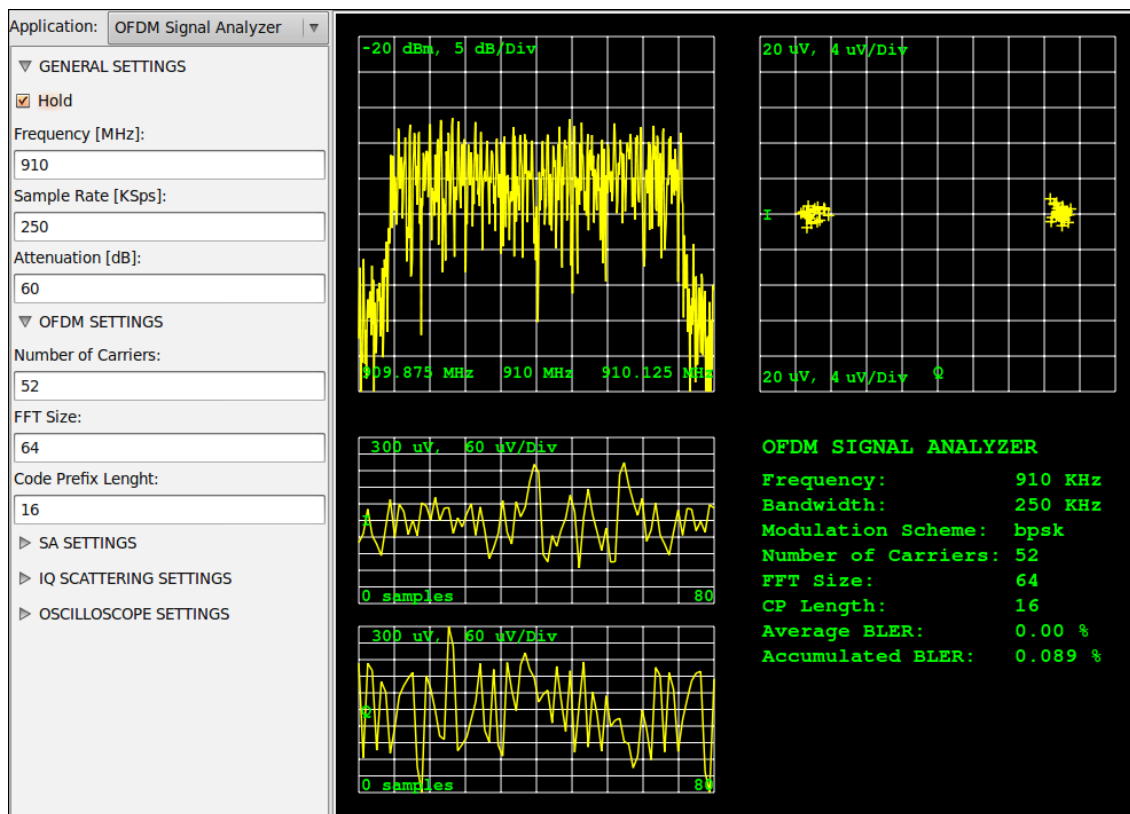


Figura 5.21. Análisis de la señal de prueba mediante la configuración en bucle

5.3 DESCRIPCIÓN Y COMENTARIOS ACERCA DE LAS HERRAMIENTAS DE SOFTWARE USADAS

El software del prototipo de análisis de señales OFDM se ha implementado utilizando Linux como sistema operativo, C++ como lenguaje de programación y diversas herramientas software de código abierto y de libre utilización. A continuación se describen cada una de las herramientas que se han utilizado durante la implementación del prototipo y los aspectos más relevantes de las herramientas al igual que los inconvenientes generados por su utilización.

5.3.1 Entorno integrado de desarrollo Eclipse

Es un entorno integrado de desarrollo para múltiples plataformas y de código abierto inicialmente desarrollado para el lenguaje de programación Java que actualmente soporta muchos lenguajes mediante la instalación de los *plugins* adecuados. Eclipse es un entorno de desarrollo maduro fácilmente extensible y altamente adaptable.

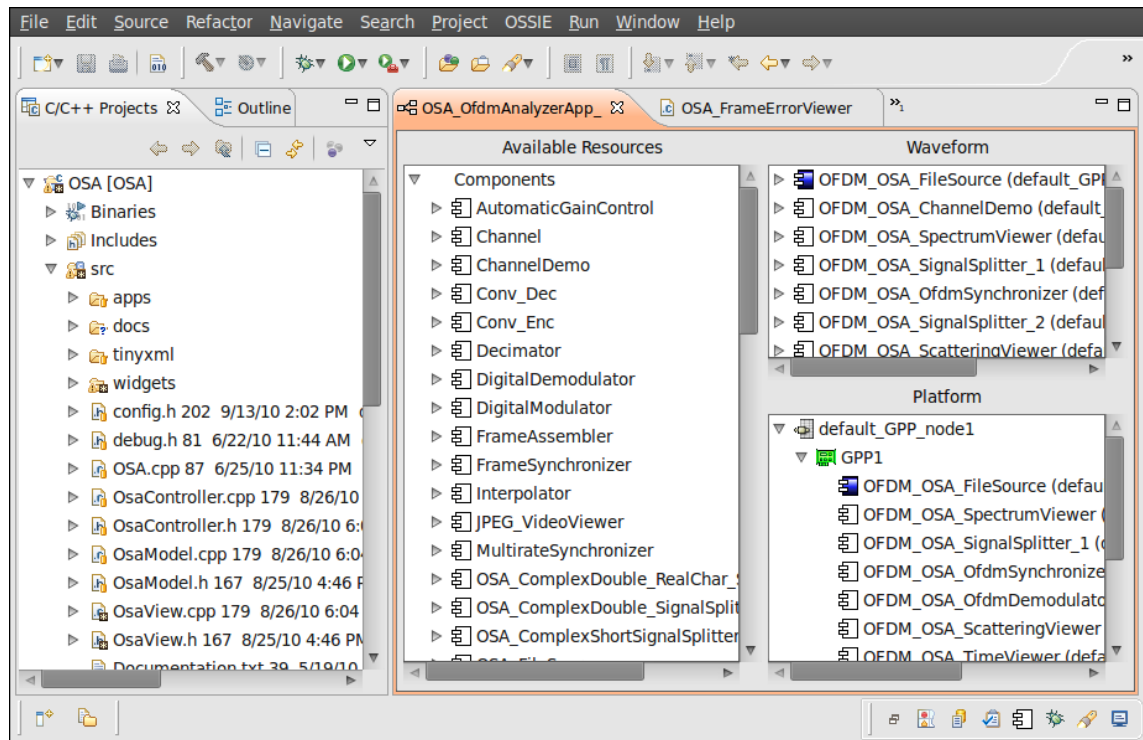


Figura 5.22. Entorno integrado de desarrollo Eclipse con el plugin OSSIE OWD instalado

Una de las características de mayor utilidad cuando se emprenden proyectos de desarrollo de software de mediano o gran tamaño es el soporte de sistemas de control de versiones, los cuales mantienen un registro de cada una de las modificaciones que se han realizado sobre los archivos de código fuente facilitando el trabajo en equipo para desarrollo de proyectos de software. En la Figura 5.22 se muestra una imagen de Eclipse con el *plugin OSSIE Eclipse Feature* que se describe en el siguiente apartado.

5.3.2 *Plugin OSSIE Eclipse Feature*

Actúa como interfaz para trabajar con las herramientas del CF de OSSIE desde el entorno de desarrollo Eclipse. Este *plugin* proporciona una interfaz gráfica para el desarrollo de aplicaciones y componentes SCA.

Las interfaces proporcionadas por el *plugin* facilitan el trabajo de desarrollo, puesto que permiten automatizar gran parte del desarrollo del código fuente como generación de los archivos descriptores de SCA, generación de plantillas de código que contienen la definición de los puertos y propiedades de cada componente según lo especifica la arquitectura SCA, de tal manera que el desarrollador solamente debe concentrarse en la implementación de los algoritmos de procesamiento digital de señales.

Sin embargo, por la falta de madurez de esta herramienta aún se presentan algunos errores e inconvenientes. En algunas ocasiones, el menú para generar las plantillas de código fuente no se habilita cuando debería hacerlo, de tal manera que es necesario reiniciar todo el entorno de

desarrollo. Cuando se introducen cadenas de texto con la letra 'ñ' o vocales tildadas en los descriptores de las propiedades o puertos de un componente SCA, se presenta un error interno que no permite la generación de las plantillas y cuando se especifican puertos del tipo *'string'* el código generado queda incompleto produciendo errores en la compilación del componente SCA.

Otro aspecto a tener en cuenta está relacionado con la inclusión de librerías de programación en el desarrollo de los componentes SCA. El entorno de desarrollo Eclipse ha definido algunas interfaces de usuario en las que se pueden agregar las referencias a los archivos de cabecera y las librerías de programación que se necesiten, sin embargo el *plugin OSSIE Eclipse Feature* utiliza las herramientas *'automake'* y *'autoconf'* ignorando la configuración de librerías definida en Eclipse de tal manera que es necesario invocar el script *'PKG_CONFIG'* en el archivo *'configure.ac'*. De igual manera sucede con los archivos de código fuente adicionales que deben ser enumerados en el archivo *'Makefile.am'*.

Finalmente este complemento de eclipse inhabilita la inclusión de los archivos generados automáticamente en el sistema de control de versiones, de tal manera que para controlar los cambios sobre estos archivos es necesario recurrir a procedimientos alternativos como agregar estos archivos con un cliente del sistema de control de versiones diferente a Eclipse.

A pesar de los inconvenientes existen muchas posibilidades para implementar nuevas funcionalidades en este complemento, por ejemplo un entorno completamente gráfico orientado al desarrollo de aplicaciones basadas en modelos, que facilitaría aún más el trabajo del desarrollador de aplicaciones.

5.3.3 Glade

Glade es una herramienta gráfica para el desarrollo de interfaces gráficas en GNOME (*"GNU Network Object Model Environment"*) utilizando la librería de gestión de ventanas GTK (*"Gimp Tool Kit"*). Esta herramienta es independiente del lenguaje de programación y no genera código fuente sino un archivo XML que contiene información acerca de la interfaz de usuario. En la Figura 5.23 se muestra el diseñador de interfaces gráficas de la herramienta Glade.

Aunque en GTK se pueden construir interfaces de usuario utilizando solamente código, es más fácil utilizar el diseñador gráfico de Glade y luego crear la interfaz de usuario en GTK a partir de la invocación del archivo XML. Este no es el caso de los componentes gráficos o *widgets* que se han implementado para la visualización de las diferentes señales, Glade solamente ofrece la funcionalidad de construir ventanas a partir de componentes previamente construidos e integrados en la herramienta de tal manera que para construir los *widgets* de análisis de señales fue necesario crearlos en su totalidad mediante código fuente y agregados a la interfaz de usuario en tiempo de ejecución mediante código fuente también.

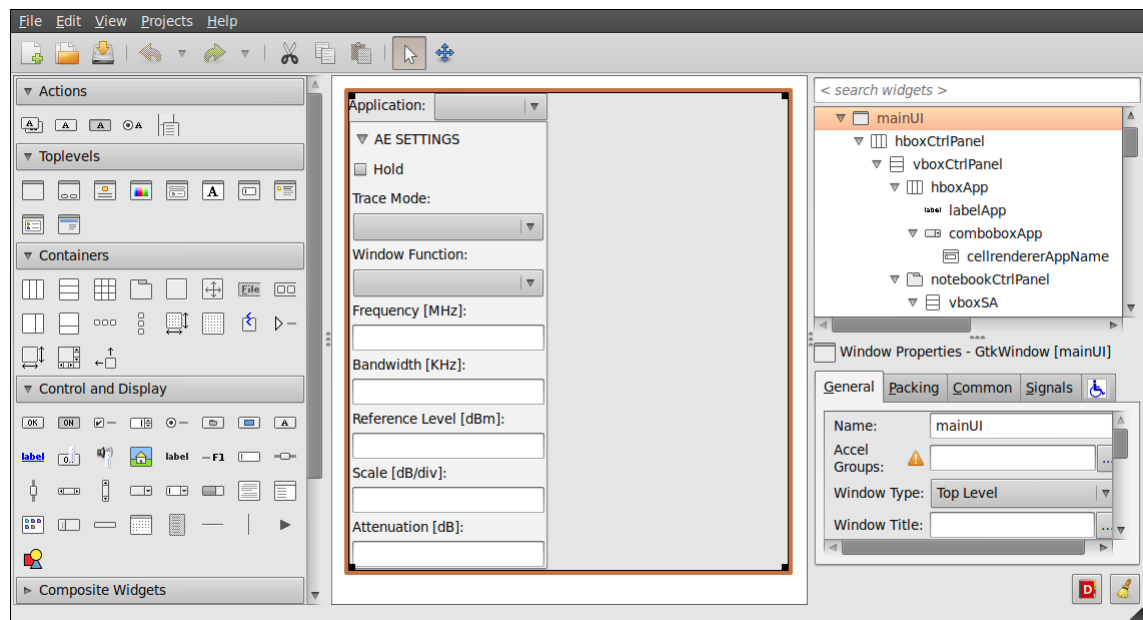


Figura 5.23. Diseñador de interfaces de usuario Glade

5.3.4 Doxygen

Doxygen es un generador de documentación para C++, C, Java y otros lenguajes de programación, que se ha utilizado en el desarrollo de este proyecto para generar la documentación del programa y sus distintos componentes a partir de los comentarios que se encuentran en el código fuente. Para generar esta documentación es necesario que el código fuente tenga los comentarios adecuados de acuerdo con el formato especificado en el manual de esta herramienta.

5.3.5 Librerías de software

Para el desarrollo de este trabajo se utilizaron diferentes librerías, algunas de propósito general y otras de aplicación específica para el desarrollo de aplicaciones de procesamiento digital de señales. A continuación se hace una breve descripción de cada una de las herramientas utilizadas empezando por la más importante que es la librería del proyecto OSSIE.

OSSIE

Implementa el CF de SCA, proporciona los mecanismos de interacción y control del CF y es el componente fundamental para el desarrollo de aplicaciones y componentes SCA. OSSIE es un proyecto que aun no llega a su primera versión estable pero tiene un equipo de desarrollo activo que está trabajando para obtener una versión conforme con la especificación SCA.

Fue necesario modificar el código fuente del componente SCA controlador del USRP para que este pudiera funcionar con la tarjeta de interfaz RFX900. Aunque el *Domain Manager* y el *Device Manager* se han utilizado sin modificación a través de la aplicación denominada '*nnodeBooter*', el *Domain Manager* presenta algunos errores en su implementación que impiden cambiar de una

aplicación SCA a otra sin reiniciar el *'noodeBooter'* y también impiden instalar e iniciar una aplicación SCA que contiene múltiples instancias del mismo componente SCA. De otro lado la implementación OSSIE solamente permite que un componente SCA se conecte al puerto de salida de otro componente, de tal manera que es necesario implementar componentes duplicadores cuando la salida de un componente SCA se debe procesar por dos o más componentes.

La encapsulación de diferentes funcionalidades de procesamiento digital de señales es uno de los puntos fuertes de SCA que se ha implementado en OSSIE, con esto se logra cierta granularidad que permite reutilizar diversos componentes en diferentes aplicaciones de procesamiento de señales. Sin embargo una gran falencia de SCA es que no permite crear componentes SCA a partir de componentes más simples, esto hace imposible la reutilización de componentes que han sido diseñados con diferentes grados de granularidad, por ejemplo, el demodulador OFDM requiere realizar una FFT además de otros procedimientos sobre la señal, sin embargo para la implementación del demodulador es necesario implementar nuevamente la FFT a pesar de que exista un componente SCA que implemente únicamente la Transformada Rápida de Fourier.

FFTW

FFTW es una librería que implementa el algoritmo de la Transformada Rápida de Fourier. Esta es una librería cuyo estado de desarrollo es bastante maduro, en la cual se han desarrollado diferentes optimizaciones que han logrado que sea catalogada como la implementación de código abierto más eficiente de la FFT, de hecho esta es la implementación que se utiliza en el producto comercial MATLAB y en este proyecto se ha utilizado en los diferentes componentes SCA en donde se requiere realizar la transformada discreta de Fourier de cualquier secuencia.

OmniORB4

SCA ha definido que todos sus componentes deben interactuar entre sí mediante los mecanismos de comunicaciones que proporciona CORBA (*"Common Object Request Broker Architecture"*). OmniORB es la implementación de la especificación 2.6 de CORBA que le permite al Modelo de Aplicación SCA comunicarse e interactuar con los diferentes componentes que conforman la Aplicación SCA para el análisis de señales y con los componentes que pertenecen al CF de SCA.

Boost

Boost es una colección de librerías diseñadas para extender la funcionalidad de C++ que se han incluido en el reporte técnico del comité de estandarización de C++ y hará parte de la especificación C++0x. De todas las librerías que pertenecen a Boost, se han utilizado las librerías *CircularBuffer* e *Inteprocess*, para manipular arreglos de memoria en forma de buffer circular y para la comunicación entre los procesos de la aplicación SCA y los componentes de la interfaz de usuario.

Libsigc++

Esta es una librería que implementa un sistema de llamadas de notificación con verificación segura de la clase de los argumentos. Se utilizó la librería Libsigc++ para implementar los mecanismos de

notificación de eventos que se exponen en los diferentes elementos de la aplicación que implementan la interfaz “observador”.

GTK+

GTK+ es un conjunto de librerías que proporcionan un conjunto de herramientas de software para el desarrollo de interfaces gráficas de usuario en entornos gráficos GNOME, aunque también se puede usar en Windows por medio de la herramienta Cigwin. GTK+ comprende las librerías Glib, GTK, GDK, Pango y Cairo que proporcionan gran cantidad de funcionalidades que se requieren para la creación y manipulación de entornos gráficos, sin embargo la documentación que existe es limitada y en muchas ocasiones es necesario recurrir al código fuente de aplicaciones existentes para descubrir la forma en que se deben utilizar sus diversos componentes.

CONCLUSIONES Y RECOMENDACIONES

A continuación se presentan las conclusiones obtenidas al finalizar el proyecto.

El USRP es una plataforma de hardware económica para aplicaciones de SDR. Durante el desarrollo del trabajo se pudo determinar que el USRP se adapta a proyectos de baja complejidad y de requerimientos simples pero no cumplió con las expectativas que existían al inicio del proyecto. Múltiples inconvenientes como el acoplamiento de las secciones de transmisión y recepción cuando el USRP opera en modo *full dúplex*, la falta de linealidad en sus componentes y la carencia de filtros de compensación en la sección de transmisión hicieron que no fuese posible recibir una señal adecuada para su demodulación, de tal manera que fue necesario replantear los objetivos y los métodos de verificación.

Por causa de estos inconvenientes, parte de la verificación del software del prototipo se hizo mediante una configuración en bucle, sin utilizar el hardware del USRP. Sin embargo, en la búsqueda de plataformas alternativas de hardware para aplicaciones de SDR se encontraron varias opciones diferentes al USRP, con excelentes especificaciones y buen desempeño, que confirman la posibilidad de utilizar plataformas genéricas de SDR para la implementación de instrumentos de análisis de señales de comunicaciones inalámbricas.

SCA es una arquitectura que se diseñó para soportar la implementación y operación de aplicaciones de comunicaciones sobre plataformas de SDR. Con el desarrollo de este trabajo se demuestra que SCA se puede integrar fácilmente en aplicaciones con requerimientos similares como sucede con los sistemas de instrumentación para el análisis de señales de comunicaciones inalámbricas y facilita enormemente su desarrollo.

En la implementación de OSSIE, parte del procesamiento se realiza sobre el procesador del computador anfitrión y la capacidad de procesamiento está limitada por este. Aunque en general los FPGAs y DSPs tienen una mayor capacidad de procesamiento que el procesador de un computador, en la práctica se comprueba que la velocidad de los procesadores actuales permite ejecutar aplicaciones de mediana complejidad con señales de un ancho de banda moderado. Para la implementación de aplicaciones que requieren mayor capacidad de procesamiento es necesario migrar el código fuente de la implementación OSSIE a una plataforma de SDR que proporcione los recursos de hardware necesarios para lograr la capacidad de procesamiento requerida.

Sobre una señal OFDM se pueden realizar diferentes tipos de análisis, la implementación de un prototipo completamente funcional es un proyecto bastante extenso que requiere mucho tiempo y recursos. En este trabajo se desarrolló una aplicación de funcionalidad limitada, con un modelo sencillo, flexible y escalable que proporciona un ejemplo de cómo se pueden implementar aplicaciones de análisis de señales de mayor complejidad.

Como futuros proyectos relacionados con el presente trabajo se podrían considerar las siguientes propuestas.

Integrar o migrar OSSIE con plataformas de hardware para radio definido por software diferentes al USRP, que cumplan con las especificaciones mínimas que se requieren en un dispositivo de instrumentación para el análisis de señales.

También se podrían implementar diferentes aplicaciones SCA de procesamiento y análisis de señales para señales generadas por diferentes tecnologías que permitan expandir la funcionalidad del prototipo desarrollado en este trabajo.

Finalmente este trabajo puede servir como referencia e introducción para quienes pretenden desarrollar proyectos que involucren la arquitectura SCA o el hardware USRP, por ejemplo la implementación de una aplicación de comunicaciones compatible con la arquitectura SCA o el desarrollo de nuevos esquemas de modulación o la implementación de nuevos protocolos de comunicaciones inalámbricas utilizando el hardware USRP.

REFERENCIAS BIBLIOGRÁFICAS

- [1] Ettus Research LLC, "USRP Family Products and Daughter Boards". [en línea]. Disponible en: <http://www.ettus.com/products>. [Accedido: 01 de Dic. de 2010].
- [2] Joint Tactical Radio System, "Software Communications Architecture (SCA)". [en línea]. Disponible en: <http://sca.jpeojtrs.mil/sca.asp>. [Accedido: 01 de Dic. de 2010].
- [3] J. Bard y V. J. Kovarick Jr, *Software Defined Radio: The Software Communications Architecture*. New York, NY: John Wiley & Sons, 2007.
- [4] Virginia Tech, "OSSIE: SCA-Based Open Source Software Defined Radio". [en línea]. Disponible en: <http://ossie.wireless.vt.edu>. [Accedido: 01 de Dic. de 2010].
- [5] S. Haykin y B. Van Veen, *Signal and Systems*, 2a ed. New York, NY: John Wiley & Sons, 2003.
- [6] R. A. Witte, *Spectrum and Network Measurements*. Atlanta, GA: Noble Publishing, 2001.
- [7] J. G. Proakis y D. G. Manolakis, *Tratamiento Digital de Señales*, 4a ed. Madrid: Pearson Education, 2007.
- [8] Agilent Technologies, "The Fundamentals of Signal Analysis", Application Note 243.
- [9] Agilent Technologies, "Spectrum Analysis Basics", Application Note 150.
- [10] M. Golio y J. Golio, *The RF and Microwave Handbook: RF and Microwave Circuits, Measurements, and Modeling*, 2a ed. Boca Raton, FL: CRC Press, 2008.
- [11] C. Rauscher, *Fundamentos del Análisis de Espectro*. Rohde & Schwarz, 2001.
- [12] Agilent Technologies, "Vector Signal Analysis Basics", Application Note 150-15.
- [13] J. Mitola, *Software Radio Architecture: Object-Oriented Approaches to Wireless System Engineering*. New York, NY: John Wiley & Sons, 2000.
- [14] J. H. Reed, *Software Radio: A Modern Approach to Radio Engineering*. Upper Saddle River, NJ: Pearson Education, 2002.
- [15] P. B. Kenington, *RF and Baseband Techniques for Software Defined Radio*. Norwood, MA: Artech House, 2005.
- [16] R. Hosking, *Digital Receiver Handbook: Basics of Software Radio*, 2a ed. Upper Saddle River, NJ: Pentek Inc, 2003.
- [17] P. Burns, *Software Defined Radio for 3G*. Norwood, MA: Artech House, 2003.

- [18] H. Meyr, M. Moeneclaey y S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York, NY: John Wiley & Sons, 1998.
- [19] F. Abbas, "The USRP under 1.5X Magnifying Lens!", 2008. [en línea]. Disponible en: <http://gnuradio.org/redmine/attachments/download/129>. [Accedido: 01 de Dic. de 2010].
- [20] M. Millhaem, "SDR Advantages for Test Equipment", White Paper.
- [21] A. Opler, "Fourth Generation Software", *Datamation*, vol. 13, pp. 22-24, Enero de 1967.
- [22] D. Dallet y J. M. da Silva, *Dynamic Characterization of Analogue-to-Digital Converters*. Holada: Springer, 2005.
- [23] M. Millhaem, "Software Defined Radio: The Next Wave in RF Test Instrumentation", White Paper.
- [24] J. Bard y V. Kovarik, *Software Defined Radio: The Software Communications Architecture*. Inglaterra: John Wiley & Sons, 2007.
- [25] Q. H. Mahmoud, *Middleware for Communications*. Inglaterra: John Wiley & Sons, 2004.
- [26] C. R. Aguayo, C. B. Dietrich y J. H. Reed, "Understanding the Software Communications Architecture", *IEEE Communications Magazine*, vol. 47, no. 9, pp. 50-57, Septiembre de 2009.
- [27] C. R. Aguayo et al., "Open-Source SCA-Based Core Framework and Rapid Development Tools Enable Software-Defined Radio Education and Research", *IEEE Communication Magazine*, vol. 47, no. 10, pp. 48-55, Octubre de 2009.
- [28] "GNU Radio". [en línea]. Disponible en: <http://gnuradio.org/redmine/wiki/gnuradio>. [Accedido: 01 de Dic. de 2010].
- [29] Communications Research Centre Canada, "Software Communications Architecture – Reference Implementation". [en línea]. Disponible en: http://www.crc.gc.ca/en/html/crc/home/research/satcom/rars/sdr/products/scari_open/scari_open. [Acceddo: 01 de Dic. de 2010].
- [30] M. Hermeling y J. Rice, "Rapid Prototyping for SCA", *SDR 06 Technical Conference and Product Exposition*, 2006.
- [31] M. Robert et al., "OSSIE: Open Source SCA for Researches", *SDR 04 Technical Conference and Product Exposition*, 2004.
- [32] S. Edwards, J. Snyder y C. Dietrich, "Towards an Open-Source Integrated Development Environment for Software-Defined Radio Work", *SDR 08 Technical Conference and Product Exposition*, 2008.

- [33] Cherrystone Software Labs, "Algorithmic Performance Comparison Between C, C++, Java and C# Programming Languages", Agosto de 2010. [en línea]. Disponible en: <http://www.cherrystonesoftware.com/doc/AlgorithmicPerformance.pdf>. [Accedido: 01 de Dic. de 2010].
- [34] L. Bass, P. Clements y R. Kazman, *Software Architecture in Practice*, 2a ed. Boston, MA: Addison Wesley, 2003.
- [35] S. J. Metsker y W. C. Wake, *Design Patterns in Java*. Westford, MA: Addison Wesley, 2006.
- [36] H. Schulze y C. Lüeders, *Theory and Applications of OFDM and CDMA Wideband Wireless Communications*. New York, NY: John Wiley & Sons, 2005.
- [37] M. Engels. *Wireless OFDM Systems: How to make them work?* Inglaterra: Springer, 2002.
- [38] O. Edfords, "Low-complexity algorithms in digital receivers", Tesis Doctoral. Universidad Tecnológica de Lulea, Lulea, Suecia, 1996.
- [39] J. van de Beek, M. Sandell y P. Börjesson, "ML Estimation of Time and Frequency Offset in OFDM Systems", *IEEE Transactions on Signal Processing*, vol. 45, no. 7, Julio de 1997, pp. 1800-1805.