Universidad Nacional de Río Cuarto Facultad de Ingeniería

Cátedra:

Sistemas y Señales II Código: 0021

Trabajo Práctico N°1

GNU Radio Companion

Fecha de Realización:

Integrantes del Grupo	
Apellido y Nombre	DNI
Dellepiane Gino	42031381
Gianmmarco Agustín	45593703

Fecha de Presentación:

Calificación:_____

Firma Docente: _____

Ejercicio 1:

Dos señales analógicas $x_{1(t)}$ y $x_{2(t)}$ son como siguen:

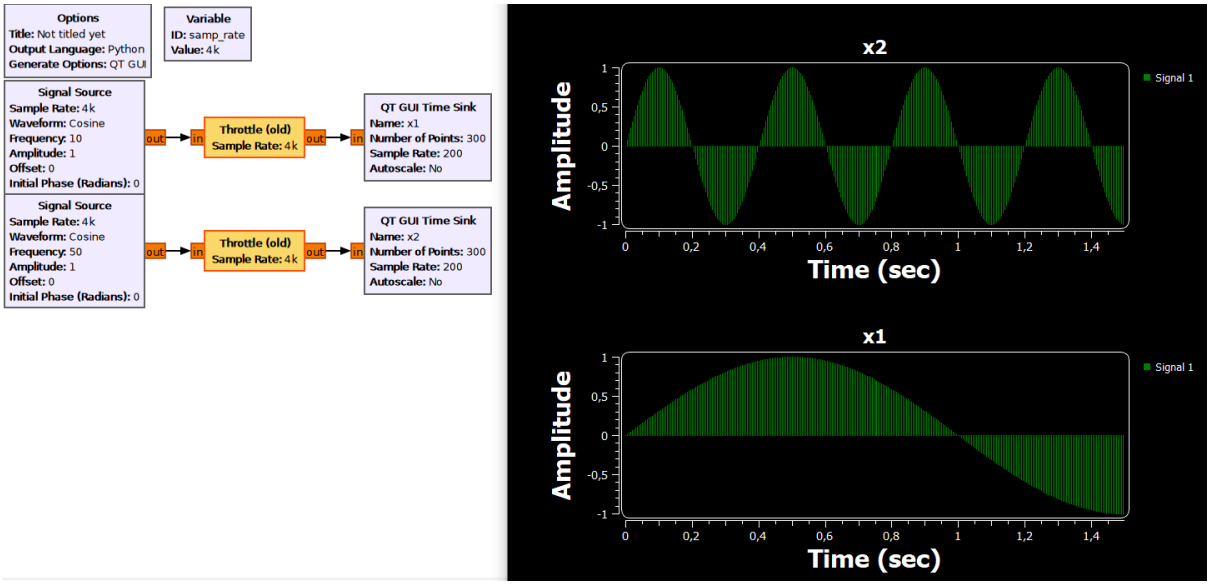
$$x_{1(t)} = cos(2\pi * 10t)$$
$$x_{2(t)} = cos(2\pi * 50t)$$

Las dos señales son muestreadas a una tasa de muestreo $f_s = 200Hz$.

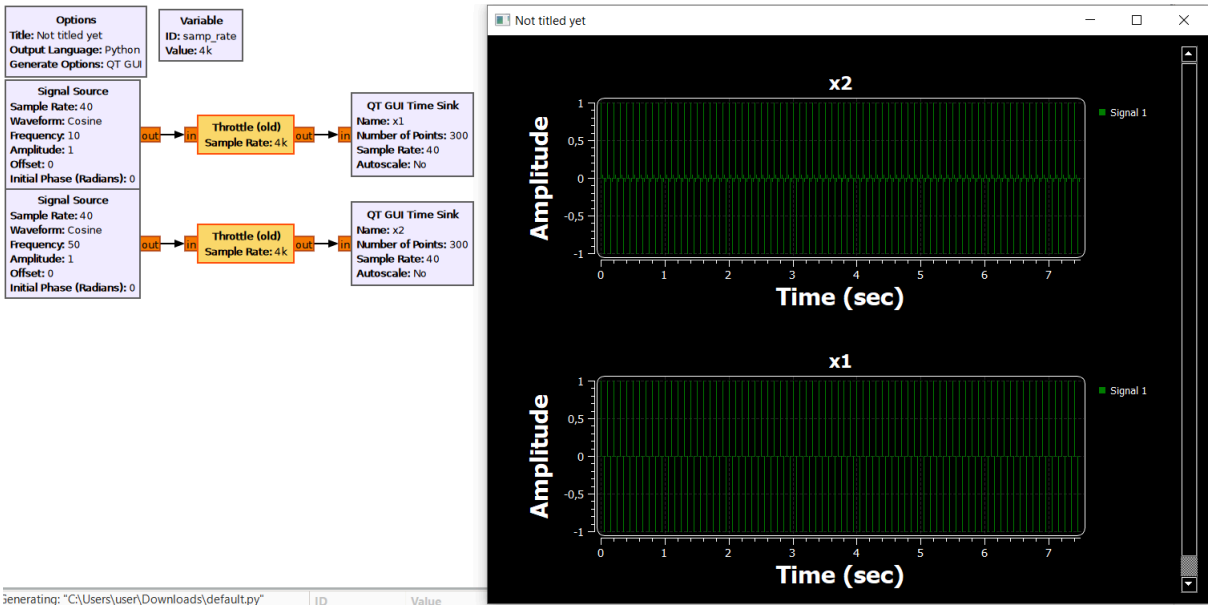
a) Encuentre las secuencias $x_{1[n]}$ y $x_{2[n]}$ de manera teórica.

$$x_{[n]} = x_{(t)}|t = \frac{n}{f_s}$$
$$x_{1[n]} = cos(\frac{2\pi*10*n}{200}) = cos(\frac{\pi n}{10})$$
$$x_{2[n]} = cos(\frac{2\pi*50*n}{200}) = cos(\frac{\pi n}{2})$$

b) Implemente usando el Companion, y visualice las señales. Utilice Stem plot, para ver los puntos muestreados en el bloque QT GUI Time Sink. Recuerde usar el bloque Throttle. Nota: Configure el Number of Points en QT GUI Time Sink en 300 muestras. Para este ejercicio particular, configure el bloque Throttle en Sample Rate=4000.



c) Ahora cambie la tasa de muestreo fs, a 40 muestras por segundo (40 Hertz). Vuelva a graficar cada señal. Explique el fenómeno de aliasing que se observa en las gráficas del inciso b.



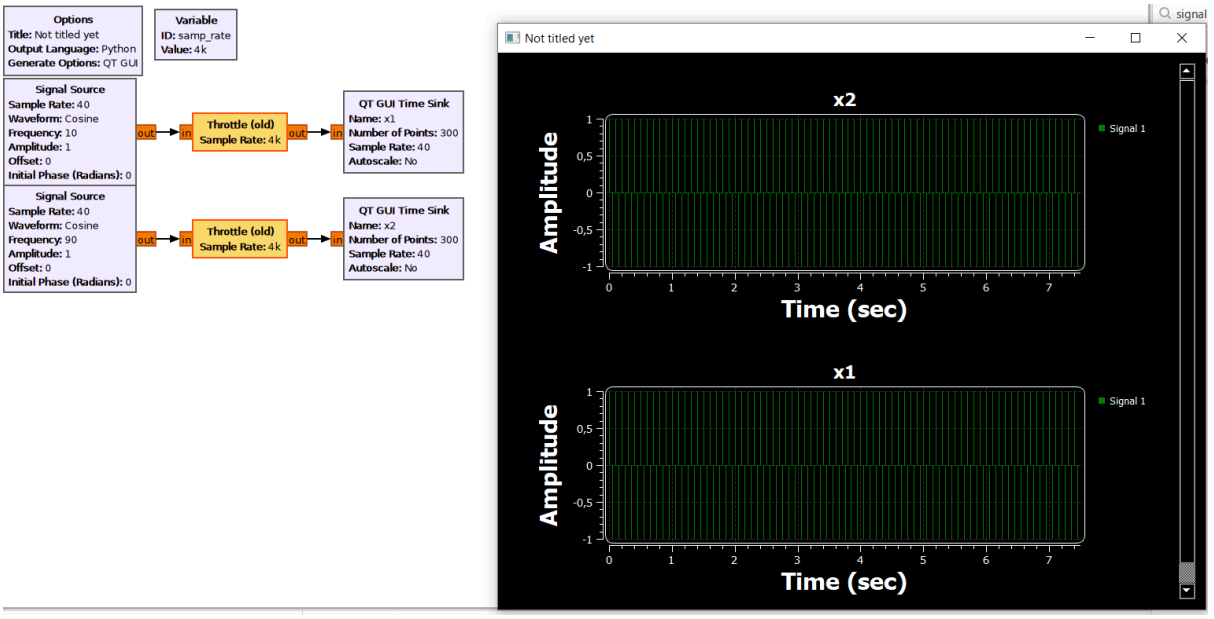
Sabiendo el teorema de Nyquist, podemos ver que en la primera frecuencia de muestreo no hay aliasing, esto es porque la regla se cumple:

$$f_o < f_s/2$$
$$f_o < \frac{200Hz}{2}$$
$$f_o < 100Hz$$

En la segunda frecuencia de muestreo vemos que ya no cumple con este teorema, ya que haciendo los mismos procedimientos podemos apreciar que solo se cumple para aquellas frecuencias que no superen los 20Hz. Así mismo podemos visualizar que la señal $x_{2(t)}$ está mal muestreada, y así mismo, esta señal se parece a la $x_{1(t)}$, con una frecuencia de 10Hz

d) ¿Qué otras señales producen un alias de la señal x1[n] cuando fs=40 Hertz? Muestre un ejemplo en GNURADIO Companion.

Toda señal que supere los 20Hz



$$x_{3(t)} = cos(2 * \pi * 90 * t)$$

e) Encuentre una ecuación matemática para determinar todas las señales tales que, al ser muestreadas con fs=40 Hertz, son indistinguibles de la señal x1[n] cuando ha sido muestreada también a esta fs.

Dos señales continuas $cos(2\pi ft + \theta)$ y $cos(2\pi f`t + \theta`)$ producen la misma secuencia muestreada si y sólo si sus argumentos muestreados difieren en un múltiplo entero de 2π o se relacionan por simetría de coseno. Esto lleva a las condiciones de aliasing.

$$f = \pm 10Hz + 40k \; con \; k \epsilon Z$$

Ejercicio 3:

Una señal de tiempo continuo x(t) tiene la forma de:

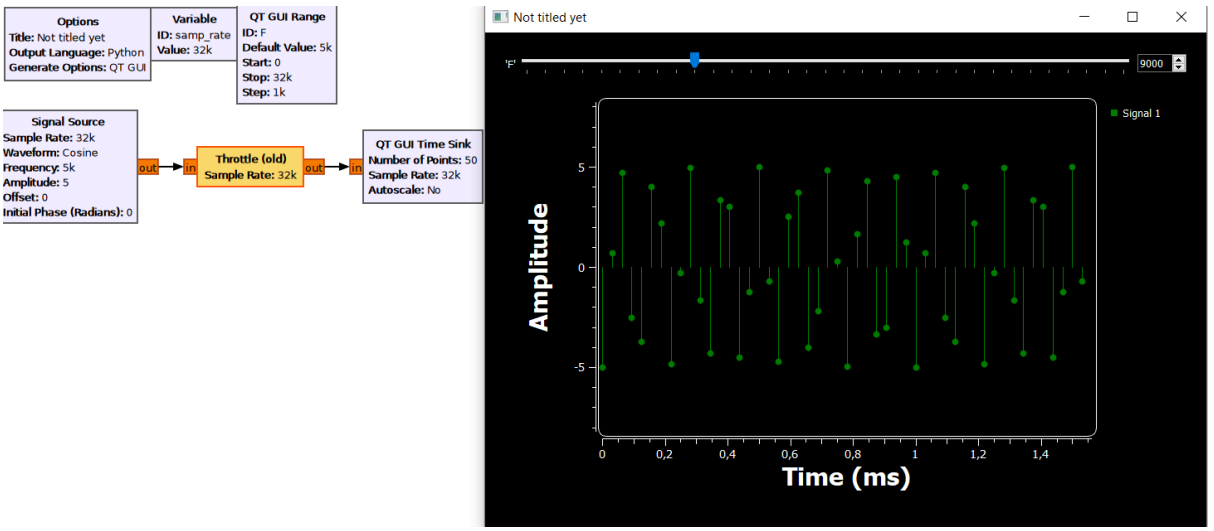
$$x_{(t)} = 5cos(2\pi 5000t)$$

Considere inicialmente una frecuencia de muestreo tal que permita una correcta reconstrucción de la señal. Utilice el bloque QT GUI Range del Companion para modificar la misma.

$f_s > 2f_o$

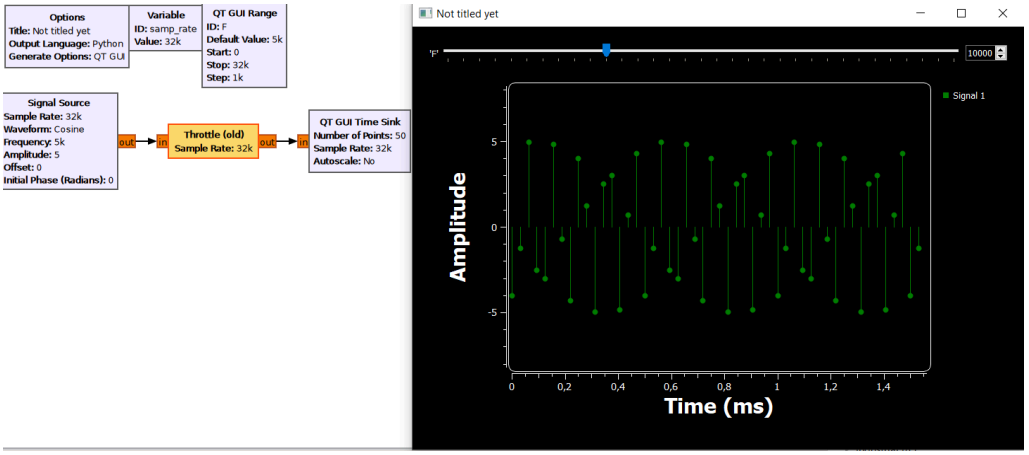
$f_s > 10000Hz$

a) ¿Qué pasa cuando usamos una fs (fs: frequency sample, o frecuencia de muestreo, o tasa de muestreo) menor a la mínima requerida por el criterio de Nyquist?



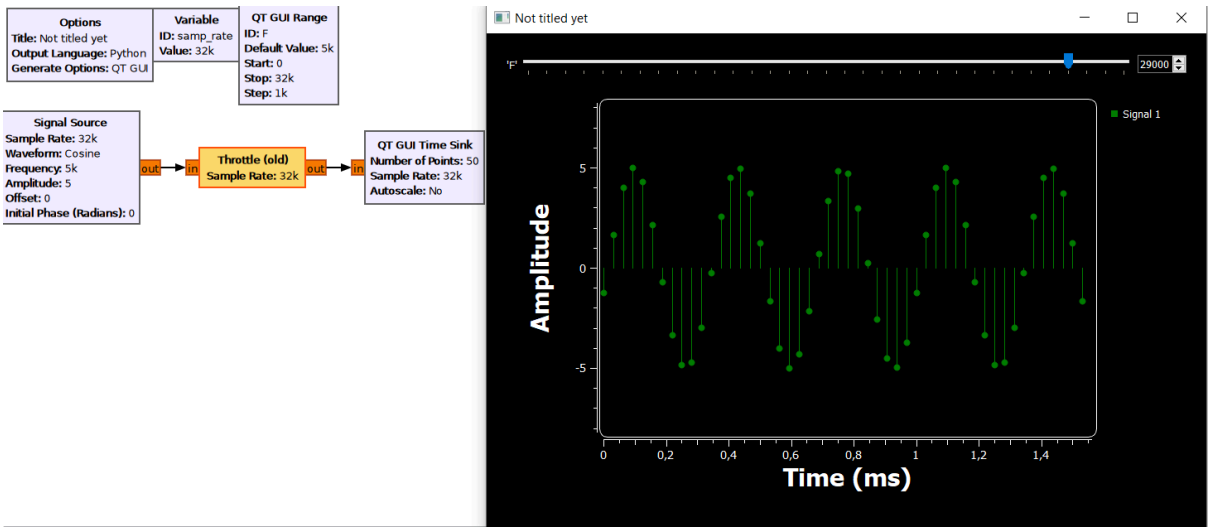
Se produce el efecto aliasing

b) ¿Una fs igual a la de Nyquist que produce?



El mismo efecto

c) ¿y una superior a la de Nyquist?, ¿qué mejora aún más al aumentarla?



Ejercicio 5:

Una señal compleja puede descomponerse y/o representarse en coordenadas ortogonales (IQ, o bien, parte real y parte imaginaria) o también, en coordenadas polares (módulo o mag, y fase). Ambas representaciones contienen la misma información para describir la señal.
Considere una señal compleja de tiempo continuo x(t) de la forma:

$$x_{(t)} = e^{j2\pi 1000t}$$

Utilice una fs=64Khz

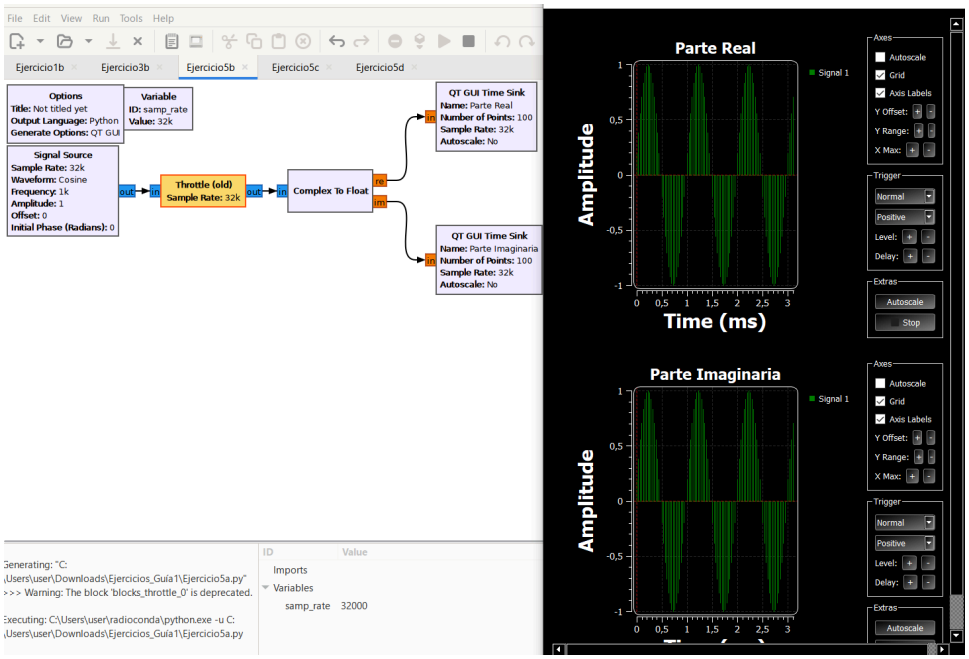
a) Considere la representación fasorial de la señal x(t) ¿A qué velocidad angular se mueve el fasor?, ¿cuántas vueltas da el fasor en el tiempo de 1 segundo?, ¿cuánto tiempo tarda el fasor en dar una vuelta?

   Se mueve a $2\pi 1000$

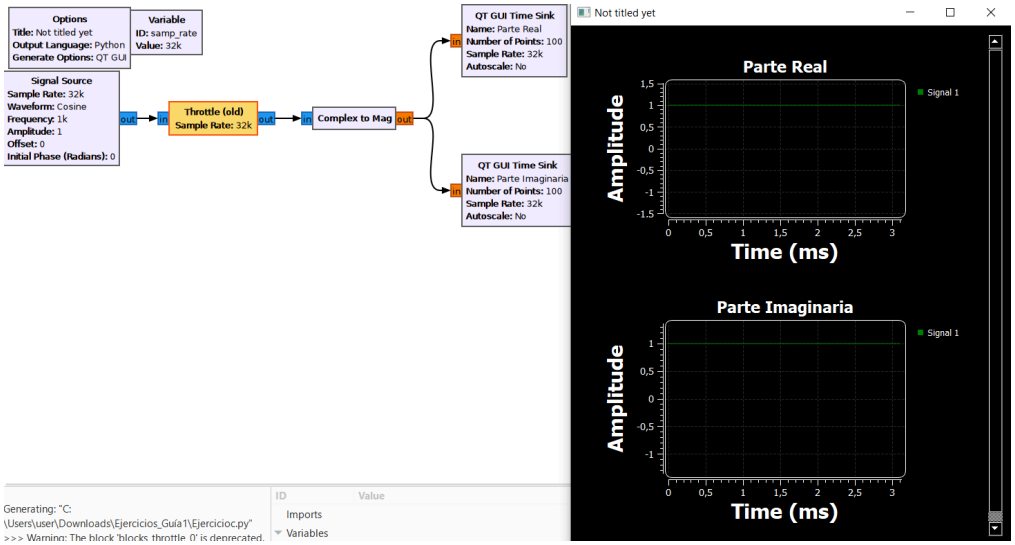
   $f_o = 1000Hz$, Da 1000 vueltas por segundo

   $T = 1/f_o = 1/1000\ [s]$, eso tarda en dar una vuelta

b) Utilice el bloque Complex to Float y grafique la parte real e imaginaria de la señal, por separado y en el tiempo. ¿Qué señal se ve en cada caso?



c) Utilice el bloque Complex to Mag Phase y grafique ambas salidas. Explique cada gráfico. ¿Cuánto vale la fase a los 10.30ms? Dibuje el fasor en este tiempo de 10.30ms. ¿Cuántas vueltas dió?



para 10.3 ms, la amplitud mide 1

Ejercicio 7:

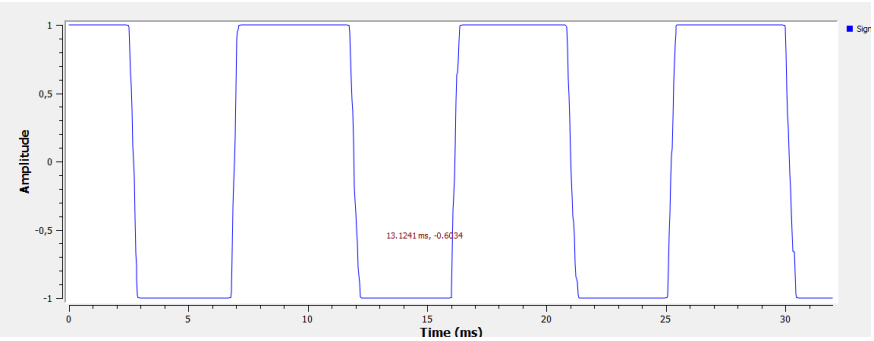
Se ha grabado el tono (el sonido del tono) que produce una cuerda de guitarra tocada al aire, afinada en el estándar A4=440Hz, a una tasa de 32Khz, en el archivo guitar.wav.

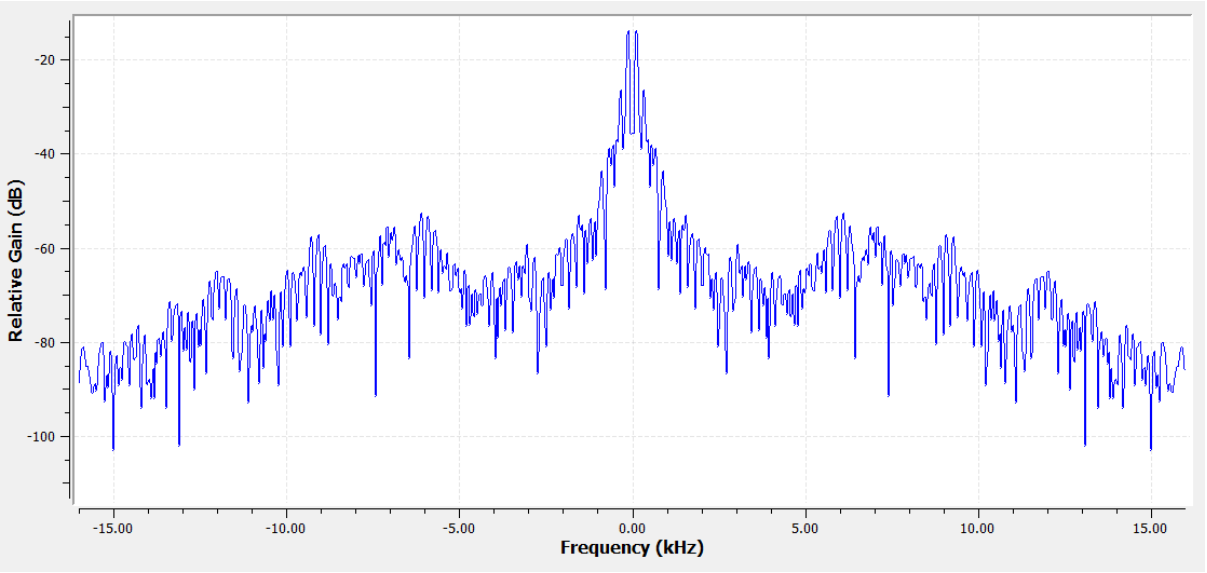
a) Usando Companion: ¿Qué forma de onda tiene el tono de la cuerda de guitarra grabada?, ¿qué frecuencia ha identificado? Nota: Si el programa arroja error en audio sink, es por la placa de sonido. En ese caso, coloque el valor: plughw:0,0 , en el campo Device Name, del bloque Audio Sink.

Tiene forma de onda cuadrada y su frecuencia es de 1KHz

b) ¿A qué nota musical de la guitarra corresponde este tono? Una nota en una guitarra, se produce cuando tocamos una sola cuerda a la vez. Una nota es un tono, es decir, se asocia con una frecuencia particular (aunque también, dependiendo del instrumento, se producen distintas frecuencias relacionadas (armónicos) que le dan el “sonido distintivo” a ese instrumento). Las notas se enumeran de la A a G (en el cifrado americano) o de DO a SI (en el sistema latino). A continuación dejamos un enlace para que investigue y pueda deducir que nota es, según el tono que identificó.

1000Hz representa casi a un DO o C





Ejercicio 9:

Recomendamos seguir el capítulo 1.4.4: Cuantificación de señales sinusoidales, del libro Tratamiento digital de señales, Proakis, para resolver este ejercicio.

Cuando generamos una señal con GNU Radio Companion, por ejemplo con el bloque Signal Source, podemos elegir el tipo de variable para la secuencia de salida (Output Type)

El tipo de variable Float, usa 32 bits para representar una muestra, que es una representación de punto decimal de precisión simple, o también llamado float32s.

a) Responda:
    i) Esta forma de representar una muestra: ¿Introduce algún error de cuantización?

        Sí, introduce un error de cuantización, aunque muy pequeño. tenemos:

        1 bit para el signo

        8 bits para el exponente

        23 bits para la mantisa (fracción significativa)

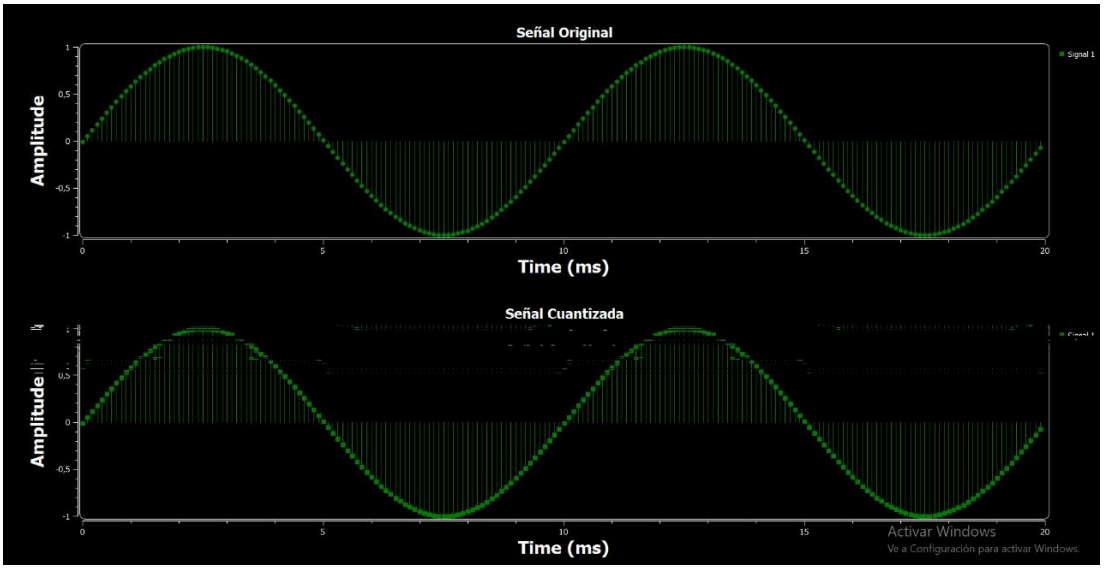
        Eso significa que no todos los números reales pueden representarse exactamente, sino solo aquellos dentro del conjunto finito de valores que puede codificar el formato float32s

    ii) ¿Hay alguna forma de evitar el error de cuantización cuando trabajamos señales digitales?

        No

b) Use el bloque Quantizer, para cuantizar señales. Cree, en Companion, una señal tipo seno, de 100 Hertz de frecuencia, salida Float, y con frecuencia de muestreo de 10 KHz, y amplitud 1. Compare gráficamente la señal original con la misma señal pero cuantizada a 12 bits. Nota: No olvide usar el bloque Throttle.

    i) ¿Observa diferencias entre ambas señales?

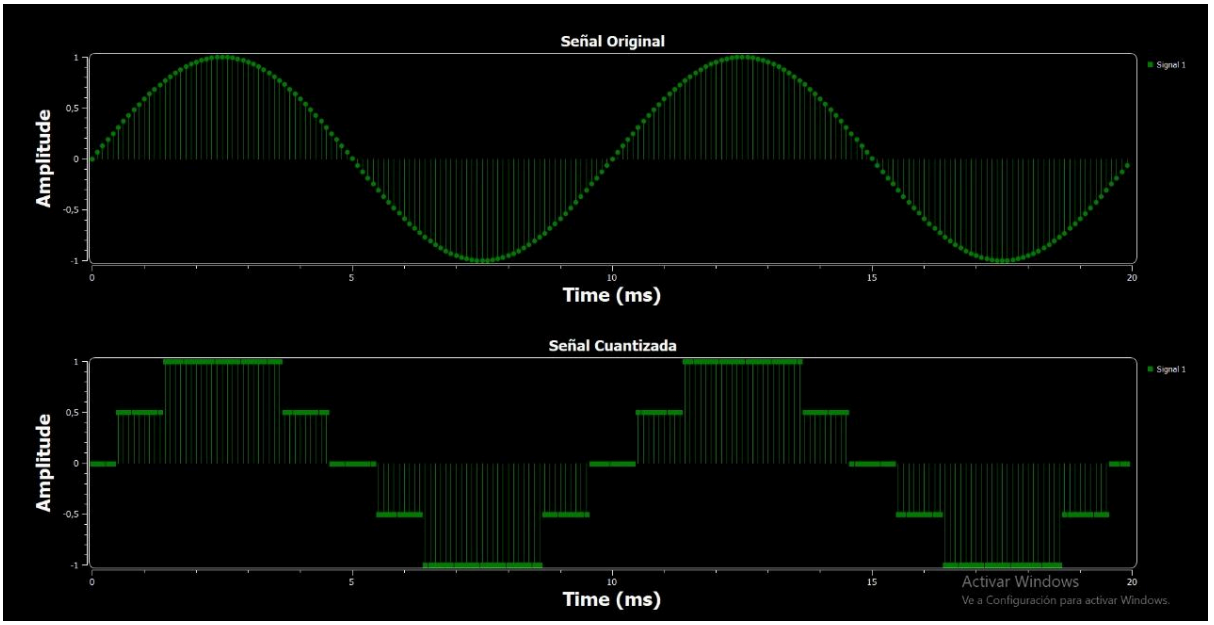


ii) ¿Cuántos niveles de cuantización son 12 bits?

$L = 2^b - 1 = 4095$

c) Ahora proponga una cuantización de 4 niveles. ¿Cuántos bits se necesitan para lograr 4 niveles? Compare gráficamente la señal sin cuantizar con la cuantizada, en Companion.

Se necesitan 2 bits:



d) Calcule de manera teórica la relación señal-ruido de cuantización (SQNR) para señales sinusoidales que usan 4 niveles de cuantización. También calcule el SQNR en dB.

i. ¿Cuánto aumenta el SQNR por cada bit que se agregue en el cuantizador?, ¿por qué?

Paso de cuantización:

$$\Delta = (x_{max} - x_{min})/(L - 1) = 2/3$$

Potencia de la señal:

$$P_s = (x_{max} - x_{min})^2/2 = 2$$

Potencia del ruido de cuantización:

$$P_q = \Delta^2/12 = 1/27$$

Relación señal-ruido de cuantización (lineal):

$$SNQR = P_s/P_q = 54$$

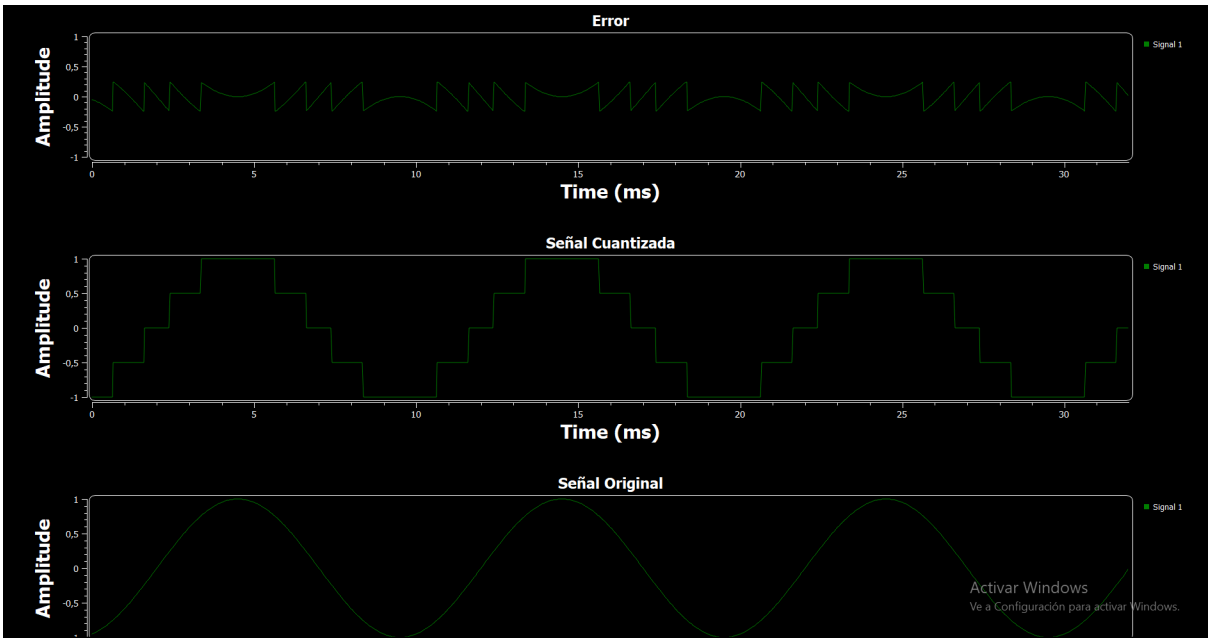
$$SNQR_{dB} = 10log_{10}(SNQR) = 17.32dB$$

Si trabajamos con las ecuaciones de las potencias teóricas llegamos a:

$$SNQR_{dB} = 6.02 * b + 1.76$$

De la cual podemos observar que el SNQR aumenta 6.02 [db] por cada bit agregado, esto es porque al aumentar el número de bits aumentan los niveles de cuantización haciendo que se minimice el error.

e) Vamos a desestimar el error de cuantización introducido por usar variables Float. Considerando una cuantización de 4 niveles, estime el error de cuantificación eq[n] para una señal sinusoidal usando bloques del Companion, y grafique eq[n] usando el bloque QT GUI Time Sink.



f) Utilizando el bloque Python Block, reemplace el código por el que se encuentra disponible en el anexo de este documento.
Al reemplazar el código, el bloque pasa a llamarse Potencia media. Este bloque toma como entrada una señal discreta, y computa la potencia media como el promedio (o valor medio) de las muestras de entrada, elevadas cada una al cuadrado.

En particular, si al bloque Potencia media le ingresamos la señal de entrada xa[n] (la salida del bloque Signal Source), a la salida obtendremos la Potencia media de la señal Px.

La potencia media de la señal $x_a(t)$ es

$$P_x = \frac{1}{T_p}\int_0^{T_p} (A\cos\Omega_0 t)^2\, dt = \frac{A^2}{2} \qquad (1.4.31)$$

Note que hemos reemplazado xa(t) por xa[n] en el cálculo de potencia media, por lo tanto la integral se convierte en una sumatoria.

Si al mismo bloque Potencia media le aplicamos a la entrada la señal eq[n], obtenemos a la salida la Potencia del error cuadrático medio Pq.

$e_q(t) = x_a(t) - x_q(t)$ se muestra en la Figura 1.4.9. En esta figura, τ indica el tiempo que $x_a(t)$ permanece dentro de los niveles de cuantificación. La potencia del error cuadrático medio P_q es

$$P_q = \frac{1}{2\tau}\int_{-\tau}^{\tau} e_q^2(t)\, dt = \frac{1}{\tau}\int_0^{\tau} e_q^2(t)\, dt \qquad (1.4.28)$$

Además, el bloque Potencia media devuelve a la salida un vector que repite el valor de potencia media computado.

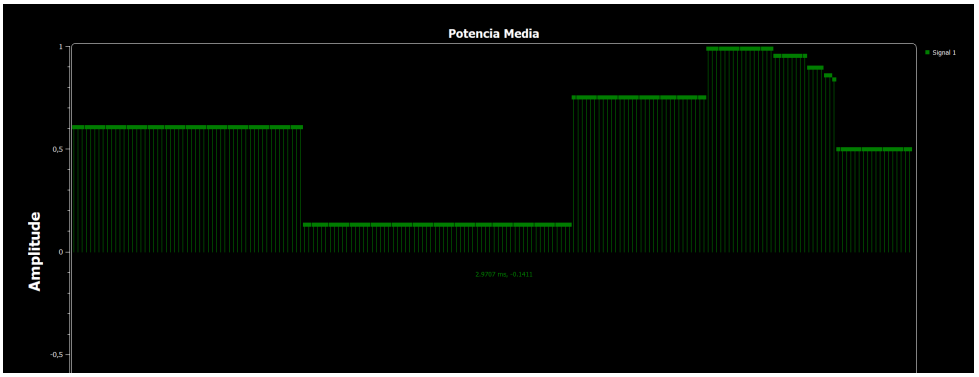
Por lo tanto, si aplicamos a la entrada xa[n], tendremos a la salida Px[n], donde:

$$Px\,[n] \;=\; Px, \quad \forall\, n$$

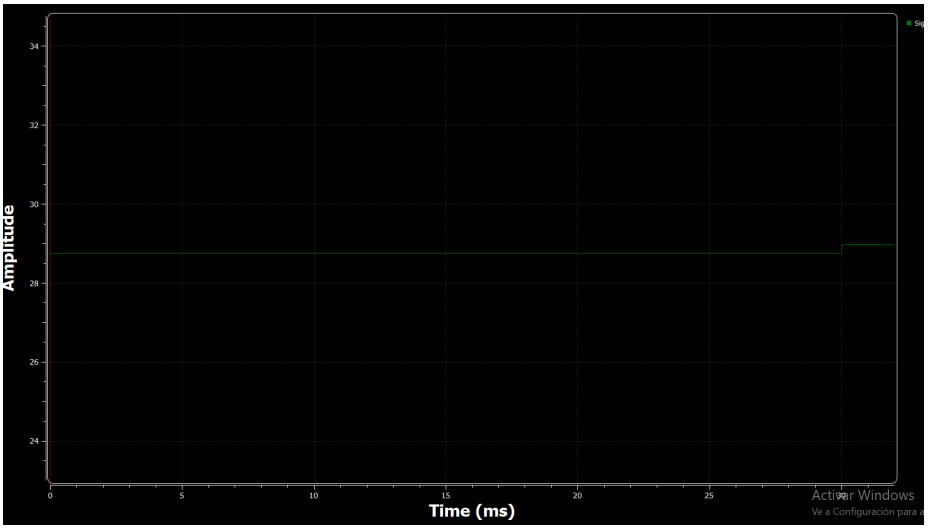
O también, si a la entrada del bloque aplicamos eq[n], a la salida obtenemos Pq[n], donde

$$Pq\,[n] \;=\; Pq, \quad \forall\, n$$

i) Grafique Px[n] y Pq[n] utilizando el bloque QT GUI Time Sink.



ii) Compute y grafique la relación SQNR en Companion, usando el bloque Potencia media.

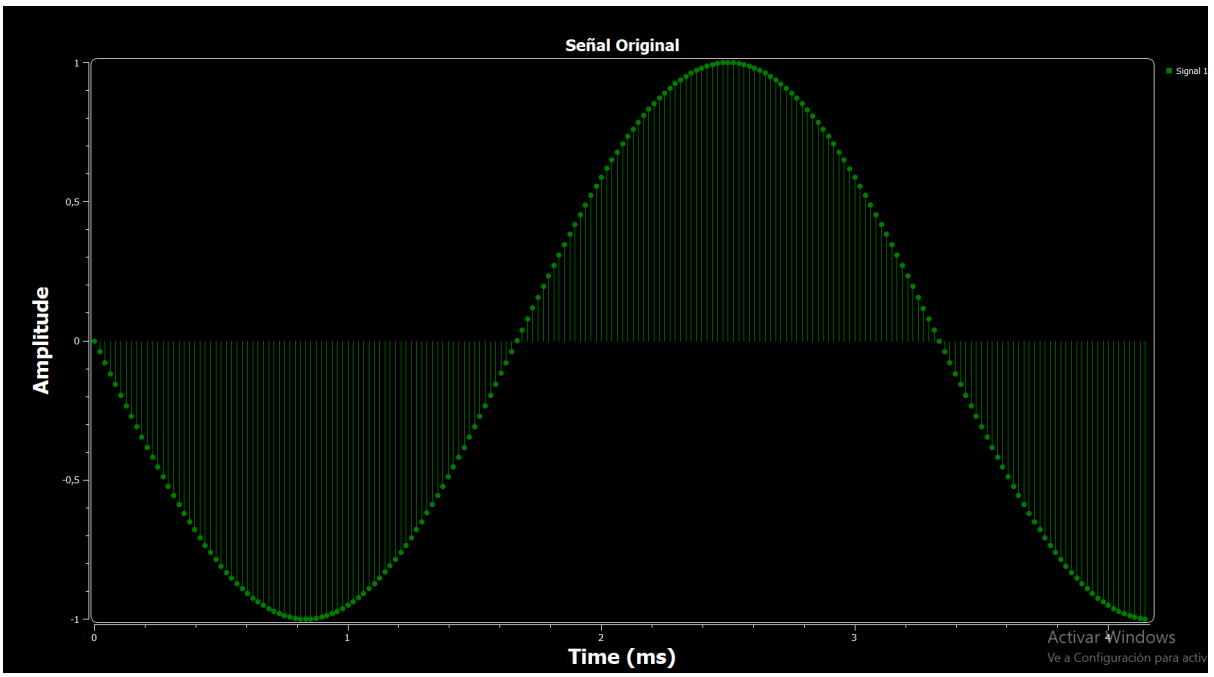


iii) Compare la relación SQNR que obtuvo teóricamente, con la relación SQNR que obtuvo en Companion. ¿A qué puede deberse las diferencias en los valores?

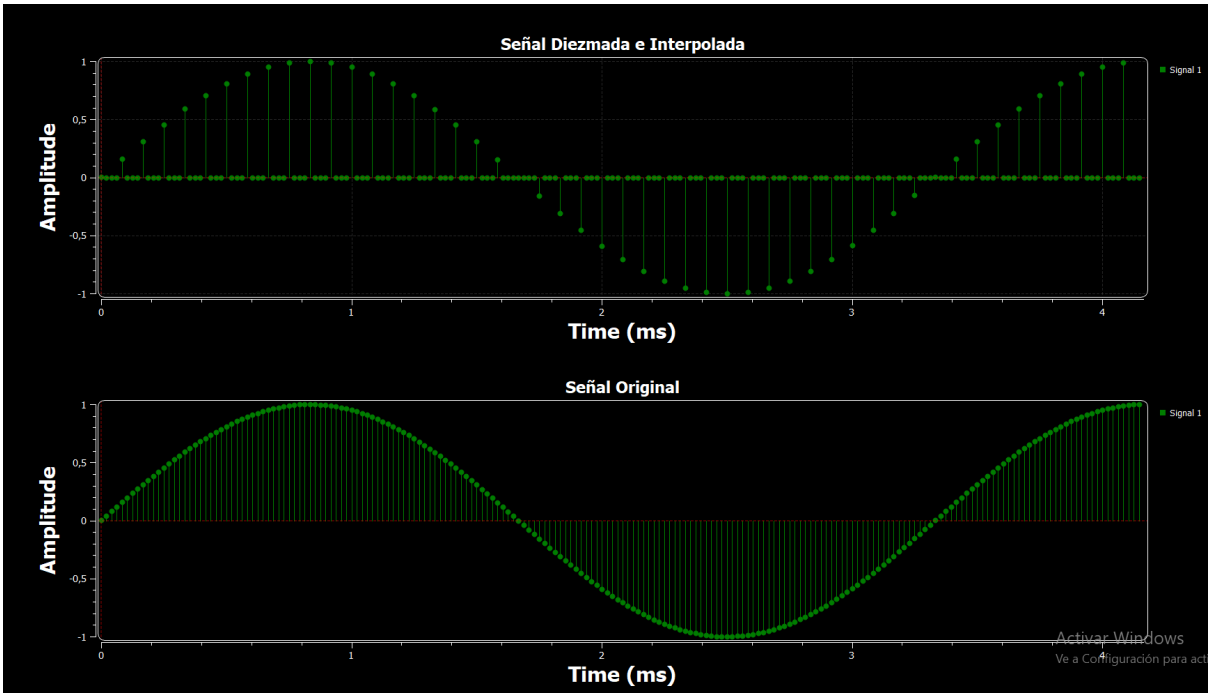
Ejercicio 11:

Cree un tono tipo seno de 300Hz y frecuencia de muestreo de 48KHz, usando el creador de tonos online disponible en el Anexo.

a. Reproduzca la señal en el Companion.

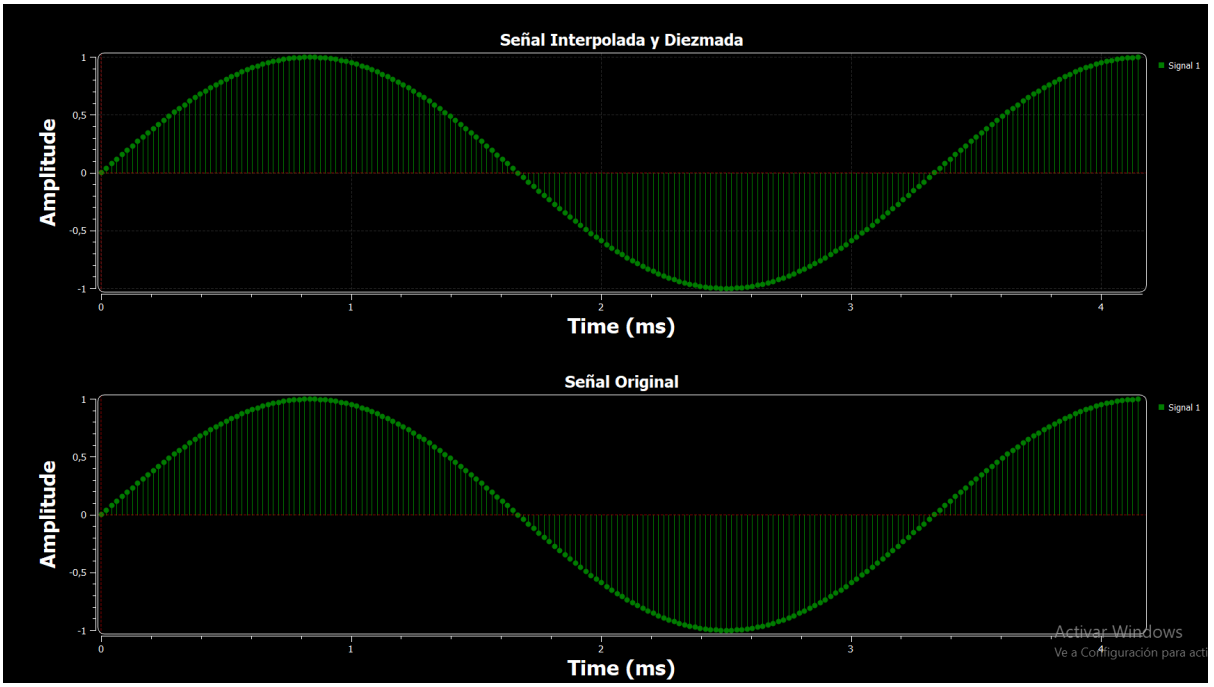


b. Utilice una combinación de bloques de GNURADIO Companion para diezmado e interpolación tal que pueda llevar la señal a una tasa de muestreo de 32Khz. Verifique gráficamente en Companion.



c. ¿Qué bloque debe ir primero (interpolación o diezmado) y por qué?

Primero interpolar la señal y luego diezmar el resultado de la interpolación, para poder recuperar la señal original



d. Reemplace la implementación por un bloque Rational Resampler. Siguiendo: https://wiki.gnuradio.org/index.php/Rational_Resampler, utilice un Fractional BW de 0.4. Vuelva a reproducir la señal luego del resampling, con el bloque Audio Sink.

Ejercicio 12:

Se necesitan obtener muestras de una RTL-SDR, a una frecuencia de muestreo de 1.2Mhz, para procesar cierta información. Considerando los dos canales I-Q:

a. Calcule en bitstream, o tasa de bits por segundo.

$f_s = 1.2MHz$

$Resolución\ por\ canal\colon\ 8\ bits\ para\ I\ y\ 8\ bits\ para\ Q \implies Nº\ total\ de\ bits\colon\ 16\ bits/muestra$

$Tasa\ de\ bits = f_s * Nº\ total\ de\ bits = 1.2MHz * 16\ bits = 19.2MB/s$

b. ¿Cuántos niveles de cuantización tienen las muestras que se toman?

La cantidad de niveles de cuantización se determina por la resolución en bits del RTL-SDR:

$L = 2^8 = 256\ niveles$

c. ¿Cuál es la frecuencia de muestreo máxima que soporta la SDR (consejo: investigue en la web las características de las RTL-SDR)?

Típica máxima teórica: $3.2MS/s$

Máxima estable recomendada: $2.4MS/s$ o $2.56MS/s$