



**FACULTAD  
DE INGENIERIA**

Universidad de Buenos Aires

# Procesamiento de señales, fundamentos

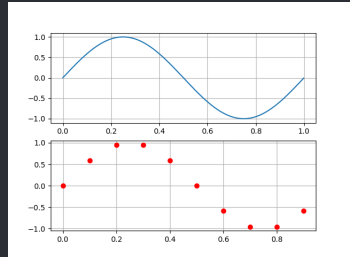
.....

Maestría en sistemas embebidos MSE2020

Universidad de Buenos Aires

## Clase 1 - Introducción

Ing. Pablo Slavkin



# Resumen de seccion 1

## 1. Señales

- 1.1 Plan de vuelo
- 1.2 Porque digital?
- 1.3 Señales
- 1.4 Generacion de señales en Python
- 1.5 Sistemas

## 2. ADC

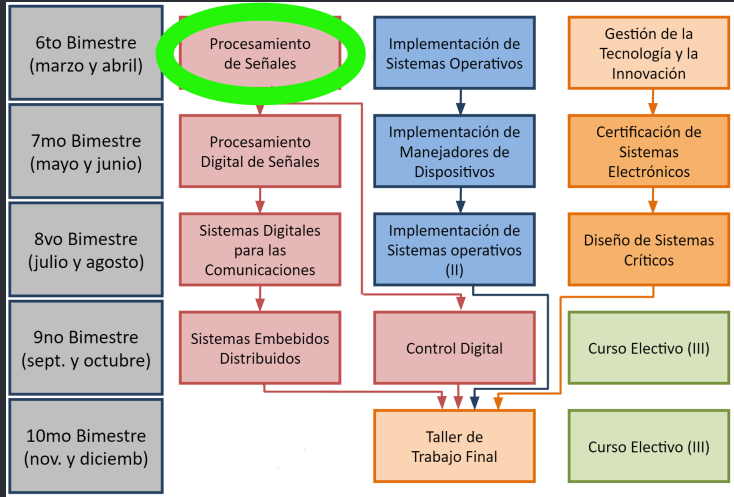
- 2.1 Aliasing
- 2.2 Teorema de Shannon

## 3. Quantizacion

- 3.1 Ejemplos
- 3.2 Modelo estadistico
- 3.3 Modelo estadistico
- 3.4 Densidad espectral
- 3.5 SNR
- 3.6 Densidad espectral

# Plan de vuelo

*Ud. esta aqui*



# Porque digital?

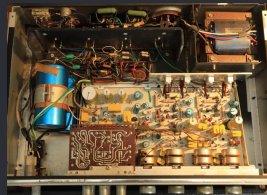
## Digital vs analogico

- 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

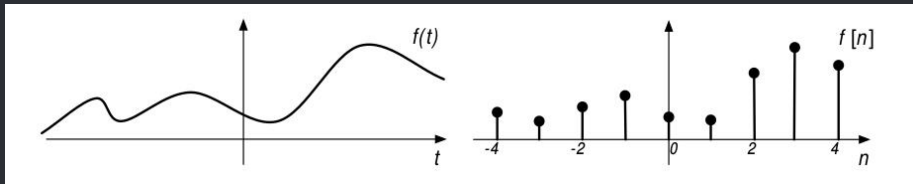
## *Tipos de señales*

- 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

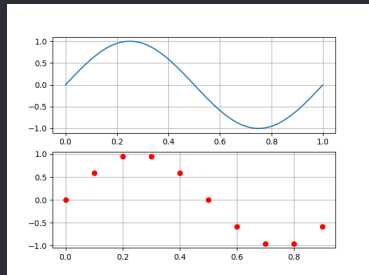
```
import matplotlib.pyplot as plt
import numpy as np

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()
```



Posrian 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 periódicas

## Continua periodica

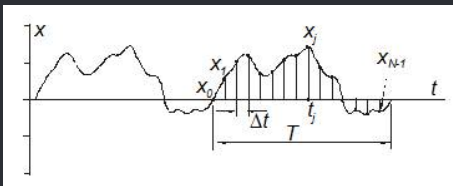
si existe un  $T_0 > 0$ , tal que  $x(t + T_0) = x(t)$ , para todo  $t$

$T_0$  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.

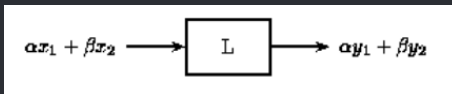
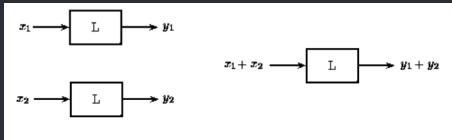
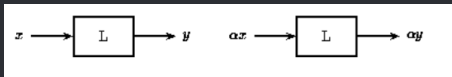
En términos de ingeniería, muy a menudo la entrada y la salida son señales eléctricas.

# Sistemas

## Lineales

### Lineal

Un sistema es lineal cuando su salida depende linealmente de la entrada. Satisface el principio de superposicion, escalado y adicicion

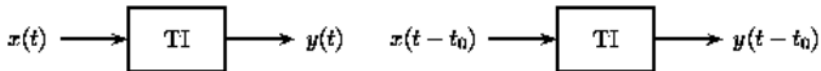


# Sistemas

## *Invariantes en el tiempo*

### 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



# Sistemas

## *Lineales invariantes en el tiempo*

### LTI

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.

# Resumen de seccion 2

## 1. Señales

- 1.1 Plan de vuelo
- 1.2 Porque digital?
- 1.3 Señales
- 1.4 Generacion de señales en Python
- 1.5 Sistemas

## 2. ADC

- 2.1 Aliasing
- 2.2 Teorema de Shannon

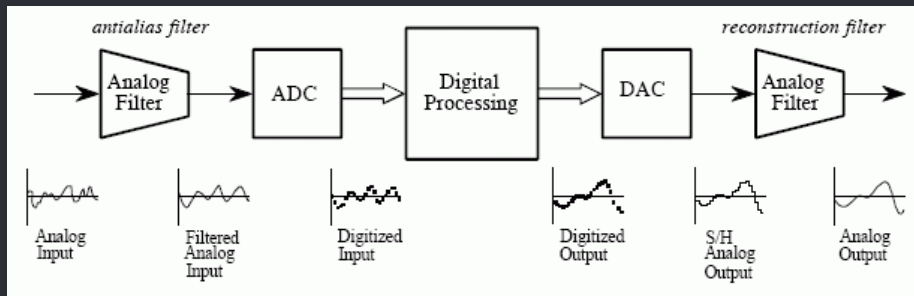
## 3. Quantizacion

- 3.1 Ejemplos
- 3.2 Modelo estadistico
- 3.3 Modelo estadistico
- 3.4 Densidad espectral
- 3.5 SNR

## 3.6 Densidad espectral

# ADC

## Bloque generico de procesamiento



Porque el filtro antialiasing?

# 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)+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])

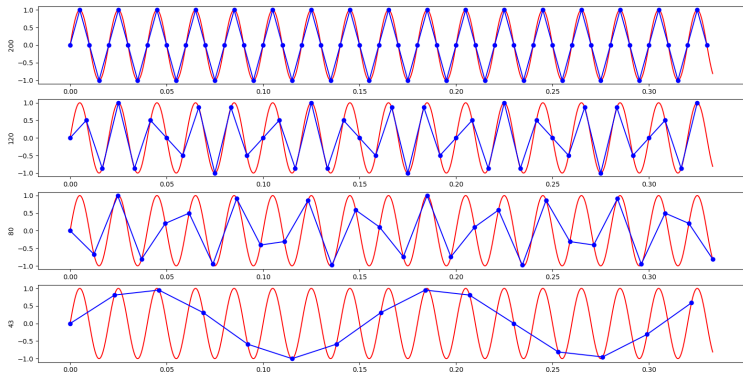
plt.show()
```



# Aliasing

## Simulando en Python

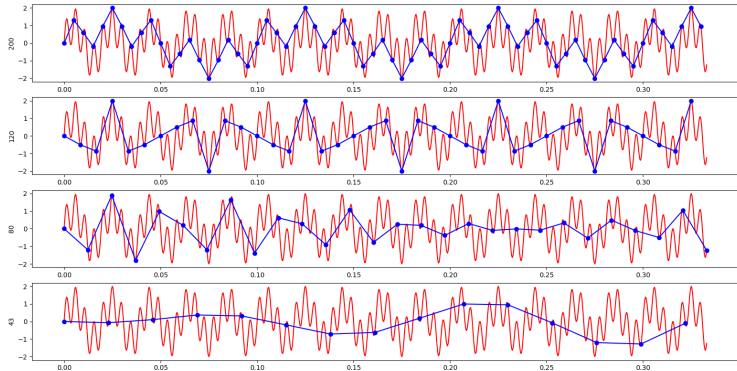
Diferentes frecuencias de sampleo para capturar una señal de 50hz



# Aliasing

## Simulando en Python

Que pasa si se suma ruido de alta frecuencia?



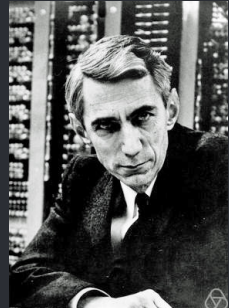
# Teorema de sampleo

## 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 = 2000
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])

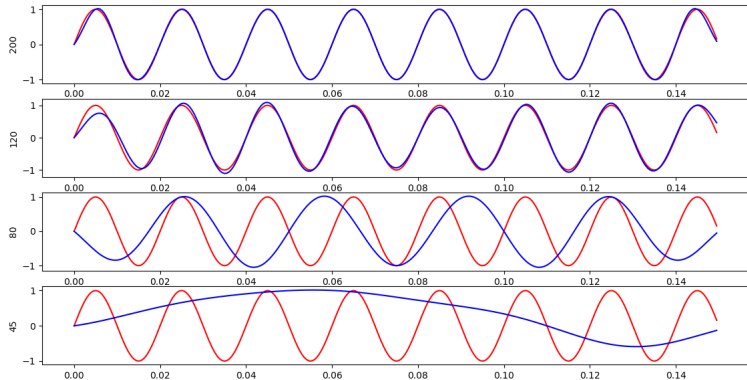
plt.show()
```

# Teorema de sampleo

## Teorema de Shannon



### Sampleo e interpolado

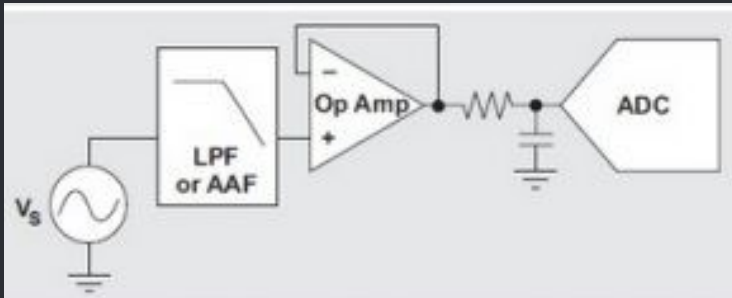


# Sampleo

## Filtro antialias

### FAA

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



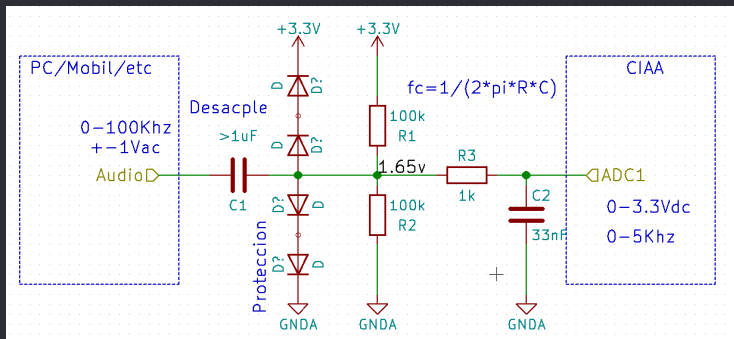
# Sampleo

## Acondicionamiento de señal



Acondicionar la señal de salida del dispositivo de sonido (en PC ronda  $\pm 1V$ ) 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 3

## 1. Señales

- 1.1 Plan de vuelo
- 1.2 Porque digital?
- 1.3 Señales
- 1.4 Generacion de señales en Python
- 1.5 Sistemas

## 2. ADC

- 2.1 Aliasing
- 2.2 Teorema de Shannon

## 3. Quantizacion

- 3.1 Ejemplos
- 3.2 Modelo estadistico
- 3.3 Modelo estadistico
- 3.4 Densidad espectral
- 3.5 SNR

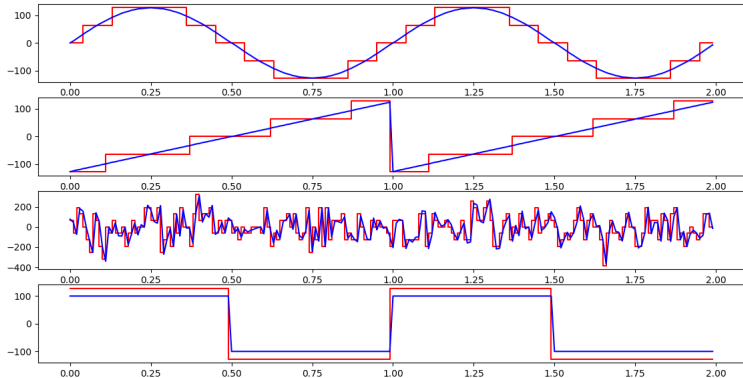
### 3.6 Densidad espectral



# 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)

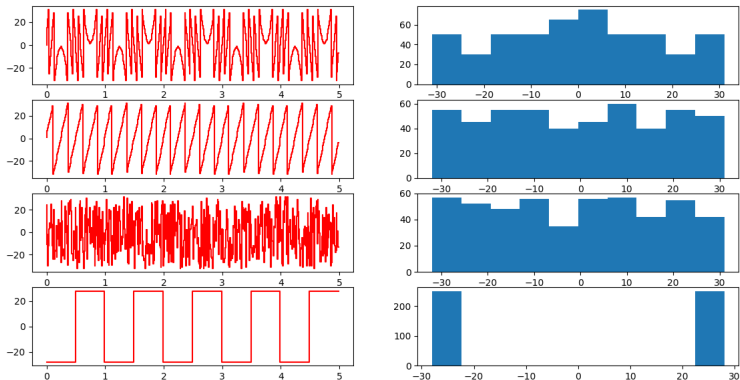
fig         = plt.figure()
for i in range(len(signalC)):
    contiAxe = fig.add_subplot(4,1,i+1)
    plt.step(tC,signalQ[i], 'r-')
    plt.plot(tC,signalC[i], 'b-')

plt.show()
```

# Ruido de cuantización

## 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()
```

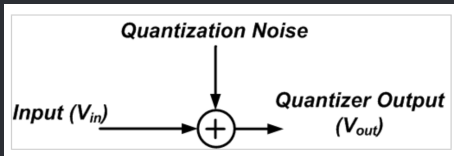
# Ruido de cuantizacion

## Modelo estadístico

En el caso de que se cumplan las siguientes premisas:

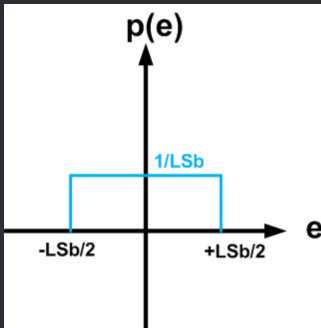
- La entrada se distancia de los diferentes niveles de cuantizacion con igual probabilidad
- El error de cuantizacion NO esta correlacionado con la entrada
- El cuantizador cuanta con un numero relativamente largo de niveles
- Los niveles de cuantizacion son uniformes

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



# Ruido de cuantizacion

*Funcion densidad de probabilidad*



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

# Ruido de cuantizacion

*Funcion densidad de probabilidad*

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

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

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

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

$$P_q = \frac{1}{lsb} \frac{lsb^3}{24} + \frac{lsb^3}{24}$$

## Potencia de ruido de cuantizacion

$$P_q = \frac{lsb^2}{12} \quad (1)$$

# Ruido de cuantizacion

## *Densidad espectral de potencia de ruido*

Si consideramos la potencia de ruido uniformemente distribuido en todo el espectro desde DC hasta la mitad de la frecuencia de sampleo, nos queda que:

$$S_{\text{medio espectro}}(f) = \frac{P_q}{\frac{F_s}{2}}$$

$$S_{\text{medio espectro}}(f) = \frac{lsb^2}{6 * F_s}$$



# Ruido de cuantizacion

*Relacion señal a ruido*

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

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

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

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

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

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

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

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

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

# Ruido de cuantizacion

*Relacion señal a ruido*

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

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

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

### Densidad espectral de potencia de ruido

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

Entonces como puedo mejorar la SNR de un sistema?