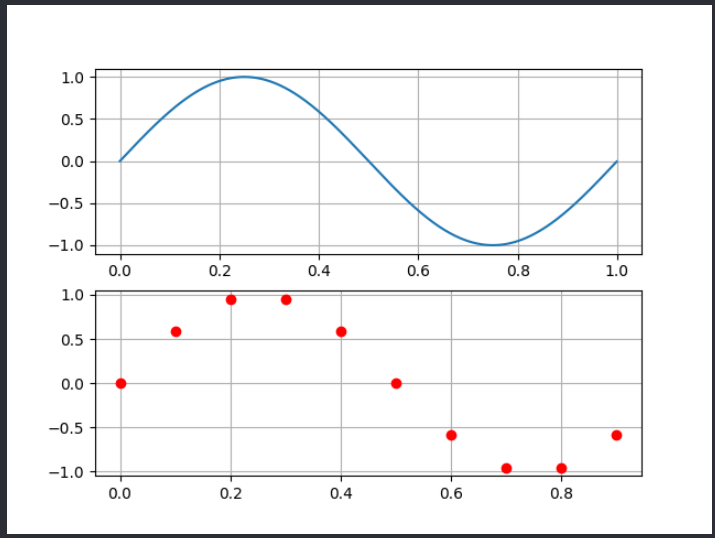
FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Procesamiento de señales, fundamentos

Maestría en sistemas embebidos MSE2020

Universidad de Buenos Aires



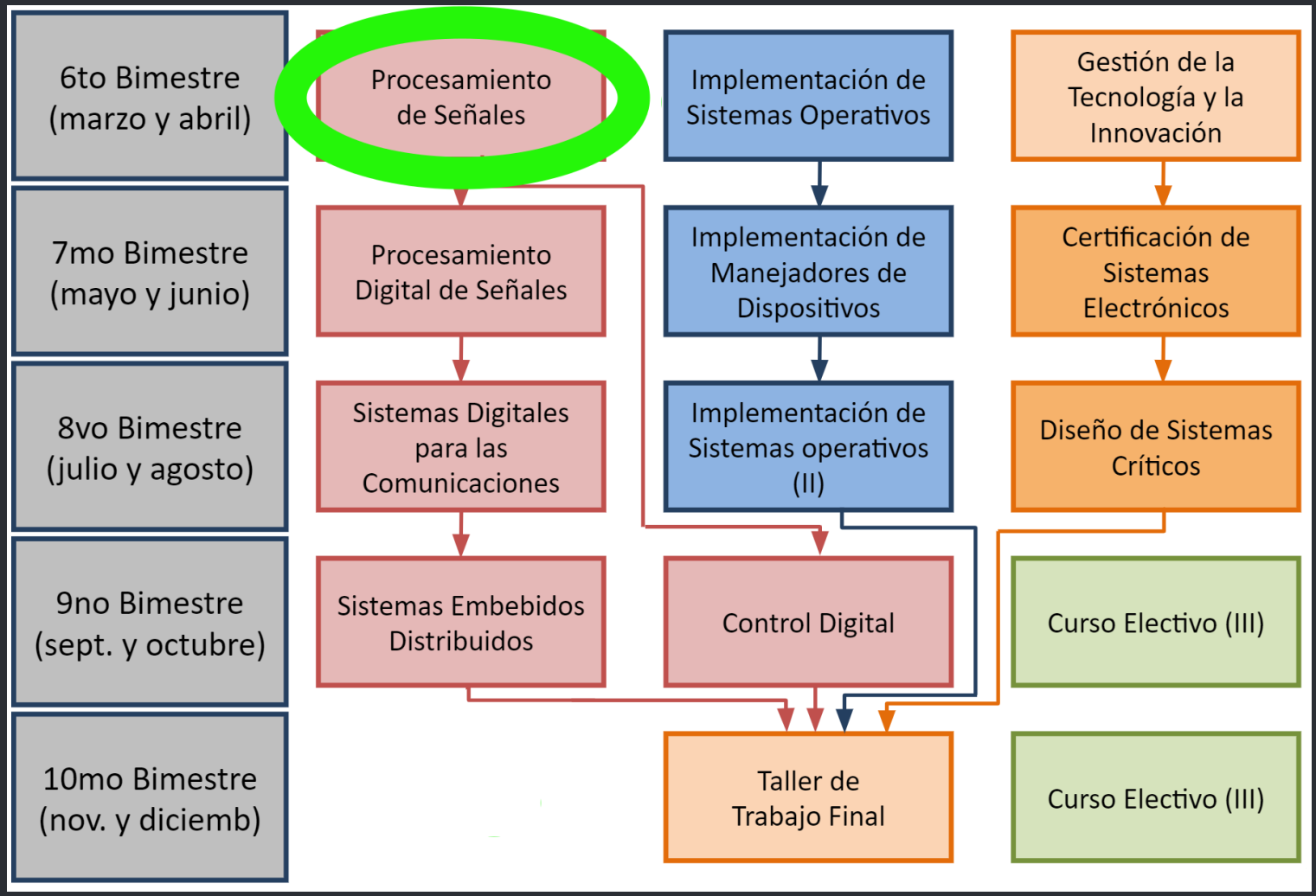
Clase 1 - Introducción

Ing. Pablo Slavkin

Resumen de seccion 1

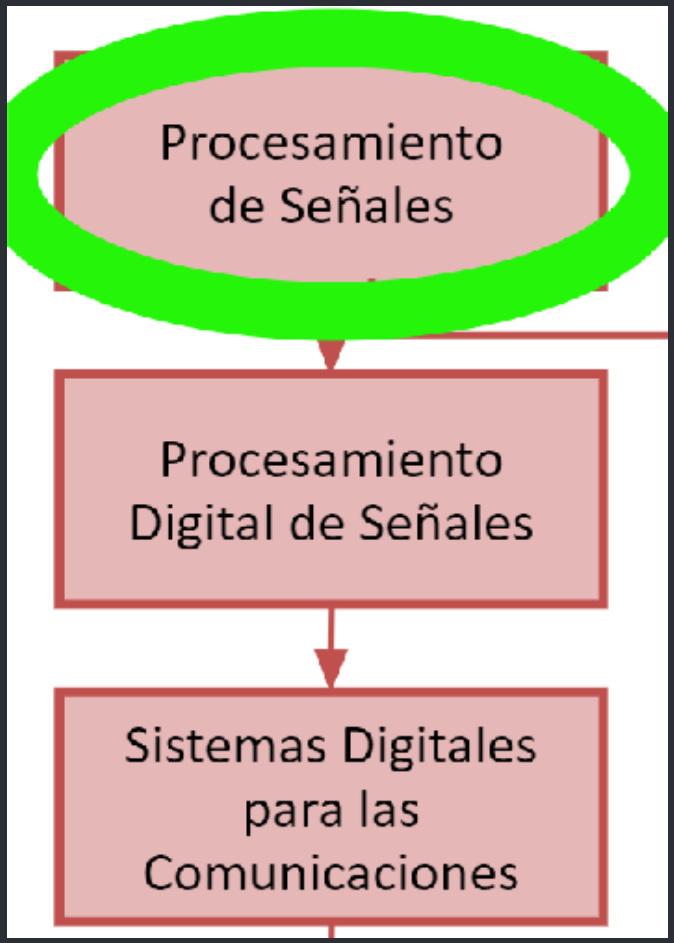
Plan de vuelo

Ud. esta aqui



Plan de vuelo

Ud. esta aqui



1.
   - Sampleo
   - Fourier, Z
   - Filtrado basico
   - CIAA
   - Python
2.
   - Estadistica
   - Canales
   - Filtrado complejo
   - FPGA?
   - Python
3.
   - Implementacion
   - Hi-Speed
   - Comunicaciones
   - FPGA
   - Python

Bibliografia

Libros, links y otro material

[1] Steven W. Smith.
The Scientist and Engineer's Guide to Digital Signal Processing
Second Edition, 1999.

[2] Allen B. Downey
Think DSP - Digital Signal Processing in Python

[3] Richard Lyons.
Understanding digital signal processing.
Third edition.

[4] Boaz Porat.
Digital Processing of Random Signals: Theory and Methods. *Digital Processing of Random Signals: Theory and Methods*.

[5] Allen B. Downey
Think Python, 2nd Edition, - How to Think Like a Computer Scientist

[6] Emmanuel C Ifeachor, Barrie W Jervis
Digital Signal Processing. *A practial approach*.

[7] NW. Taylor, Francis Group, LLC.
Introduction to Python Programming.

[8] Matt Harrison
Illustrated guide to python 3

porque digital?

digital vs analogico

- digital
  - reproducibilidad
  - tolerancia de componentes
  - partidas todas iguales
  - componentes no envejecen
  - facil de actualizar
  - soluciones de un solo chip
- analogico
  - alto ancho de banda
  - alta potencia
  - baja latencia

Señales y sistemas

Que son?

Señal

Una señal, en función de una o más variables, puede definirse como un cambio observable en una entidad cuantificable

Sistema

Un sistema es cualquier conjunto físico de componentes que actúan en una señal, tomando una o más señales de entrada, y produciendo una o más señales de salida.

Señales y sistemas

Tipos de señales

- De tiempo continuo
- Pares
- Periódicas
- De energía
- Reales

- De tiempo discreto
- No deterministas
- Impares
- Aperiódicas
- De potencia
- Imaginarias

Señales y sistemas

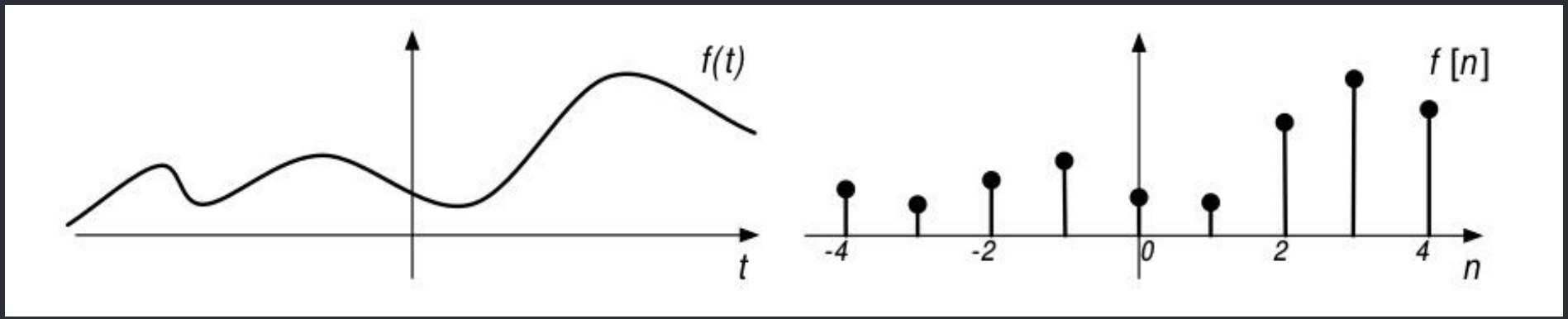
Tipos de señales

- De tiempo continuo

Tiene valores para todos los puntos en el tiempo en algún intervalo (posiblemente infinito)

- De tiempo discreto

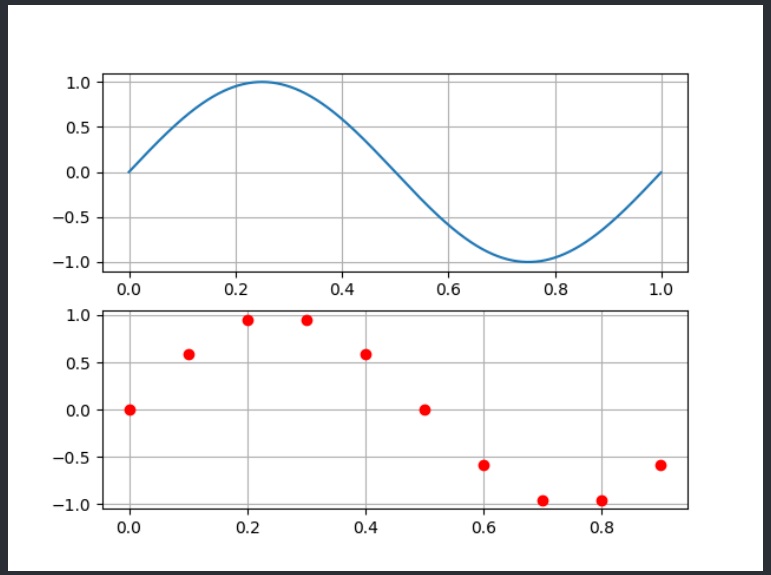
Tiene valores solo para puntos discretos en el tiempo



Generacion de señales en Python

Continuo? vs discreto

```
import matplotlib.pyplot as plt
import numpy as np

fig = plt.figure(1)

TC=0.001
tc  = np.arange(0.0, 1.0, TC)
ax1 = fig.add_subplot(211)
ax1.plot(tc, np.sin(2*np.pi*tc),'b-')
ax1.grid(True)

TD=0.1
td  = np.arange(0.0, 1.0, TD)
ax2 = fig.add_subplot(212)
ax2.plot(td, np.sin(2*np.pi*td),'ro')
ax2.grid(True)

plt.show()
```



Podrian pensarse como muestras de una señal de tiempo continuo

$x[n] = x(nT)$ donde n es un número entero y **T** es el período de muestreo.

Señales periodicas

Continua periodica

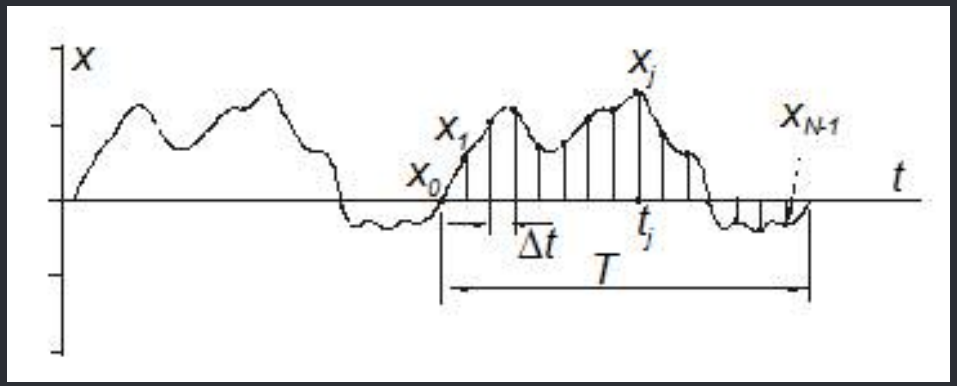
si existe un $T_0 > 0$, tal que $x(t + T_0) = x(t)$, para todo t

T_0 es el período de $x(t)$ medido en tiempo, y $f_0 = 1/T_0$ es la frecuencia fundamental de $x(t)$

Continua discreta

si existe un entero $N_0 > 0$ tal que $x[n + N_0] = x[n]$ para todo n

N_0 es el período fundamental de $x[n]$ medido en espacio entre muestras y $F_0 = \Delta t/N_0$ es la frecuencia fundamental de $x[n]$



Sistemas

Sistema

Un sistema es cualquier conjunto físico de componentes que actúan en una señal, tomando una o más señales de entrada, y produciendo una o más señales de salida.

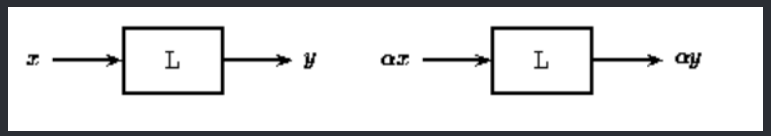
En ingeniería, a menudo la entrada y la salida son señales eléctricas.

Linealidad

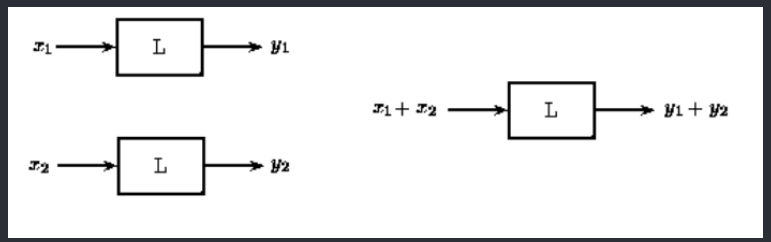
Un sistema es lineal cuando su salida depende linealmente de la entrada.

Satisface el principio de superposicion.

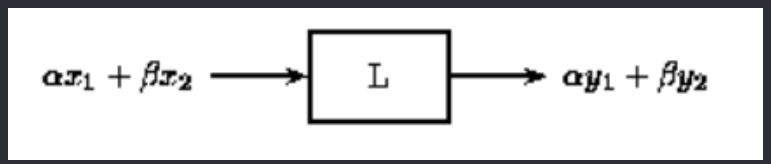
escalado



adicion



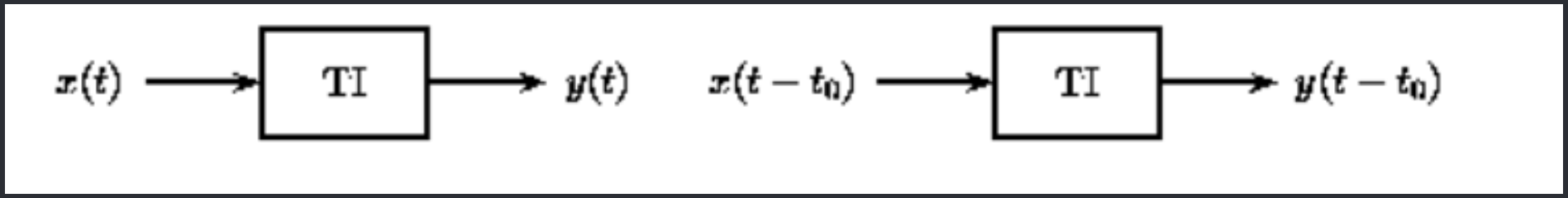
superposicion



$y(t) = e^{x(t)}$ $y(t) = \frac{1}{2}x(t)$

Invariantes en el tiempo

Un sistema es invariante en el tiempo cuando la salida para una determinada entrada es la misma sin importar el tiempo en el cual se aplica la entrada

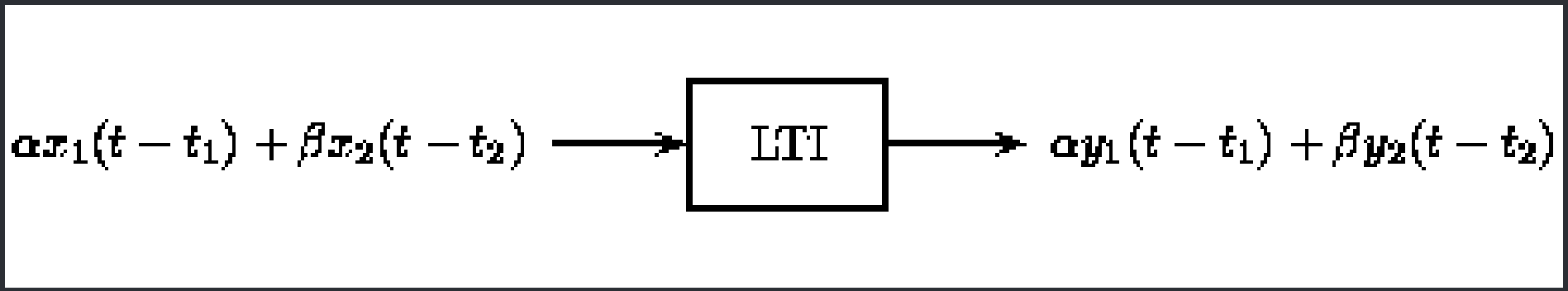


$y(t) = x(t) * \cos(t)$ $y(t) = \cos(t)$

Sistemas

Lineales invariantes en el tiempo

Un sistema es LTI cuando satisface las 2 condiciones anteriores, de linealidad y de invariancia en el tiempo.



*** LTI ***

En este curso, solo estudiaremos sistemas lineales invariantes en el tiempo.

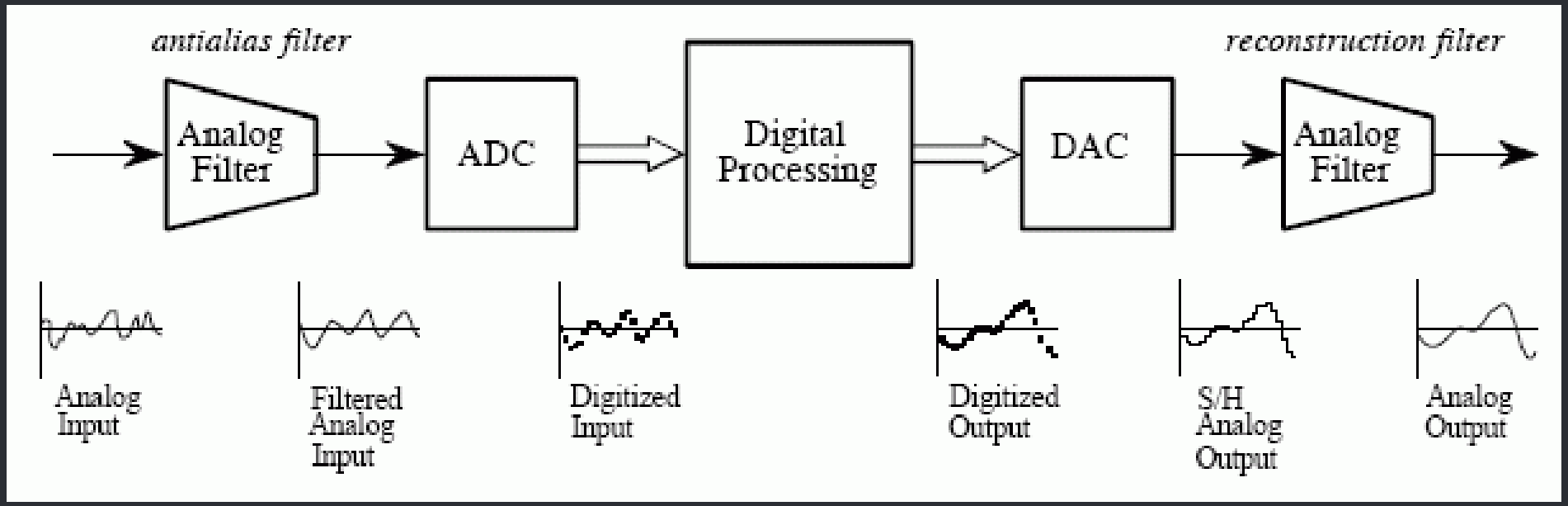
Senos y cosenos en LTI

En todo sistema LTI para una entrada senoidal la salida es siempre senoidal.

Resumen de seccion 2

ADC

Bloque generico de procesamiento



Porque el filtro antialising?

Aliasing

Simulando en Python

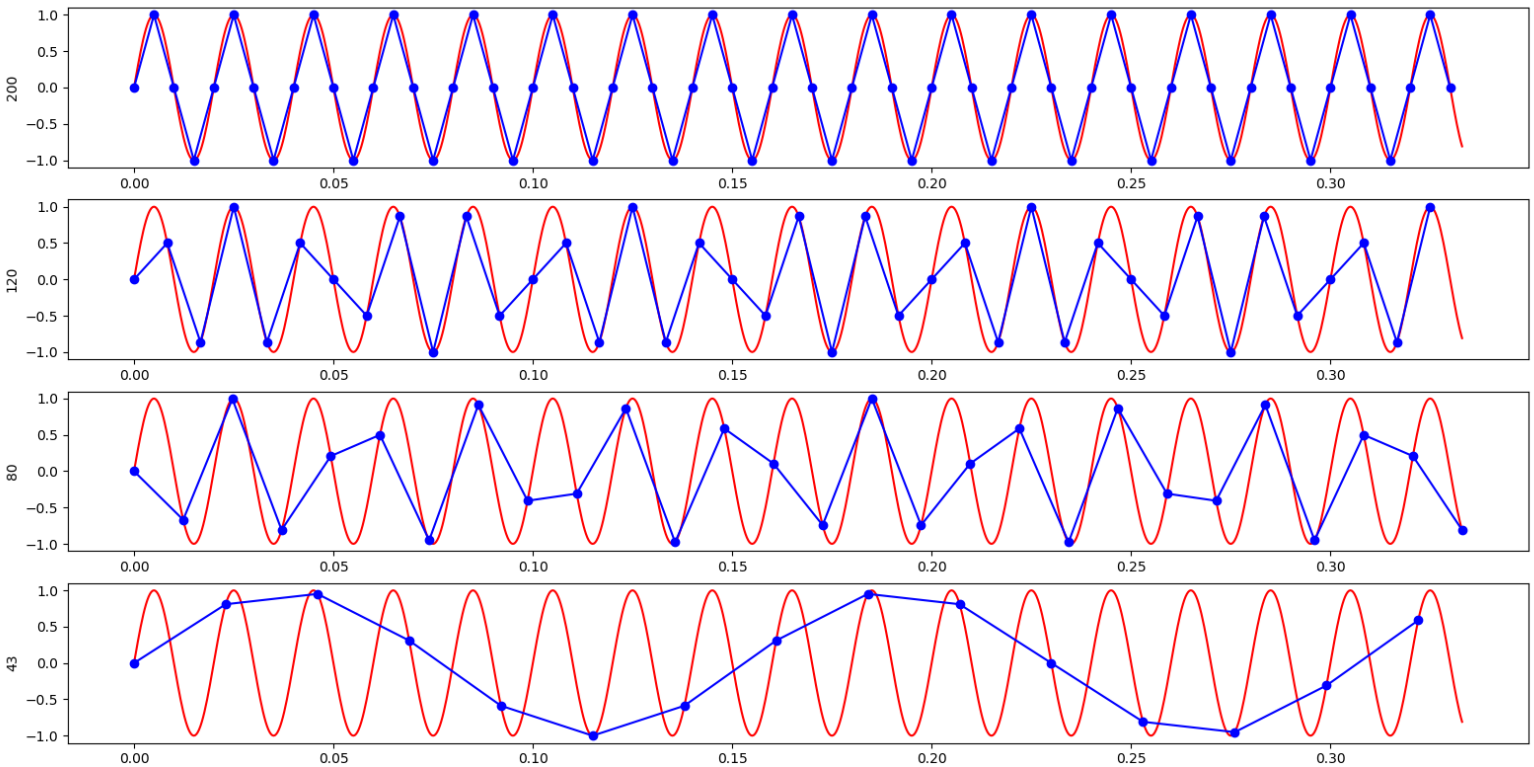
Diferentes frecuencias de sampleo para capturar una señal de 50hz

```
import numpy as np
import matplotlib.pyplot as plt

signalFrec = 50
NC         = 1000
fsC        = 3000
tC         = np.arange(0,NC/fsC,1/fsC)
signalC    = np.sin(2*np.pi*signalFrec*tC)
fsD        = [200,120,80,43]

fig      = plt.figure()
#signalC    = np.sin(2*np.pi*signalFrec*tC)+0.5*np.sin(2*np.pi*210*
    tC)

for i in range(len(fsD)):
    contiAxe = fig.add_subplot(4,1,i+1)
    plt.plot(tC,signalC,'r-',tC[::fsC//fsD[i]],signalC[::fsC//fsD[i
        ]],'b-o')
    contiAxe.set_ylabel(fsD[i])

plt.show()
```

Aliasing

Simulando en Python

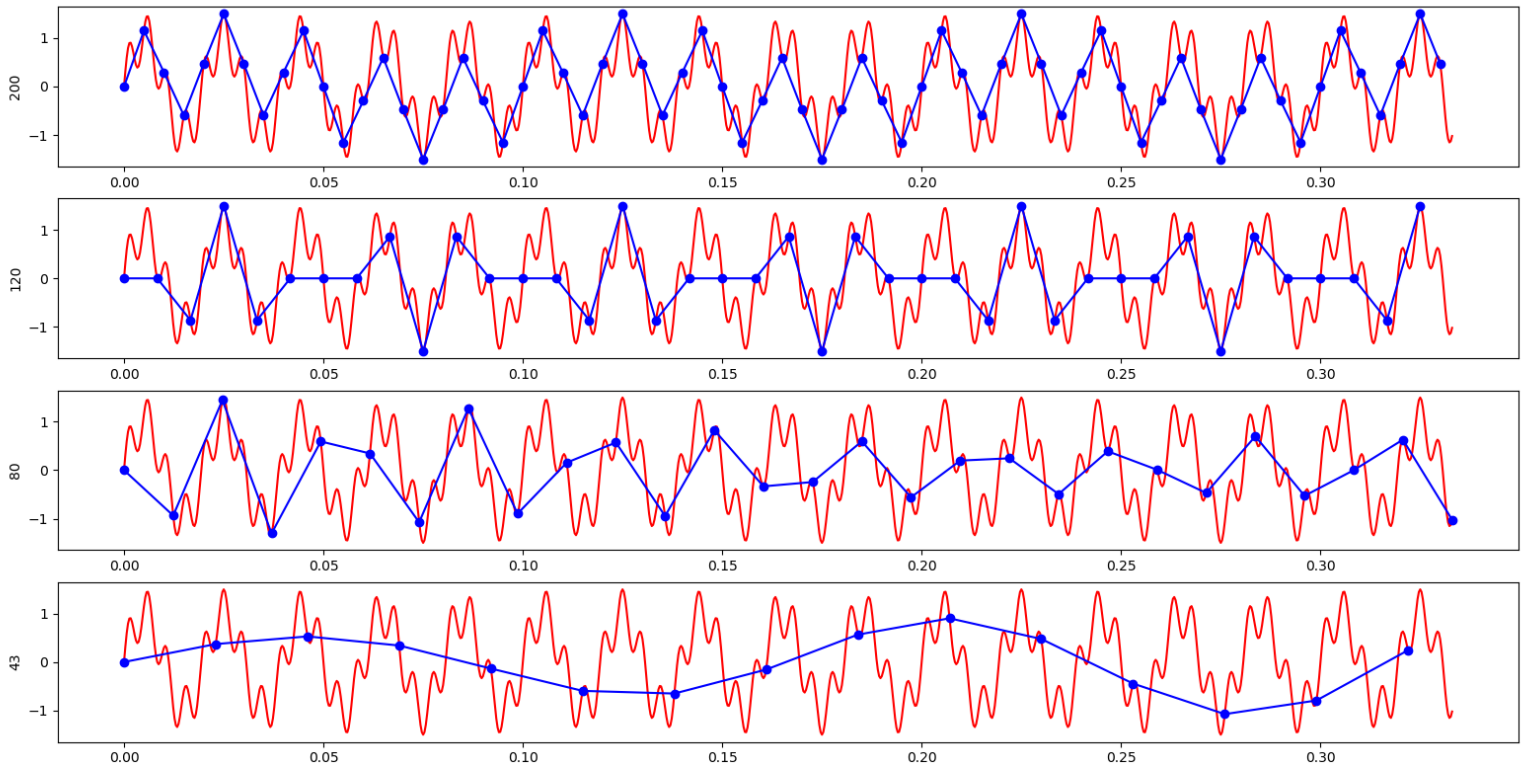
Diferentes frecuencias de sampleo para capturar una señal de 50hz



Aliasing

Simulando en Python

Que pasa si se suma ruido de alta frecuencia?



Aliasing

Disco Giratorio

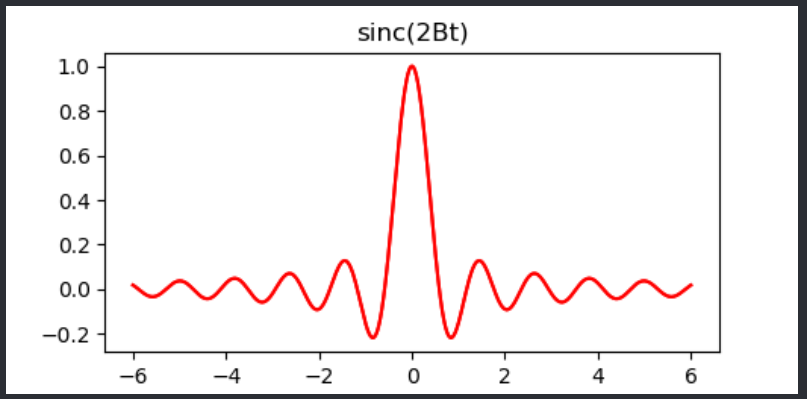
Teorema de sampleo

Teorema de Shannon

Teorema

La reconstrucción exacta de una señal periódica continua en banda base a partir de sus muestras, es matemáticamente posible si la señal está limitada en banda y la tasa de muestreo es superior al doble de su ancho de banda

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \frac{\sin \pi(2Bt - n)}{\pi(2Bt - n)}.$$



Teorema de sampleo

Teorema de Shannon

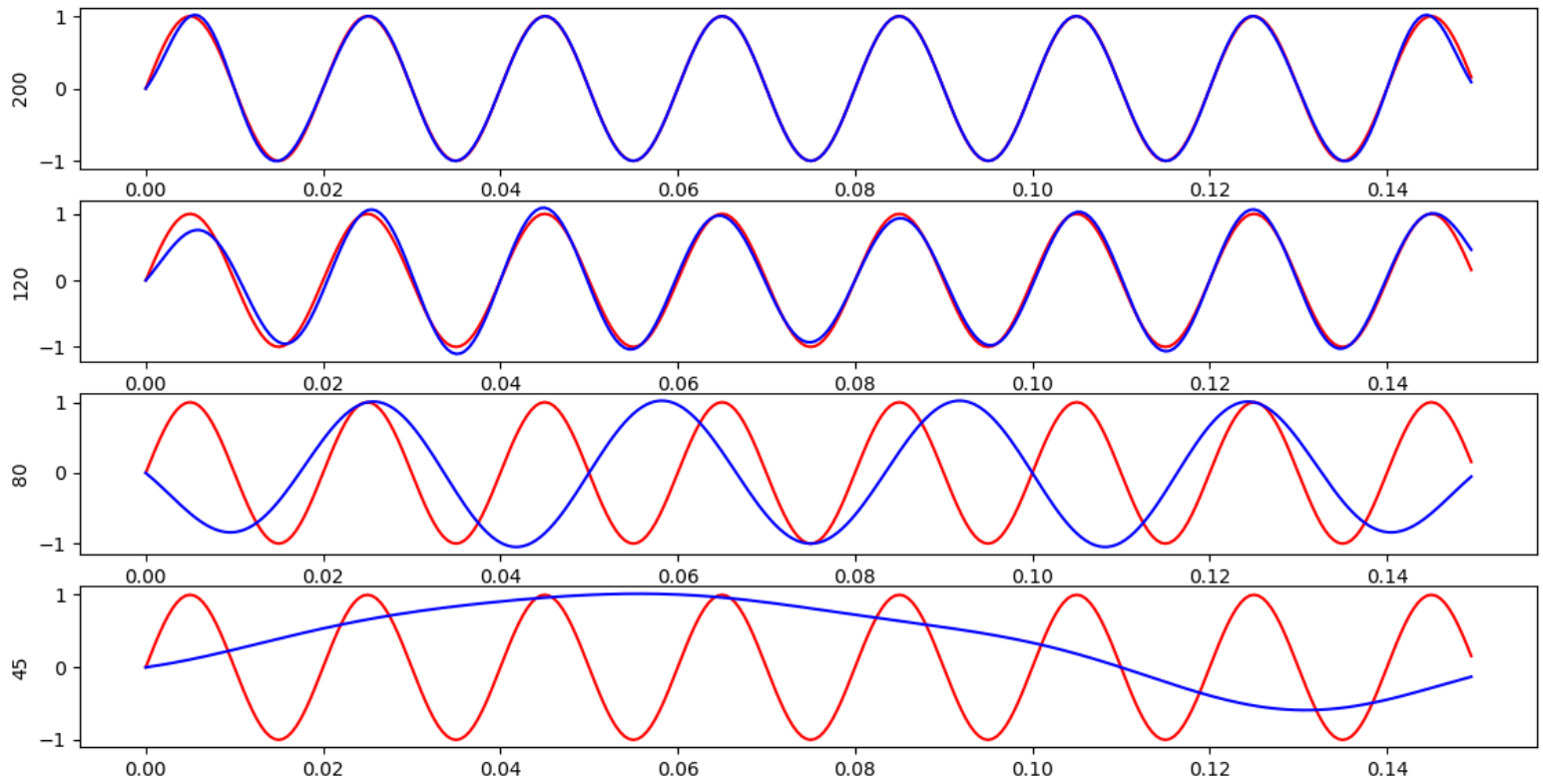
Sampleo e interpolado

```
import numpy as np
import matplotlib.pyplot as plt

signalFrec = 50
NC         = 300
fsC        = 1000
tC         = np.arange(0,NC/fsC,1/fsC)
signalC    = np.sin(2*np.pi*signalFrec*tC)
#signalC    = np.sin(2*np.pi*signalFrec*tC)+0.5*np.sin(2*np.pi*210*tC)
fsD        = np.array([200,120,80,45])

fig        = plt.figure()

def interpolate(x, s, u):
    y=[]
    B = 1/(2*(s[1] - s[0]))
    for t in u:
        prom=0
        for n in range(len(x)):
           prom+=x[n]*np.sinc(2*B*t-n)
        y.append(prom)
    return y

for i in range(len(fsD)):
    contiAxe = fig.add_subplot(4,1,i+1)
    Xt=interpolate(signalC[::fsC//fsD[i]],tC[::fsC//fsD[i]],tC)
    plt.plot(tC,signalC,'r-',tC,Xt,'b-')
    contiAxe.set_ylabel(fsD[i])

plt.show()
```

Teorema de sampleo

Teorema de Shannon

Sampleo e interpolado

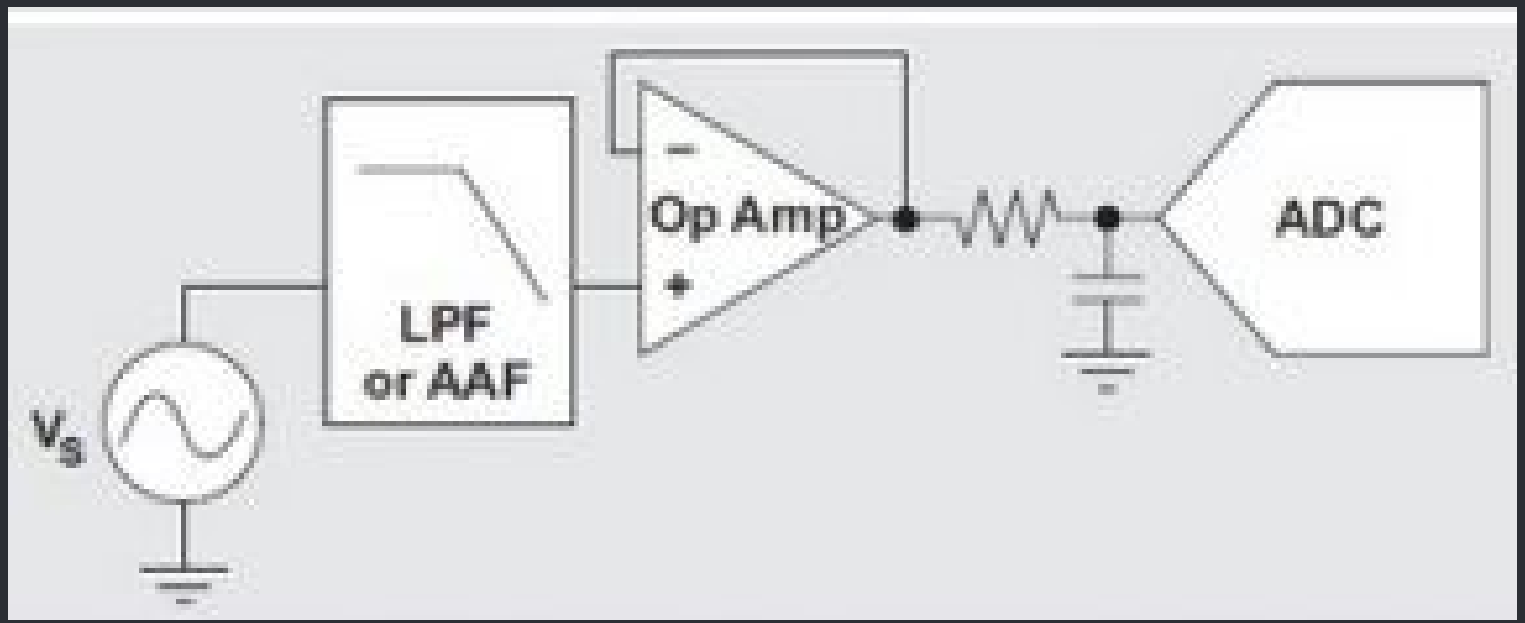


Sampleo

Filtro antialias

FAA

Filtro analogico pasabajos que elimina o al menos mitiga el efecto de aliasing

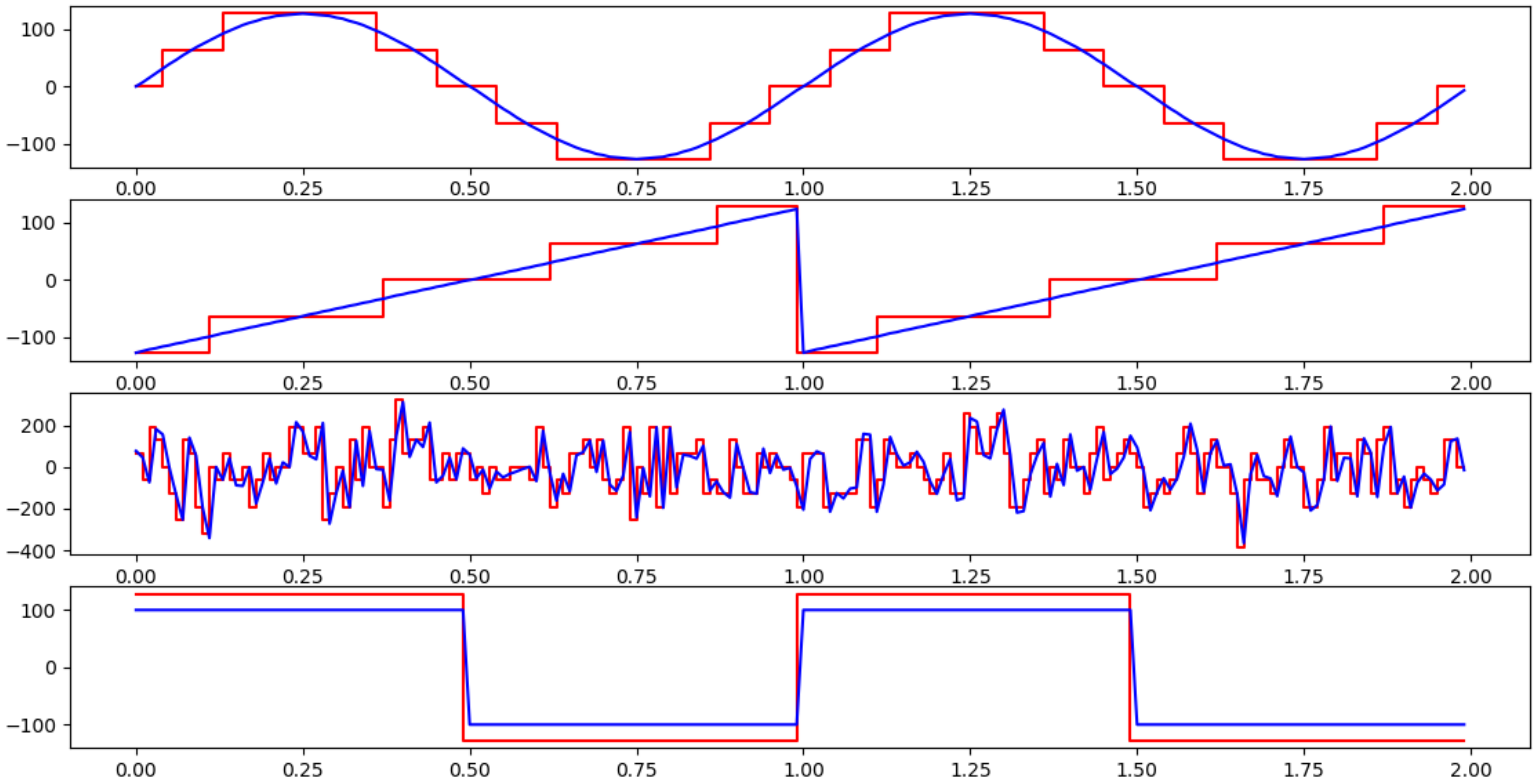


Resumen de seccion 3

Ruido de cuantizacion

Ejemplo de cuantizacion

Diferentes formas de onda cuantizadas



Ruido de cuantizacion

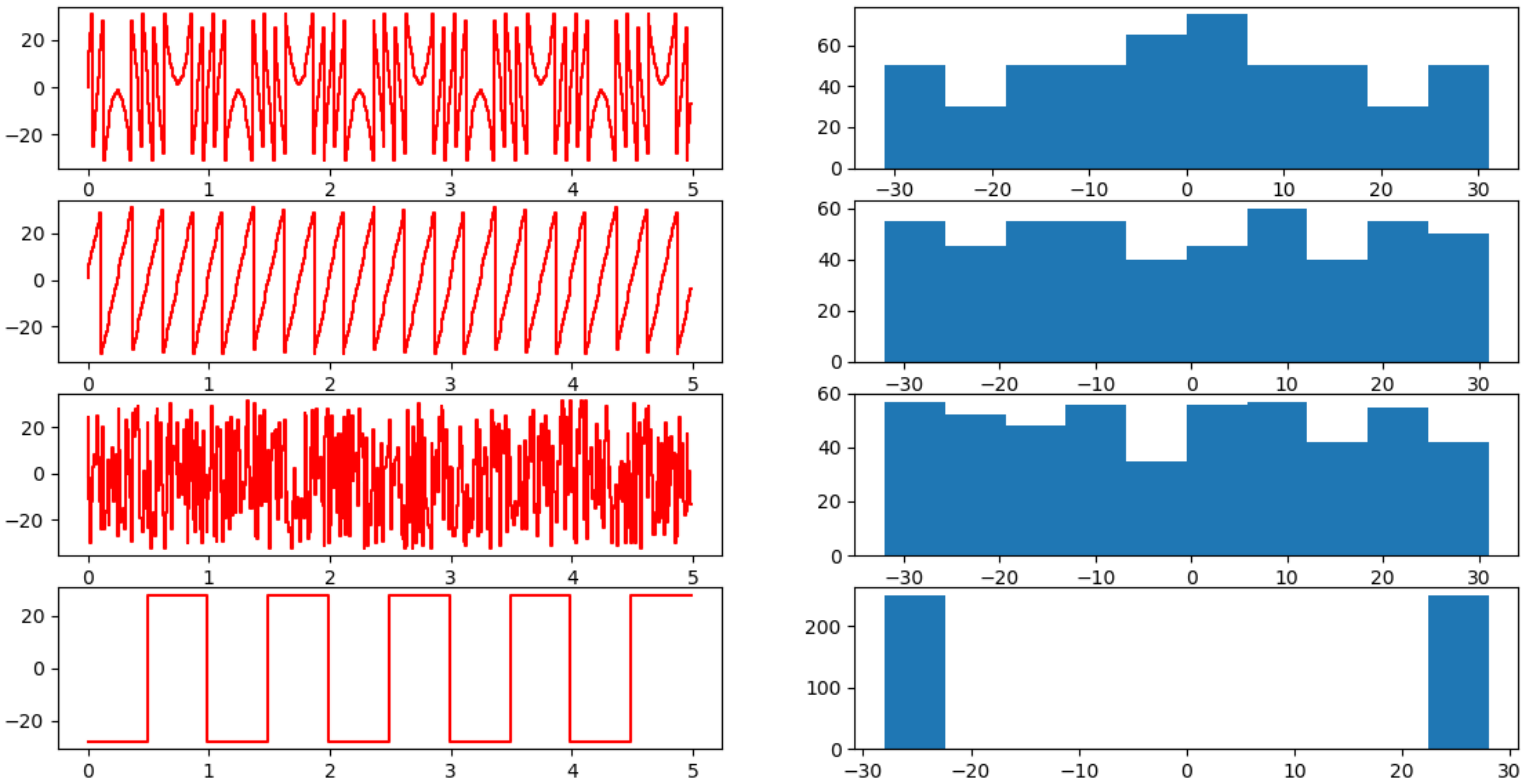
Cuantizacion en python

```
import numpy as np
import scipy.signal as sc
import matplotlib.pyplot as plt

signalFrec = 1
NC         = 200
fsC        = 100
Bits       = 2
tC         = np.arange(0,NC/fsC,1/fsC)
signalC    = np.array([(2**7-1)*np.sin(2*np.pi*signalFrec*tC),
             (2**7-1)*sc.sawtooth(2*np.pi*tC,1),
             (2**7-1)*np.random.normal(0,1,len(tC)),
             100*sc.square(2*np.pi*tC,0.5)],dtype='int16')

signalQ  = np.copy(signalC)
signalQ += (2**(8-Bits))//2
signalQ  &= 0xFFFF<<(8-Bits)

fig      = plt.figure()
for i in range(len(signalC)):
    contiAxe = fig.add_subplot(4,1,i+1)
    plt.step(tC,signalQ[i],'r-')
    plt.plot(tC,signalC[i],'b-')

plt.show()
```

Ruido de cuantizacion

Histogramas

Histogramas de ruido para cada señal



Ruido de cuantizacion

Histogramas

Histogramas en Python

```
import numpy as np
import scipy.signal as sc
import matplotlib.pyplot as plt

signalFrec = 1
NC         = 500
fsC        = 100
Bits       = 2
tC         = np.arange(0,NC/fsC,1/fsC)
signalC    = np.array([(2**7-1)*np.sin(2*np.pi*signalFrec*tC),
             (2**7-1)*sc.sawtooth(2*np.pi*tC,1),
             (2**7-1)*np.random.normal(0,1,len(tC)),
             100*sc.square(2*np.pi*tC,0.5)],dtype='int16')

signalQ  = np.copy(signalC)
signalQ += (2**(8-Bits))//2
signalQ  &= 0xFFFF<<(8-Bits)

fig      = plt.figure()
for i in range(len(signalC)):
    contiAxe = fig.add_subplot(4,2,2*i+1)
    plt.step(tC,signalC[i]-signalQ[i],'r-')
    contiAxe = fig.add_subplot(4,2,2*i+2)
    plt.hist(signalC[i]-signalQ[i])

plt.show()
```

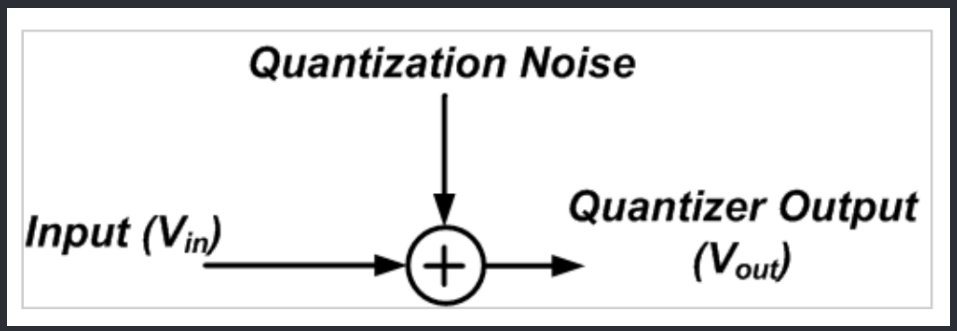
Ruido de cuantizacion

Modelo estadistico

En el caso de que se cumplan las siguientes premisas:

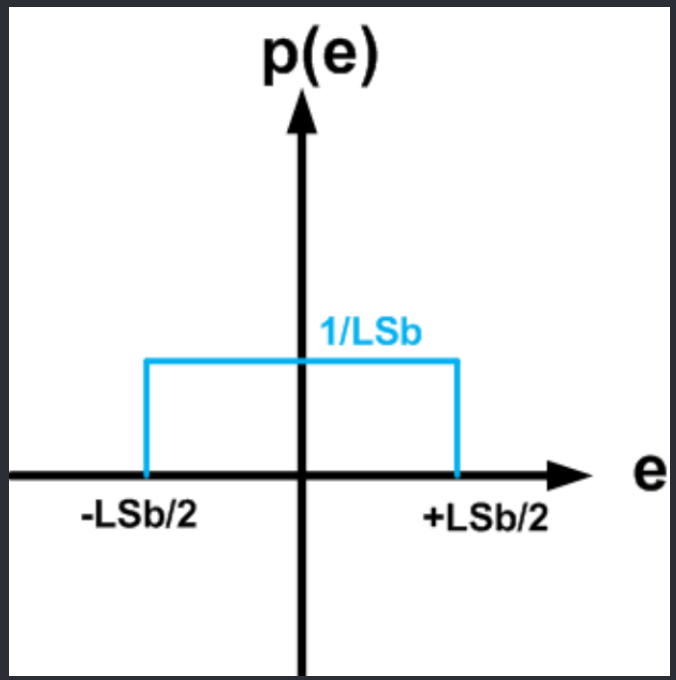
- La entrada se distancia de los diferentes niveles de cuantizacion con igual probabilidad
- El error de cuantizacion NO esta correlacionado con la entrada
- El cuantizador cuanta con un numero relativamente largo de niveles
- Los niveles de cuantizacion son uniformes

Se puede considerar la cuantizacion como un ruido aditivo a la señal segun el siguiente esquema:



Ruido de cuantizacion

Funcion densidad de probabilidad



$$\int_{-\frac{lsb}{2}}^{\frac{lsb}{2}} p(e)de = 1$$

Ruido de cuantizacion

Potencia de ruido de cuantizacion

$$P_q = \int_{-\frac{lsb}{2}}^{\frac{lsb}{2}} e^2 p(e) de$$

$$P_q = \int_{-\frac{lsb}{2}}^{\frac{lsb}{2}} e^2 \frac{1}{lsb} de$$

$$P_q = \frac{1}{lsb}\left(\frac{e^3}{3}\Big|_{-\frac{lsb}{2}}^{\frac{lsb}{2}} \right)$$

$$P_q = \frac{1}{lsb}\left(\frac{(\frac{lsb}{2})^3}{3} - \frac{(\frac{-lsb}{2})^3}{3} \right)$$

$$P_q = \frac{1}{lsb}\left(\frac{lsb^3}{24} + \frac{lsb^3}{24} \right)$$

Potencia de ruido de cuantizacion

$$P_q = \frac{lsb^2}{12} \tag{1}$$

Ruido de cuantizacion

Relacion señal a ruido

$$input = \frac{Amp}{2}\sin(t)$$

$$P_{input} = \frac{1}{T}\int_0^T \left(\frac{Amp}{2}\sin(t)\right)^2 dt$$

$$P_{input} = \frac{1}{T}\left(\frac{Amp}{2}\right)^2 * \left(\frac{t}{2} - \frac{\sin(2t)}{4}\right)\Big|_0^T$$

$$P_{input} = \frac{Amp^2}{4T}\frac{T}{2}$$

$$P_{input} = \frac{Amp^2}{8}$$

$$lsb = \frac{Amp}{2^N}$$

$$P_{ruido} = \frac{lsb^2}{12}$$

$$P_{ruido} = \frac{\left(\frac{Amp}{2^N}\right)^2}{12}$$

$$P_{ruido} = \frac{Amp^2}{12 * 2^{2N}}$$

Ruido de cuantizacion

Relacion señal a ruido

$$SNR = 10 \log_{10} \left(\frac{P_{input}}{P_{ruido}} \right)$$

$$SNR = 10 \log_{10} \left(\frac{\frac{Amp^2}{8}}{\frac{Amp^2}{12 * 2^{2N}}} \right)$$

$$SNR = 10 \log_{10} \left(\frac{3 * 2^{2N}}{2} \right)$$

$$SNR = 10 \log_{10} \left(\frac{3}{2} \right) - 10 \log_{10} \left(2^{2N} \right)$$

SNR

$$SNR = 1{,}76 + 6{,}02 * N \quad (2)$$

$$SNR_{N=10} \approx 60dB$$

$$SNR_{N=11} \approx 66dB$$

Ruido de cuantizacion

Densidad espectral de potencia de ruido

Si condideramos la potencia de ruido uniformemente distribuido en todo el espectro desde $-Fs$ hasta $+Fs$, nos queda que:

Densidad espectral de potencia de ruido

$$S_{espectral}(f) = \frac{P_q}{Fs}$$

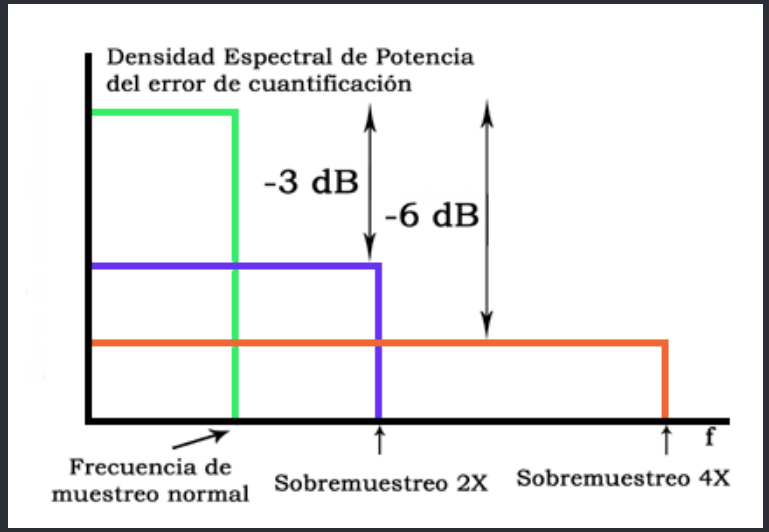
Entonces como puedo mejorar la SNR de un sistema?

Sobremuestreo

Densidad espectral de potencia de ruido

Oversampling x4

$$S_{espectral}(f) = \frac{P_q}{4 * Fs}$$



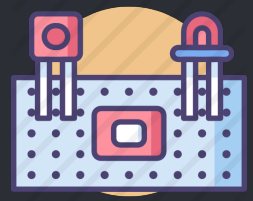
Que hago si tengo un AD de 10bits y deseo una SNR de 72dB? $SNR_{10} \approx 66dB$

Pero si sobremuestreo a 4x obtengo 6dB extras

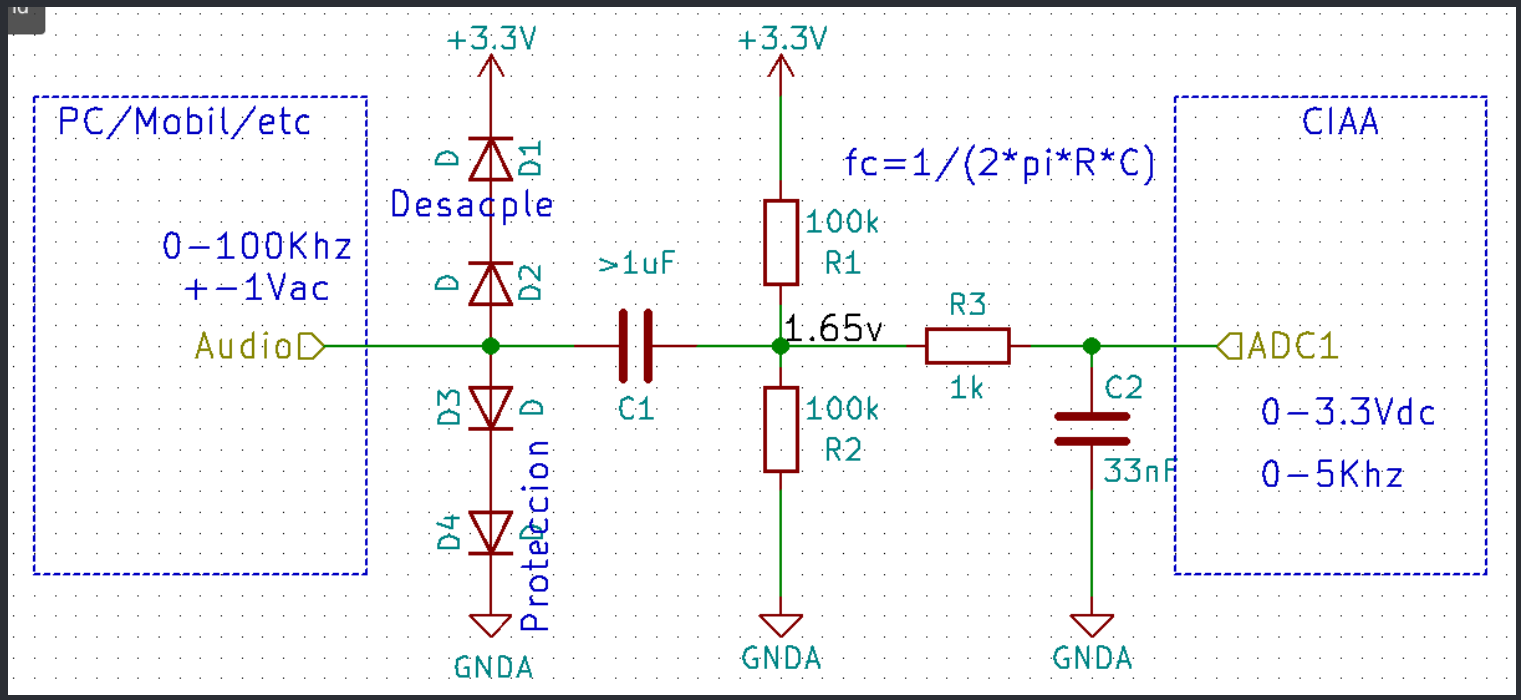
Resumen de seccion 4

Sampleo

Propuesta para acondicionamiento de señal



Acondicionar la señal de salida del dispositivo de sonido (en PC ronda ±1V) al rango del ADC del hardware. En el caso de la CIAA sera de 0-3.3V.
Se propone el siguiente circuito, que minimiza los componentes sacrificando calidad y agrega en filtro anti alias de 1er orden.



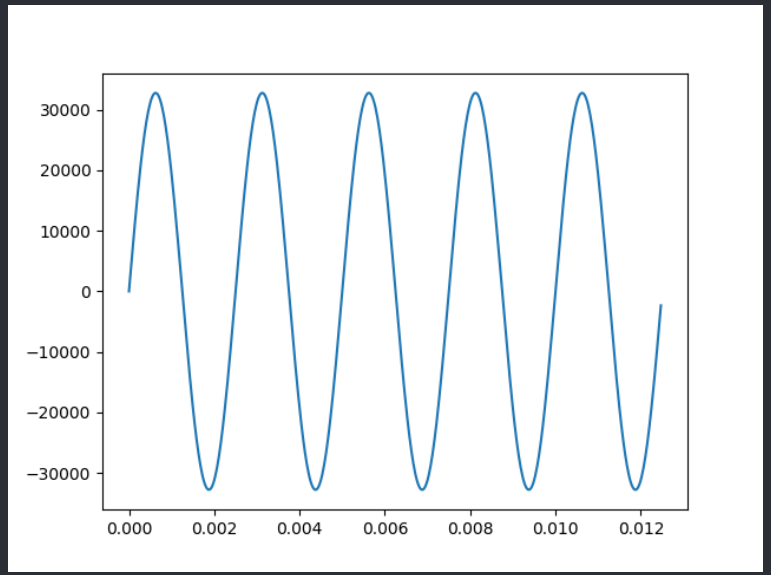
Resumen de seccion 5

Generacion de audio con Python

simpleaudio lib

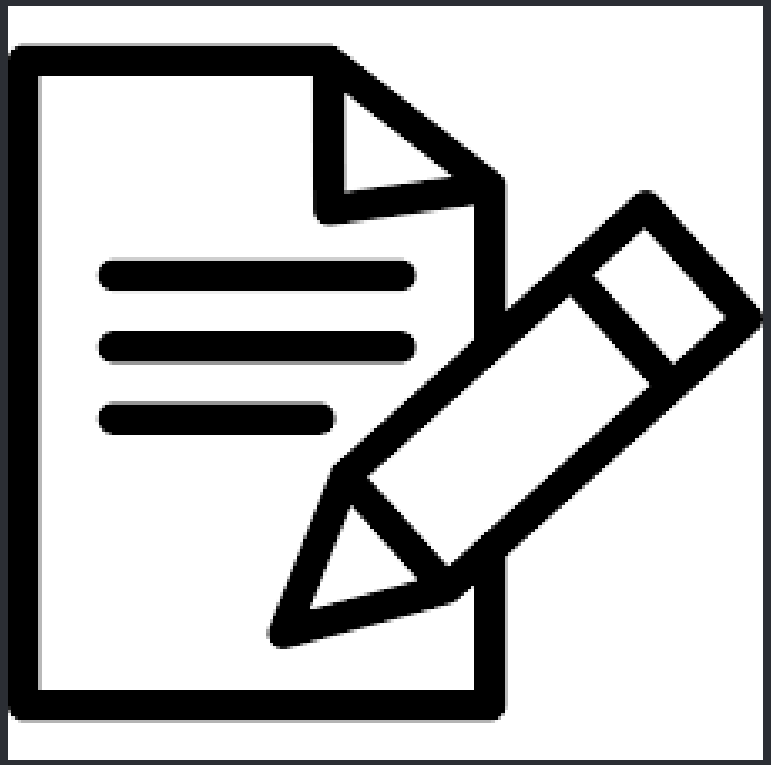
Como herramienta para la generacion de audio para capturar con la CIAA se propone el uso de simpleaudio con el siguiente template de codigo como base

```
import matplotlib.pyplot as plt
import numpy as np
import scipy.signal as sc
import simpleaudio as sa

f          = 400
fs         = 44100
sec        = 10
t = np.arange(0,sec,1/fs)
note = (2**15-1)*np.sin(2 * np.pi * f * t)
#jnote = (2**15-1)*sc.sawtooth(2 * np.pi * f * t)
#jnote = (2**15-1)*sc.square(2 * np.pi * f * t)

fir=plt.figure(1)
plt.plot(t[0:5*fs//f],note[:5*fs//f])
plt.show()

audio = note.astype(np.int16)
for i in range(100):
    play_obj = sa.play_buffer(audio, 1, 2, fs)
    play_obj.wait_done()
```



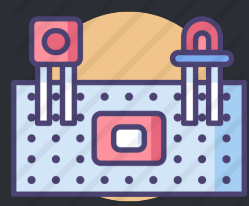
Metodo de evaluacion

- 3 pts - Examen
- 3 pts - TP Python
- 4 pts - Proyecto final



Evaluacion

Proyecto final



- Debera incluir algun tipo de procesamiento en hardware. ej. fft, fir, iir
- Puede utilizar el ADC para samplear, DAC para reconstruir y/o canales de comunicacion para adquirir datos previamente digitalizados
- presentacion de 10 minutos.
- debera funcionar!

Ejemplos:

- Filtrado y/o procesamiento de audio, señales biomedicas, etc.
- Tecnicas de compresion en dominio de la frecuencia
- Aplicaciones con acelerometro, magnetometro, T+H