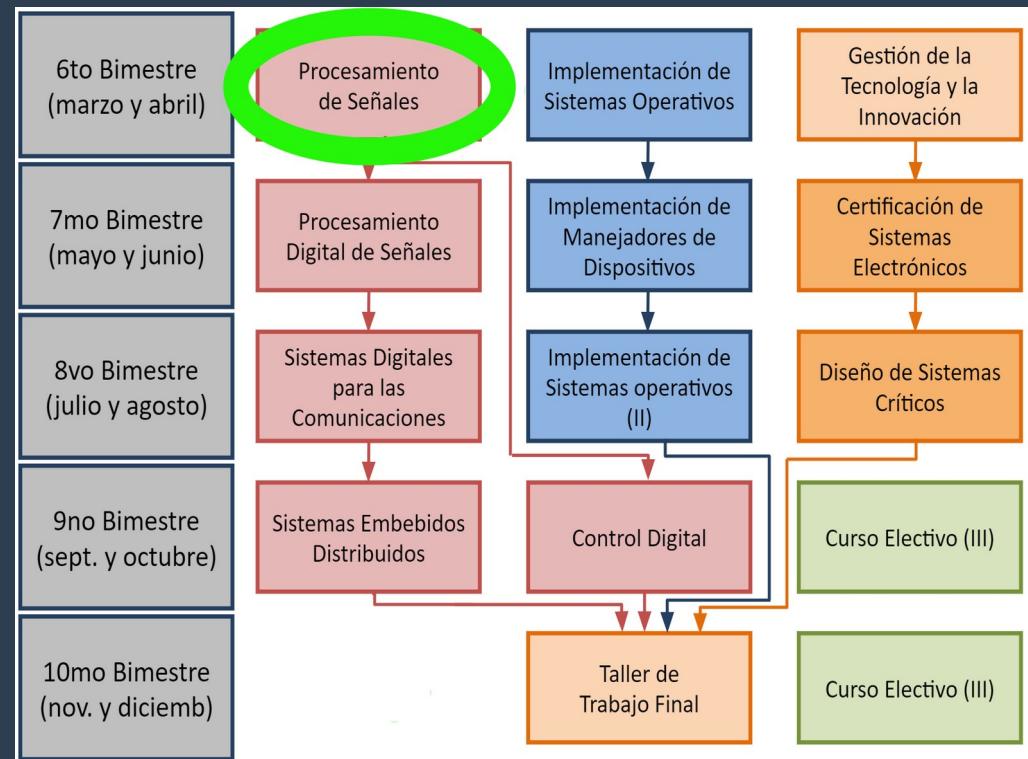


Procesamiento de señales. Fundamentos

Docentes

- 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, sistemas, adquisición y visualización con CIAA y Python
- 2 Reconstrucción, Nyquist, Números Q y generación con Python
- 3 Euler + Fourier + DFT
- 4 IDFT, FFT con numpy
- 5 Respuesta al impulso | Convolución
- 6 Filtrado | FIR
- 7 Repaso para TP final y tips & tricks
- 8 Temas varios y presentación de TP's



Procesamiento de señales. Fundamentos

Objetivos generales

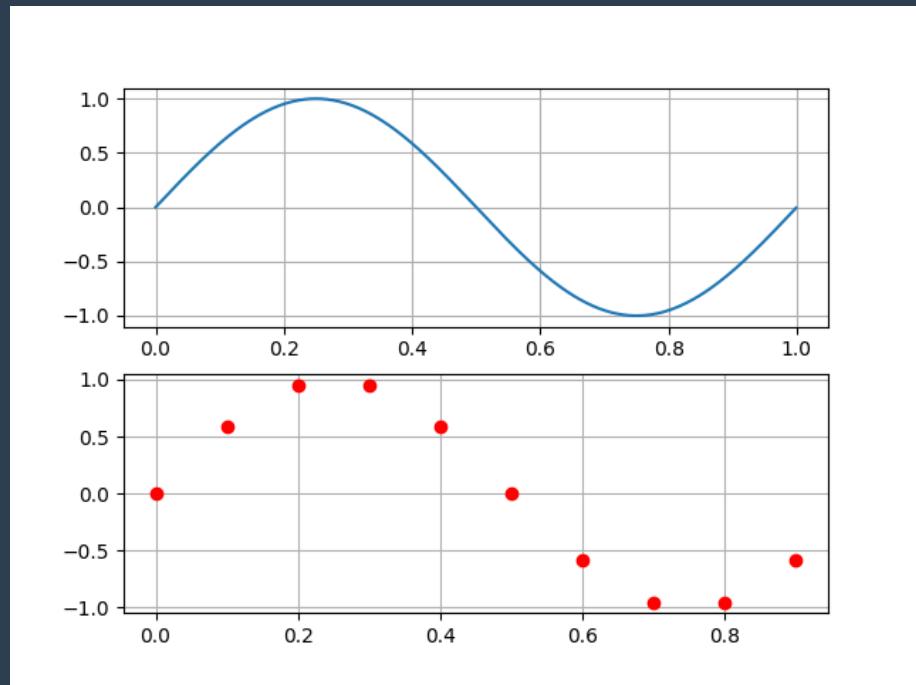
- 1 Poder dimensionar una cadena de adquisición, frecuencias, uC y definir la arquitectura
- 2 Entender los fundamentos de pasar del dominio del tiempo al dominio de la frecuencia y poder instanciarlos en un sistema embebido
- 3 Poder implementar algoritmos simples de procesamiento utilizando python/mlab/octave pero principalmente utilizando CMSIS-DSP o similar en un sistema embebido
- 4 Entender y poder disenar un filtro digital FIR



Procesamiento de señales. Fundamentos

Clase 1 – Señales y sistemas

- Señales y sistemas. LTI
- Adquisición
 - Fenómeno de aliasing
- Cuantización
 - Dithering
 - Sobre muestreo
- Preparación de hardware
- CIAA<>Python
- 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
- Los filtros son muy superiores



● Analógico

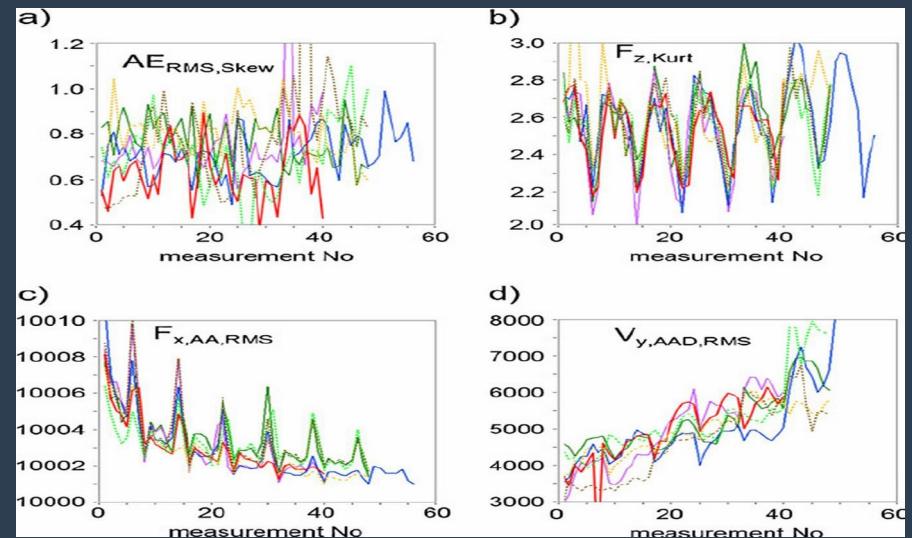
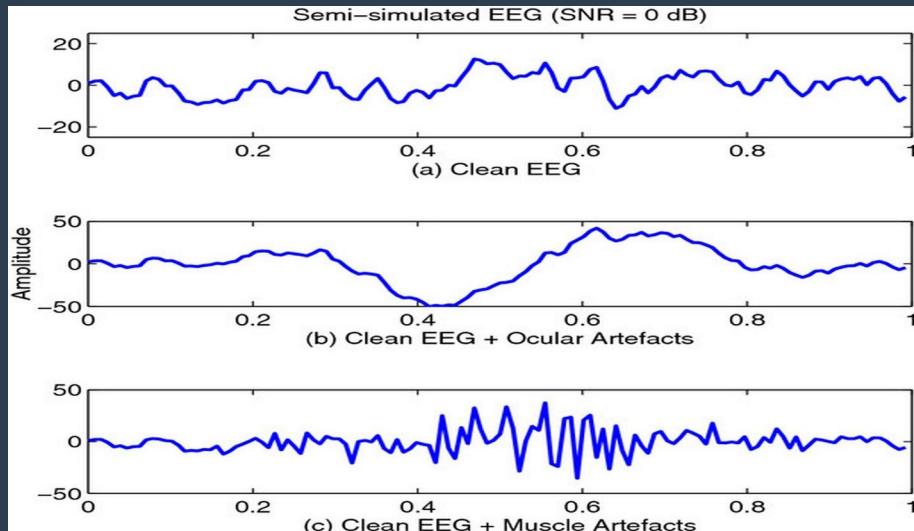
- Gran rango dinámico de entrada Ej.: 1mv to 20v
- Alto ancho de banda Ej. 0.001Hz to 10Mhz
- Alta potencia Ej: Amplificadores, RF
- Baja latencia Ej. Fibra óptica, RF
- Distorsiones no lineales en audio artístico



Señales

● Señales

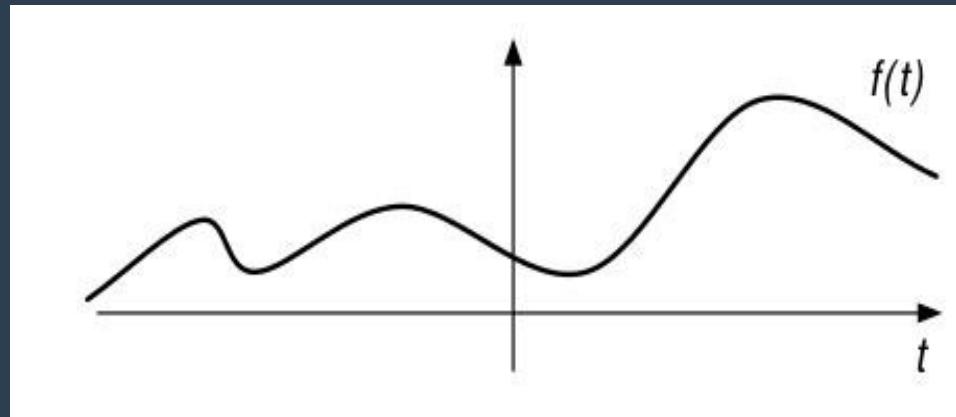
- Una señal, en función de una o más variables, puede definirse como un cambio observable en una entidad cuantificable
- Ej: Sonido, imágenes, video, biológicas.



Tipos de señales

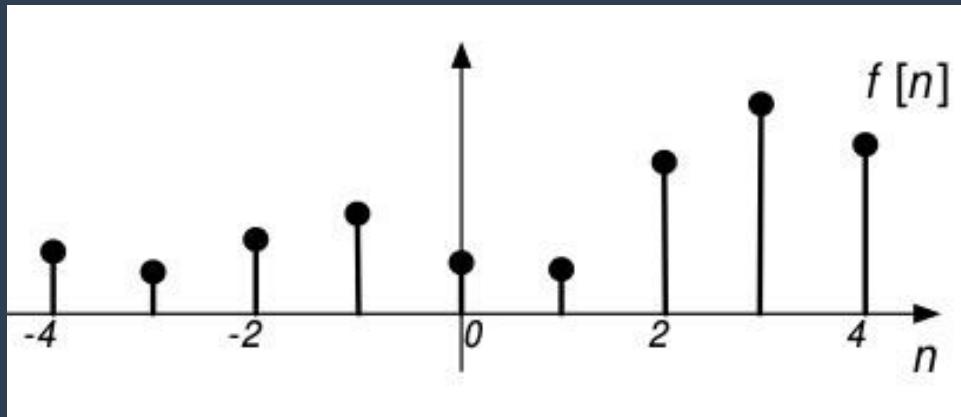
● Tiempo continuo

- 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??



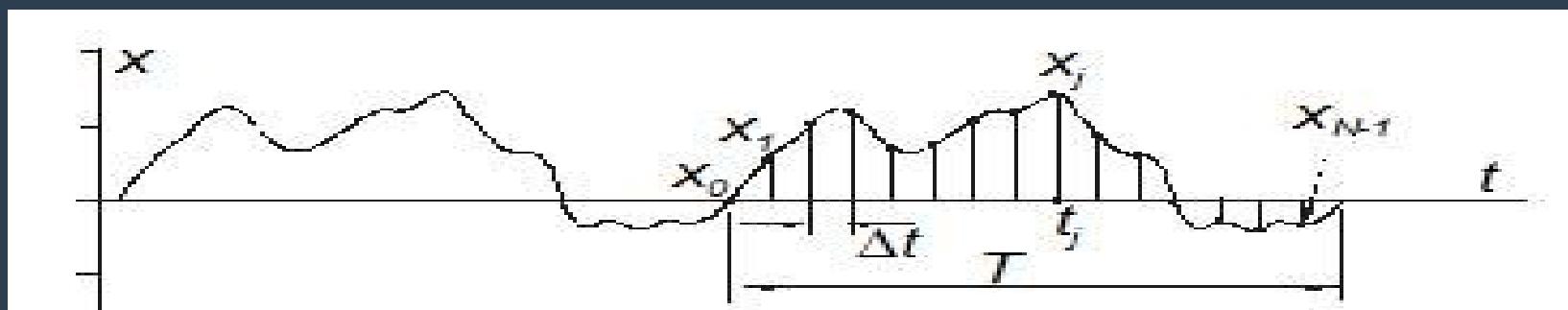
Tipos de señales - Periodicidad

● 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
- $f_0 = 1 / (\Delta t * N_0)$ es la frecuencia fundamental de $x[n]$



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?

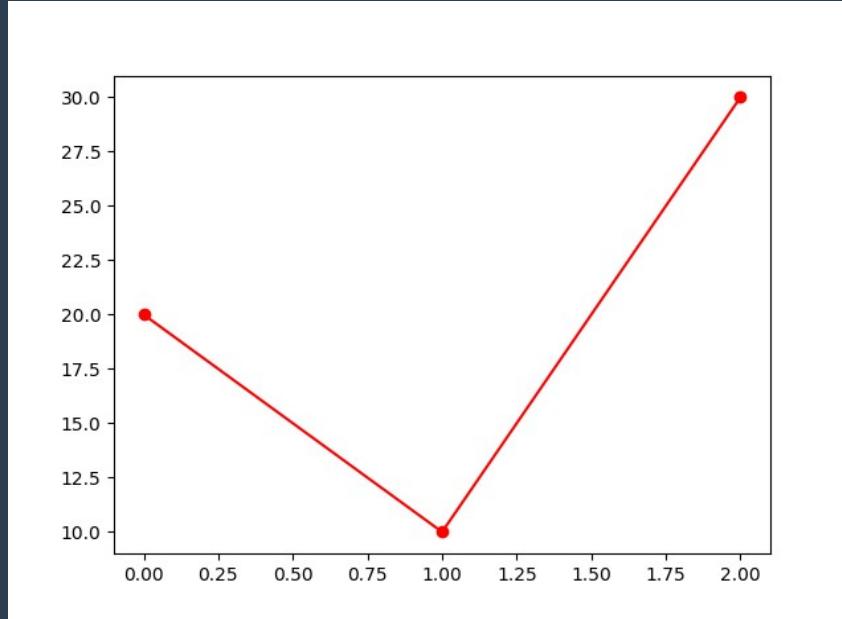
- $S= ?$
- $T= ?$

Ver código: senales_periodicas.py

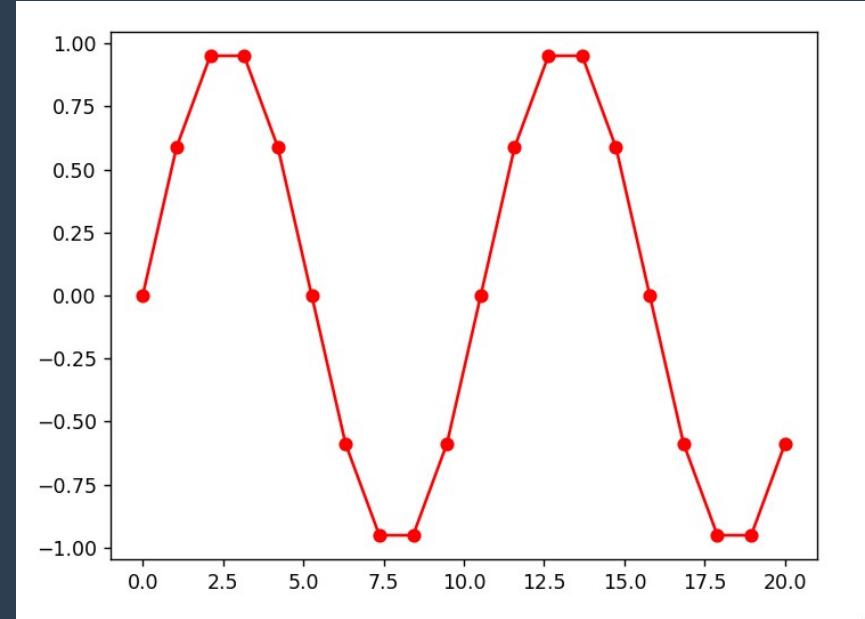
Señales. Grafica con matplotlib

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

```
s = [np.sin(2*np.pi*3*t*1/30) for t in range(20)]  
  
t = np.linspace (0 ,len(s) ,len(s))  
plt.plot(t,s,'ro-')  
plt.show()
```



Ver código: senales_python.py



Ver código: senales_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: $0 < E_x < +\infty$.

- *Unidad: joules*

- *De potencia finita (discreta)*

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

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

- *Unidad: watts*

Tipos de señales – Potencia finita

- *Cálculo de potencia del sin(x)*

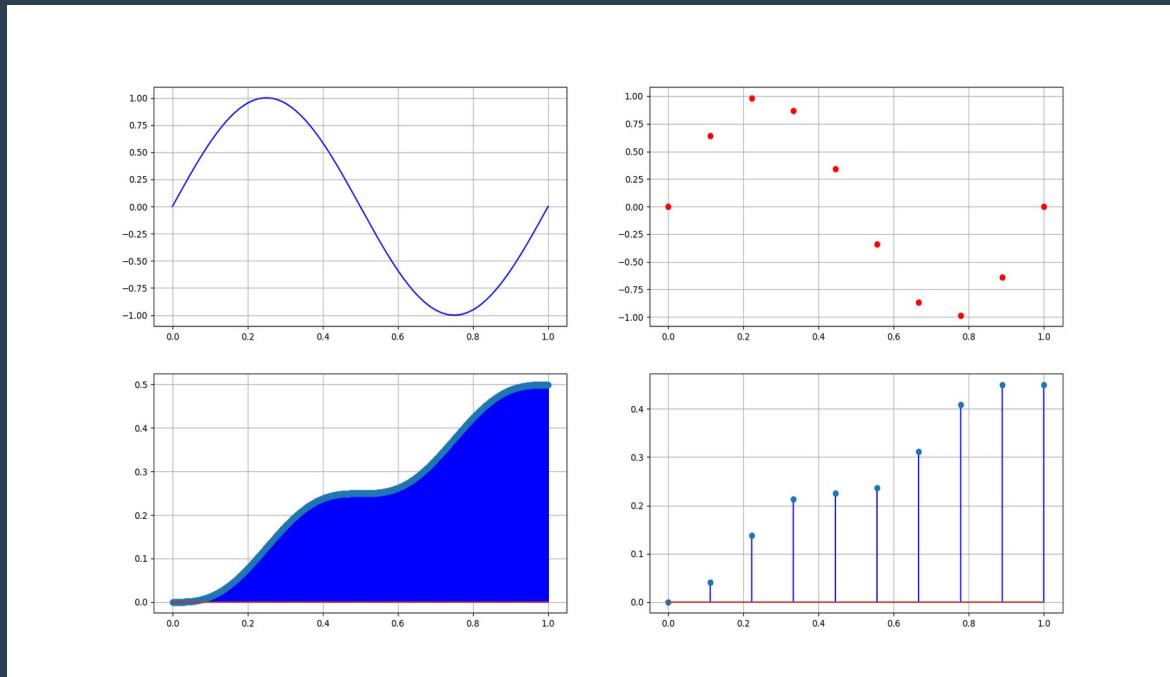
$$P = \frac{1}{2\pi-0} \int_0^{2\pi} |\sin(t)|^2 dt$$

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

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

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

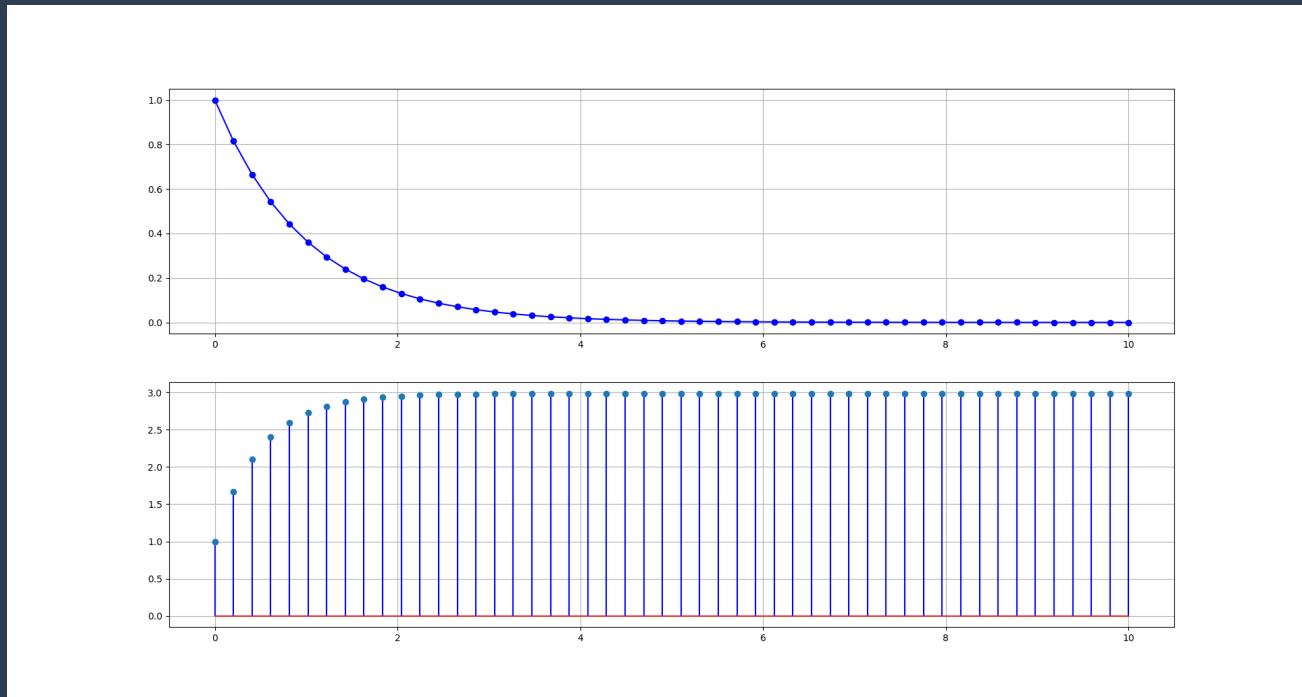
- *Cálculo de potencia de Sin(n) con Python*
- *Notar que la energía tiende a infinito*



Ver código: power.py

Tipos de señales – Energía finita

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



- Ver código: `energy.py`

Sistemas

- *Definición*

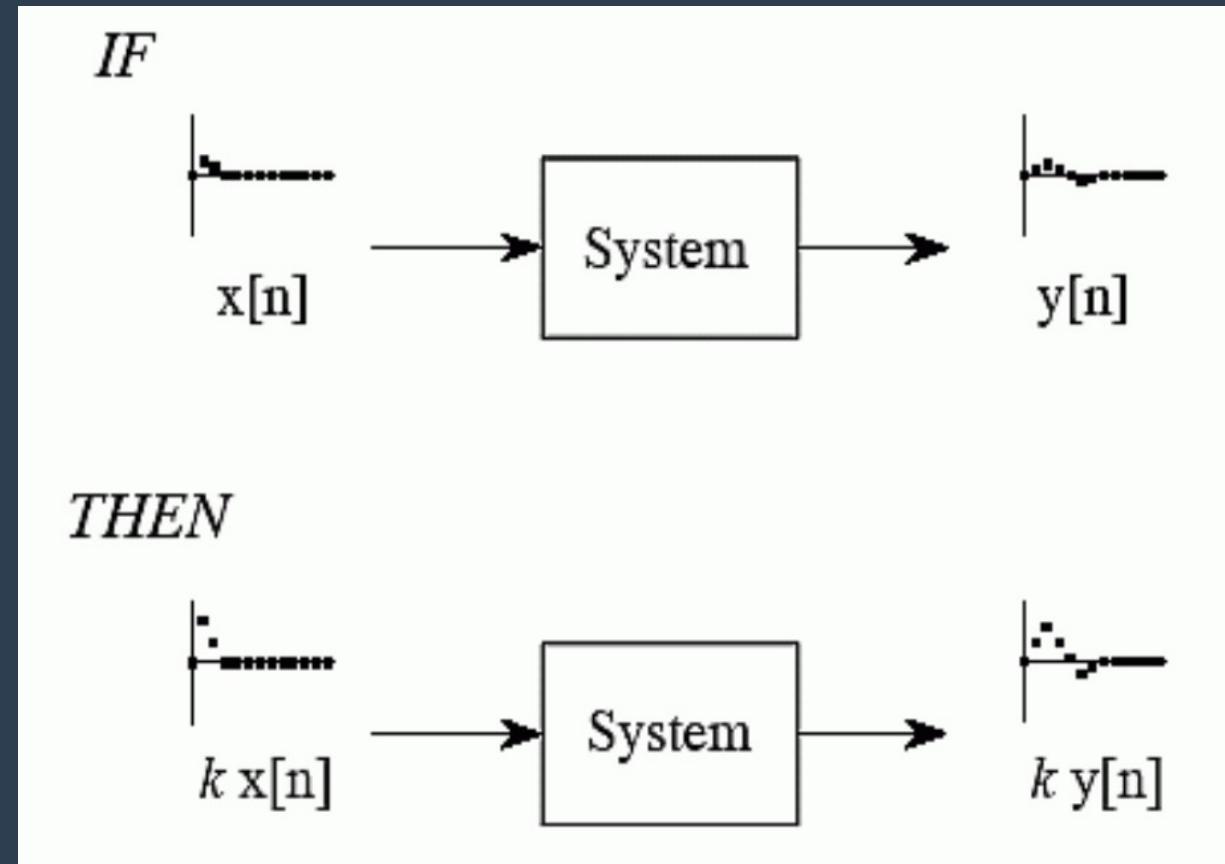
Un sistema es cualquier conjunto físico de componentes que actúan en una señal, tomando una o más señales de entrada, y produciendo una o más señales de salida



Sistemas homogéneos

Escalado

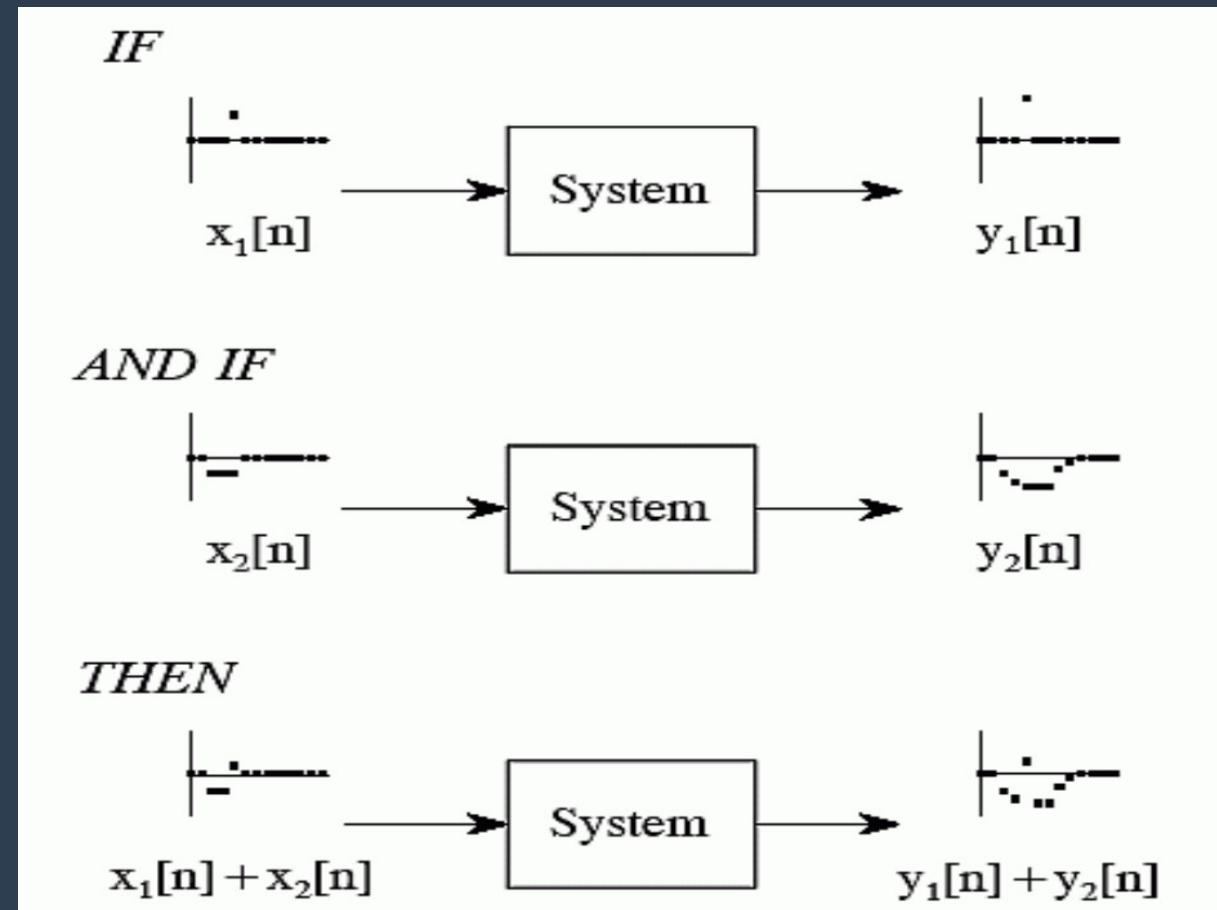
- Un cambio en la entrada repercute en la salida en la misma proporción



Sistemas aditivos

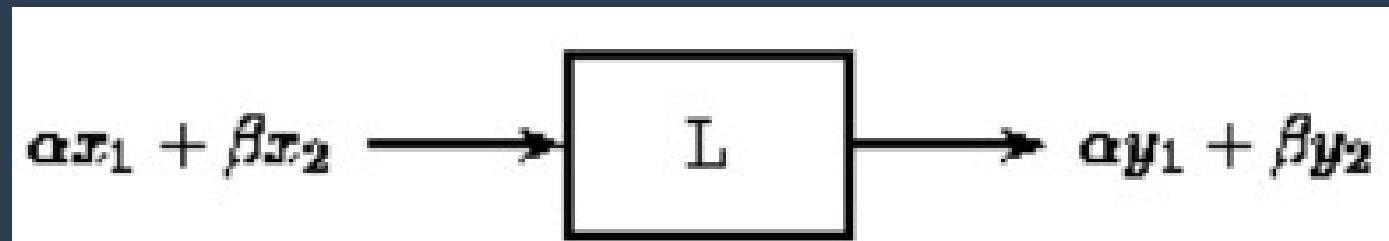
Adición:

- 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:

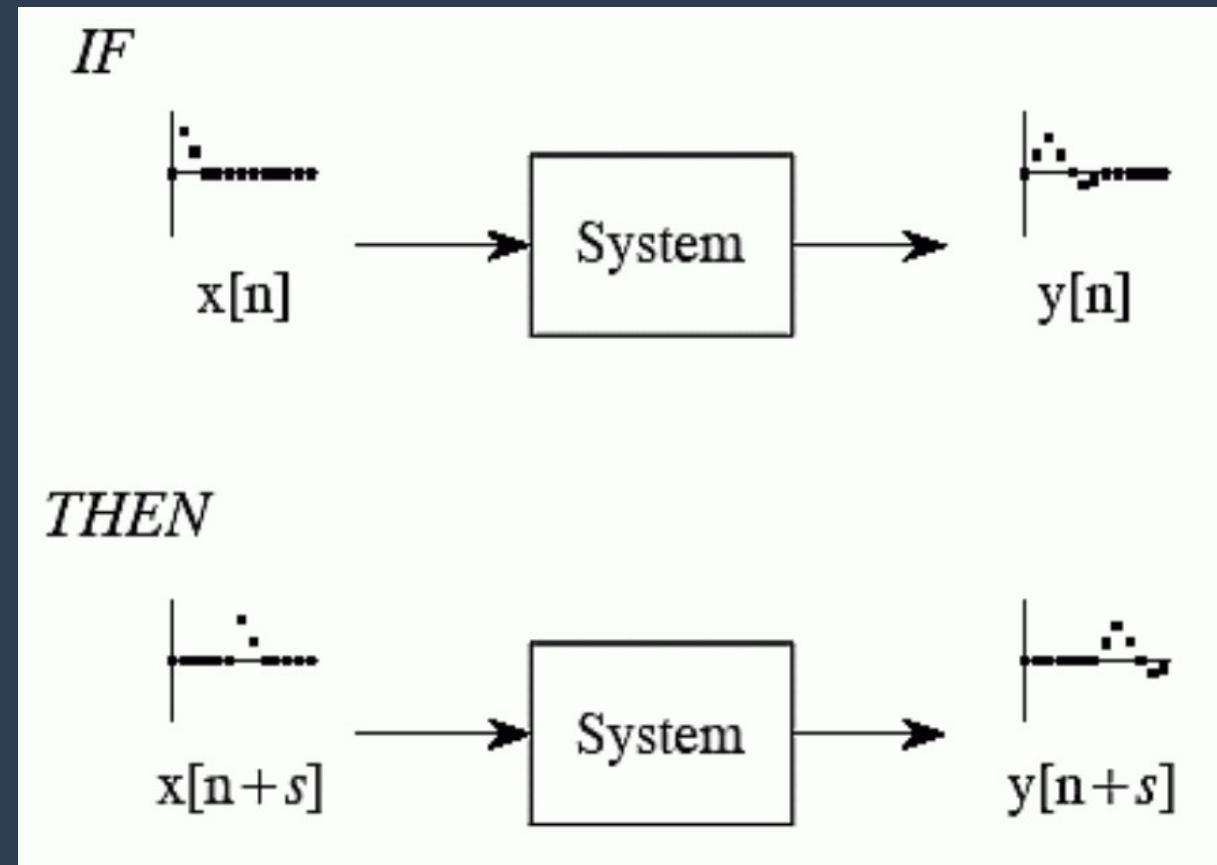
- $y(t) = e^{x(t)}$ => no lineal
- $y(t) = 1/2 x(t)$ => lineal

Sistemas de tiempo invariante

- Si la entrada se demora, la salida también

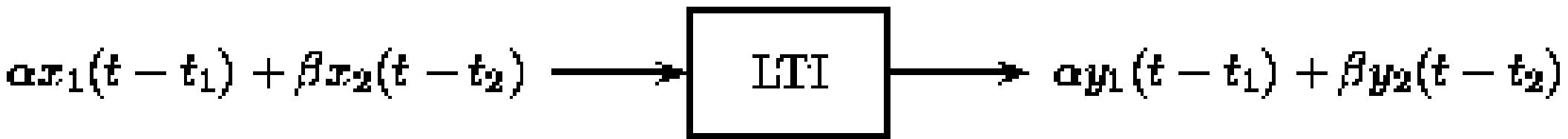
Ej:

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

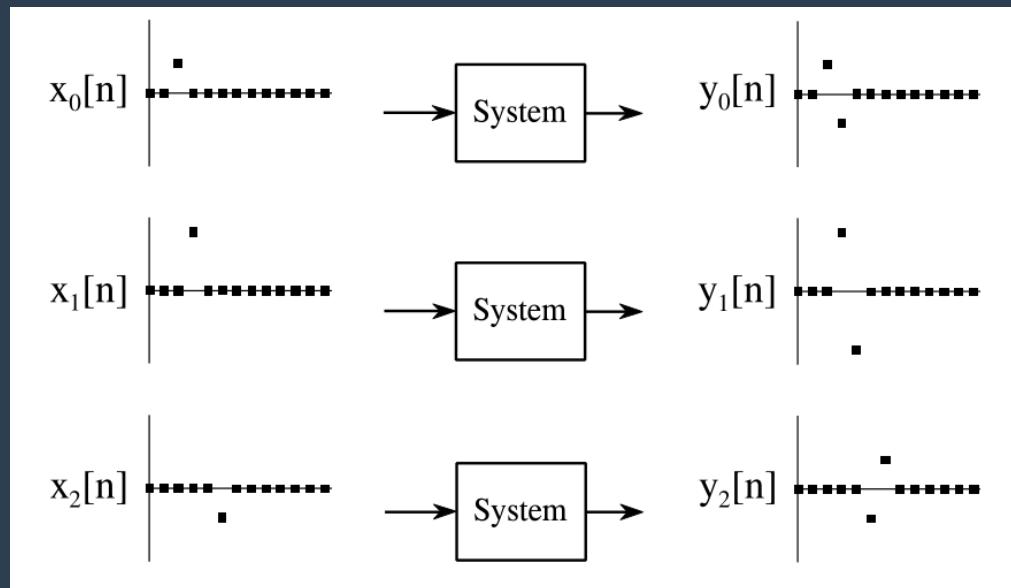
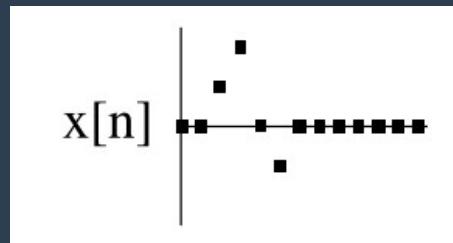


En este curso, solo estudiaremos sistemas lineales invariantes en el tiempo.

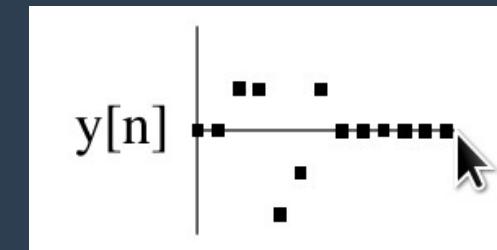
Sistemas LTI – Descomposición

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

Descomposición ->



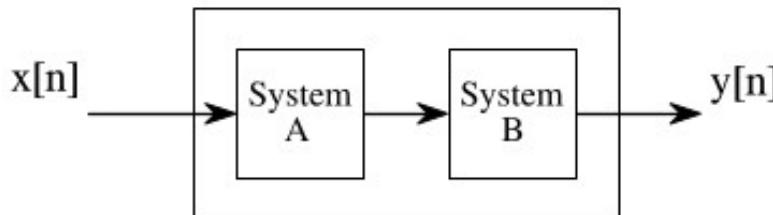
Síntesis ->



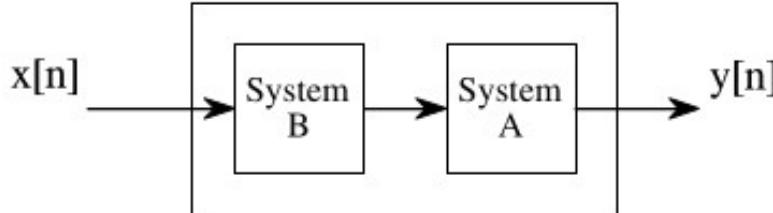
Sistemas LTI - Conmutación

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

IF



THEN



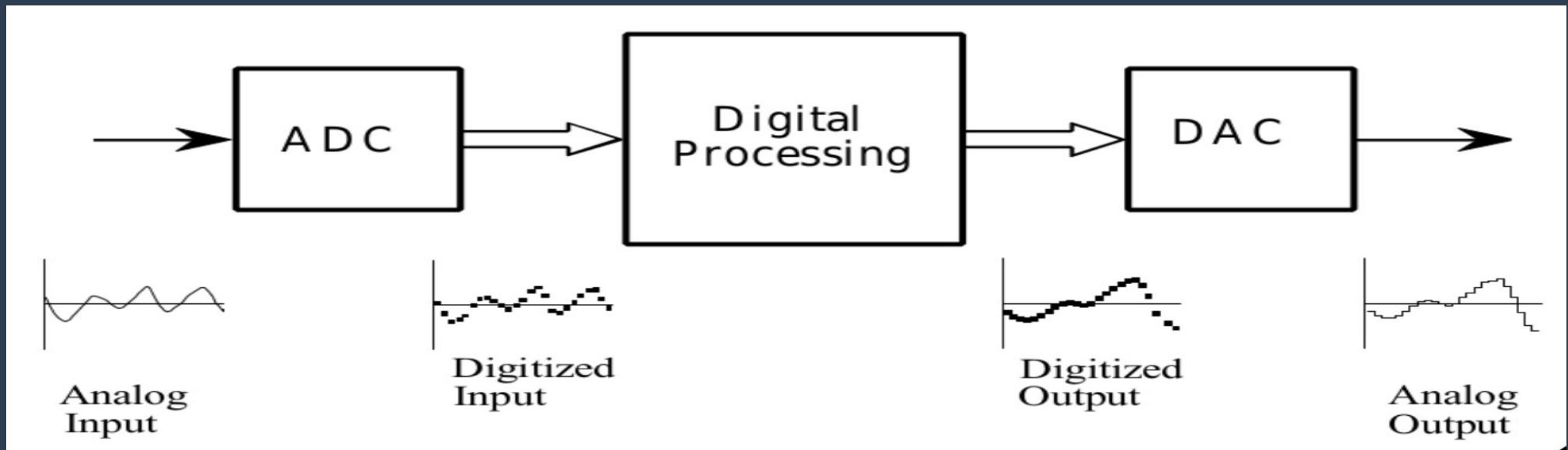
Sistemas LTI - Características

- Fidelidad senoidal
 - En todo sistema LTI para una entrada senoidal $a \times \text{Hz}$ la salida es **siempre** senoidal de $x \text{ Hz}$.
- 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/8YnhmWVu5K21mpYc8>



Efecto de aliasing en un disco giratorio

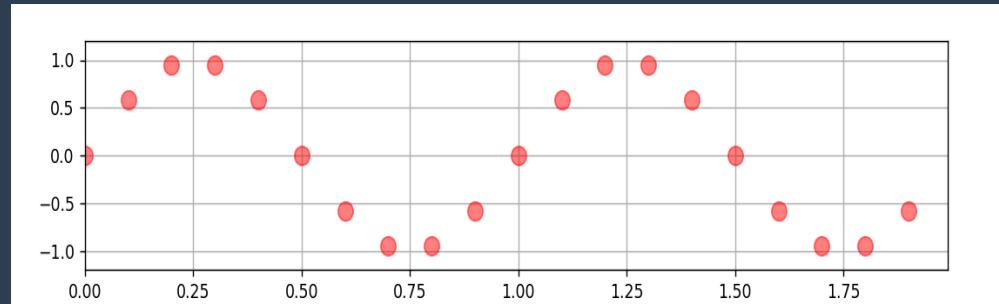
<https://youtu.be/-XiWEq8MIKc>



Efecto de aliasing en Python

- $F_s=10 | N=20$
- A simple vista parece una senoidal de 1Hz
- Pero podria esconder una señal multiplo de F_s , como 10Hz 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 $F>F_s/2$

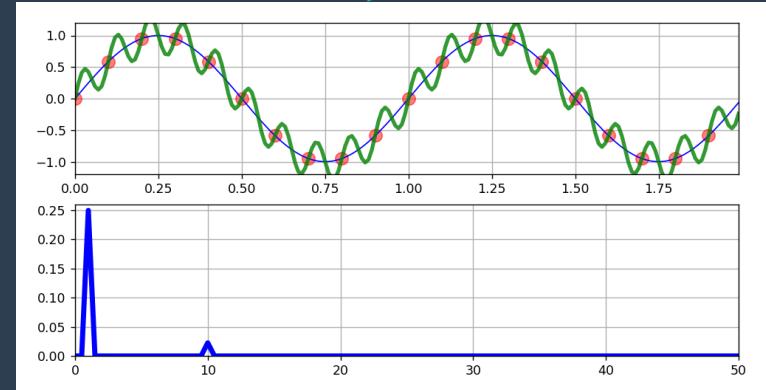
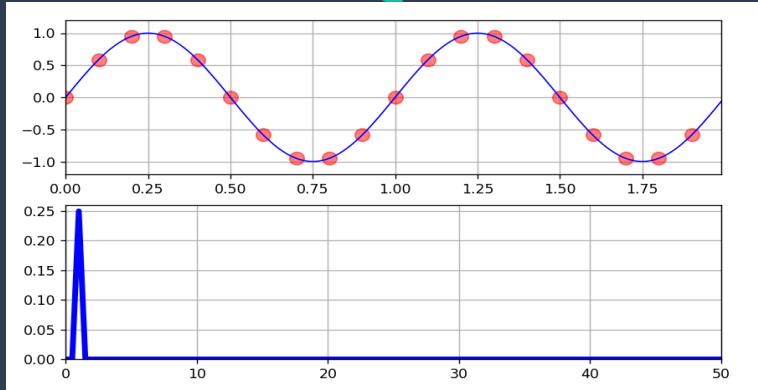
• *Que señal esconde esta secuencia de puntos?*



Ver código: teorema_sampleo.py

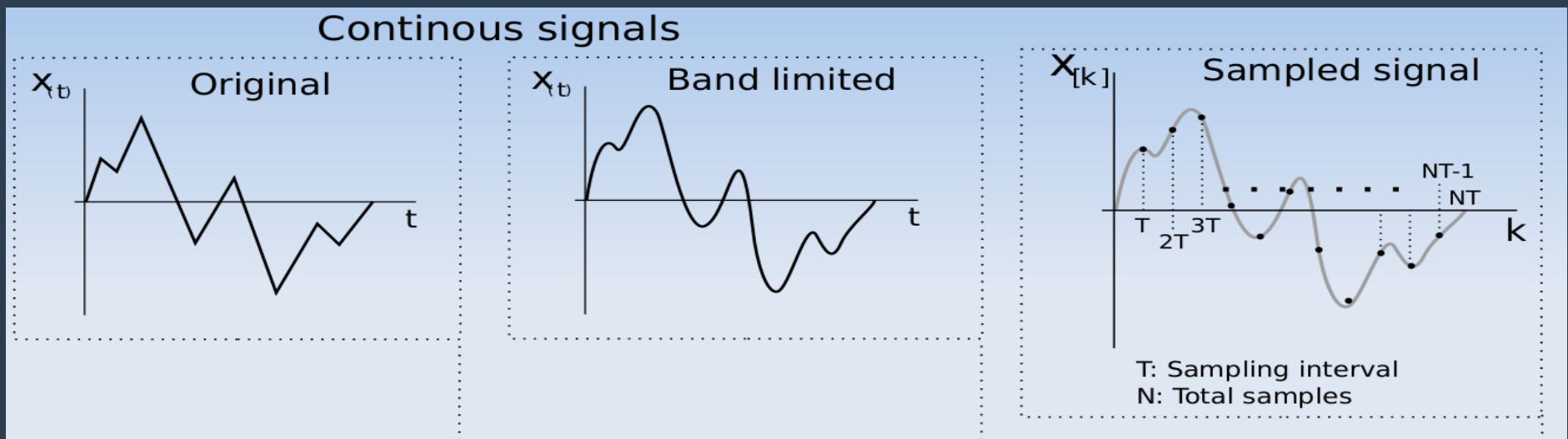
1Hz ?

1Hz + 10Hz ?



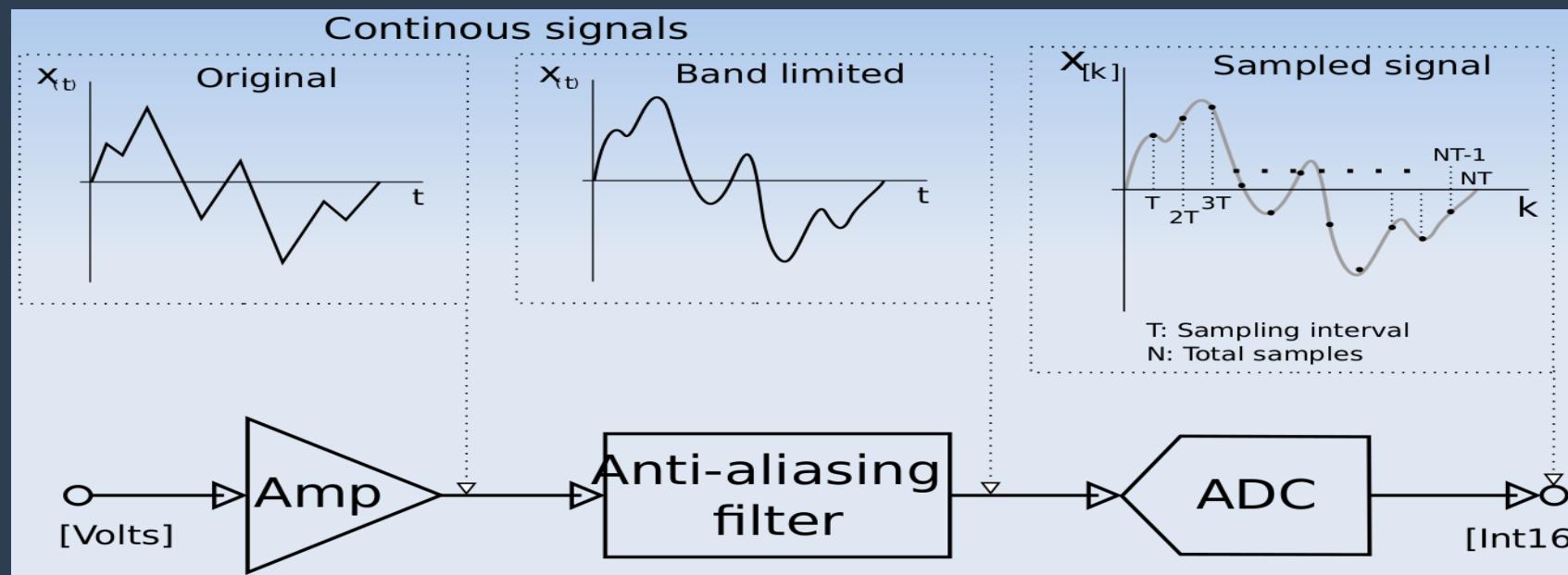
Filtro anti alias (FAA) - Pasa bajos

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



Digitización - Filtro anti alias

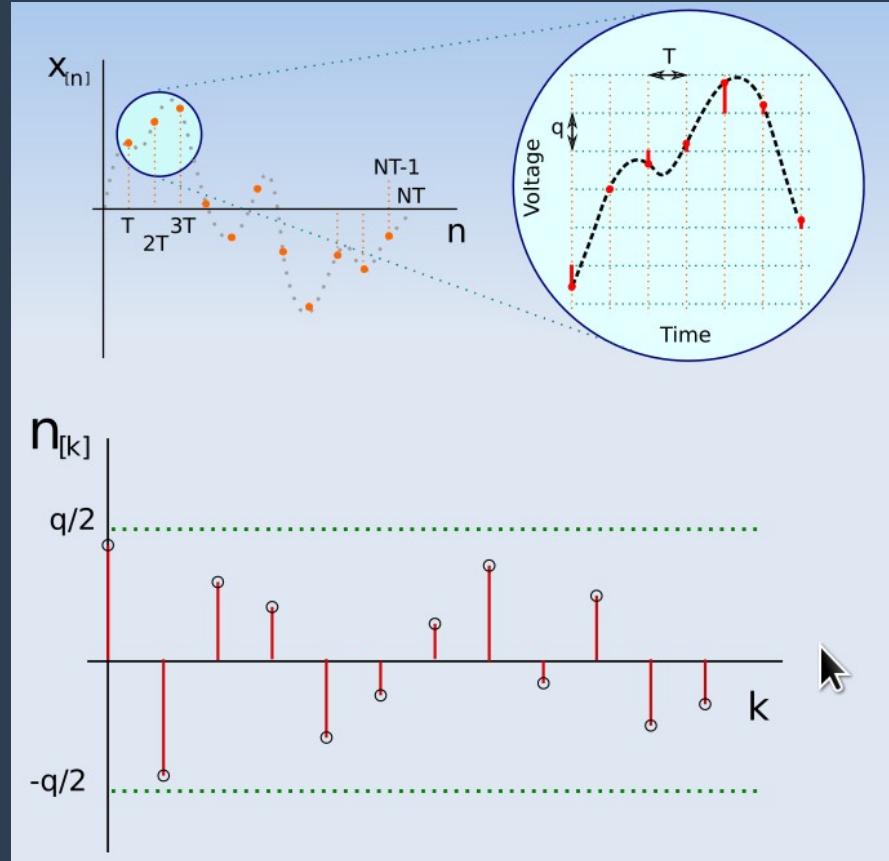
- Secuencia de adquisición utilizando un filtro anti alias
- Se podría poner el bloque de antialiasing luego del ADC en el dominio digital ? Porque?
- Que estrategias puedo usar para simplificar el FAA analogico?



Cuantización

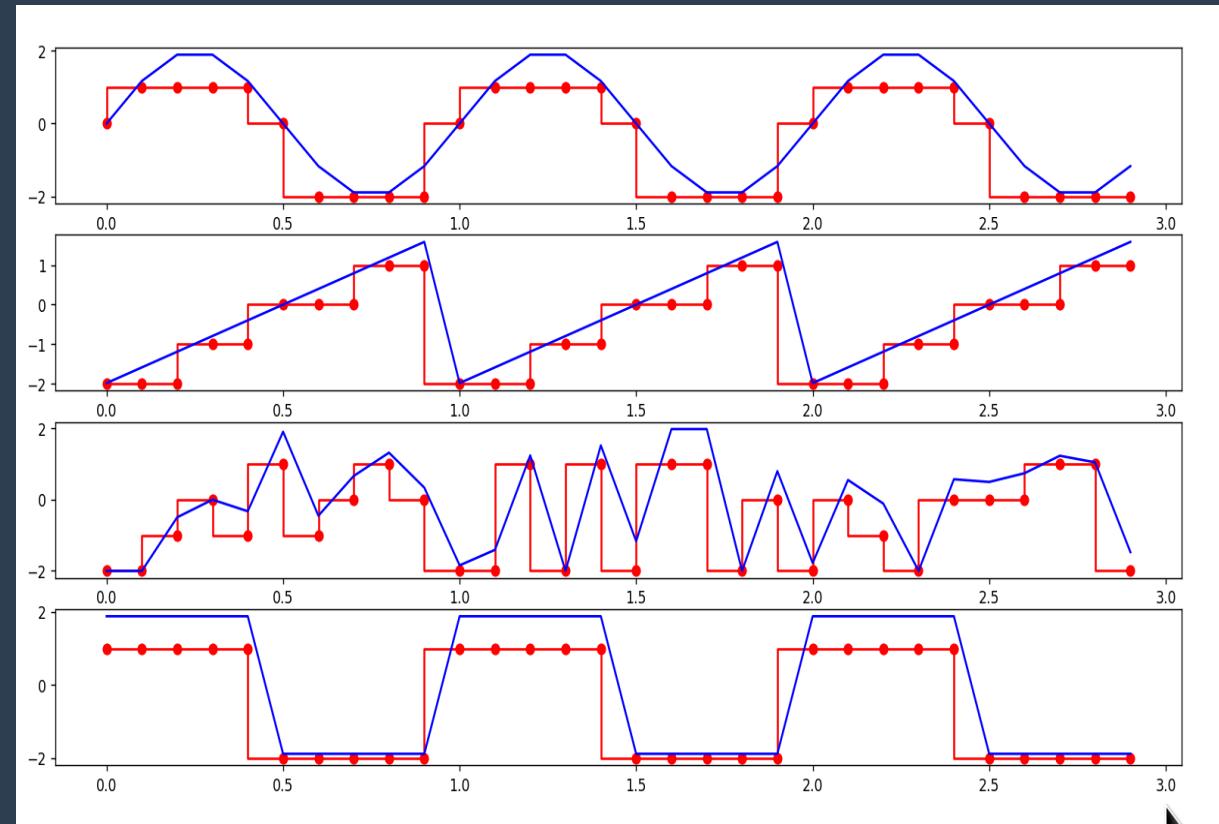
Error en la adquisición

- 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 en Python

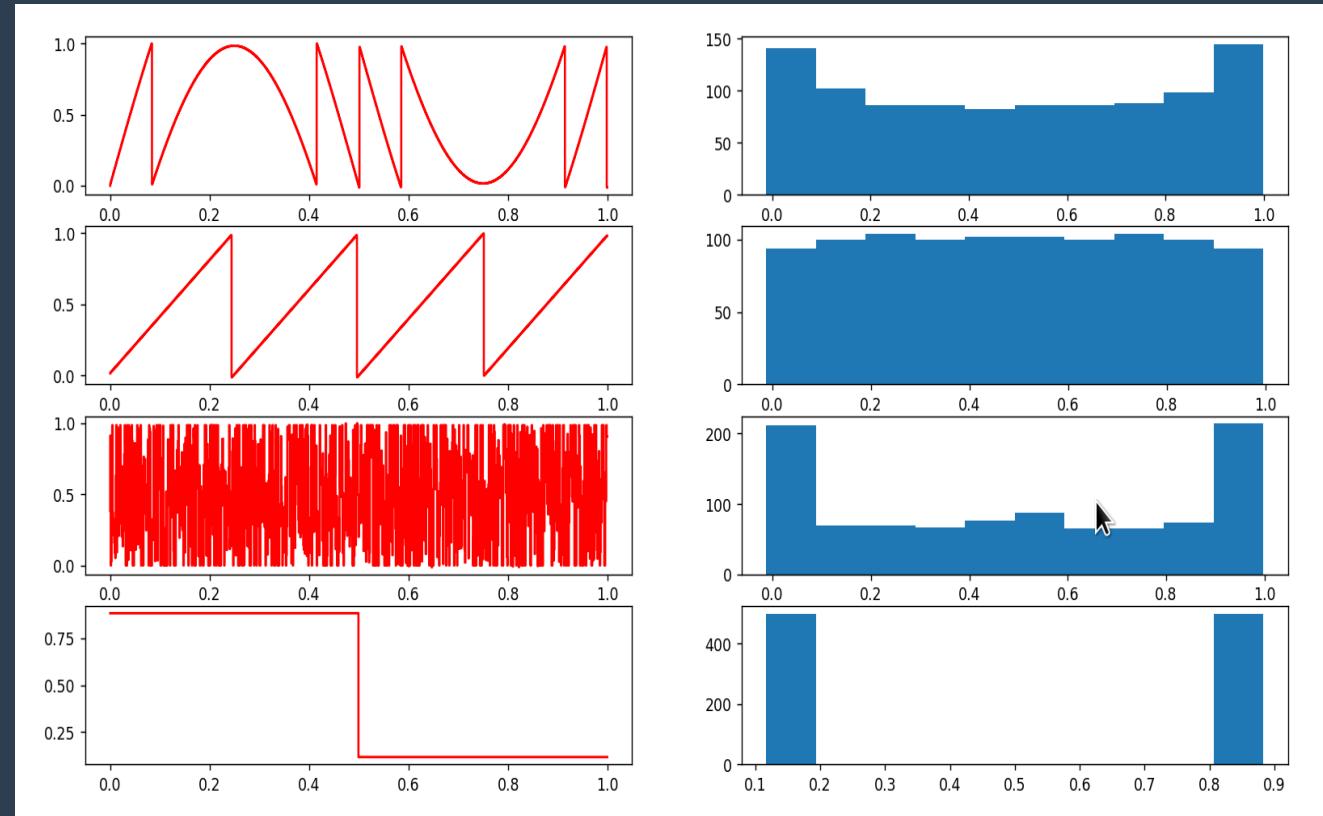
- En azul la señal sampleada con precisión 'infinita'
- En rojo la señal escalonada con precisión de 2 bits
- Ej: Calcule el error máximo para una señal de 1V, ADC de 10b entre 0-1V:
 $e=1/1024=0,000976562$ volts



Ver código: ruido_cuantizacion.py

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

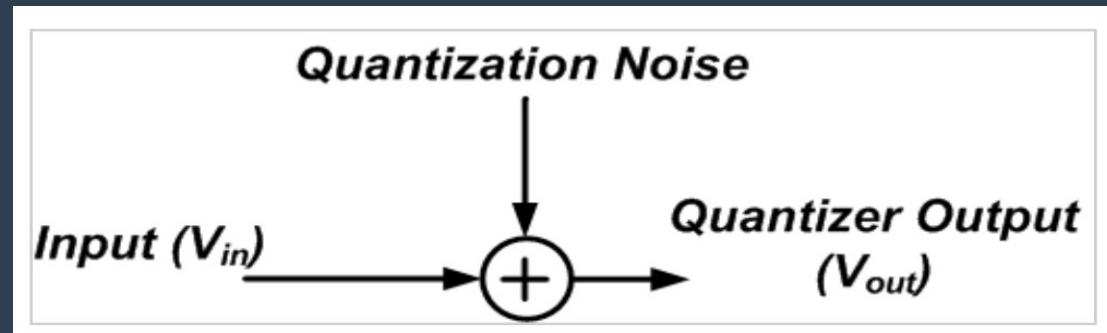


Ver código: [ruido_histograma.py](#)

Ruido de cuantización

En el caso de que se cumplan las siguientes premisas:

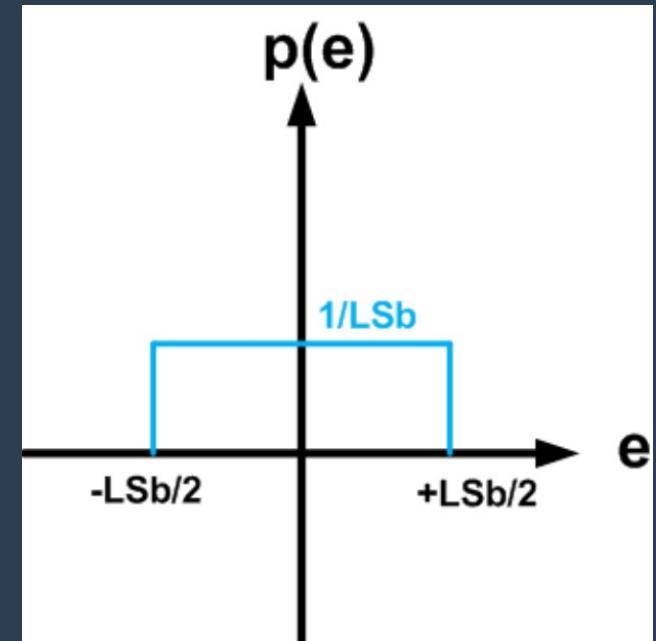
- La señal de entrada varia lo suficiente para que los diferentes niveles de cuantización tengan igual probabilidad de ocurrencia
- El error de cuantización NO esta correlacionado con la entrada
- El cuantizador cuanta con un numero relativamente largo de niveles
- Los niveles de cuantización son uniformes
- 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 máximo al cuadrado, NO es el numero 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

- Cálculo de la relación entre una señal senoidal de amplitud máxima y el ruido de cuantización en función de la cantidad de bits N
- Tip: $\cos 2x = \sin^2 x - \cos^2 x$

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

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

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

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

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

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

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

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

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

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

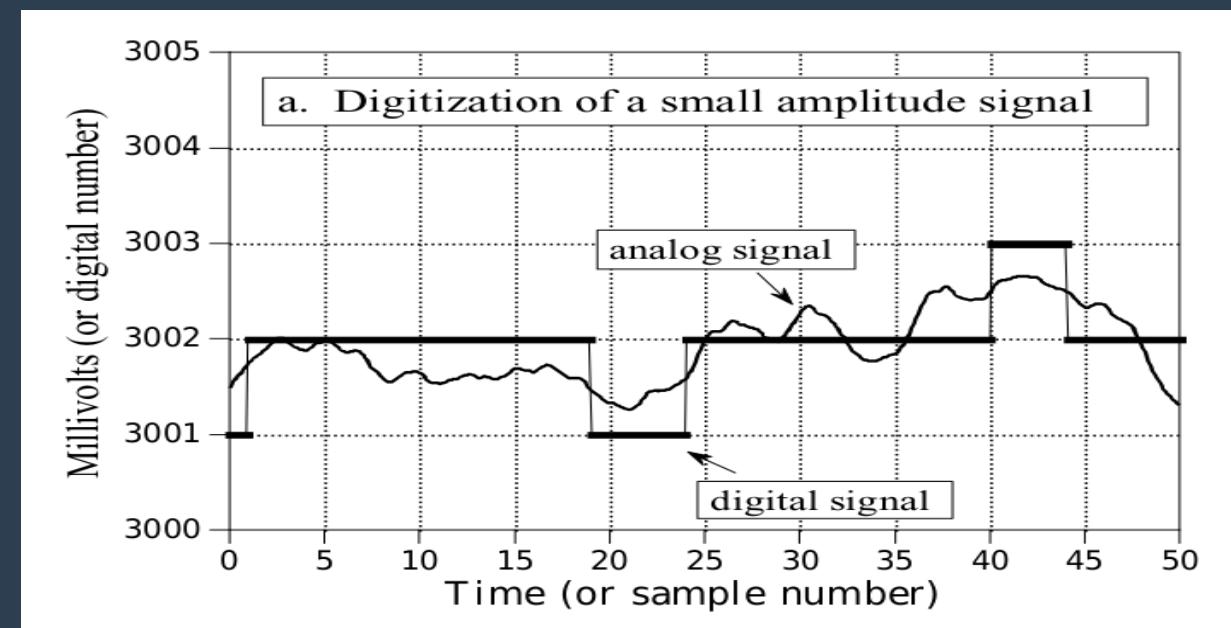
$$SNR = 1,76 + 6,02 * N$$

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

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

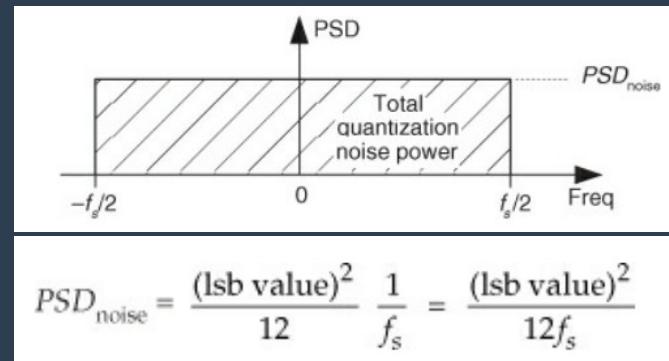
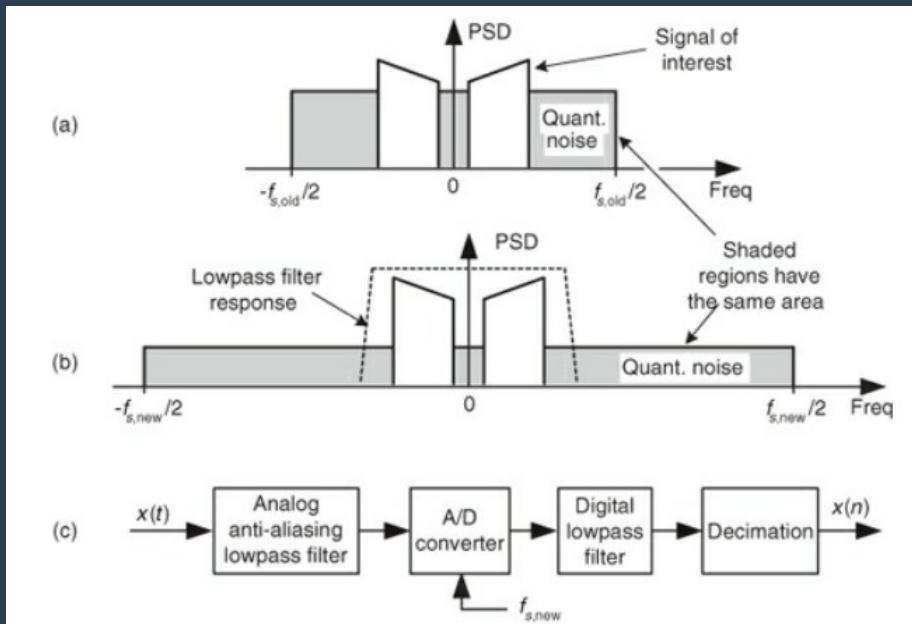
Dithering

- Técnica de agregado de ruido antes del ADC para prevenir que señales con poca variación sean sampleadas siempre con el mismo valor
 - Se suele utilizar una señal triangular
 - app note



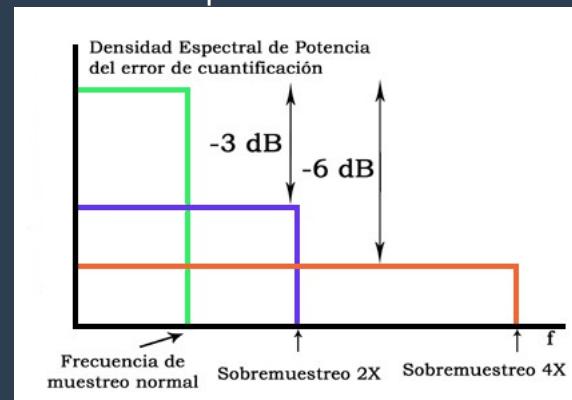
Densidad espectral de potencia de ruido

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



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

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

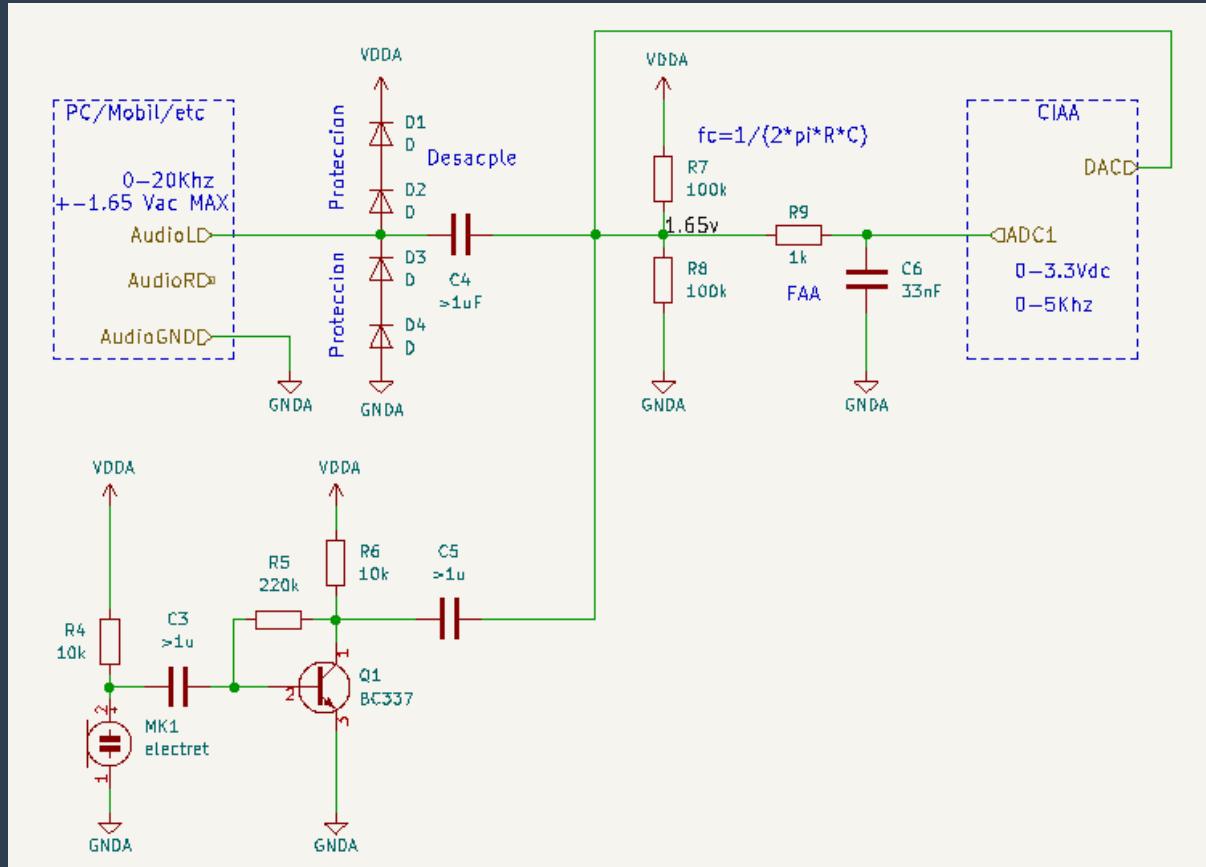


Hardware

Hardware – CIAA <> Python

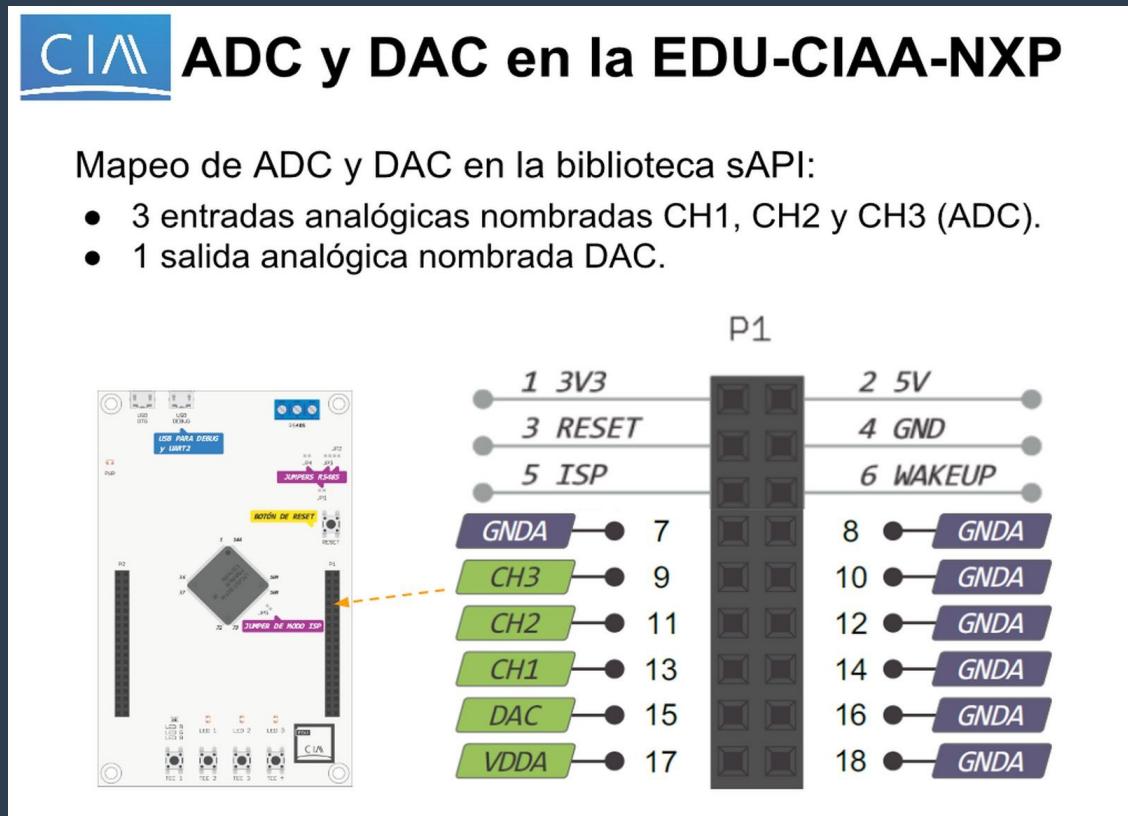
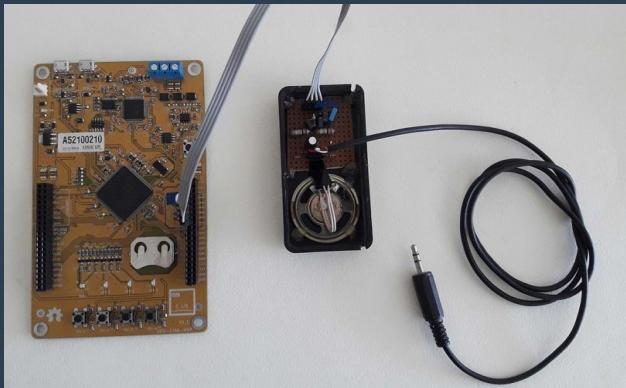
● Armar el circuito

- ADC1 para samplear (buscar Fs max y cantidad de bits)
- DAC para sintetizar (buscar Fs max y cantidad de bits)
- Micrófono
- Audio L como output o como input
- Protección de audio out en +- 1.65v
- Se puede agregar por comodidad un jumper al DAC y/o mic



Hardware – CIAA pinout

- Pinout CIAA y circuito de ejemplo armado en tarjeta multipropósito



Visualización

Superloop de adquisición

- Se propone capturar con el ADC
- Enviar los datos por la UART
- Visualizar con matplotlib
- Se envía un header y luego los samples uno a uno en int16_t
- Ver código: psf.c

```
struct header_struct {  
    char mark[8];  
    uint32_t id;  
    uint16_t length;  
    uint16_t fs ;  
} header={"*header*",0,256,20000};
```

```
00000030: 3201 1601 8e00 cbff 1bff bffe dcfe 65ff 2.....e.  
00000040: 2700 d700 3201 1401 8a00 c9ff 19ff c0fe '...2.....  
00000050: dcfe 68ff 2b00 da00 3301 1301 8700 c6ff ..h.+..3.....  
00000060: 16ff befe dffe 6aff 2e00 db00 3201 1101 ..j....2.....  
00000070: 8600 c2ff 15ff befe e0fe 6fff 3200 df00 .....o.2.....  
00000080: 2a68 6561 6465 722a 643e 0000 8000 1027 *header*d>....  
00000090: 656e 642a a300 e8ff 30ff c5fe d1fe 4dff end*....0....M.  
000000a0: 0d00 c300 2d01 2001 a300 e4ff 2dff c5fe ....-.....-  
000000b0: d1fe 4fff 0f00 c500 2d01 1f01 a100 e1ff ..0.....-.....  
000000c0: 2bff c3fe d3fe 53ff 1200 c800 2e01 1e01 +.....S.....  
000000d0: 9d00 deff 28ff c3fe d4fe 55ff 1400 ca00 ....(.....U.....  
000000e0: 3001 1d01 9b00 daff 27ff c2fe d5fe 57ff 0.....'.....W.  
000000f0: 1800 cc00 2f01 1b01 9800 d8ff 23ff c2fe ...../.....#.....  
00000100: d7fe 5aff 1c00 cf00 3101 1b01 9500 d4ff ..Z.....1.....  
00000110: 21ff c1fe d7fe 5dff 1f00 d100 3101 1901 !.....].....1.....  
00000120: 9300 d1ff 1fff c1fe d9fe 61ff 2200 d300 .....a.".....  
00000130: 3101 1701 9100 ceff 1dff c0fe dbfe 63ff 1.....c.....  
00000140: 2500 d500 3201 1501 8d00 ccff 1bff bffe %...2.....  
00000150: dcfe 66ff 2900 d800 3301 1601 8a00 c8ff ..f.)..3.....  
00000160: 18ff bffe ddfe 69ff 2c00 da00 3201 1301 .....i.,..2...  
00000170: 8800 c5ff 16ff bdfe dffe 6bff 2f00 dd00 .....k./...  
00000180: 3401 1201 8500 c2ff 15ff bffe e0fe 6eff 4.....n.....  
00000190: 3200 de00 2a68 6561 6465 722a 653e 0000 2...*header*e>..  
000001a0: 8000 1027 656e 642a a000 e3ff 2dff c4fe ...'end*....-  
000001b0: d1fe 4fff 0e00 c500 2d01 1f01 a000 e1ff ..0.....-.....
```

0x1234 (dos bytes por sample)

header

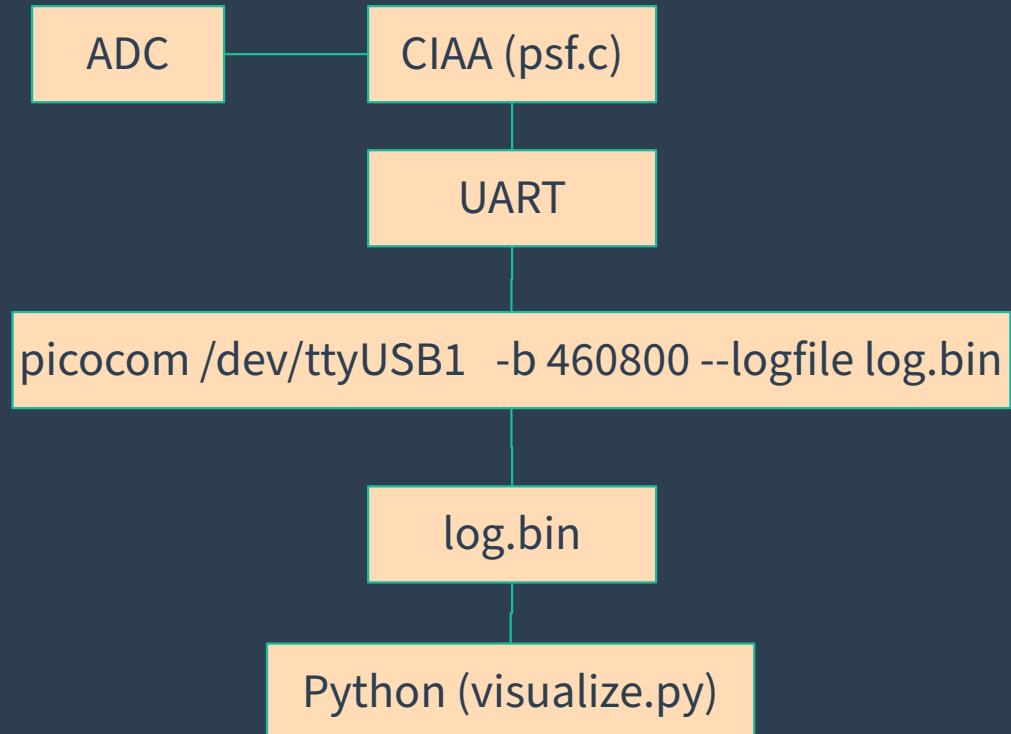
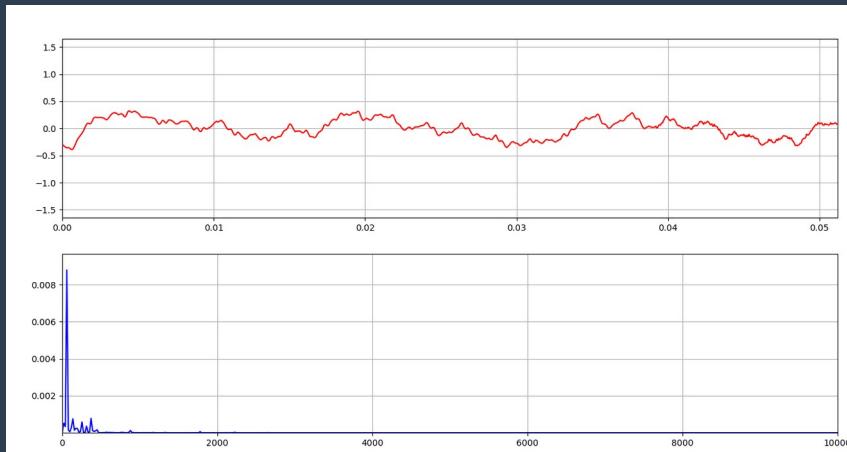
sample

sample

sample

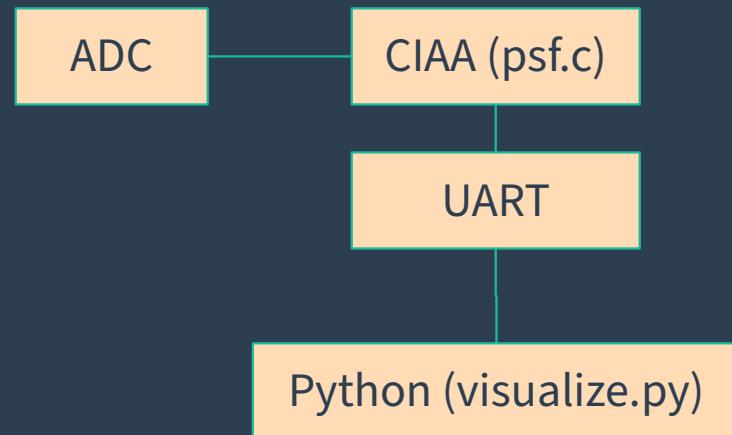
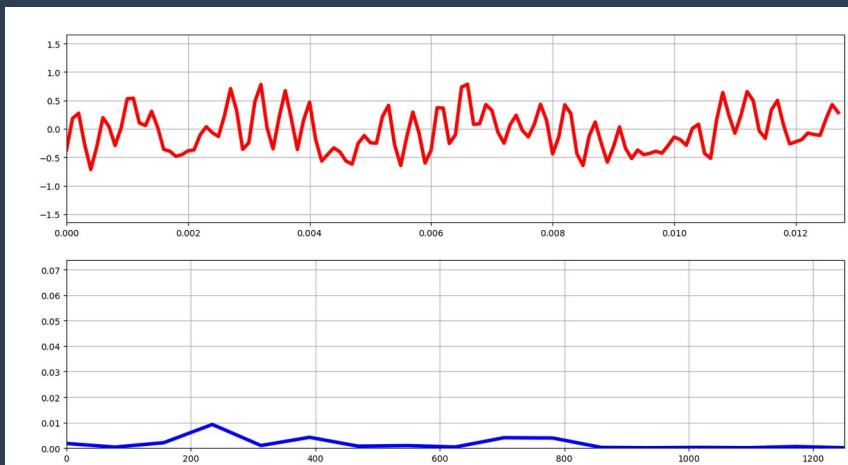
Envío por UART y grabación a archivo

- Se propone utilizar picocom v3.1 que permite reenviar los datos a un archivo
- Luego se lee el archivo desde Python
- Se grafica con matplotlib
- Tiene la ventaja de poder volver a procesar los datos offline.
- En Linux demostró no perder ningún byte



Envío por UART y recepción con pyserial

- Se propone leer directamente los datos utilizando la biblioteca pyserial.
- Se procesan los datos y se grafican con matplotlib
- En ciertos casos se encontró que agrega bytes en cero. Se mitiga el problema con flush, pero se pierden tramas.



Ancho de banda máximo por UART@460800bps

- Se calcula el ancho de banda máximo para transmitir datos desde la CIAA a la PC. Se determino empíricamente que el baudrate máximo que soporta la CIAA es de 460800bps limitado por el conversor USB.
- En función de esa tasa se podrá predecir cual sera la capacidad del sistema de visualización en tiempo real, sin perdida de tramas.
- Sin embargo es posible capturar datos a mayor velocidad si se utiliza otro método de visualización, se acepta el descarte de tramas, compresión, menos bits por muestra o almacenamiento

$$USB <> \text{UART}_{\text{maxbps}} = 460800 \text{ bps}$$

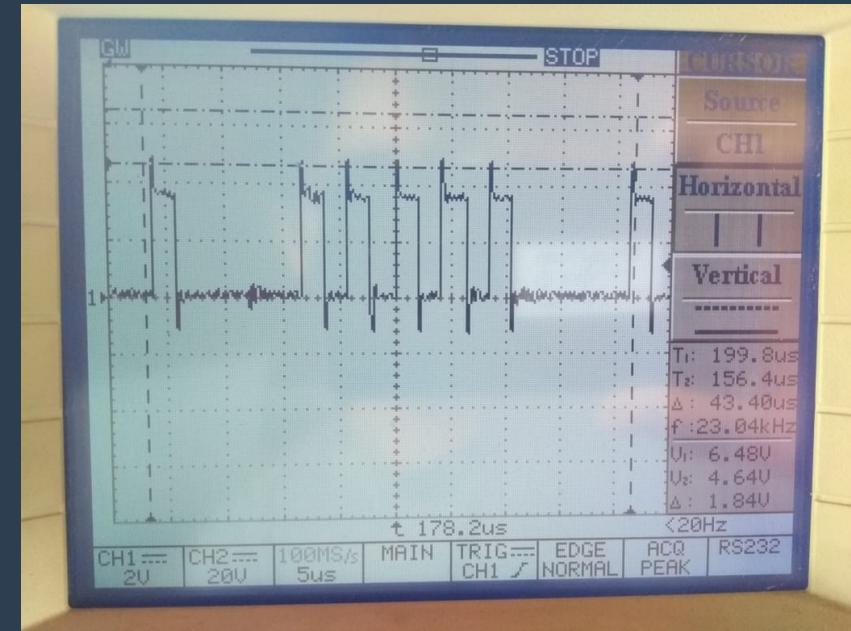
$$\text{Eficacia} = \frac{10b}{8b} = 0,8$$

$$\text{bits muestra} = 16$$

$$\text{Tasa efectiva} = \frac{460800 \text{ bps} * 0,8}{16} = 23040$$

Máxima señal muestreable y reconstruible

11520hz



Calculo del FAA @Fs=20K y B=10K

- Considerando Fs=20Khz entonces se espera un ancho de banda máximo de 10Khz
- Se muestra el calculo de un filtro RC de 1er orden para mitigar el efecto de sanaless mayores a 10K.
- Sin embargo, dado que la pendiente de atenuación del filtro no es muy abrupta, se puede optar por aumentar la Fs o reducir B para minimizar el efecto del aliasing.
- Otra alternativa es aumentar el orden del filtro

Calculo del filtro antialias 1er orden R-C

$$B = 10\text{ kbps}$$

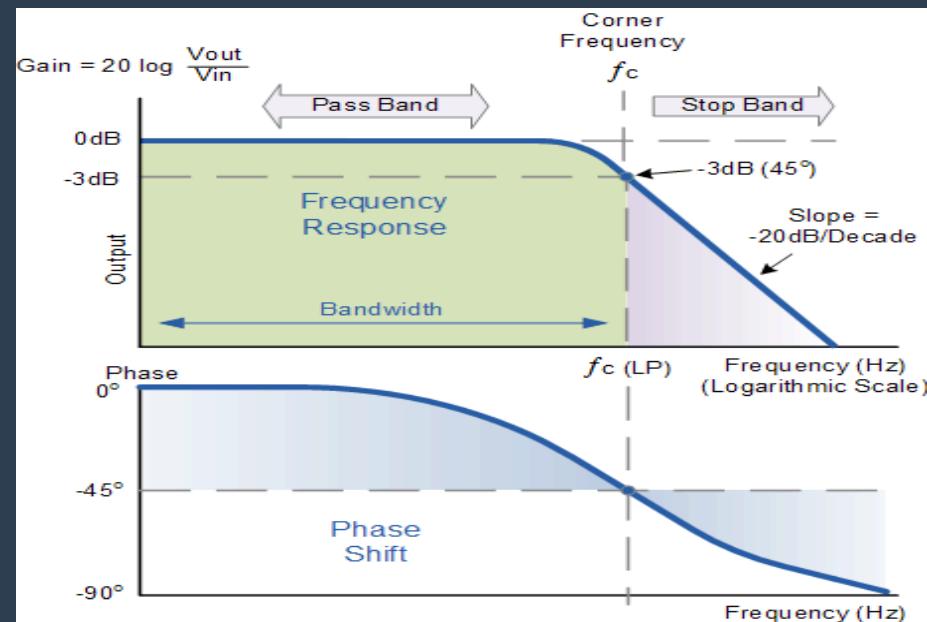
$$f_{corte} = \frac{1}{2 * \pi * R * C}$$

$$R = 1\text{k}\Omega$$

$$C = \frac{1}{f_{corte} * R * 2 * \pi} \approx 15\text{nF}$$

Máxima señal muestreable y reconstruible

11520hz



Hardware – CIAA config

- Para compilar entrar en el directorio ciaa
- Setear el nombre del la carpeta y el nombre del proyecto en el archivo Makefile
- make download para cargar el codigo

```
10 # Board default value
11 BOARD = edu_ciaa_nxp
12
13 # Board from an external board.mk file
14 -include board.mk
15
16 # Program path and name -----
17
18 # Program path and name default values
19 PROGRAM_PATH = ../clases/1_clase/ciaa
20 PROGRAM_NAME = psf1
21
```

- Para visualizar en python posicionarse en el directorio 1_clase/ciaa/psf
- Abrir el archivo visualize.py y setear el puerto de comunicaciones
- Lanzar python3 visualize.py

```
!visualize.py > [No Name] >
1  #!/usr/bin/python3
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from matplotlib.animation import FuncAnimation
5  import os
6  import io
7  import serial
8  STREAM_FILE=("/dev/ttyUSB1","serial")
9  #STREAM_FILE=("log.bin","file")
10
11 header = { "head": b"head", "id": 0, "N": 128, "fs": 1000}
12 fig = plt.figure(1)
13
14 adcAxe = fig.add_subplot(2,1,1)
15 adcAxe.set_title("ADC Data")
```

Organización

Evaluación – Trabajos prácticos

- TP1 - 5 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 - 5 pts

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

- TP final – factor de escala que aplica a la suma de las notas de los TP's de 0 a 1,5 veces
 - Deberá incluir algún tipo de procesamiento en hardware. ej. DFT/IDFT, Convolución, filtrado, etc.
 - Puede utilizar el ADC para samplear, DAC para reconstruir y/o canales de comunicación para adquirir datos previamente digitalizados.
 - Acelerómetros, IMU's, microfono, señales biomedicas, detectores, etc. son señales validas
 - Presentación de 10 minutos.
 - Deberá funcionar!
- Entrega fuera de fecha multiplica x 0.9

Calculo de nota final

- Se suman las notas de cada tp y se multiplica por el factor obtenido en el TP final
 - El tp final otorga entre
 - 0 en el caso de que no presente el tp y
 - 1.5 en caso de que sea sobresaliente.
 - $1,3 * 0,9$ en caso de que sea muy bueno, pero entrega fuera de fecha
 - Ej:
 - $Tp1=3, Tp2=4, Tpf=1.2 \Rightarrow (3+4)*1.2 = 8,4$
 - $Tp1=5, Tp2=5, Tpf=0 \Rightarrow (5+5)*0 = 0$
 - $Tp1=1, Tp2=1, Tpf=1,5 \Rightarrow (1+1)*1,5 = 3$

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
- Sistemas de control automático

Material de la materia y links afines

- Material de la materia en github:

- https://github.com/pslavkin/psf_2022

- Encuesta anónima

- <https://forms.gle/1j5dDTQ7qjVfRwYo8>

- Encuesta tecnologías:

- <https://forms.gle/rHRbXMerj6npjkdm7>

- Trabajos presentados en las ediciones previas

- 2020

- <https://drive.google.com/drive/folders/1fWSO5NvWQLMzBDe1wcpmfjd1JFiERG5?usp=sharing>
 - Afinador guitarra

- 2021

- https://docs.google.com/presentation/d/1FjsH7fxjbQoOANveXR_8EkVMCgVolYAOoWWv6my4350/edit?usp=sharing
 - <https://docs.google.com/presentation/d/1CD4qitEbdFZf2u4pPqeQ-W4AcgALzL2mi9EwG63tAhA/edit?usp=sharing>
 - https://docs.google.com/presentation/d/1Q5KuCzW1wUFnUZAteKZMK9PZ_8jZrnCDOOZQEjR8f8/edit?usp=sharing
 - https://docs.google.com/presentation/d/1cBTe6nbgt1t_GfjSMcmi7jsiq0HnF9490Al3KjBgBOXA/edit?usp=sharing
 - <https://drive.google.com/file/d/1VZ4QuVARYMVQJDQiCEYzRGNShzql-Pdk/view?usp=sharing>
 - <https://drive.google.com/file/d/1rMImP6DEmocispCR7e540cW4QSXYshtP/view>
 - https://docs.google.com/presentation/d/1BUse3eIplogLIYDkQMBox1At7RaYhi3xM60fQdNRG/edit#slide=id.g1f87997393_0_782

- 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.**

- 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/>

- **Explicación intuitiva de convolución**

- <https://betterexplained.com/articles/intuitive-convolution/>

- **Pyfdax install**

- <https://github.com/chipmuenk/pyfda>