SDR con GNU Radio: de la teoría a la aplicación

Eduardo Gonzalez*, Carlos Zerbini*†, Guillermo Riva*†, Daniel Rosso*, Fernanda Suarez*, Luis Guanuco*
*Grupo de Investigación y Transferencia en Electrónica Avanzada (GInTEA)
†Laboratorio de Comunicaciones (LdC)
Universidad Tecnológica Nacional, Facultad Regional Córdoba (UTN-FRC)
czerbini@electronica.frc.utn.edu.ar

***Resumen*—Actualmente, las técnicas de radio definida por software permiten implementar sistemas de comunicaciones reconfigurables y eficientes. En este contexto, la combinación de hardware de bajo costo con las librerías GNU radio se presenta como una opción con gran potencial tanto didáctico como de aplicación. Para aprovechar este potencial, es necesario comprender cómo los conceptos teóricos se aplican en estas herramientas, así como los compromisos de implementación que se plantean. El presente trabajo propone un análisis exhaustivo de estos aspectos a través de dos implementaciones de complejidad moderada, una de ellas de carácter didáctico y la otra más cercana a la aplicación. De este modo, se busca contribuir con bases sólidas a la difusión de estas técnicas.**

I. Introducción

Las técnicas de radio definida por software *(Software Defined Radio, SDR)* permiten implementar un amplio espectro de casos de procesamiento en comunicaciones en forma totalmente digital. SDR se concibe con el advenimiento de las técnicas digitales, si bien su evolución ha estado fuertemente ligada a la tecnología disponible [1] [2] [3]. Mediante estas técnicas los sistemas de comunicaciones pueden adaptarse automáticamente a distintas formas de modulación, utilizando combinaciones de software libre, etapas de conversión analógico-digital de alta velocidad y hardware reconfigurable (FPGAs) para lograr sistemas de altas prestaciones.

Actualmente la información sobre este tópico es bastante dispersa, y en particular en nuestro idioma es bastante escasa, lo que dificulta su adopción como herramienta didáctica y de desarrollo. Sobre esta base, el presente trabajo pretende brindar una aproximación a los sistemas SDR, demostrando sus beneficios desde el campo puramente didáctico hasta el de aplicación pura, mediante casos bien fundamentados que abarcan distintos niveles de abstracción. El tratamiento dado a los conceptos sirve a su vez de base para abordar casos de mayor complejidad.

Se selecciona en este trabajo un caso de modulación de complejidad moderada como es la modulación de frecuencia *(Frecuency Modulation, FM)*, en particular su versión de banda ancha *(WideBand FM, WBFM)*. WBFM transporta a su vez en su banda base información modulada; permitiendo así trabajar con diferentes técnicas de modulación como doble banda lateral con portadora suprimida *(Double Sideband-Suppressed Carrier, DSB-SC)* y modulación binaria por desplazamiento de fase *(Binary Phase Shift Keying, BPSK)*.

II. Metodología y herramientas utilizadas

II-A. Front-end de RF

Para nuestro trabajo se utiliza como hardware receptor o front-end de RF un dongle RTL2832U diseñado para TV digital (DVB-T) [2]. El driver utilizado, *librtlsdr*, accede a las muestras no procesadas *raw mode*, permitiendo implementar una gran variedad de aplicaciones a bajo costo. Este equipo posee un rango de recepción de 24 a 1500 MHz, mientras que su tasa de muestreo compleja (I/Q) es de 2,56 MS/s, la cual determina el máximo ancho de banda de la señal recibida. Los principales componentes de este hardware son:

- Mezclador en cuadratura basado en el chip R820T, el cual traslada la frecuencia central seleccionada al valor de IF para su posterior procesamiento.
- Conversor analógico-digital (ADC) y filtro diezmador, ambos contenidos en el chip RTL2832U.

II-B. Procesamiento digital

El procesamiento digital se puede efectuar mediante una variedad de herramientas tales como Matlab, Labview o la más reciente *Gnuradio* [4]. Gnuradio ofrece numerosos módulos de procesamiento digital parametrizables orientados a comunicaciones, así como métodos para definir interconexiones entre ellos. Estas librerías constan de funciones altamente optimizadas para implementación de sistemas SDR, y puede utilizarse mediante el entorno *gnuradio companion*, el cual brinda una interfaz gráfica para la implementación y verificación de tales sistemas. Tanto las interconexiones como algunas de sus implementaciones se realizan mediante el lenguaje Python, mientras que las funciones con mayor exigencia de procesamiento se implementan mediante el lenguaje C++. La interacción entre ambos lenguajes se logra mediante el paquete *Simplified Wrapper and Interface Generator (SWIG)* [5].

La compilación de gnuradio es bastante laboriosa como se comprobó en el presente trabajo, en particular cuando se requieren características no soportadas por defecto. Por ejemplo, al inicio de nuestros ensayos se intentó trabajar con el front-end *Signalhound SA 44B*, comercializado como analizador de espectro e implementado mediante SDR, lo cual demandó considerable tiempo con resultados poco satisfactorios. Recientemente, Gnuradio fue incorporado a repositorios de linux, facilitando notablemente la tarea de instalación. Algunas de las librerías más importantes utilizadas para procesamiento

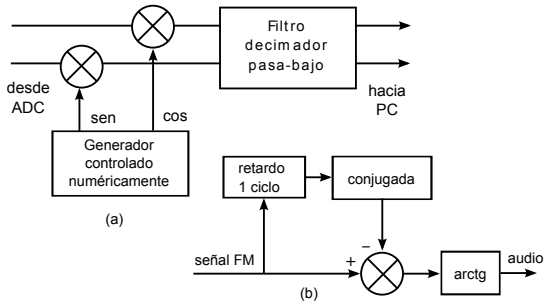

Figura 1. (a) Esquema del DDC, (b) demodulador *quad_demod*

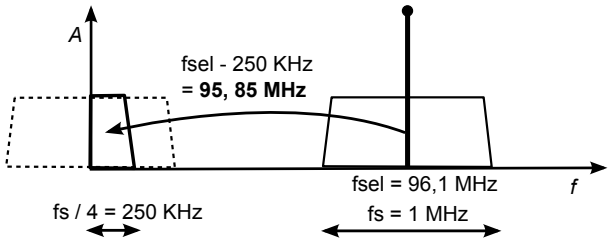

Figura 2. Procesamiento de señal *pasabanda* hacia señal *pasabajo* (banda base modulada)

digital en GNU radio son *Vector Optimized Library of Kernels (VOLK)* [6] y *BOOST* [7].

Un diagrama de flujo típico en gnuradio utiliza dos archivos principales. Uno de ellos, de extensión *.grc*, es un archivo XML relacionado a la representación gráfica del diagrama; mientras que el otro, de extensión *.py*, es el código python que instancia los objetos de GNU radio. El código fuente de estos objetos se puede encontrar en el árbol de compilación de gnuradio, lo que permite un completo control y optimización del flujo implementado.

III. Desarrollo

III-A. Casos planteados en FM

Durante la etapa de investigación de este trabajo, se relevaron numerosas implementaciones de FM en gnuradio. Estos ejemplos, sin embargo, se limitan a enumerar pasos de implementación sin brindar análisis de su operación [8] [9]. A partir de estas observaciones, se propone un análisis exhaustivo de las técnicas utilizadas, así como su estudio teórico a fin de aportarles utilidad desde el punto de vista didáctico.

Con fines de clasificación, se propone dividir los casos estudiados en dos categorías principales:

1. *Caso didáctico:* en este apartado se busca una implementación mediante bloques simples, que permitan llevar los conceptos teóricos de FM a su forma digital. Este caso, más allá de implementar un receptor completo de FM, pretende mostrar las funciones de demodulación desde un punto de vista muy cercano a la teoría. Se plantean dos sub-casos, que llamaremos (1a) y (1b). Asimismo, se tratan casos donde las muestras son simuladas así como los bloques de interfaz con hardware. Finalmente se muestra cómo, utilizando gnuradio, se puede escalar la implementación mediante el uso de *bloques jerárquicos*.
2. *Caso práctico:* este caso demuestra la utilidad de gnuradio en aplicaciones prácticas, mediante la implementación completa de un receptor para FM de banda ancha, recuperando toda la información que es posible transportar en banda base.

III-B. Caso didáctico: procesamiento de señales FM

En este caso se analiza en detalle el proceso de demodulación de una señal FM, así como su implementación directa mediante bloques de GNUradio. A fin de introducir nuestro análisis, es necesario un breve repaso de los conceptos fundamentales de FM. Encararemos nuestro trabajo considerando dos enfoques didácticos de FM.

La señal de FM es un caso especial de *modulación angular*, donde la *fase* de la señal modulada depende de la *integral* de la señal modulante mas una cierta fase inicial. Esto diferencia a FM de la otra variante de modulación angular, modulación de fase (Phase Modulation, PM), donde la fase depende *linealmente de la señal modulante*. La forma general de la señal modulada se expresa en la Ec. 1, mientras que su fase está dada por la Ec. 2.

$$\begin{aligned} s(t) &= A_c\ cos[\theta_i(t)] = A_c cos[2\pi f_i(t) \cdot t + \theta_0] \\ &= A_c\ cos\{2\pi[f_c + k_f m(t)] \cdot t + \theta_0\} \end{aligned} \quad (1)$$

$$\theta_i(t) = 2\pi \int_{-\infty}^{t} f_i\ d\tau = 2\pi f_c t + 2\pi k_f \int_{-\infty}^{t} m(\tau)\ d\tau \quad (2)$$

De este razonamiento, se observa que para recuperar la banda base $m(t)$ de una señal $s(t)$ modulada en frecuencia se requieren dos pasos principales:

1. Eliminar la frecuencia portadora f_c de la Ec. 1, obteniendo una banda base con frecuencia instantánea proporcional al mensaje original $m(t)$.
2. De la Ec. 2, obtener la frecuencia instantánea $f_i(t)$ de la banda base derivando la fase respecto al tiempo. A partir de $f_i(t)$, se obtiene $m(t)$ de forma directa.

En el *caso 1a*, la primera operación se realiza mediante un *Digital Downconverter (DDC)* en el chip RTL820 del receptor utilizado, controlado por el módulo de gnuradio *RTL-SDR Source*. Como se ilustra en la Fig. 1(a), este bloque consta de un oscilador controlado numéricamente *(Numerically Controlled Oscillator, NCO)*, que genera señales en fase y en cuadratura a la frecuencia f_c, y un filtro decimador FIR que toma sólo el producto $f_i - f_c$. Ya que en este caso se utiliza $f_c = f_i$, se obtiene directamente la banda base en un rango de 0 a f_{max}, resultando un receptor de *Frecuencia Intermedia Nula*. Opcionalmente se puede aplicar *decimación* en la salida, a fin de adecuar la tasa de datos al ancho de banda requerido. Este procesamiento se muestra en la Fig. 2 para un caso de f_s = 1 MHz, factor de decimación = 4 y $f_{central}$ = 96,1 MHz.

A partir de una señal pasabanda *real* recibida por la antena, el mezclador en cuadratura obtiene una banda base *compleja* con componentes I/Q. Es por ello que la tasa de datos obtenida

luego de realizar la decimación es llamada *tasa en cuadratura*, y es *igual* al ancho de banda analógico recibido [10].

En la Fig. 3 se muestra el diagrama implementado en GNUradio para este caso, mientras que la Tabla I resume los bloques principales utilizados en todos los casos presentados. El bloque de procesamiento *quadrature_demod_cf*, implementa demodulación en frecuencia en forma digital, mediante la derivada de la fase. En primer término, se toma la diferencia de fase entre muestras sucesivas. Para una muestra compleja en tiempo discreto $s[i]$, esta fase se obtiene calculando el valor $arctg\{s[i] * conj(s[i-1]\}$, como se muestra en la Fig. 1(b). La derivada se aproxima entonces dividiendo esta diferencia de fase por el intervalo de muestreo, lo que equivale a multiplicarla por la tasa de muestreo $quad\_rate$. Finalmente se afecta por un factor de escala que depende de la desviación máxima de frecuencia Δf, obteniendo muestras de la señal modulante $m[i]$ como se observa en la Ec. 3. A la salida de este bloque se obtiene la banda base compuesta de FM.

$$m[i] = \frac{quad\_rate}{2\pi\Delta f} \cdot arctg\{s[i] \cdot s[i-1]\} \tag{3}$$

Los bloques de *fuente (source)* y *destino (sink)*, que vinculan a gnuradio con el hardware de E/S, merecen especial consideración. En nuestro caso, el bloque tipo *source* que recibe las muestras entregadas por el driver *rtl_sdr* puede ser *osmosdr* o *rtl_sdr*. Este bloque básicamente controla la configuración del hardware de front-end, por lo que está bastante ligado a él. Con algunas variantes específicas a cada modelo, estos bloques requieren los parámetros *frecuencia central, frecuencia de muestreo* y *factor de decimación*.

El bloque *sink*, en tanto, está íntimamente relacionado al hardware que recibe las muestras procesadas en GNU radio, en nuestro caso una placa de audio, cuyo principal parámetro es su tasa de muestreo. Respecto a este bloque, el parámetro *OK_to_block* representó un interrogante en nuestras implementaciones ya que se encuentra sólo superficialmente documentado [9]. Su objetivo es aplicar contención a todo el flujo de datos que lo alimenta *(stream backpressure)*, a fin de controlar la tasa de datos de todo el diagrama cuando la placa de audio es el único hardware involucrado en el flujo. Sin embargo, cuando se utilizan dos relojes, uno proveniente de la fuente (receptor de radio) y otro del destino de datos (placa de audio), la teoría dicta que este parámetro se deshabilite en la placa de audio a fin de que el receptor de radio sea quien fije la tasa de datos. Al aplicar esta política en nuestros ensayos, se presentó el problema de *buffer underrun* (representado por el error *aU* en la consola de GNU radio). Éste indica que la tasa de arribo de muestras a la placa de audio es menor a la tasa de procesamiento de ésta, lo que en la práctica se traduce en cortes de sonido. Al aplicar contención, sin embargo, el problema se soluciona. Desde el aspecto teórico esto planteó un interrogante: por un lado, la aplicación de contención no es adecuada cuando existen dos relojes en el sistema; por otro lado, la contención debería requerirse siempre que la tasa de arribo sea *mayor* a la soportada por el hardware de destino. Finalmente, se arribó a la conclusión de que la aplicación de contención soluciona el problema de *buffer underrun* como consecuencia secundaria de la absorción de defasajes entre la tasa procedente del resampler y la tasa de la placa de audio, dado que ambos relojes son inestables en el hardware de bajo costo utilizado.

Otro enfoque didáctico de la demodulación FM, que llamaremos *caso 1b*, parte de considerar la recepción de dos señales en cuadratura I y Q, y su procesamiento puramente en software. Este enfoque es de suma utilidad en la transición hacia modulaciones vectoriales más complejas. En la Ec. 4 se muestran dos expresiones para la señal pasabanda real general de entrada a un receptor de radio. La expresión $[I(t)+jQ(t)]$, ampliada en la Ec. 5, es la banda base compleja, que consta de una componente de amplitud $a(t)$ y fase $\phi(t)$, en general todas ellas dependientes del tiempo. La Ec. 6 muestra las componentes de esta banda base para el caso de FM.

$$\begin{aligned} s(t) &= [I(t) + jQ(t)][cos2\pi f_c t + jsen2\pi f_c t] \\ &= a(t)cos[2\pi f_c t + \phi(t)] \\ &= I(t)cos2\pi f_c t - Q(t)sen2\pi f_c t \end{aligned} \tag{4}$$

$$\tilde{s}(t) = I(t) + jQ(t) = a(t)e^{j\phi(t)} \tag{5}$$

$$\begin{aligned} I(t) &= A_c cos2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau \\ Q(t) &= A_c sen2\pi k_f \int_{-\infty}^{t} m(\tau)d\tau \end{aligned} \tag{6}$$

La banda base compleja en modulación angular contiene la información o mensaje en el parámetro $\phi(t)$. En particular, para el caso de FM se debe recuperar $\phi(t)$ y luego derivarla, como se observa en la Ec. 7.

$$\begin{aligned} arg[\tilde{s}(t-1) \cdot \tilde{s}(t)] &= arg[a(t-1)e^{j\phi(t-1)} \cdot a(t)e^{j\phi(t)}] \\ &= \phi(t-1) - \phi(t) \approx \frac{d\phi}{dt} = 2\pi k_f m(t) \end{aligned} \tag{7}$$

Un receptor de FM definido según estas ecuaciones se puede modelar en GNU radio como se observa en la Fig. 4. En primer término, se multiplican las componentes de $s(t)$ por el seno y el coseno de la portadora respectivamente, extrayendo las componentes diferencia y combinando los resultados de ambas en el bloque *float_to_complex*. Esto equivale a multiplicar la señal recibida por la conjugada de la portadora compleja, extrayendo la banda base compleja $\tilde{s}(t)$. En el caso de FM, $a(t) = A_m = cte.$, por lo que se trabaja sobre el argumento de $e^{j\phi(t)}$. Posteriormente, se multiplica la señal resultante por la conjugada compleja retardada un ciclo y se obtiene el argumento de esta operación.

Un punto de especial importancia se refiere a la tasa de procesamiento. En casos como (1b), donde se trabaja con muestras previamente almacenadas o simuladas, la tasa de procesamiento se encuentra en principio regida por la máxima velocidad del procesador. Por ello, es esencial el uso del

Tabla I
BLOQUES GENERALES UTILIZADOS EN LOS CASOS PRESENTADOS

Bloque	**Función**
Freq. xlating filter	Desplazar en frecuencia, filtrar y sub-muestrear
Quadrature demod	Producto de la muestra actual por la conjugada de la muestra anterior
Signal Source	Sintetizar funciones en software
firdes	Herramienta de diseño de filtros FIR
Power Squelch	Interrumpe o anula el flujo de salida según un umbral, un filtro de promediación, así como una curva de respuesta
Rational resampler	Combina submuestreo con interpolación para obtener relaciones decimales entre tasa de E/S
De-énfasis	Compensa la disminución de relación señal-ruido al aumentar la frecuencia
Filtro FIR	Aplica coeficientes y ventanas obtenidos mediante la herramienta *firdes*. Puede además aplicar sub-muestreo o interpolación
Mult. const.	Multiplica por una expresión controlada mediante parámetros
SRRC	Filtro diseñado para disminuir la ISI en RDS
Binary slicer	Cuantificador de 1 bit
Keep one in N	Decimador a nivel de símbolo digital
Differential decoder	Utiliza el módulo de un alfabeto digital para calcular la distancia entre dos símbolos

bloque de gnuradio *throttle*, el cual se encarga de regular la carga que gnuradio impone al procesador. En casos donde se utiliza hardware de adquisición de datos, la velocidad de procesamiento estará regida por la tasa de procesamiento de dicho hardware, por lo que no deben utilizarse bloques *throttle*.

Para escalar hacia implementaciones más complejas, como las del caso (2), resulta fundamental el concepto de *bloques jerárquicos*. Esto se demuestra para el caso de FM de banda ancha mediante el bloque *wfm_rcv*, el cual es un bloque jerárquico que agrupa las funciones *demodulación, de-énfasis y filtrado pasabajo*. Este bloque jerárquico se ilustra en la Fig. 3, donde el resto de los bloques permanece invariable.

III-C. Caso práctico: bloques de alto nivel

En el caso (1) se arriba a un punto donde se dispone de la banda base de FM; sin embargo, esta banda base es *compuesta*, es decir, consta de diferentes zonas, algunas de ellas a su vez señales moduladas, que deben separarse y recuperarse en forma individual [8]. En el caso (1) se recupera el canal mono de audio (I+D) simplemente mediante el filtrado pasabajo que impone la tasa de muestreo de audio. En el presente caso se va un paso más allá, recuperando toda la información, tanto digital como analógica, contenida en la banda base de FM. Para ello es necesario incrementar el nivel de abstracción, agrupando las funciones básicas en bloques jerárquicos y agregando funciones más complejas.

Este caso se analiza e implementa sobre la base de dos diagramas de flujo principales. El primer diseño es [11], desarrollado por un colaborador de Gnuradio y Ettus Research. Ya que fue realizada en una versión obsoleta de Gnuradio, no es posible utilizar esta implementación en forma directa. A raíz de esto, se estudió exhaustivamente el archivo fuente XML y se migró utilizando instancias y configuraciones actualizadas, lo que aportó valiosa experiencia en el funcionamiento del flujo. El segundo diseño, basado en [12], adiciona decodificación del canal digital *Radio Data System (RDS)*, que se detalla en la Sec. III-D.

Cabe destacar que en este caso de estudio se utiliza el bloque *freq_xlating_fir_filter_scf* para implementar el procesamiento del DDC de la Fig. 2 a partir de un muestreo en hardware de 1 MHz, donde el sufijo *scf* indica que el bloque toma un flujo de *shorts* en su entrada, entrega un flujo de *complex* en su salida, y utiliza un filtro con coeficientes tipo *float*. El factor de decimación utilizado es 4, entregando datos a 250 Hz en su salida. En contraste, el caso (1) implementa el DDC *dentro* del dongle, por lo que no es necesario el bloque *freq_xlating_fir_filter_scf*, reduciendo asimismo la carga impuesta al procesador.

En la Fig. 5 se observa el diagrama implementado, en forma conceptual por razones de espacio. Las cadenas superiores están dedicadas a recuperar la información analógica de canal derecho (R) e izquierdo (L), mientras que la cadena inferior recupera información digital del canal RDS. Los bloques *mult const* actúan aquí como *conmutadores controlados por funciones* para definir dinámicamente el comportamiento del diagrama según controles del panel de usuario. Esta es una característica muy utilizada en Gnuradio.

III-D. Procesamiento de canal digital RDS

El sistema *Radio Data System (RDS)* permite incluir información variada a baja velocidad, como emisora, programación, datos climáticos, tráfico, localización, alertas, etc [13] [14]. Utiliza una modulación en subportadora de 57 KHz. Esta frecuencia se encuentra en sincronismo con el tercer armónico del piloto para transmisiones stereo. La modulación utilizada es BPSK con desviación de fase de $\pm 90\,^\circ$ y tolerancia de $\pm 10\,^\circ$.

La banda base en el sistema RDS se codifica en dos etapas, en primer lugar con un codificador diferencial a una tasa de $1187{,}5\ bps$, con el objetivo de recuperar la información aún si la señal llega invertida al receptor. En segunda instancia se utiliza un codificador bifase con el objetivo de incorporar la información del reloj de sincronismo de datos, esta operación se realiza a 2 veces el reloj de transmisión, es decir 2375 Hz. Para obtener la banda base del sistema *RDS*, se propone un flujo de señal como el de la Fig. 5.

Luego de demodular la banda base mediante el bloque jerárquico WBFM como en la Fig. 3, se aplica un filtro pasa bajos a fin de extraer la porción con información digital de la banda base centrada en 57 KHz. Posteriormente se aplica un filtro de raíz cuadrada de coseno realzado *(Square root raised cosine filter, SRRC)*, a fin de reducir al mínimo la interferencia intersímbolo (Inter-Symbol Interference, ISI). Seguidamente se utiliza el bloque receptor *MPSK-Receiver* configurado en nivel dos, es decir PSK de dos niveles (BPSK). Con el fin de discretizar los datos, se implementa el bloque *Binary Slicer*. Finalmente se realiza un diezmado a nivel de bits mediante la implementación del bloque *Keep 1 in N* y se decodifica la

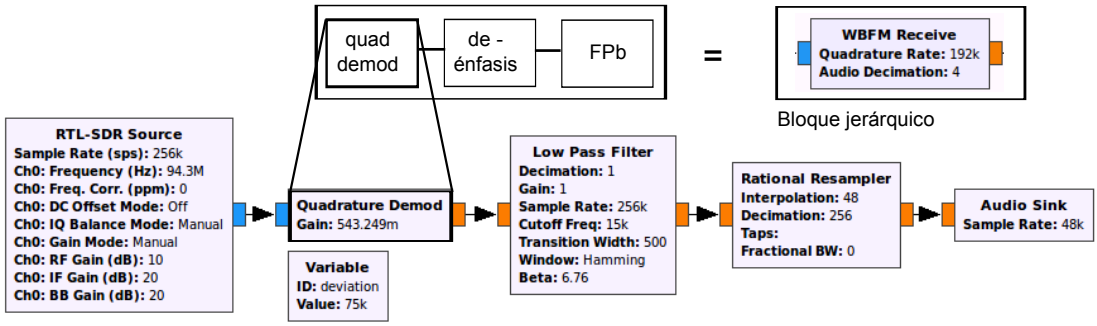

Figura 3. Ejemplo de implementación del caso 1a

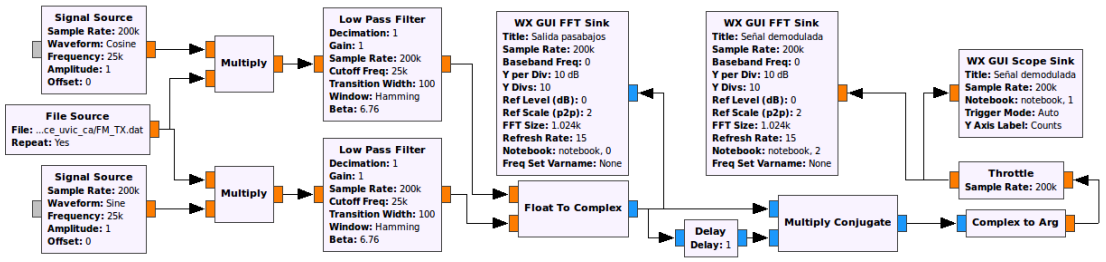

Figura 4. Ejemplo de implementación del caso 1b

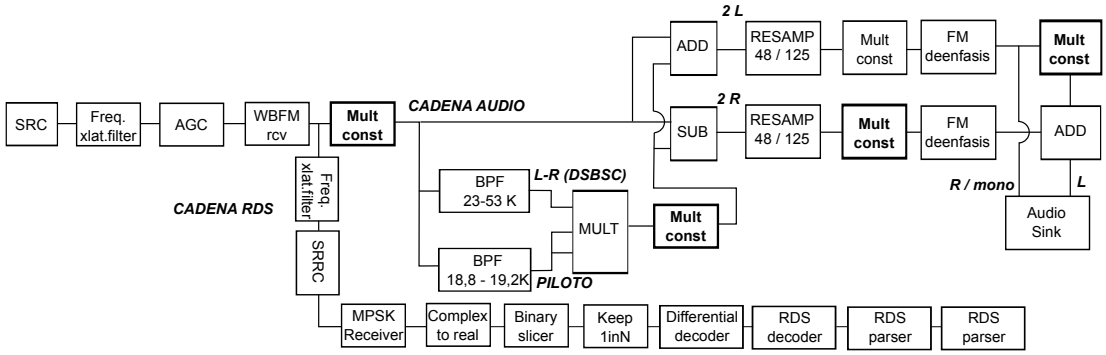

Figura 5. Ejemplo de implementación del caso 2

información a texto mediante el bloque *Differential Decoder*, el cual produce la decodificación de códigos binarios, *RDS Decode* y *RDS Parser*.

Para todo el proceso se utiliza una frecuencia de muestreo f_s = 1 MHz, pero al realizar un diezmado por 4 en el filtro pasa bajos, se debe reducir dicha frecuencia en el mismo factor en el filtro de raíz cuadrada de coseno realzado y receptor MPSK. Las Tablas II y III representan los parámetros más importantes de estos bloques respectivamente.

Tabla II
PARÁMETROS FUNDAMENTALES DEL BLOQUE *Root raised cosine filter*

Parámetro	**Función: valor**
Relación de símbolo (Symbol Rate)	Baudios de la transmisión: 2375
Alpha α	Factor de Roll-off: 1
Num Taps	Número de etapas:100

Tabla III
PARÁMETROS FUNDAMENTALES DEL BLOQUE *MPSK Receiver*

Parámetro	**Función: valor**
Nivel de modulación (M)	Para modulaciones BPSK:2
Loop Bandwith	Rango de captura del bucle de costas: $2\pi/50$
Omega	Relación entre la frecuencia de muestreo y los simbolos de la transmisión :f_s/baudrate

IV. RESULTADOS Y CONCLUSIONES

En las señales resultantes se observa claramente el concepto del ancho de banda de una señal en cuadratura (datos complejos) con respecto a una señal real. Por ejemplo, en las Figs. 6 (a) y 6 (b), donde las señales son complejas, se observa que la frecuencia de muestreo f_s define el ancho de banda *total* de la señal pasabanda, mientras que en la Fig. 6 (c) la señal es el canal I+D real, que se distribuye en el rango 0 - 24 KHz,

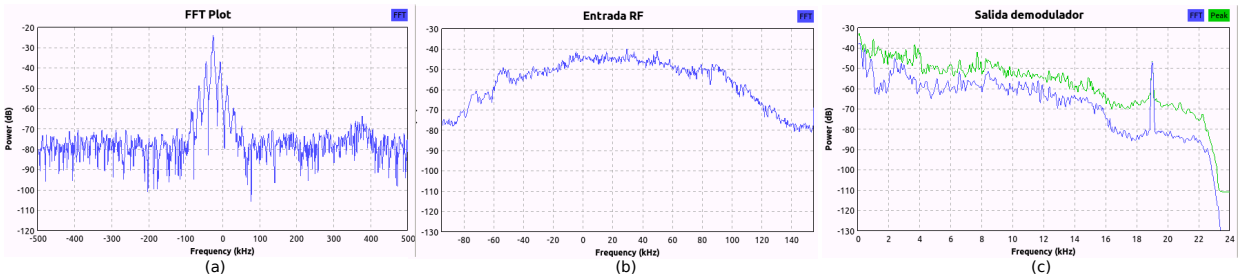

Figura 6. Señales resultantes - caso 1c: (a) $f_s = 1MHz$, (b) $f_s = 250KHz$, (c) Señal de audio

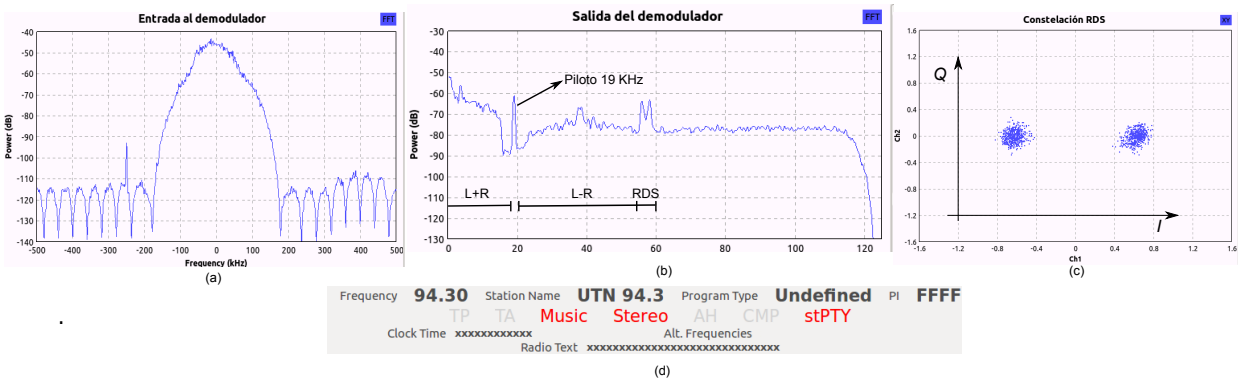

Figura 7. Señales resultantes - caso 2: (a) señal FM compleja, (b) banda base compuesta, (c) constelación canal RDS, (d) información RDS

que es la *mitad* de la tasa de muestreo de la placa de audio, 48 KHz.

Otra observación importante se refiere al concepto teórico ilustrado en la Fig. 2. En el caso (1), la fuente RTL-SDR de ajusta con una tasa de muestreo ajustada a la aplicación (por ejemplo 256 KHz en la Fig. 3, mientras que en el caso (2) la tasa de muestreo se mantiene en un valor sobre-dimensionado (por ejemplo 1 MHz) y luego se ajusta en software mediante el bloque *freq_xlating_filter*. Es decir, los primeros dos casos implementan el DDC de la Fig. 2 en el hardware de adquisición, mientras que el caso (3) lo hace totalmente en software. Esto permite al caso (2) trabajar con hardware que no cuente con un DDC, sin embargo impone una carga adicional al procesador del sistema. Para anchos de banda considerables, esta diferencia puede ser notable.

En la Fig 7(a) se observa la tasa de datos compleja de 250 Hz a la salida del DDS por software. La Fig. 7(b), en tanto, ilustra claramente la banda base compuesta de FM, donde se observa el canal L+R, el canal L-R modulado en doble banda lateral con portadora suprimida (DSBSC), y el canal RDS modulado en BPSK, todos ellos procesados mediante las cadenas del diagrama de la Fig. 5. Finalmente, las Figs. 7(c) y 7(d) muestran respectivamente la constelación $I - Q$ RDS y la información típicamente transportada en este canal.

Mediante los simples casos presentados se demuestra cómo, mediando cuidadoso análisis teórico previo y correspondiente interpretación de las implementaciones, SDR y Gnuradio ofrecen gran potencial como herramienta didáctica de amplia difusión, así como para aplicaciones prácticas en investigación y desarrollo.

BIBLIOGRAFÍA

[1] T. Ulversoy, "Software Defined Radio: Challenges and Opportunities, " in IEEE Communications Surveys & Tutorials, vol. 12, no. 4, pp. 531-550, Fourth Quarter 2010.

[2] R. Machado-Fernandez, "Software Defined Radio: Basic Principles and Applications, " Fac. Ing. vol.24, n.38 pp.79-96, 2015.

[3] Matthias Fähnle, "Software-Defined Radio with GNU Radio and USRP/2 Hardware Frontend: Setup and FM/GSM Applications, " Bachelor Thesis, TU Ulm, 2010.

[4] E. Blossom, "GNU radio: tools for exploring the radio frequency spectrum, " Linux Journal, no. 122, p. 4, 2004.

[5] M. Beazley, "SWIG : An Easy to Use Tool For Integrating Scripting Languages with C and C++, " Proc. Fourth Annual USENIX Tcl/Tk Workshop, 1996.

[6] T. Rondeau, N. McCarthy, and T. O'Shea. "SIMD programming in GNURadio: Maintainable and user-friendly algorithm optimization with VOLK, " Conference on Communications Technologies and Software Defined Radio (SDR'12). Brussels, Belgium: Wireless Innovation Forum Europe, 2012.

[7] Boris Schling, *The Boost C++ Libraries*, XML Press, 2011.

[8] T. Rondeau, "GNU Radio for Exploring Signals. Talk Hard: A technical, historical, political, and cultural look at FM, " Proc. FOSDEM 2016.

[9] B. Seeber, "GNU Radio Tutorials, " Ettus Research, 2014.

[10] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed., New York, USA: McGraw-Hill, 2007.

[11] M. Leech. (2012, May) Simple FM receiver [Online] Available: https://github.com/croisez/sdr/tree/master/simple_fm_rcv

[12] B. Bloessl, "First Steps in Receiving Digital Information with RDS, " Free Open Source Developers' European Meeting (FOSDEM), 2015.

[13] A. Zuloaga Izaguirre (1996, July) Radio Data System [Online] Available: http://www.redesmadrid.com/descargas/rds/upv.pdf

[14] P. I. Martos and J. L. Bonadero, "FPGA-based Digital Demodulation, " Proc. XII RPIC, 2007.