COMUNICACIONES - AÑO 2018

Laboratorio 1: Demodulación de señales con Dongle SDR[1]

1 Introducción

Este laboratorio, previsto como una de las actividades con nota de la materia Comunicaciones (E0311), consiste en la utilización de un *Dongle* USB para analizar e implementar algunos aspectos de lo que se conoce como *Software-Defined Radio* (SDR) o Radio Definida por Software. Esta término refiere a un elemento de un sistema de comunicaciones (receptor o transmisor) en el cual parte del procesamiento que habitualmente se realizaba de manera analógica (filtrado, conversión de frecuencia, demodulación) se implementa de manera digital mediante un software operando en una PC o sistema embebido.

La realización de las actividades del laboratorio se realizará en el tiempo y lugar elegidos por los alumnos ya que se entregará un Dongle USB a cada uno. Para la calificación del Laboratorio cada estudiante deberá entregar un breve informe con los resultados y presentar los mismos colectivamente para discutir. La fecha límite para esta discusión es a convenir pero se realizará antes del cierre formal del cuatrimestre (y preferiblemente previo al receso invernal).

2 Objetivos

Los objetivos de este laboratorio, vinculados con lo que concebimos como parte de las cualidades de un buen profesional de la ingeniería y/o con los objetivos de formación de nuestra asignatura son:

- Presentar un caso de procesamiento digital aplicado a las comunicaciones (el procesamiento digital tiene mucho para aportar a las comunicaciones, saber de temas de comunicaciones es muy útil para hacer procesamiento digital).
- Analizar las ventajas que ofrece, así como las dificultades que presenta la implementación de receptores SDR.
- Vincular diferentes temáticas estudiadas en la materia: sistemas de modulación lineal y exponencial, modelo pasabanda de señales, densidad espectral de potencia.
- Fomentar la utilización de herramientas de procesamiento para complementar la formación en los temas incluidos en la currícula.
- Fortalecer las habilidades para exposición en público.

3 Breve descripción del Dongle SDR

El Dongle a utilizar se encuentra formado por dos integrados principales:

- Un sintonizador de señales **R820T2**: Puede sintonizar un rango de frecuencias de 24 MHz a 1700MHz. Posee una figura de ruido de 3.5 dB y consume alrededor de 180 mA.
- Un DSP y controlador USB **RTL2832U**: A pesar de que la hoja de datos de este chip no es accesible para usuarios convencionales, se sabe que posee un ADC y un DSP que realiza la conversión de frecuencia intermedia a banda base a través de mezcladores en fase y cuadratura (I/Q), el filtrado pasa bajos, el re-muestreo en fase y cuadratura y el envío de dichas muestras por el puerto USB.

[1]Agradecemos la generosidad de los docentes de la materia Comunicaciones E0214, en cuyos materiales *nos inspiramos impunemente* para este laboratorio.

En la Fig. 1 se muestra el circuito impreso del Dongle junto a un diagrama esquemático de conexión.

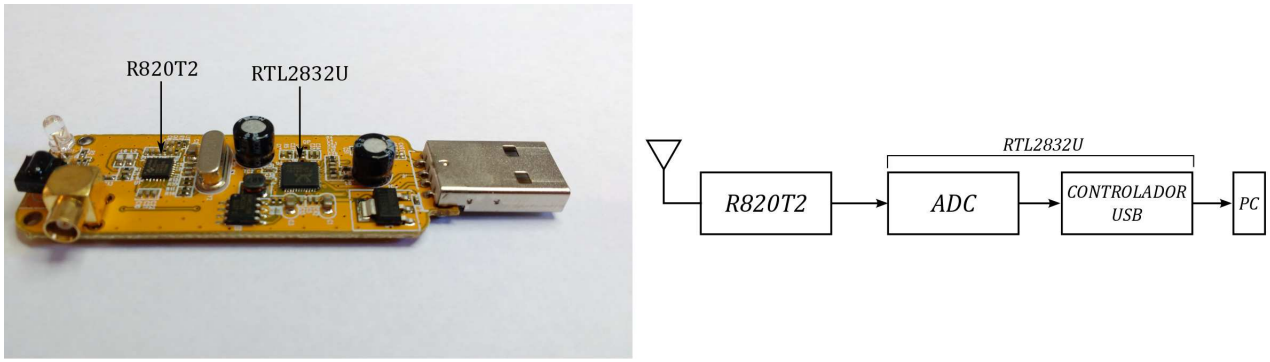

Fig. 1: PCB del Dongle. Diagrama en bloques simplificado.

4 Un poco de práctica con el SDR#

Este ejercicio busca introducir nociones básicas del Dongle. Se utilizará la librería **rtl-sdr-release** y el software **SDR#**, disponibles en la web de la cátedra. La idea será sintonizar algunas estaciones comerciales y notar algunas características básicas de las mismas. Los materiales necesarios serán: Una PC con Windows o Linux con el SDR# intalado, un Dongle y auriculares o parlantes.

1. Abra el SDR# teniendo el Dongle conectado a la PC y conectese utilizando la opción RTL-SDR(USB). Tenga en cuenta que primero debió instalar los drivers utilizando zadig.exe como se explica en la guía de instalación [1].

2. Haga un barrido por el espectro, desde el comienzo de la banda de FM comercial hasta el final, dando saltos de a 100 kHz. Las distintas estaciones de radio deberían distinguirse a simple vista. Coloque la opción WFM en la pestaña radio y elija las estaciones que posean mayor potencia. Utilice los parlantes o auriculares para escuchar.

Al elegir la opción WFM, el programa comienza a ejecutar un código que demodula la señal de FM comercial. Por defecto, supone a las estaciones con un ancho de banda de 250 kHz.

3. Note que puede modificar el ancho de banda de la señal colocando el cursor sobre el límite del ancho de banda del filtro actual haciendo click y arrastrando. Verifique con los auriculares que al achicar el ancho del filtro empieza a “escucharse mal”.

A su vez, tanto el chip que hace de sintonizador como el chip que muestrea y controla el USB posee ganancia regulable y hasta control automático de ganancia. Desde el SDR# (ícono de engranaje) es posible controlar los mismos.

4. Verifique las diferencias en el espectro si se activa/desactiva el control automático de ganancia. Cambie los valores de ganancia manualmente y observe los cambios. Quizás note que a veces “desaparecen” algunas estaciones.

Dado que la frecuencia mínima de operación del Dongle es superior a la máxima frecuencia de una estación de AM comercial, no es posible recibir estas señales. En su lugar “transmitiremos” una señal AM en una banda no ocupada por otras señales (por ejemplo 110 MHz).

4. Analice cómo resulta el espectro y la señal demodulada en este caso. También resulta útil analizar qué ocurre al activar el control automático de ganancia y al configurar manualmente diferentes valores de ganancia.

Todo lo anterior debería ser de utilidad para implementar las rutinas de procesamiento que permitan realizar la demodulación de estas señales.

5 Tarea

La tarea consiste en demodular muestras obtenidas con el Dongle USB de alguna estación de FM comercial a elección de cada alumno. Debe implementar una función en MATLAB o Python que realice la demodulación de dichas muestras y permita escuchar el mensaje (señal de audio).

5.1 Bonus

Como Bonus que promete elevar la nota final del alumno que lo concrete, se establece la tarea de realizar la demodulación de la estación comercial de FM en tiempo real y en lo posible, implementar también una interfaz gráfica que permita cambiar de estación, modificar el volumen y cualquier otra función que le parezca interesante. Todo suma ;).

6 Algunas ayudas

Para resolver el inciso anterior, conviene construirse algunas funciones que sirvan como herramientas para ir depurando y visualizando los resultados y avances.

1. Puede generar un archivo de muestras desde la librería rtl-sdr-release. Por ejemplo, para generar un registro de 21×10^6 muestras provenientes de una estación en 99.5 MHz, utilizando la frecuencia de muestreo por defecto de 2.048 MHz y guardando las muestras en un archivo *Samples* debe abrirse una consola y escribir:

   ```
   rtl_sdr.exe -f 99.5e6 -n 21e6 Samples
   ```

   Ahora en la misma carpeta se tiene un archivo Samples que posee muestras en fase y cuadratura en banda base de la señal centrada en 99.5 MHz.

2. Para generar una función en MATLAB que lea este archivo puede guiarse por los siguientes comandos

   ```
   function y = loadFile(filename)
   % y = loadFile('nombrearchivo')
   %
   % Lee el archivo generado por rtl_sdr.exe y entrega un vector
   % complejo (I y Q)

   fid = fopen(filename,'rb');
   y = fread(fid,'uint8=>double');
   y = y-127;
   y = y(1:2:end) + i*y(2:2:end);

   end
   ```

3. Ahora que tiene el vector de muestras, grafique el espectro a ver qué tal se ve. Contemplando que a lo largo del trabajo habrá muchas señales a las cuáles se requiere observar el espectro, es sumamente útil construirse una función que realice la estimación del mismo y grafique el resultado detalladamente. Puede usar como ejemplo la función siguiente, aunque puede modificar algunos parámetros según le sea más conveniente.

   ```
   function [f, Sxx] = DEP(x,fs,signal_name)
   % [f, Sxx] = DEP(x,fs,signal_name)
   ```

```
% INPUTS:
%    x  = señal de entrada
%    fs = frecuencia de muestreo
%    signal_name = cadena de caracteres para el titulo. Ej. 'Señal antena'
% OUTPUTS:
%   f = vector de frecuencias donde se evaluó la DEP
%   Sxx = Estimacion de la DEP

[Sxx,f] = pwelch(x,ones(1,8192),0,[],fs,'twosided');
Sxx = fftshift(Sxx); f = fftshift(f);
f(1:floor(length(f)/2)) = f(1:floor(length(f)/2)) - fs;

handle = figure('units','normalized','outerposition',[0 0 1 1]);
plot(f,Sxx); xlim([-fs/2 fs/2]);
set(gca, 'FontSize', 18);
legend('S_{XX}(f)','Location','NorthEast');grid on;
xlabel('f [Hz]','Interpreter','Latex','FontSize',20);
title(['DEP de ' signal_name],'FontSize',20);
grid on;

end
```

4. Dado que en los registros puede haber más de una estación, o haber tomado un ancho de banda excesivo, siempre resulta interesante realizar el filtrado. Para ello son de utilidad los comandos `butter`, `fdatool` y `filter`. En cada caso, resulta conveniente analizar que el filtrado fue exitoso, graficando el espectro con la función DEP dada anteriormente.

5. Un esquema de un demodulador típico de FM se muestra en la Fig. 2. Siguiendo el diagrama de la misma, lo primero es configurar al Dongle para obtener N muestras a una tasa f_s de una estación de radio FM centrada en f_c y guardar dichas muestras en un archivo. Luego en Matlab se genera el vector complejo de la señal utilizando la función *loadfile.m* provista anteriormente.

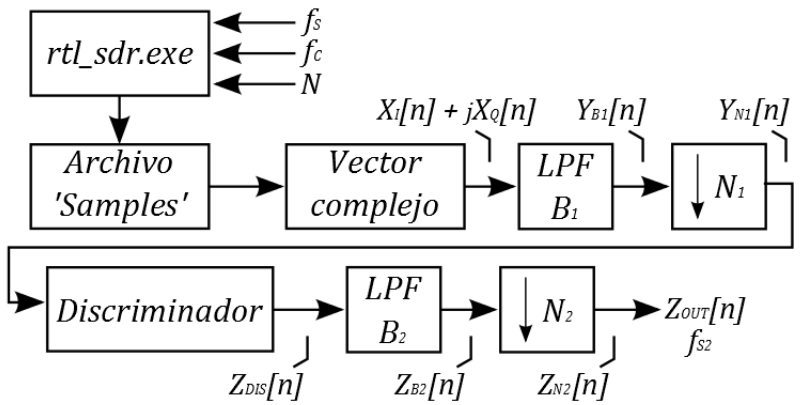

Fig. 2: Esquema de un demodulador de FM por procesamiento digital.

Debido a que en las muestras puede haber más de una estación, se debe realizar un filtrado pasa bajos con un ancho de banda acorde B_1 tal que sólo deje pasar la estación querida. En procesamiento digital es muy importante procesar el mínimo número de muestras posible, para ahorrar operaciones y tiempo de ejecución. Si se toman muestras a $f_s = 2.048$ MHz (frecuencia

de muestreo por defecto del Dongle) durante 10 segundos, se tienen $N = 20.48 \times 10^6$ muestras. El espectro de estas muestras irá de $-fs/2$ a $fs/2$, es decir de -1.024 MHz a 1.024 MHz. Pero el ancho de banda B_1 en general es bastante menor que este rango, por lo que podría bajarse la tasa de muestreo en un factor N_1 tal que nuestra señal filtrada quede dentro del rango $-\frac{f_s}{2N_1}$ a $\frac{f_s}{2N_1}$ con la ventaja de que ahora hay que procesar N/N_1 muestras a la *nueva* frecuencia de muestreo $f_{s1} = f_s/N_1$. Esta operación se conoce como **diezmado** y puede ejecutarse utilizando en matlab:

```
y_N1 = decimate(y_B1,N1,'fir');
```

El valor de N_1 se debe elegir teniendo en cuenta lo explicado en el párrafo anterior.

La implementación del discriminador de frecuencia es similar a la presentada en la Práctica 5. Recuerden que pueden serle de utilidad los comandos `atan2`, `unwrap` y `diff`. Después de la demodulación se obtendría la señal marcada en la Fig. 2 como $Z_{dis}[n]$. Dibuje el espectro de esta señal a ver qué contenidos frecuenciales ve. Debería obtener un espectro de "audio". Dependiendo la estación de radio que haya elegido, puede que vea un tono (piloto) en 19 kHZ. ¿A qué se debe? Busque en internet. ¿Qué permite obtener este piloto?

Finalmente, es posible realizar un filtrado más apropiado para nuestraseñal $Z_{dis}[n]$ eligiendo un filtro pasa bajos de ancho B_2. Para verificar el correcto funcionamiento del algoritmo hasta aquí, debería poder escuchar el audio obtenido utilizando la función **sound** de Matlab. El problema que puede presentarse aquí es que las placas de sonido no reproducen cualquier frecuencia de muestreo, pero se sabe que funcionan bien para frecuencias cercanas a 48 kHz. Por lo tanto, quizás necesite hacer un segundo diezmado por un factor de N_2. Ahora sí, escuche el mensaje! Quizás sea bueno normalizar el mensaje por su máximo para que la amplitud máxima sea 1 antes de enviar la señal a la placa de sonido.

```
Z_out = Z_out ./ max(abs(Z_out));
sound(Z_out,f_s2);
```

Bibliografía

[1] Instalación de Drivers Dongle SDR 820T2. *Web de la cátedra.*

[2] Lab 6 Software Defined Radio and the RTL-SDR USB Dongle. *ECE 4670 Spring 2014.*