

Procesamiento de señales. Fundamentos

Presentación

● Docente:

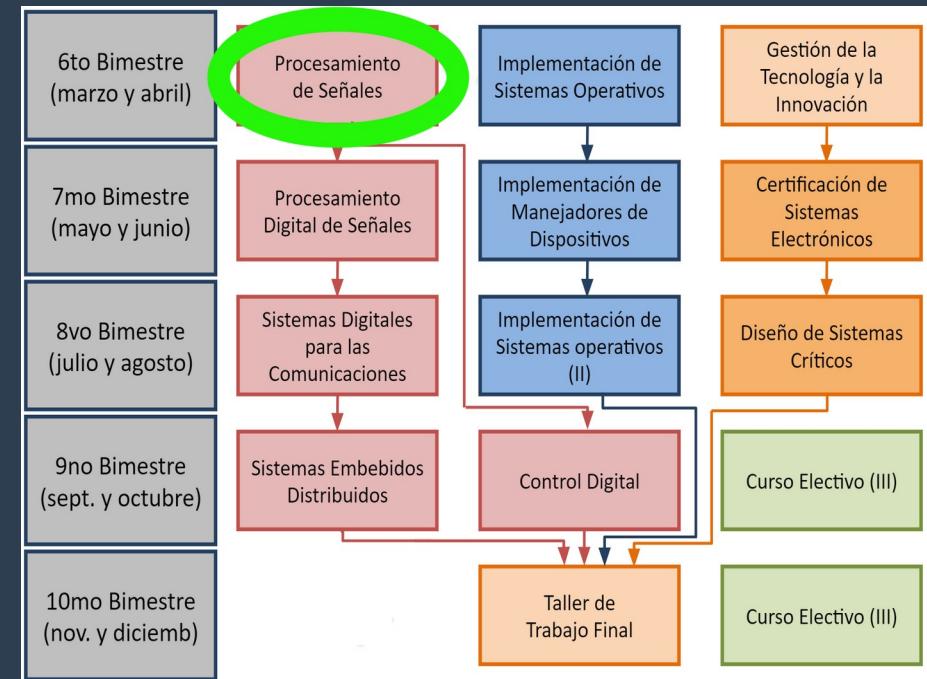
- **Pablo Slavkin**
 - <slavkin.pablo@gmail.com>

● Colaboradores:

- **Gonzalo Lavigna**
 - <gonzalolavigna@gmail.com>
- **Federico Giordano Zacchigna**
 - <federico.zacchigna@gmail.com>

● Correo grupal:

- <psf_m07@cursoscapse.com>



Procesamiento de señales. Fundamentos

Roadmap

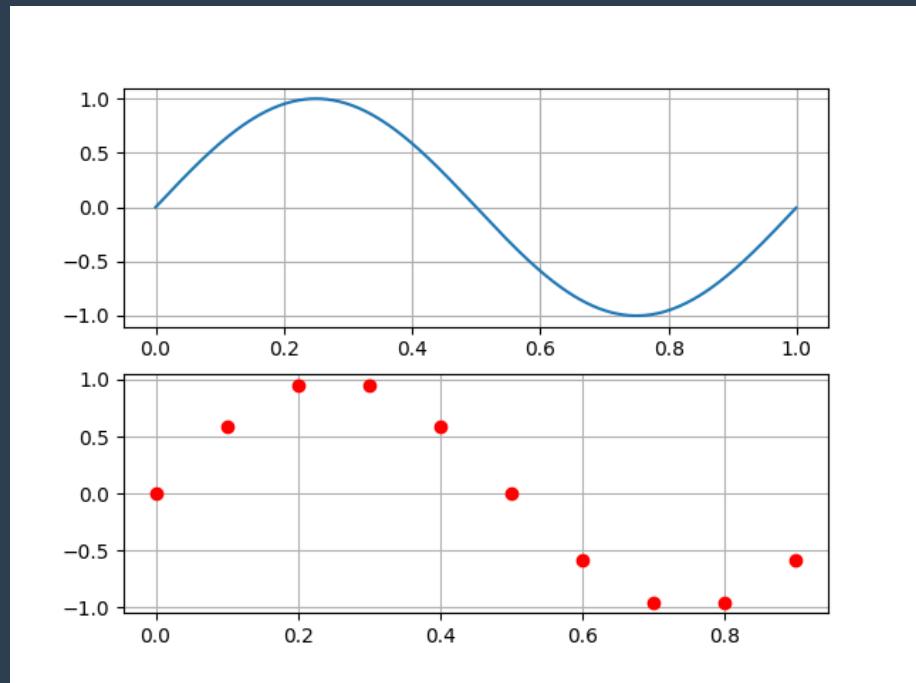
- 1 Señales y sistemas
- 2 CIAA<>Python | Hardware
- 3 Euler + Fourier + DFT
- 4 IDFT
- 5 Respuesta al impulso | Convolución
- 6 Filtrado | FIR
- 7 Repaso, TP final y tips & tricks
- 8



Procesamiento de señales. Fundamentos

Clase 1 – Señales y sistemas

- Señales y sistemas. LTI
- Adquisición
 - Fenómeno de aliasing
- Cuantización
 - Dithering
 - Sobre muestreo
- Reconstrucción
 - Teorema de sampleo
- Preparación de hardware
- Temas administrativos



Digital Vs analógico – Criterios

● Digital

- Reproducibilidad
- Tolerancia de componentes
- Partidas todas iguales
- Componentes no envejecen
- Fácil de actualizar
- Soluciones de un solo chip
- Bajo costo de producción



● Analógico

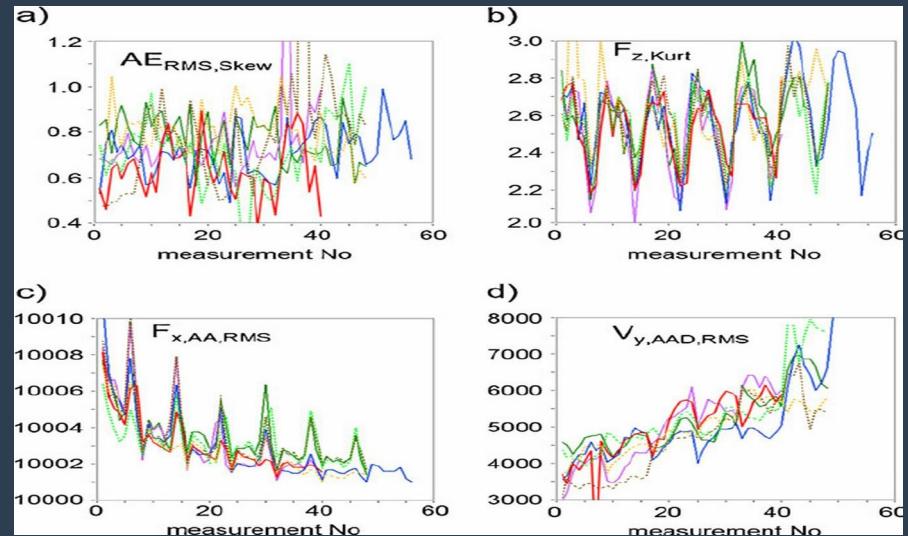
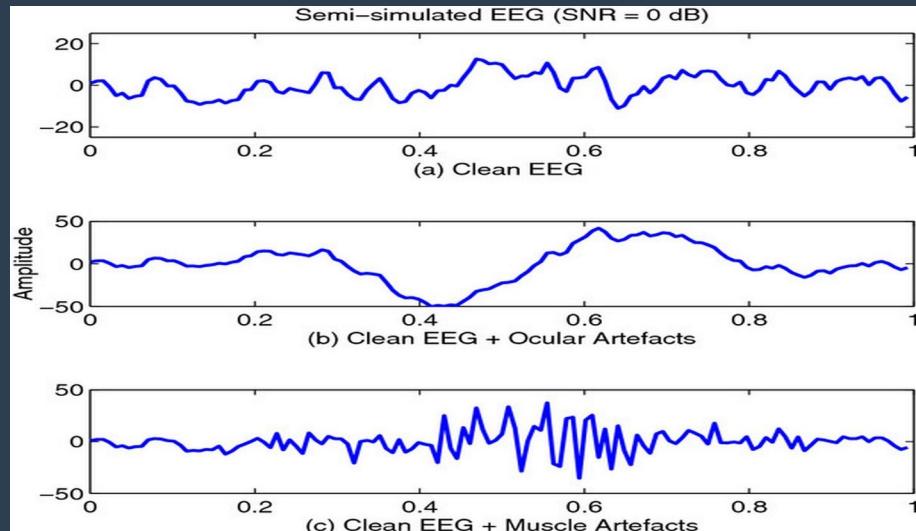
- Gran rango dinámico de entrada Ej.: 1mv to 20v
- Alto ancho de banda Ej. 0.001Hz to 10Mhz
- Alta potencia Ej: Amplificadores, RF
- Baja latencia Ej. Fibra óptica, RF



Señales

● Señales

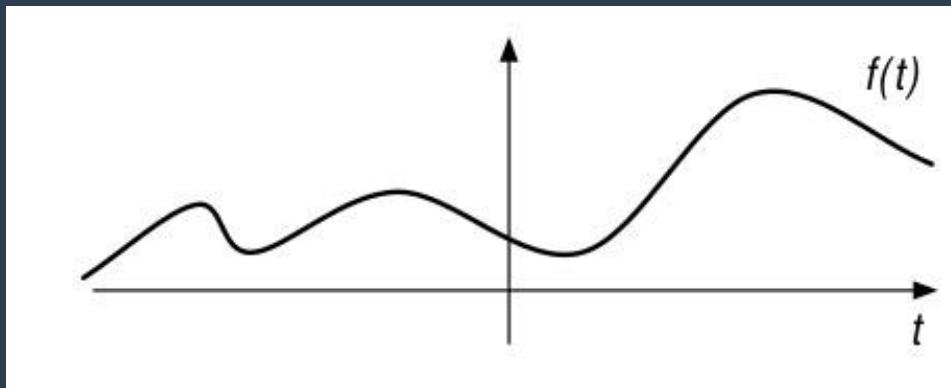
- Una señal, en función de una o más variables, puede definirse como un cambio observable en una entidad cuantificable
- Ej: Sonido, imágenes, video, biológicas.



Tipos de señales

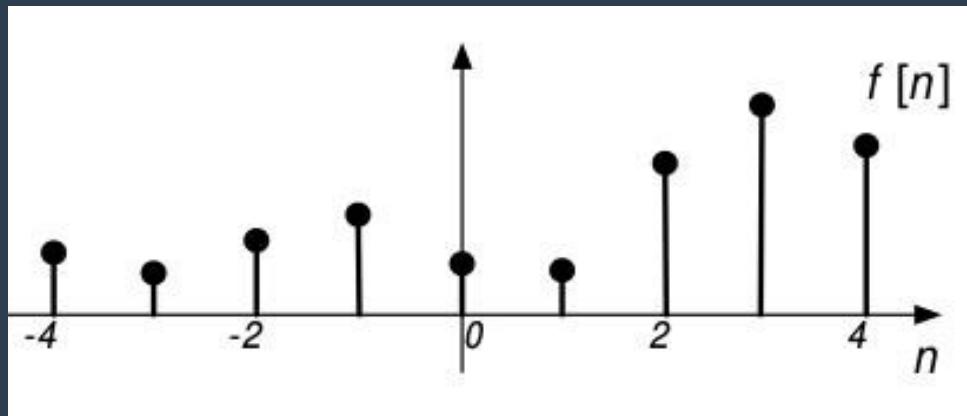
● Tiempo continuo

- Tiene valores para todos los puntos en el tiempo al menos en algún intervalo
- Ej: $y=\sin(x)$



● Tiempo discreto

- Tiene valores solo para puntos discretos en el tiempo
- El resto esta indefinido
- La distancia entre puntos debe ser siempre la misma??



Señales en Python - Calentamiento

- Genere las siguientes señales discretas en Python

- $S = [1, 5, 7, 0, -10.2]$
- $T = [-1, 19, 27.7, 4, 0.2, 50]$

- Con numpy

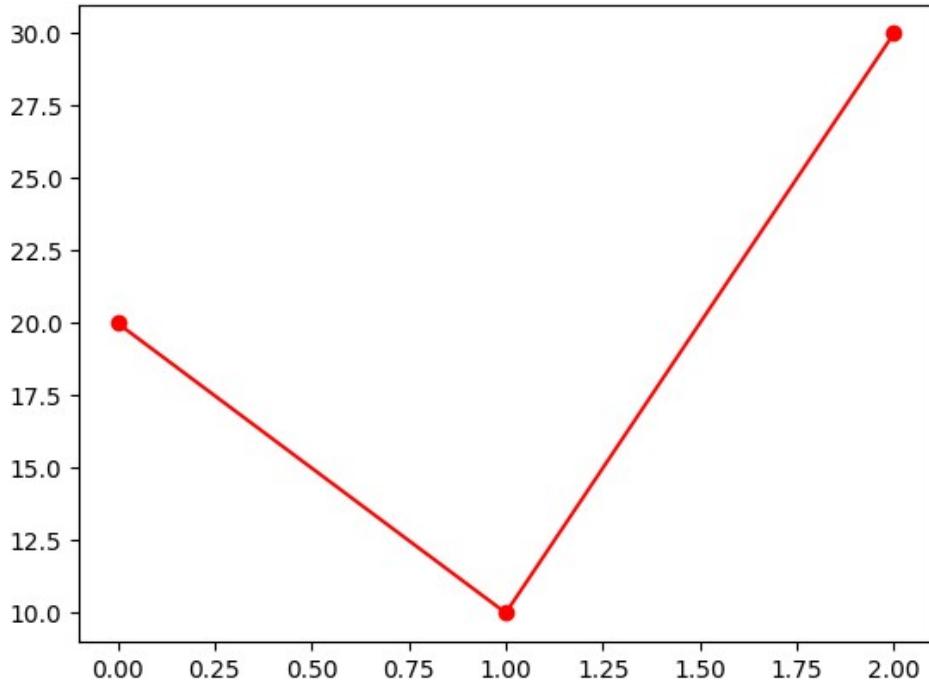
- $t = np.arange(0, 1, 0.1)$ # samples arrancando en 0 inclusive hasta 1 NO inclusive en pasos de 0.1
- $Array([0., 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9])$
- $s=np.zeros(len(t))$
- $print(type(s))$
- $t = np.linspace(0, 1, 10)$ #10 samples de 0 a 1 inclusives
- $array([0. , 0.11111111, 0.22222222, 0.33333333, 0.44444444, 0.55555556, 0.66666667, 0.77777778, 0.88888889, 1.])$
- $s=np.sin(2*np.pi*t)$
- $print(type(s))$

- Tiempo continuo en Python. Es posible?

- $S = ?$
- $T = ?$

Señales. Grafica con matplotlib

```
import matplotlib.pyplot as plt  
T=[0,1,2]  
s=[20,10,30]  
plt.plot(t,s,'ro-')  
plt.show()
```



Ver código:
senales_python.py

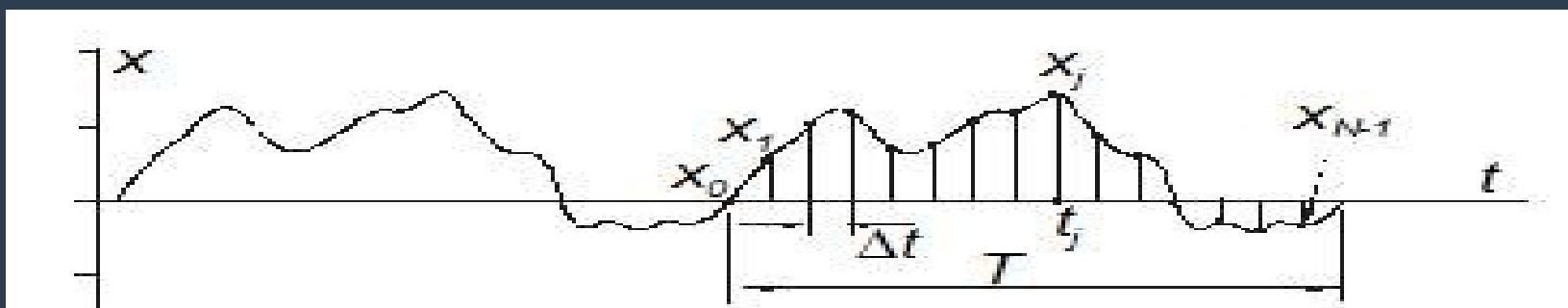
Tipos de señales

● *Continua periódica*

- 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 $x(t)$

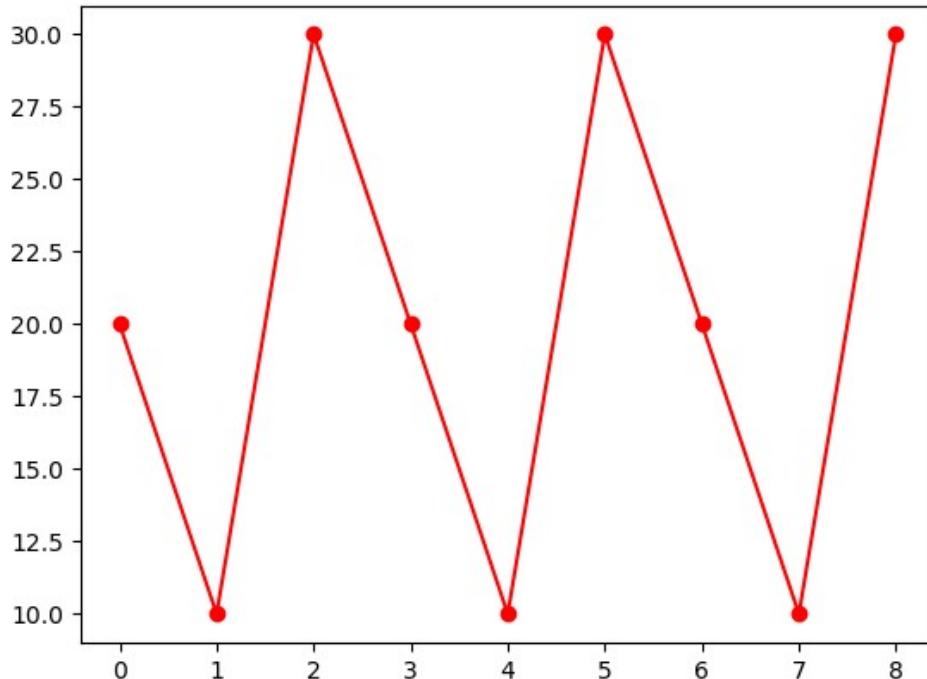
● *Discreta periódica*

- 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 $x[n]$



Señales periódicas + matplotlib

```
import numpy as np  
  
import matplotlib.pyplot as plt  
  
S1=[20,10,30]  
s=np.tile(s1,3)  
t=np.arange(0,len(s),1)  
plt.plot(t,s,'ro-')  
plt.show()
```



Ver código: seniales_periodicas.py

Tipos de señales – Potencia vs Energía

- *De energía finita (continua)*

$$0 < \int_{-\infty}^{\infty} |x(t)|^2 dt < +\infty$$

- *De potencia finita (continua)*

$$0 < \lim_{T \rightarrow +\infty} \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} |x(t)|^2 dt < +\infty$$

- *De energía finita (discreta)*

$$E_x = \sum_{n=-\infty}^{\infty} |x(n)|^2$$

Si: $0 < E_x < +\infty$.

- *De potencia finita (discreta)*

$$P_x = \lim_{N \rightarrow \infty} \frac{1}{2N+1} \sum_{n=-N}^{n=+N} |x(n)|^2$$

Si: $0 < P_x < +\infty$.

Tipos de señales – Potencia finita

- *Ejemplo del cálculo de potencia de Sin(X)*

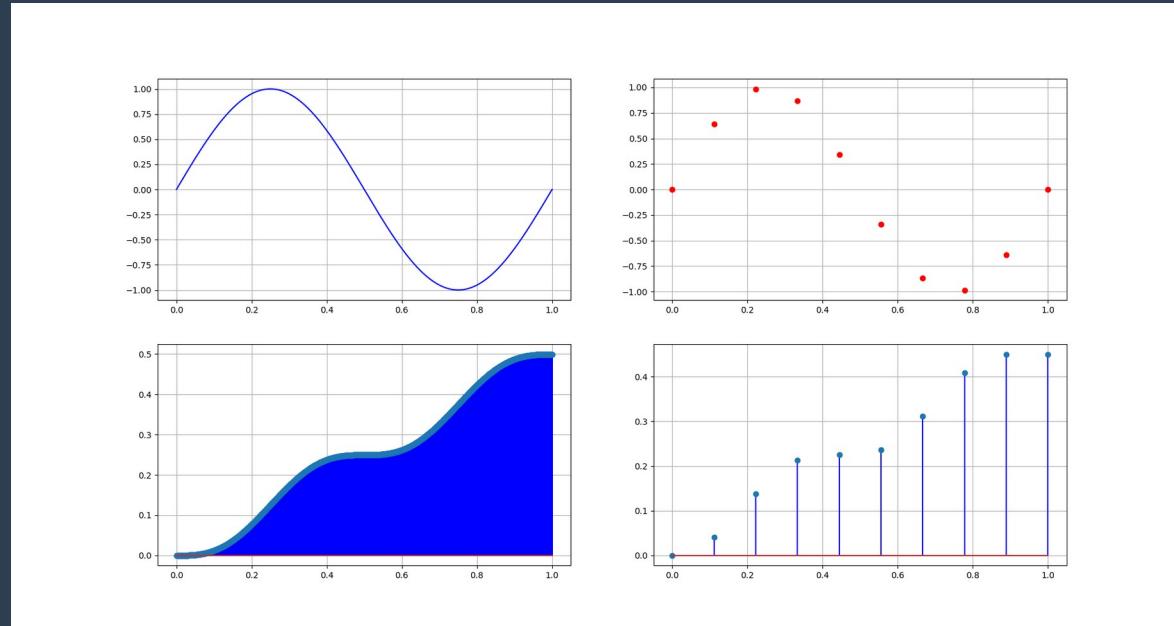
$$P = \frac{1}{2\pi-0} \int_0^{2\pi} |\sin(t)|^2 dt$$

$$= \frac{1}{2\pi} \frac{1}{2} \int_0^{2\pi} (1 - \cos(2t)) dt$$

$$= \frac{1}{4\pi} (t - \frac{1}{2} \sin(2t))|_{t=0}^{t=2\pi}$$

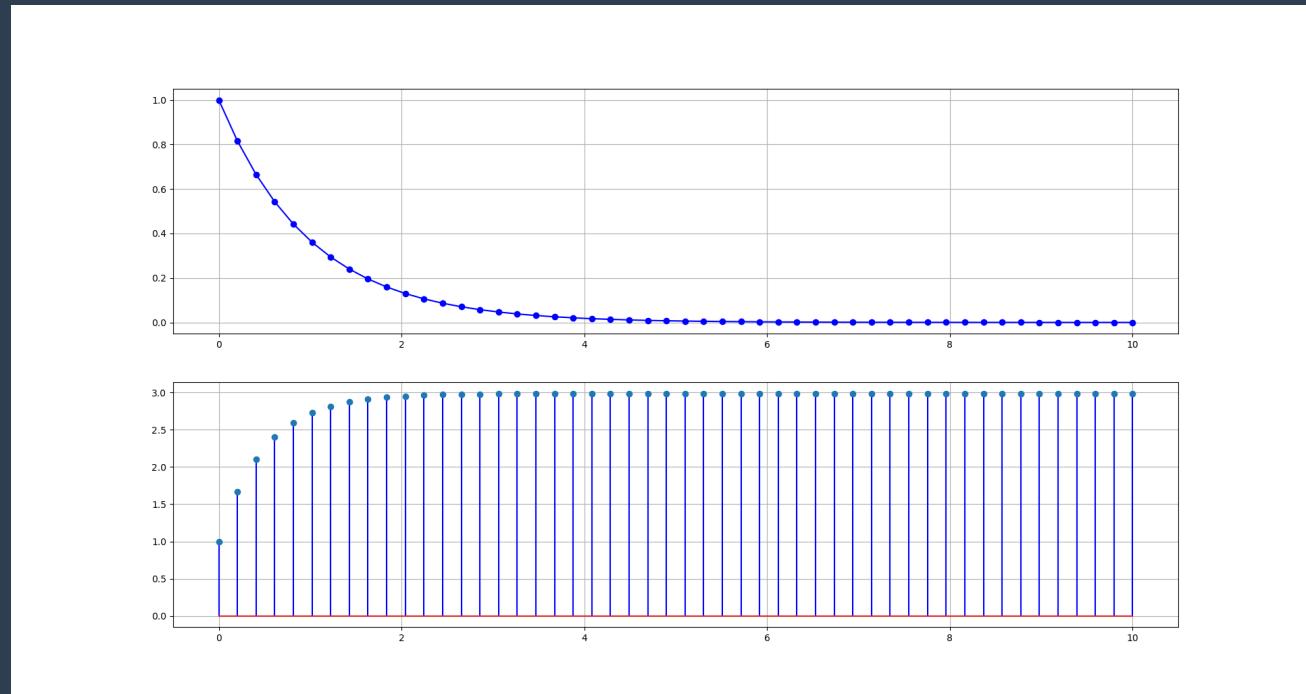
$$= \frac{1}{4\pi} (2\pi) = \frac{1}{2}$$

- *Ejemplo del cálculo de potencia de Sin(n) con Python*
- *Ver código: sine.py*



Tipos de señales – Energía finita

- *Ejemplo del cálculo de energía de $1/e^n$*
- *Notar que la potencia es 0 (cero).*
- *Ver código: energia.py*



Sistemas

- *Definición*

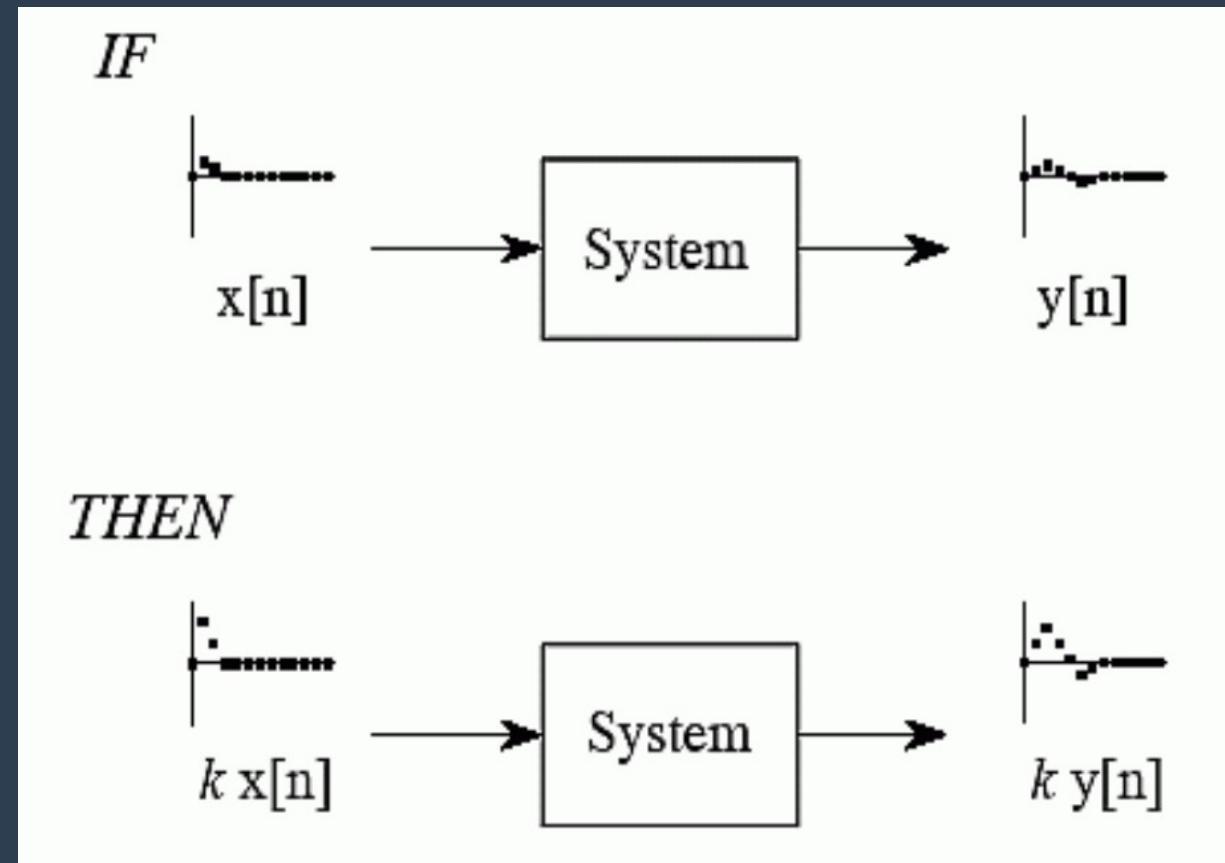
Un sistema es cualquier conjunto físico de componentes que actúan en una señal, tomando una o más señales de entrada, y produciendo una o más señales de salida



Sistemas homogéneos

● Escalado

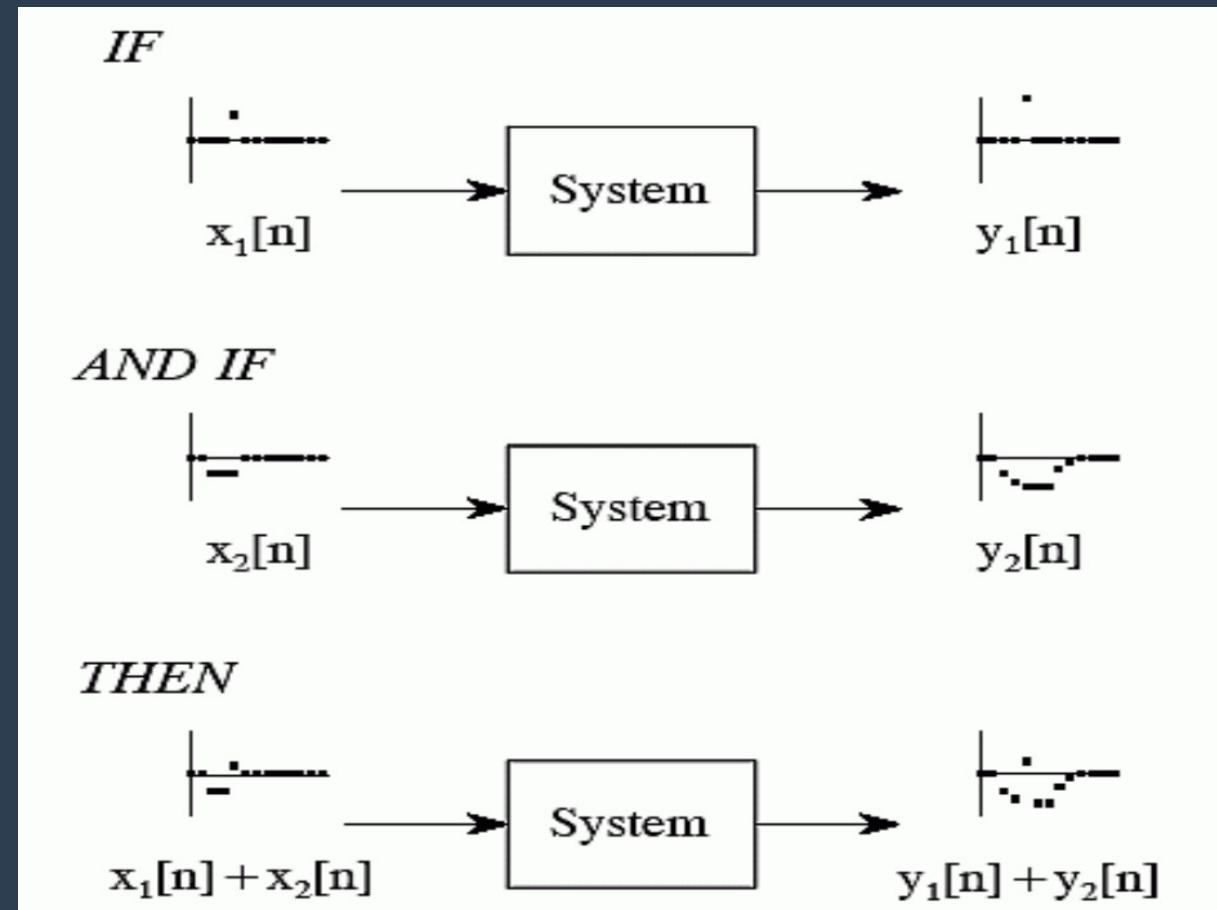
- *Un cambio en la entrada repercute en la salida en la misma proporción*



Sistemas aditivos

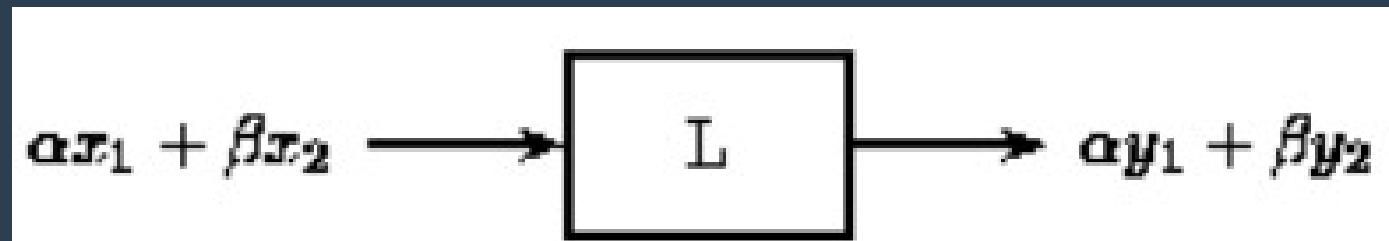
● Adición:

- *Si las entradas atraviesan el sistema sin interacción entre ellas*



Sistemas lineales – Superposición

- Un sistema es lineal cuando su salida depende linealmente de la entrada.
- Satisface el principio de superposición.



Ej:

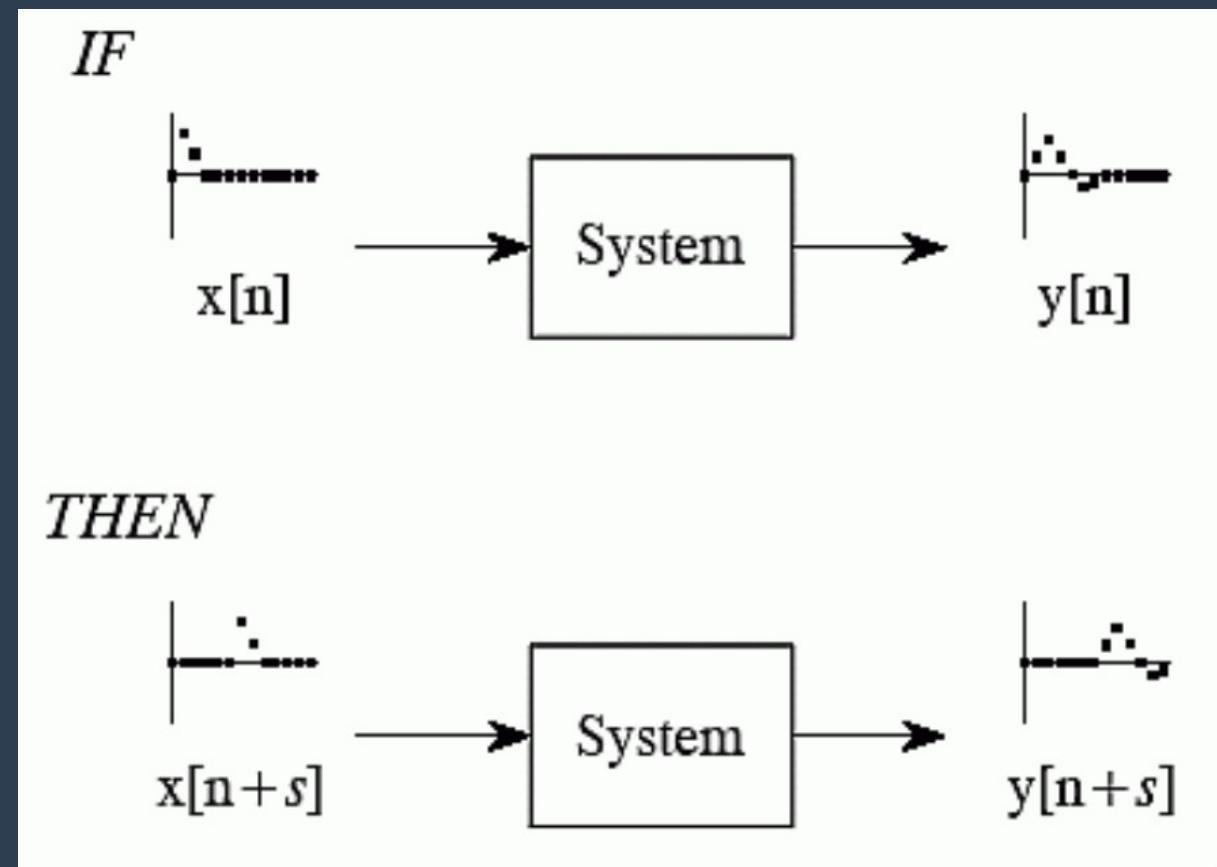
- $y(t) = e^{x(t)}$ => no lineal
- $y(t) = 1/2 x(t)$ => lineal

Sistemas de tiempo invariante

- *Si la entrada se demora, la salida también*

Ej:

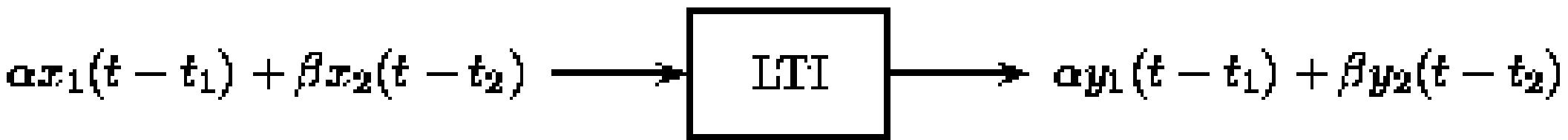
- $y(t) = x(t) * \cos(t)$ => *no invariante*
- $y(t) = \cos(x(t))$ => *invariante*



Sistemas - LTI

● *Lineales invariantes en el tiempo*

- Un sistema es LTI cuando satisface las dos condiciones:
 - 1) linealidad
 - 2) Invariancia en el tiempo

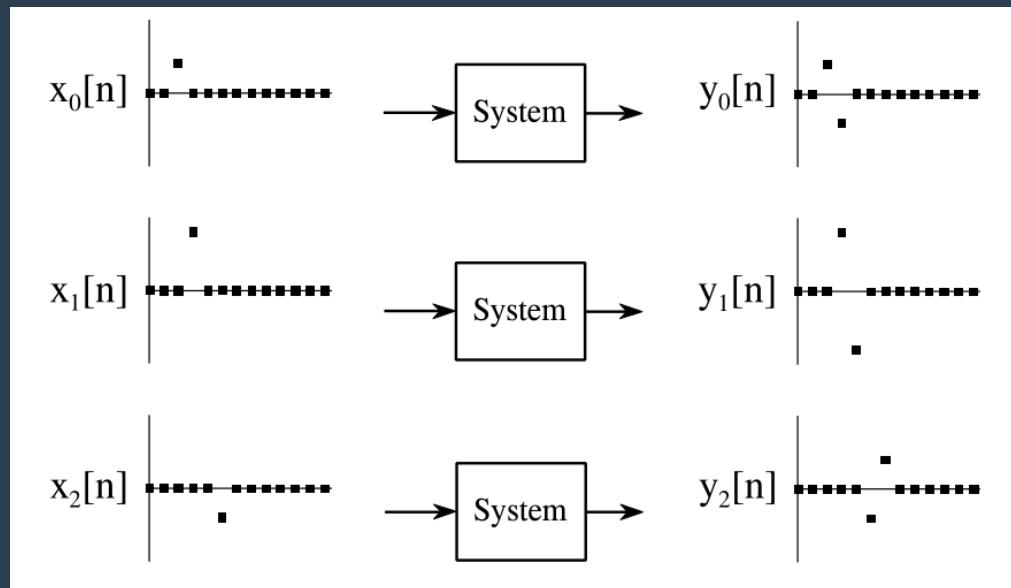
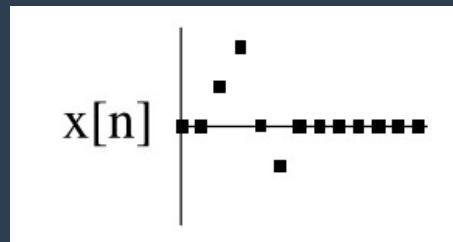


En este curso, solo estudiaremos sistemas lineales invariantes en el tiempo.

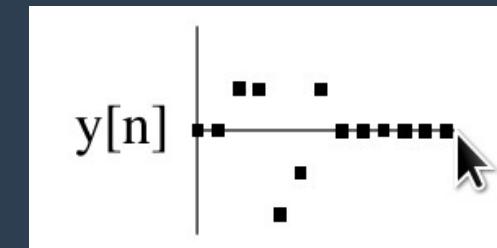
Sistemas LTI – Descomposición

- Podemos descomponer una señal compleja en una suma de señales simples.
- Hacer pasar cada una por el sistema
- Y finalmente sumar los resultados

Descomposición ->



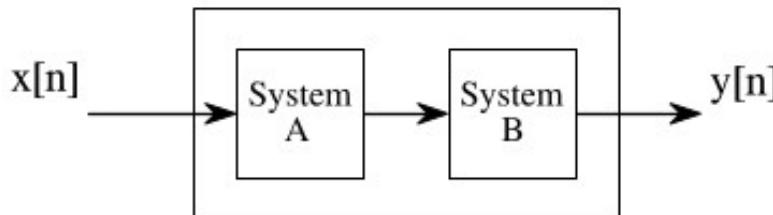
Síntesis ->



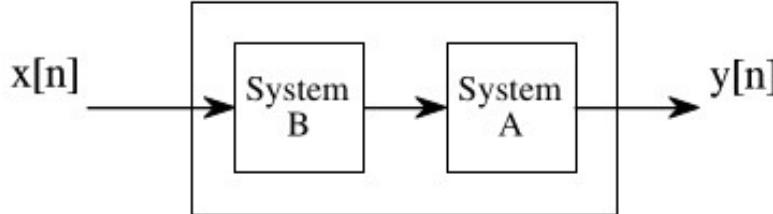
Sistemas LTI - Conmutación

- Si el sistema es LTI, perfectamente LTI, entonces es commutable
- En la practica, atención con los efectos de precisión, desborde, clipping, etc.

IF



THEN



Sistemas LTI - Características

● *Fidelidad senoidal*

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

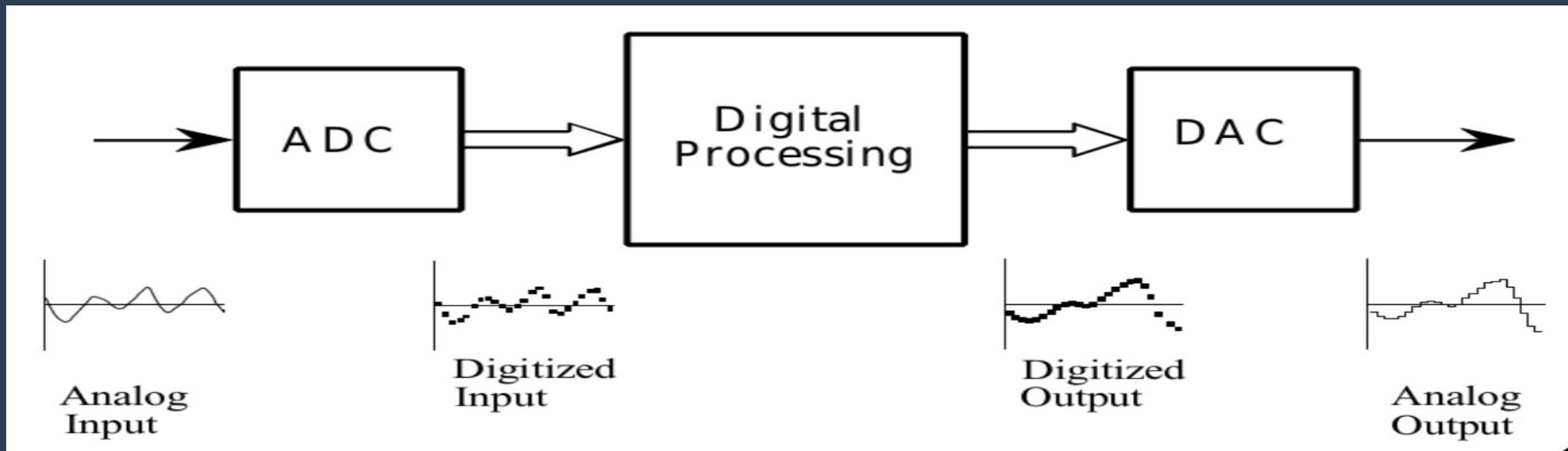
● *Linealidad estática*

- En todo sistema LTI para una entrada constante (DC) la salida es **siempre** la entrada multiplicada por una constante.

Adquisición

Secuencia de adquisición - Encuesta

- Secuencia de digitización, procesamiento y reconstrucción.
- Es correcta la secuencia? Falta algo? Sobra algo?
- <https://forms.gle/Z2e7MuApjYVmSqED6>



Efecto de aliasing en un disco giratorio

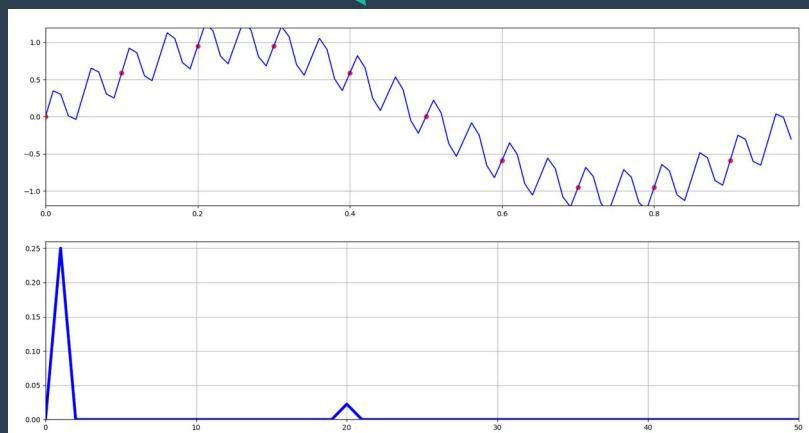
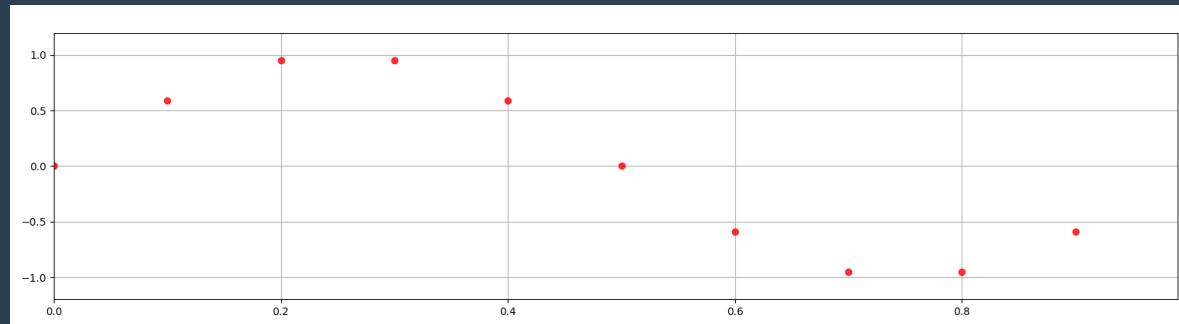
<https://youtu.be/-XiWEq8MIKc>



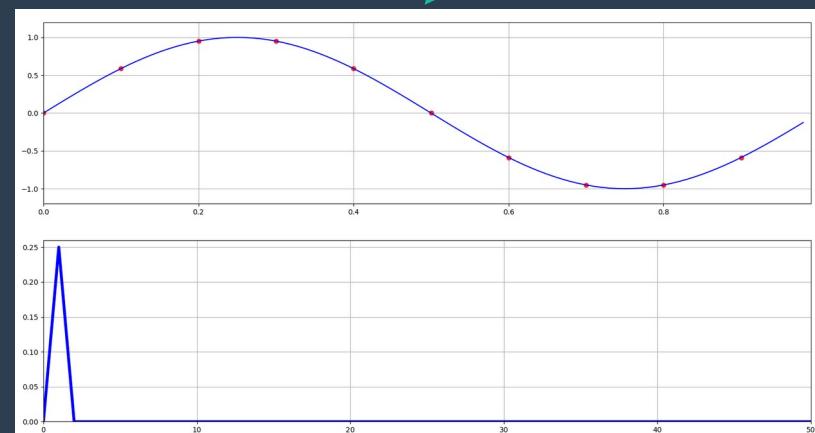
Efecto de aliasing en Python

- $F_s=10$ | $N=10$
- A simple vista parece una senoidal de 1Hz
- Pero podria esconder una señal multiplo de F_s , como 20Hz en el ejemplo
- Al samplear a F_s , es imposible distinguir señales mayores a $F_s/2$
- Por eso es necesario el FAA que evita que entren al sistema señales no deseadas de $F>F_s/2$
- Ver código teorema_sampleo.py

• *Que señal esconde esta secuencia de puntos?*



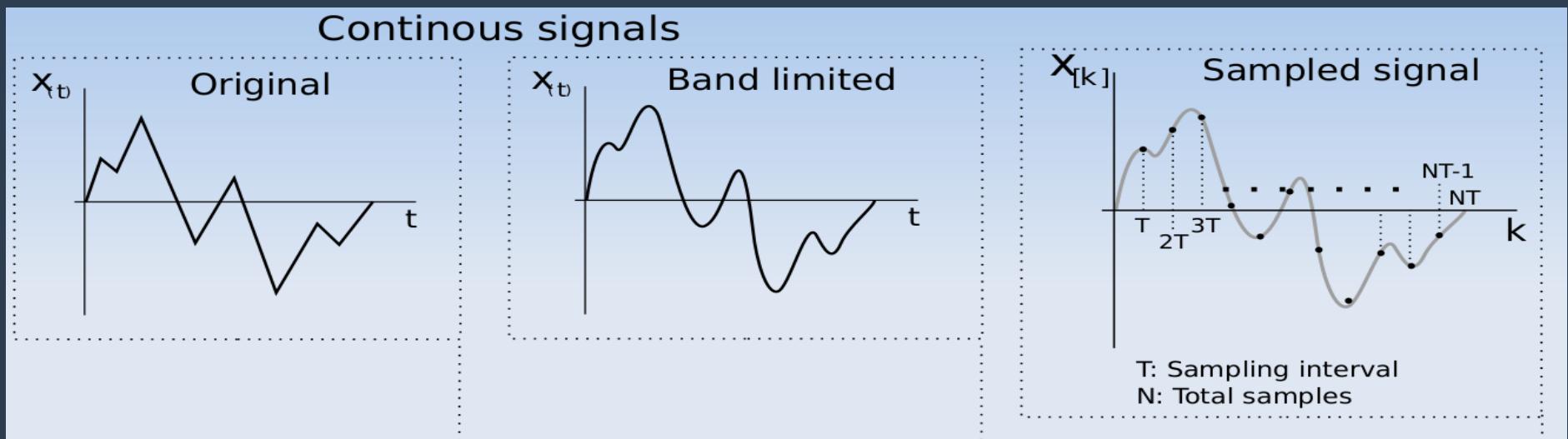
1Hz+20Hz ?



1Hz ?

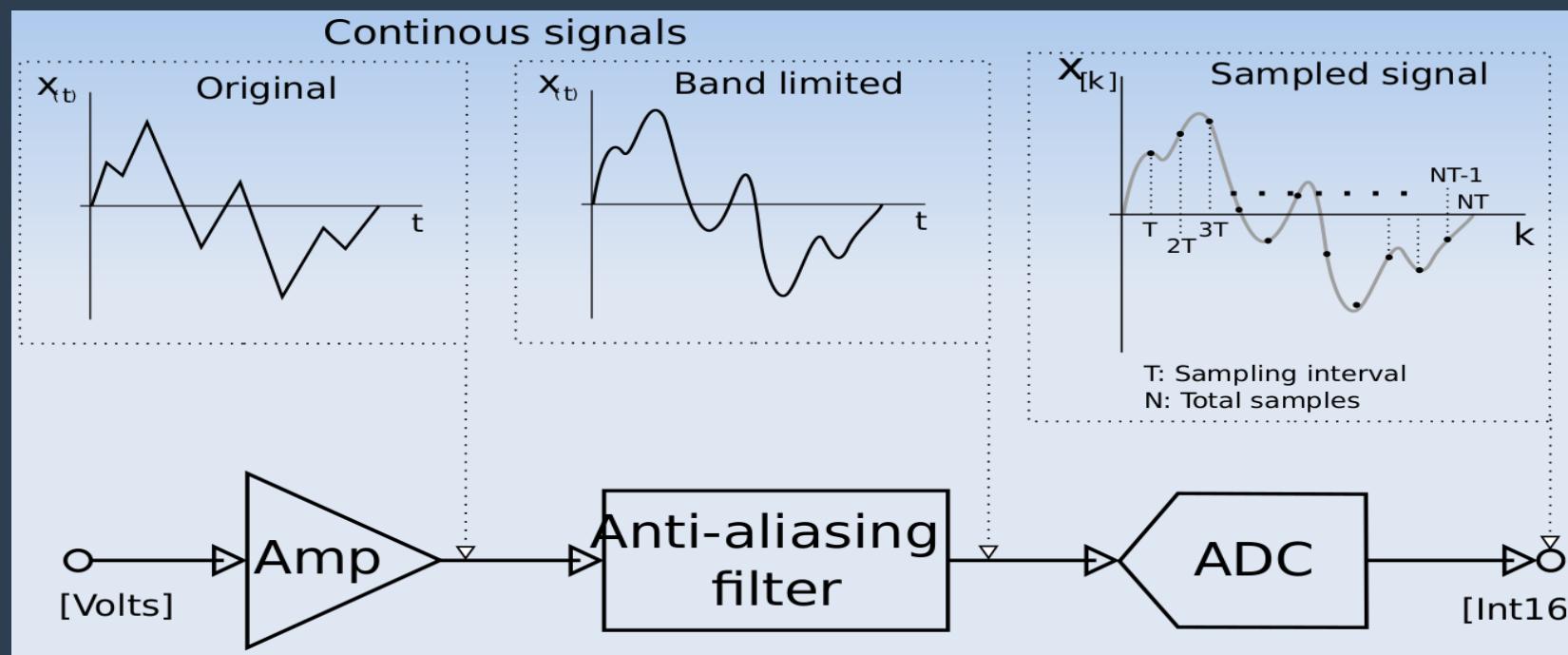
Filtro anti alias (FAA) - Pasa bajos

- El FAA es un filtro **analógico** pasa bajos que elimina o al menos **mitiga** el efecto de aliasing



Digitización - Filtro anti alias

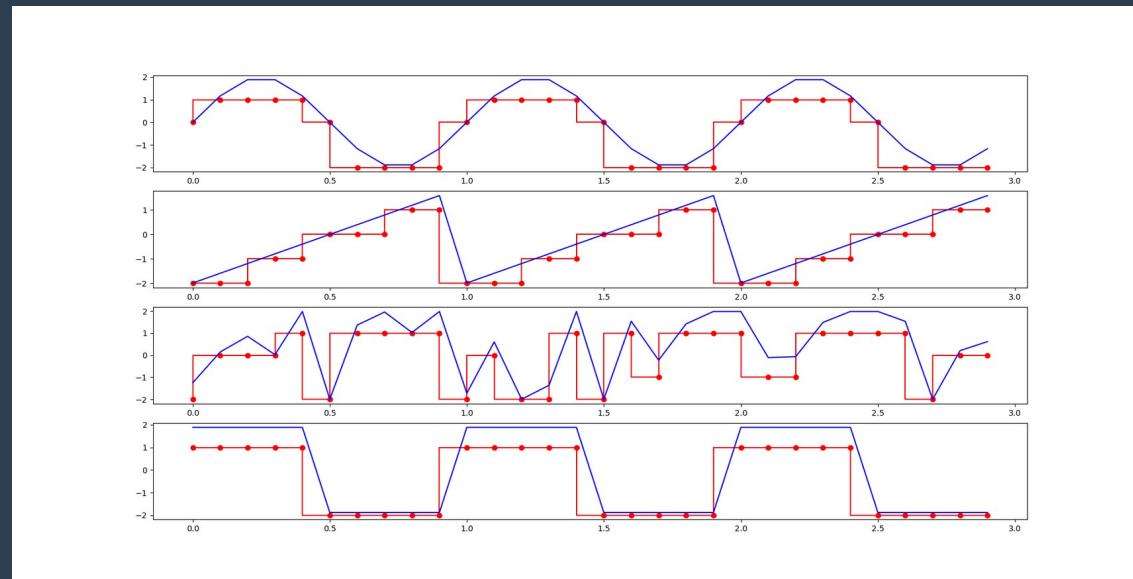
- Secuencia de adquisición utilizando un filtro anti alias
- Se podría poner el bloque de antialiasing luego del ADC?? Porque?
- Que estrategias puedo usar para simplificar el FAA analogico?



Cuantización

Ruido de cuantización en Python

- En azul la señal sampleada con precisión 'infinita'
- En rojo la señal escalonada con precisión de 2 bits
- Ver código:
`ruido_cuantizacion.py`
- Ej: Calcule el error máximo para una señal de 1V, ADC de 10b entre 0-1V:
 $e=1/1024=0,000976562$ volts

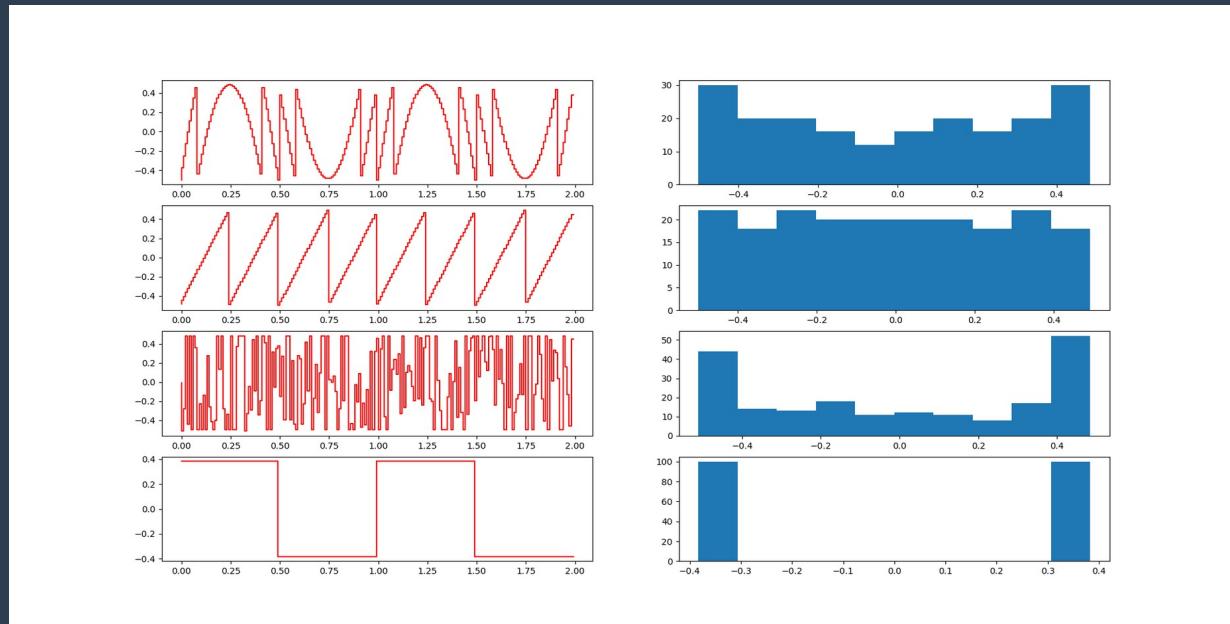


Cuantización

- La conversión analógica a digital implica cuantizar la entrada en un rango de valores discretos de precisión finita
- Esta precisión depende de la cantidad de símbolos (bits) almacenados para cada muestra
- Esto genera un error en la conversión que se denomina error de cuantización.
- Si este error cumple con determinadas premisas, podría considerarse como ruido en la señal, y en este caso se lo denomina ruido de cuantización

Ruido de cuantización histograma

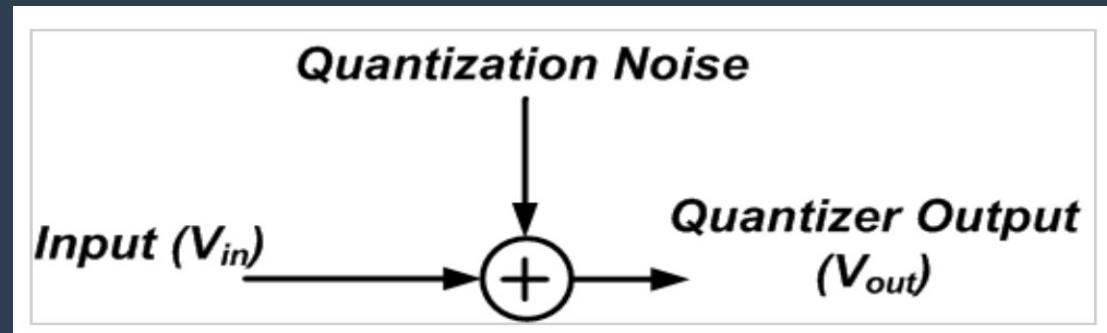
- Observando los histogramas se puede determinar si el error de cuantización se puede modelar o no como ruido
- En particular el error en una cuadrada no se comporta como ruido
- Ver código:
`ruido_histograma.py`



Ruido de cuantización

En el caso de que se cumplan las siguientes premisas:

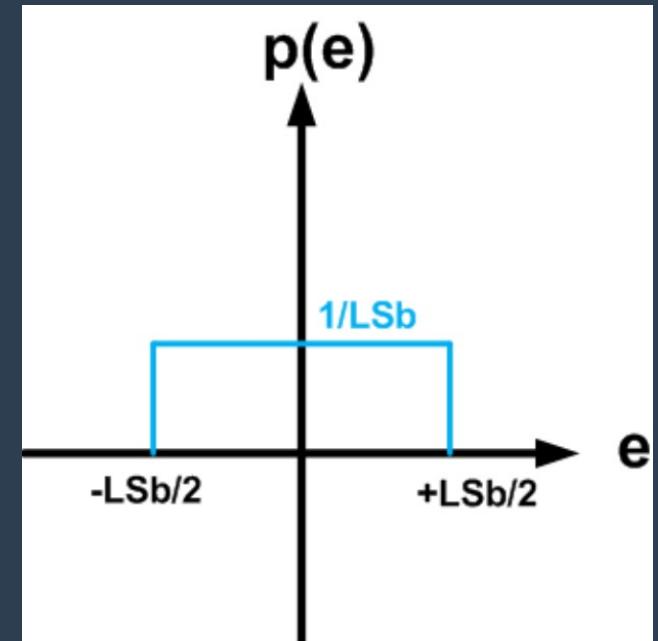
- La entrada se distancia de los diferentes niveles de cuantización con igual probabilidad
- El error de cuantización NO esta correlacionado con la entrada
- El cuantizador cuanta con un numero relativamente largo de niveles
- Los niveles de cuantización son uniformes
- Se puede considerar la cuantización como un ruido aditivo a la señal según el siguiente esquema:



Densidad de ruido de cuantización

- Si el error de cuantización se considera ruido, la densidad de probabilidad del error es constante entre +- la mitad del bit menos significativo
- Como en toda densidad, el área vale 1
- Esto implica que el valor de la densidad para todo el error sera de:
 $1/\text{LSB}$

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



Potencia de ruido de cuantización

- Calculo de la potencia del ruido, en función del LSB
- (Nota: e^2 significa error maximo al cuadrado, no es e de Euler)

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

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

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

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

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

Potencia de ruido de cuantización

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

Relación señal a ruido

- Calculo de la relación entre una señal senoidal de amplitud máxima y el ruido de cuantización

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

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

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

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

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

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

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

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

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

Relación señal a ruido

- Conociendo la potencia de señal y la del ruido se calcula su relación
- Que se puede hacer para mejorar la SNR en un sistema?

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

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

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

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

SNR

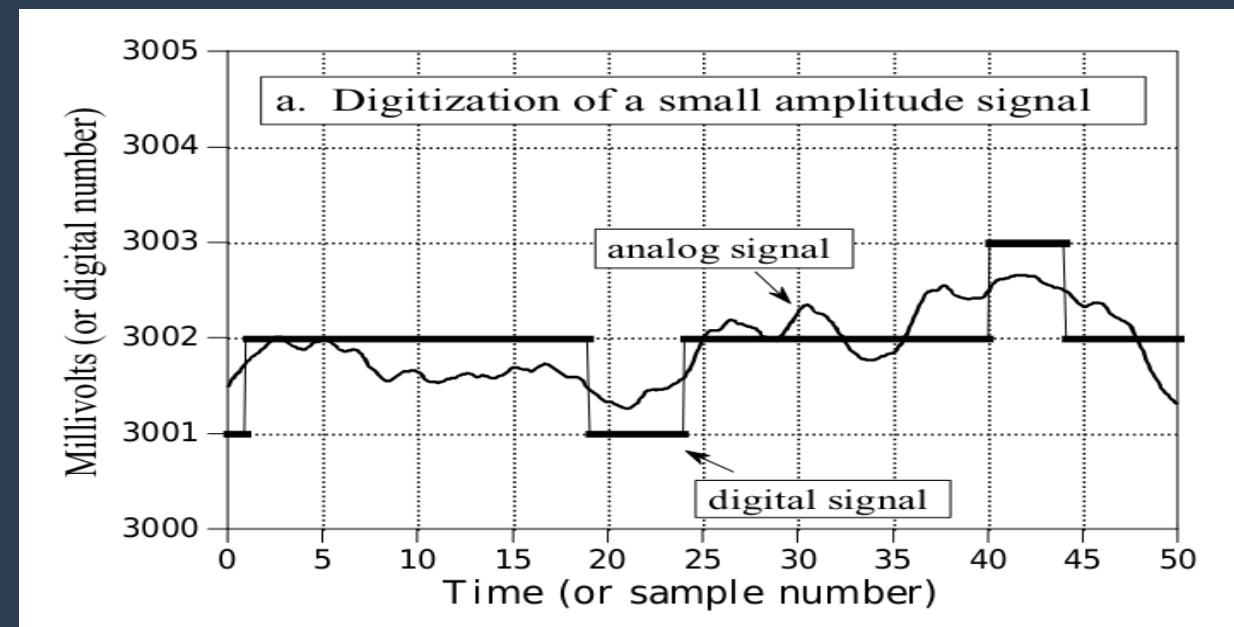
$$SNR = 1,76 + 6,02 * N$$

$$SNR_{N=10} \approx 62dB$$

$$SNR_{N=11} \approx 68dB$$

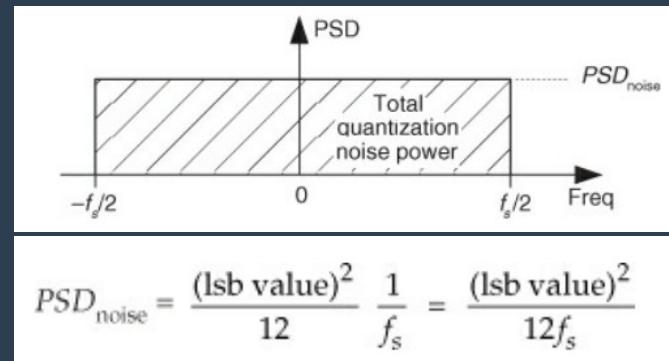
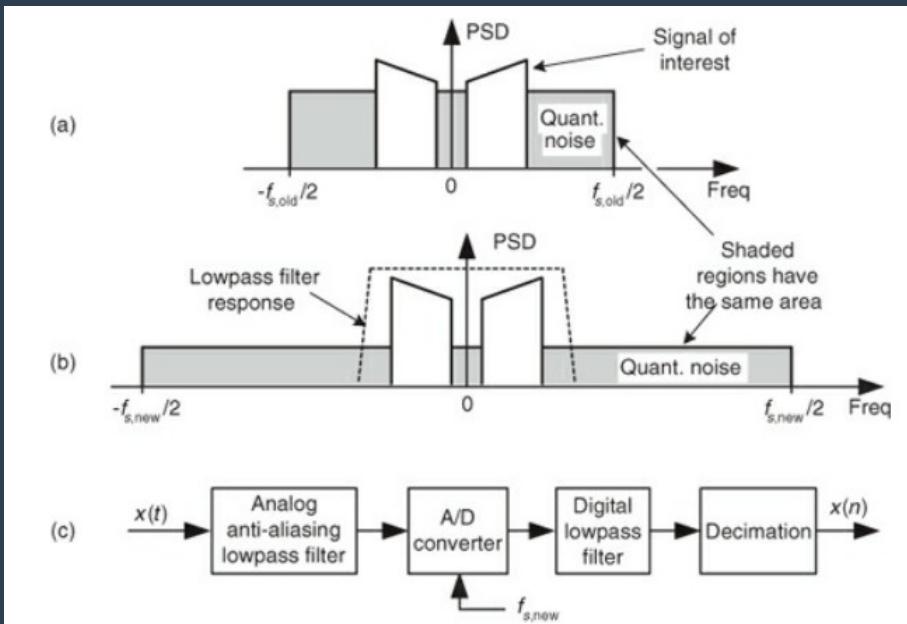
Dithering

- Técnica de agregado de ruido antes del ADC para prevenir que señales con poca variación sean sampleadas siempre con el mismo valor



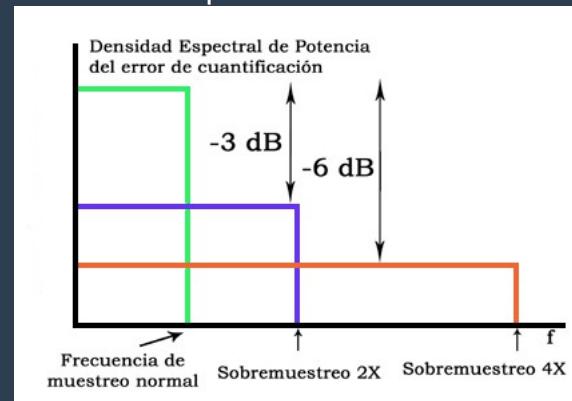
Densidad espectral de potencia de ruido

- Que se puede hacer para mejorar la SNR en un sistema?
- Se puede reducir el LSB
- Pero también se puede incrementar f_s !



$$SNR_{A/D-gain} = 10\log_{10}(f_{s,new}/f_{s,old}).$$

- $10 \log(4) = 6.02\text{dB}$
- 4x f_s equivale a 1 bit extra!



Reconstrucción

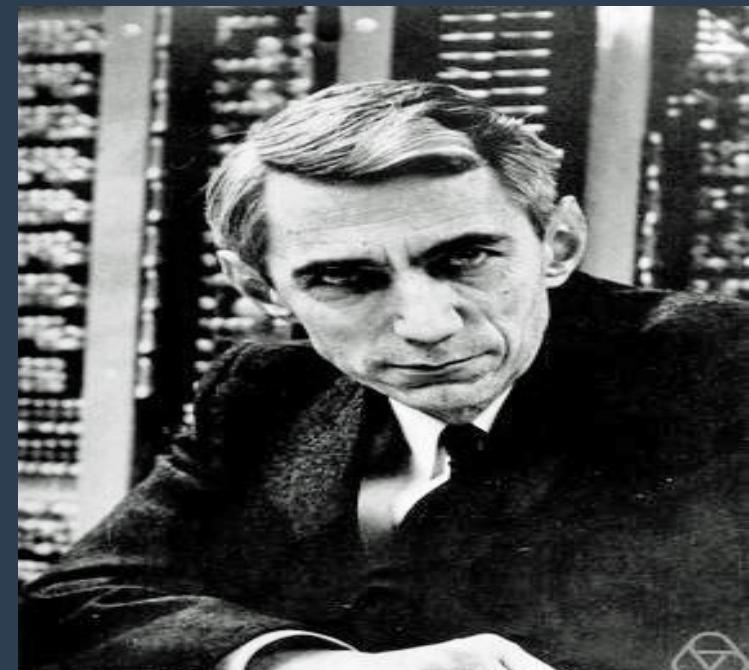
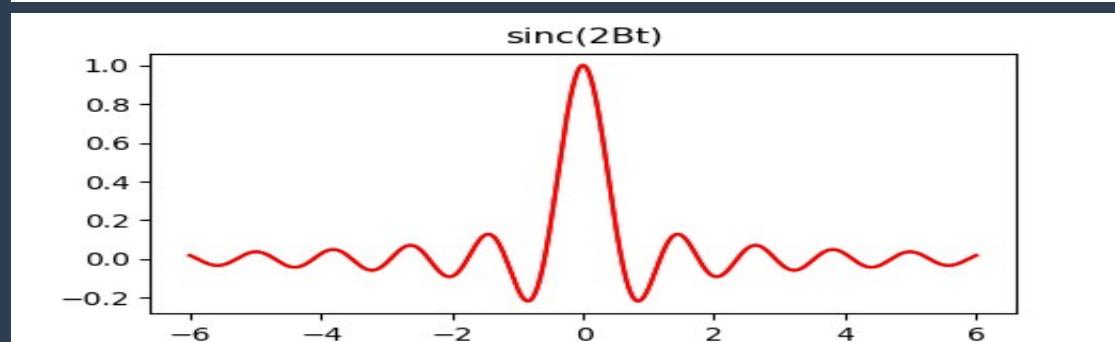
Filtro reconstructor (FAA) - Pasa bajos

- Consiste en obtener una señal continua a partir de una discreta
- Si es continua => el bloque reconstructor deberá ser analógico
- Habrá que llenar/interpolar los infinitos puntos intermedios
- Cuantos puntos como mínimo serian necesarios para una interpolación correcta?
- Cual seria la función de interpolación optima?

Teorema del sampleo - Shannon-Nyquist

- 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

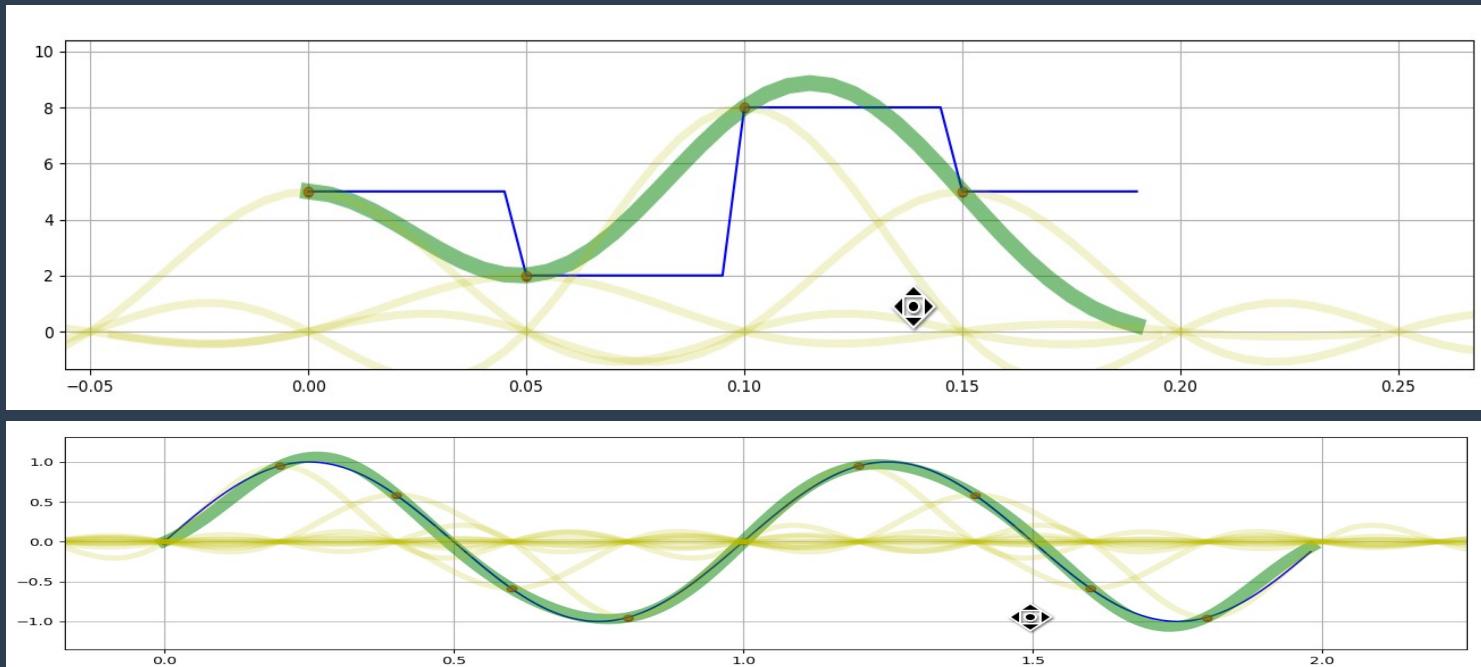
$$x(t) = \sum_{n=-\infty}^{\infty} x_n \frac{\sin \pi(2Bt - n)}{\pi(2Bt - n)}.$$



- Ver código `sinc.py`

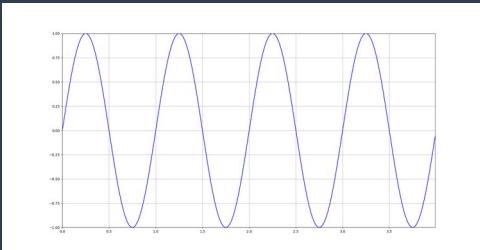
Teorema del sampleo - Python

- Reconstrucción utilizando el promedio de funciones Sinc centradas en cada muestra. Teorema de Shannon
- Probar diferentes escenarios cambiando la cantidad de samples
- Ver código: teorema_sampleo.py

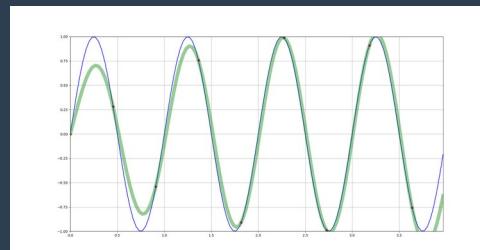


Efecto de aliasing en Python

Azul: 1hz fs=inf

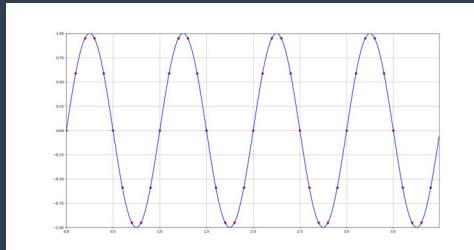


Rojo: samples fs=2.2
Verde Shannon

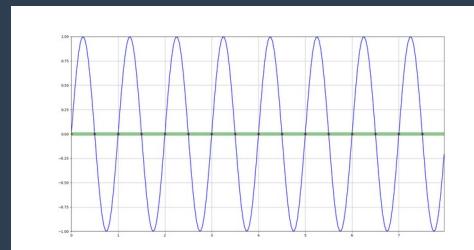


Se puede distinguir una señal de 1Hz sampleada a $f_s=1.8$ de una señal de -0.8Hz sampleada a la misma F_s ? Porque? Como se resuelve?

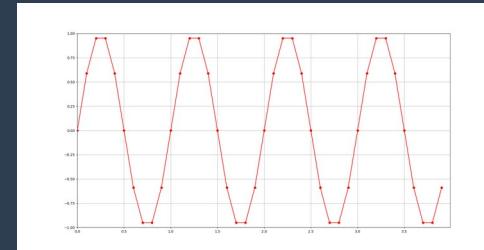
Azul: 1hz fs=inf
Rojo: samples fs=10



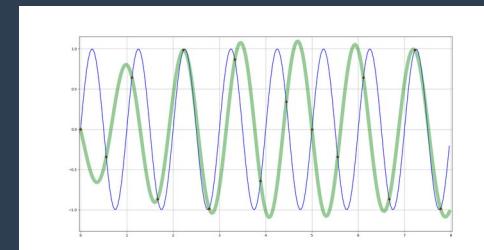
Rojo: samples fs=2.0
Verde: Shannon



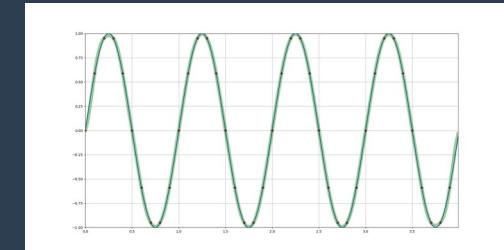
Reconstrucción con
lineas



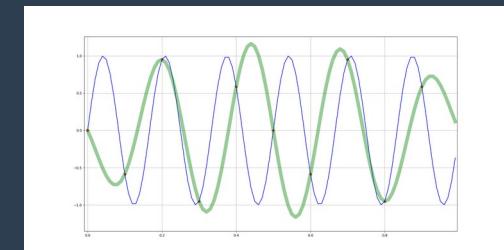
Rojo: samples fs=1.8
Verde: alias f=-0.8



Reconstrucción con
Shannon



Azul: f=6 fs=10
Verde: f=-4

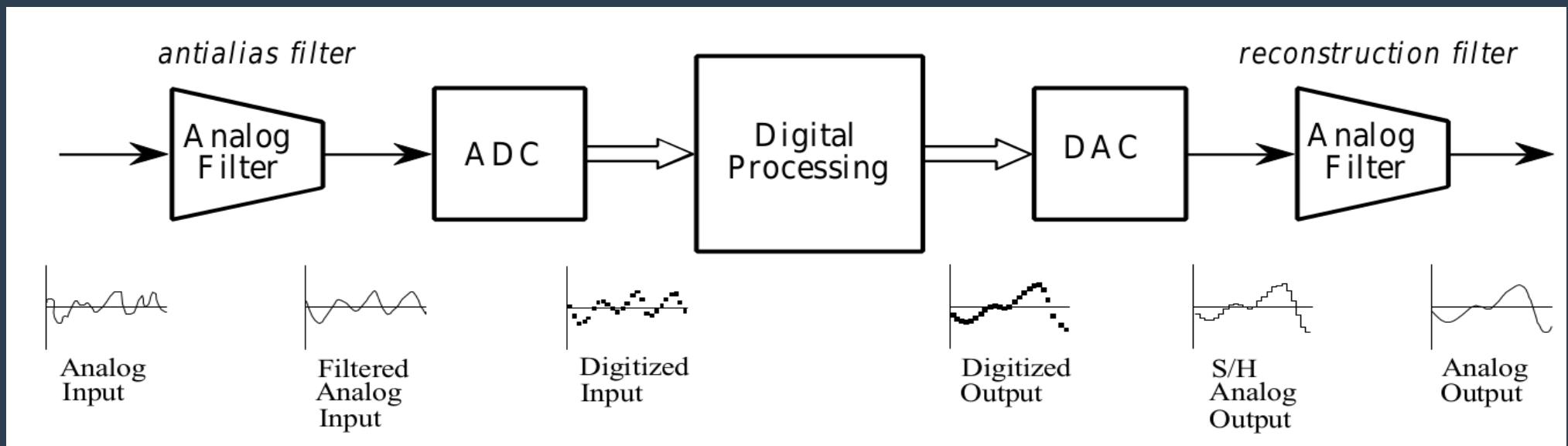


Auxiliar f=-0.8 fs=10



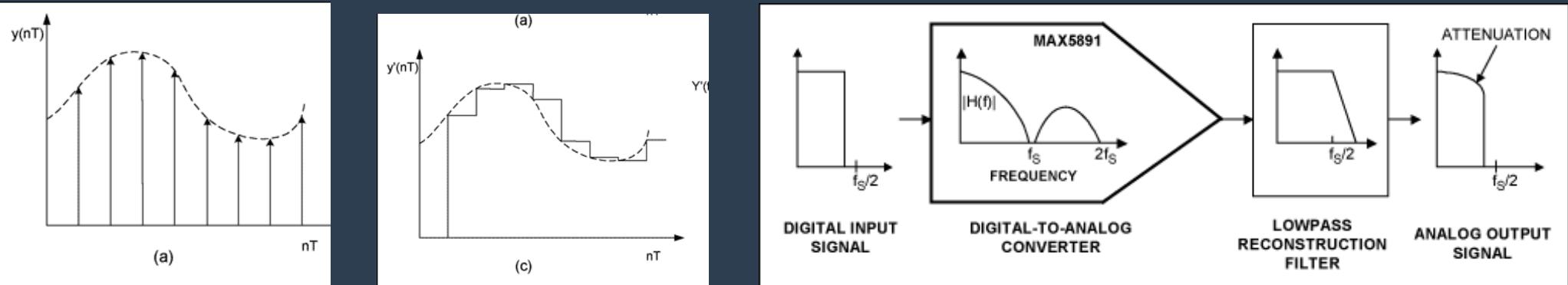
Digitalización - Secuencia completa

- Secuencia de digitizacion, procesamiento y reconstrucción.
- Se agrega el bloque (filtro) anti alias.
- Se agrega el bloque (filtro) de reconstrucción



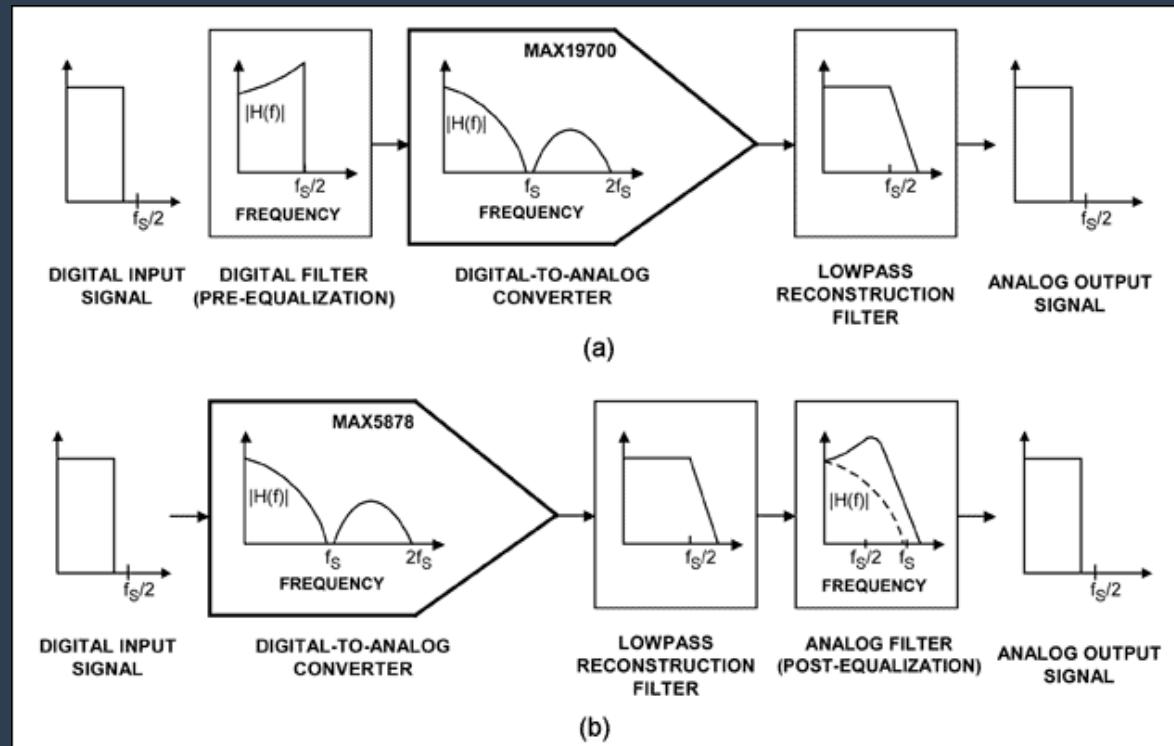
Reconstrucción - Respuesta del DAC - ZOH

- En el análisis teórico la salida del DAC debería ser un pulso de duración 0 para cada muestra.
- En general el DAC no genera pulsos de corta duración sino que mantienen un valor constante por un lapso de tiempo igual a $1/f_s$
- Esta diferencia se traduce en una atenuación de la salida para frecuencias cercanas a $f_s/2$



Reconstrucción - Mitigación efecto ZOH

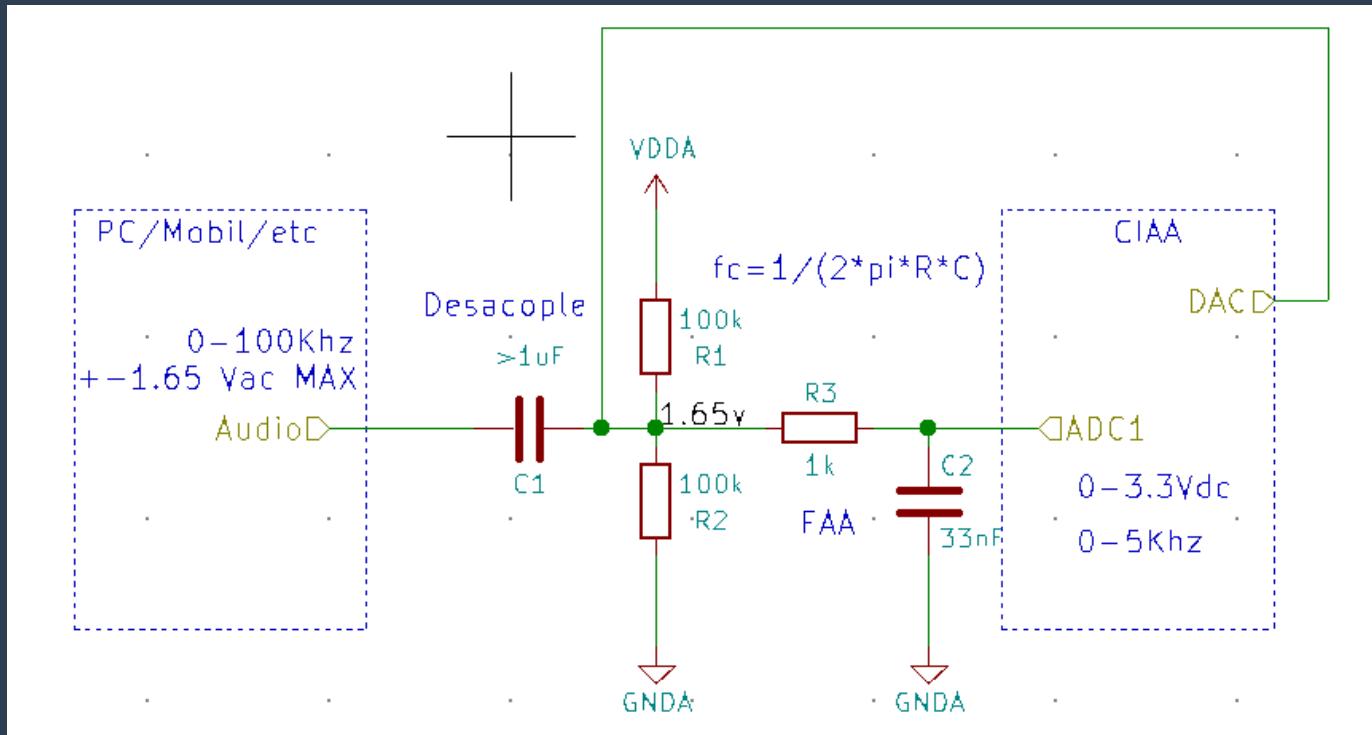
- Se puede mitigar el efecto de la atenuación con diferentes técnicas
 - Pre-equalización
 - Post-equalización
 - Interpolación agregando datos en el DAC
 - Aumentando F_s



Hardware

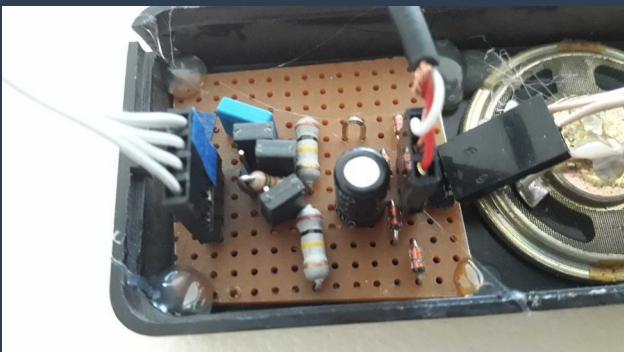
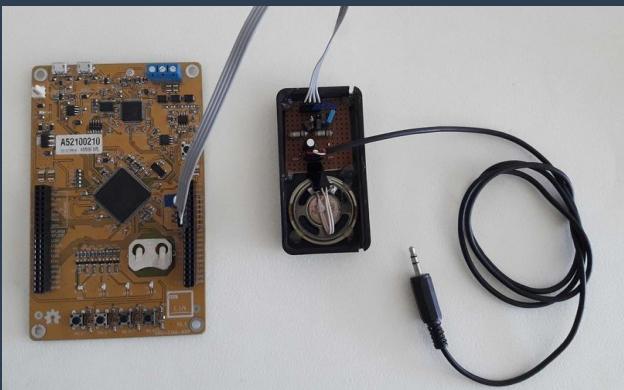
Hardware – CIAA <>> Python

- Armar el siguiente circuito
- ADC1 para samplear audio desde PC
- DAC para auto generar
- Audio out +1.65v



Hardware – CIAA pinout

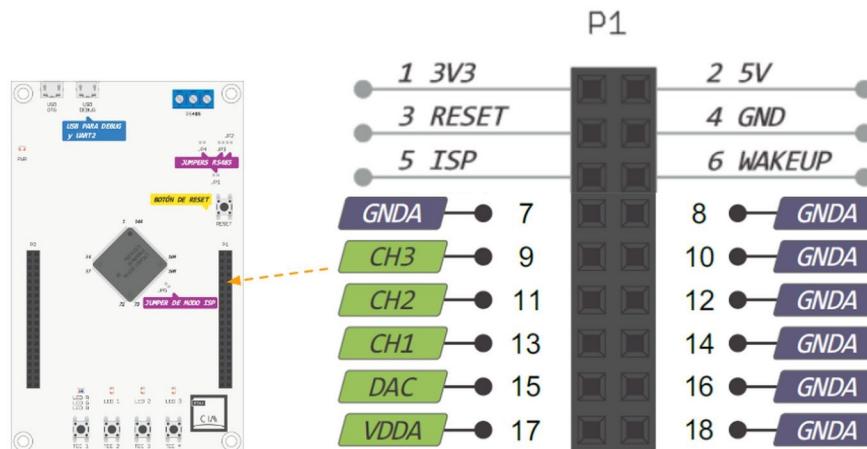
● Pinout CIAA



CIAA ADC y DAC en la EDU-CIAA-NXP

Mapeo de ADC y DAC en la biblioteca sAPI:

- 3 entradas analógicas nombradas CH1, CH2 y CH3 (ADC).
- 1 salida analógica nombrada DAC.



Administrativos

Evaluación – Trabajos prácticos

- **TP1 - 3 pts**

- Señales y sistemas LTI
- Teorema del sampleo
- Ruido de cuantización
- Generación y simulación en Python
- Adquisición y reconstrucción con la CIAA
- Sistema de números Q vs Float

- **TP2 - 3 pts**

- Transformada / antitransformada de Fourier.
- Convolución
- Filtrado
- En PC y CIAA / CMSIS-DSP

- **TP final - 4 pts**

Deberá incluir algún tipo de procesamiento en hardware. ej. DFT/IDFT, Convolución, etc.

Puede utilizar el ADC para samplear, DAC para reconstruir y/o canales de comunicación para adquirir datos previamente digitalizados

Presentación de 10 minutos.

- Deberá funcionar!

TP final – Proyectos - Ideas

- Sintetizador y/o reproductor de audio con el DAC
 - app note
- Afinador de guitarras (v2)
 - Esta desarrollado por Pablo Slavkin, pero se puede mejorar o utilizar de base para otro instrumento
- Análisis de vibraciones mecánicas
 - Diagnóstico mecánico basado en análisis de vibraciones
- Agregar eco, reverb, al audio
 - procesar audio y generar efectos
- Distorsionador de guitarra/instrumento
 - Efectos para guitarra y otro instrumento
- Busca llave con silbido (featured!)
 - Se trata de activar un sonido cuando se detecta un silbido humano.
 - Util para encontrar las llaves de la casa, la mascota, o el control remoto
- Aplicaciones con acelerómetro, magnetómetro, T+H
- Análisis de señales biomédicas

Calculo de nota final

- Calculo de la nota final según la siguiente ecuacion:
 - Las notas de los TP's pueden tener decimales y no se redondean.
 - La nota final se redondea hacia arriba, si es ≥ 0.5 pasa a 1, si es < 0.5 a cero
 - Ej:
 - $Tp1=2.4$
 - $Tp2=2.7$
 - $Tp3=3.5$
 - $\Rightarrow \text{Nota} = 8,6$ pasa a 9pts

Material de la materia y links afines

- Material de la materia en github:
 - git@github.com:pslavkin/psf_2021.git
- Encuesta anónima
 - <https://forms.gle/1j5dDTQ7qjVfRwYo8>
- Encuesta tecnologías:
 - <https://forms.gle/SZDVNX3i1cq3tioK6>
- Trabajos presentados en la edición 2020
 - <https://drive.google.com/drive/u/0/folders/1MXgBLfxebU1ZPLc7-lM0x0hq4MJHs2HJ>
- Video presentación de afinador de guitarras 2019:
 - [clases/ciaa_guitar_tunner](#)

Bibliografía

- ***The Scientist and Engineer's Guide to Digital Signal Processing***
 - Steven W. Smith.
- ***Think DSP - Digital Signal Processing in Python***
 - Allen B. Downey
- **Understanding digital signal processing.**
 - Richard Lyons.
- **Digital Processing of Random Signals: Theory and Methods.**
Digital Processing of Random Signals: Theory and Methods.
 - Boaz Porat
- **Think Python, 2nd Edition, - How to Think Like a Computer Scientist**
 - Allen B. Downey
- **Digital Signal Processing. A practical approach.**
 - Emmanuel C Ifeachor, Barrie W Jervis
- **Introduction to Python Programming.**
 - NW. Taylor, Francis Group, LLC.
- **Illustrated guide to python 3**
 - Matt Harrison

- **Numpy guide**
 - <https://numpy.org/doc/stable/user/index.html>
- **Anaconda**
 - <https://www.anaconda.com/products/individual>
- **CMSIS-DSP**
 - https://arm-software.github.io/CMSIS_5/DSP/html/index.html
- **Q numbers**
 - Yates, Randy, "Fixed-Point Arithmetic: An Introduction"
 - <http://www.digitalsignallabs.com/fp.pdf>
- **Generador de coordenadas XY a partir de SVG**
 - <https://spotify.github.io/coordinator/>
- **Explicacion intuitiva de convolucion**
 - <https://betterexplained.com/articles/intuitive-convolution/>
- **Pyfdax install**
 - <https://github.com/chipmuenk/pyfda>