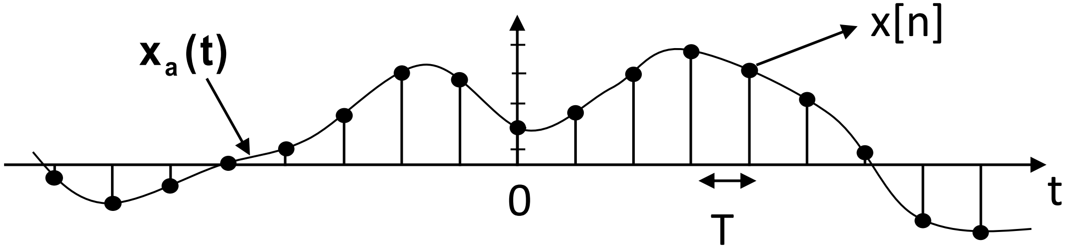
Tema 5. Conversión A/D y D/A y cambio de la frecuencia de muestreo

1. Muestreo y conversión A/D (3/12/2019)
2. Reconstrucción y conversión D/A (10/12/2019)
3. Conversión A/D y D/A de imágenes
4. Cambio de la frecuencia de muestreo
   4.1 Diezmado (downsampling) (13/12/2019)
   4.2 Interpolación (upsampling) (17/12/2019)

Muestreo (1)

- Muestreo en el dominio temporal

T = Periodo de muestreo, $f_s=1/T$ = Frecuencia de muestreo



$x[n] = x_a(nT)$

- Relación entre transformadas: $X(F)|_{F=\frac{f}{f_s}} = f_s \sum_{k=-\infty}^{\infty} X_a(f-k\cdot f_s)$

- Criterio de Nyquist: $1/_T = f_s \geq 2f_{max}$

Muestreo (2)

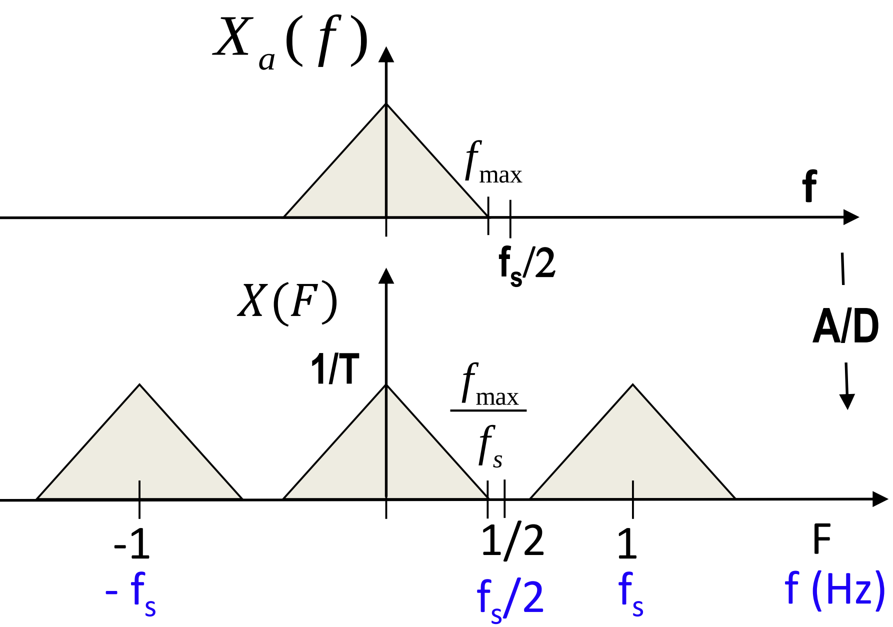
U5

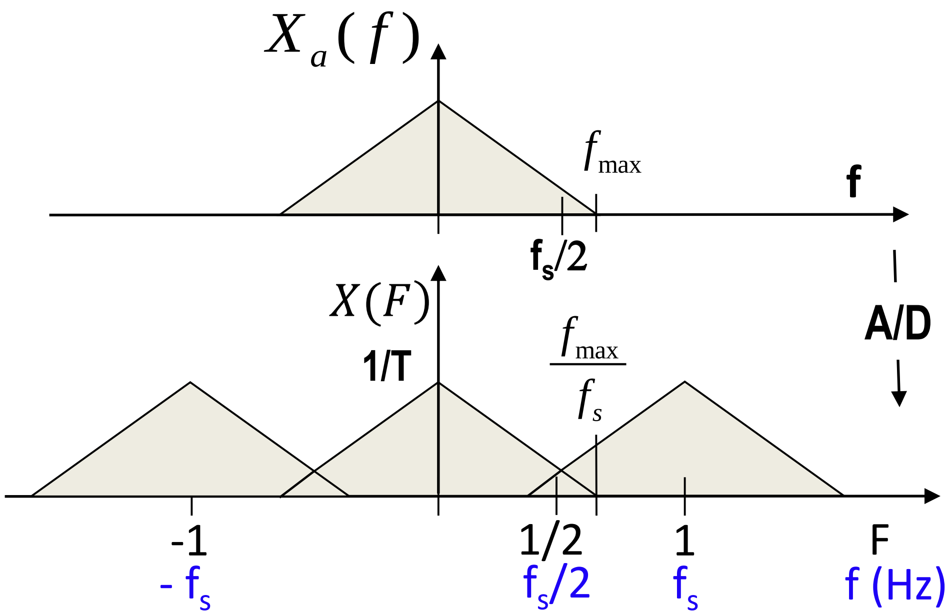
Caso 1: Se cumple el criterio de Nyquist (no hay aliasing)

$$\boxed{f_s > 2f_{max}}$$

Caso 2: No se cumple el criterio de Nyquist (hay aliasing)

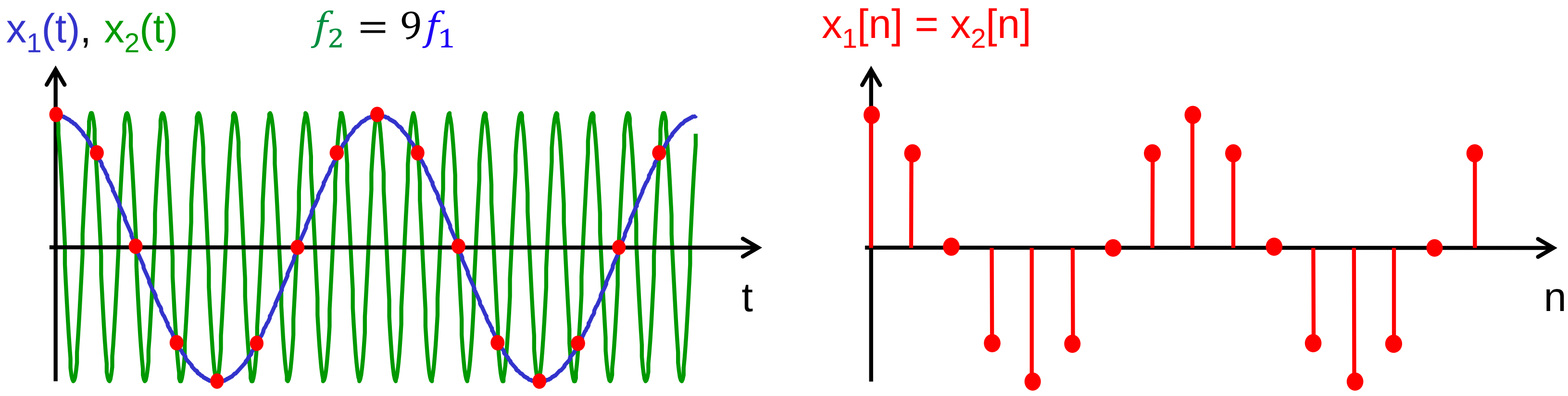
$$\boxed{f_s < 2f_{max}}$$





Aliasing de sinusoides (1)

Aliasing: sinusoides analógicas diferentes producen secuencias idénticas



$$f_s = 8f_1 = \frac{8}{9}f_2$$

$$\cos(2\pi f_1 t)\Big|_{t=nT_s} = \cos\left(2\pi\frac{1}{8}n\right)$$

$$\cos(2\pi f_2 t)\Big|_{t=nT_s} = \cos\left(2\pi\frac{9}{8}n\right) = \cos\left(2\pi\frac{1}{8}n\right)$$

Aliasing de sinusoides (2)

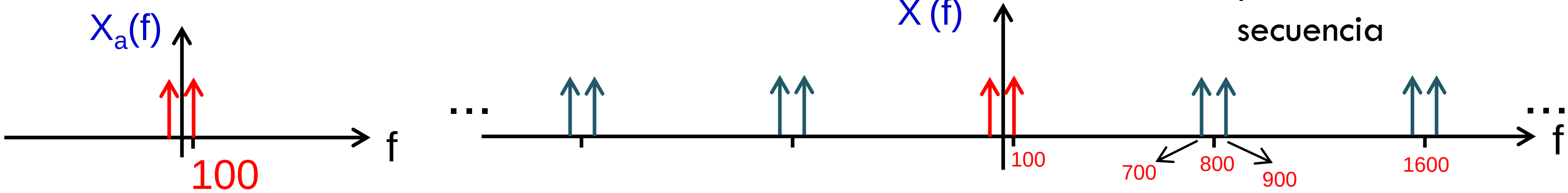
$$X(f) = f_s \sum_{k=-\infty}^{\infty} X_a(f - k \cdot f_s)$$

T=1.25ms=1/800 s, f_s=800 Hz

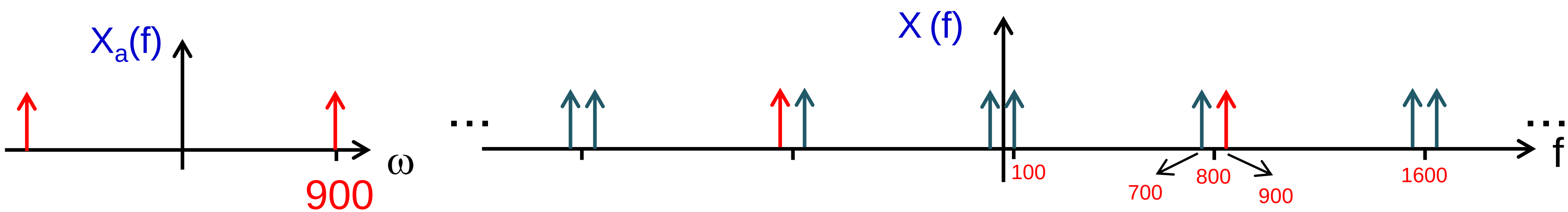
$x_a(t) = \cos(2\pi f_0 t)$ —— $x[n]$

Aliasing: diferentes sinusoides analógicas producen la misma secuencia

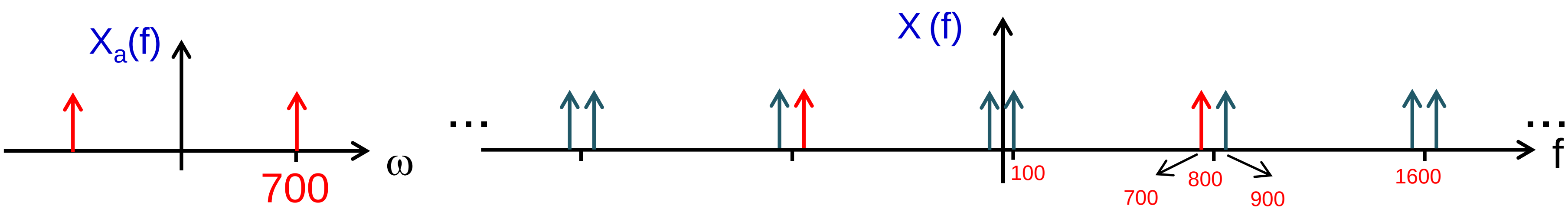
a) $f_0 = 100\ Hz$



b) $f_0 = 900\ Hz$

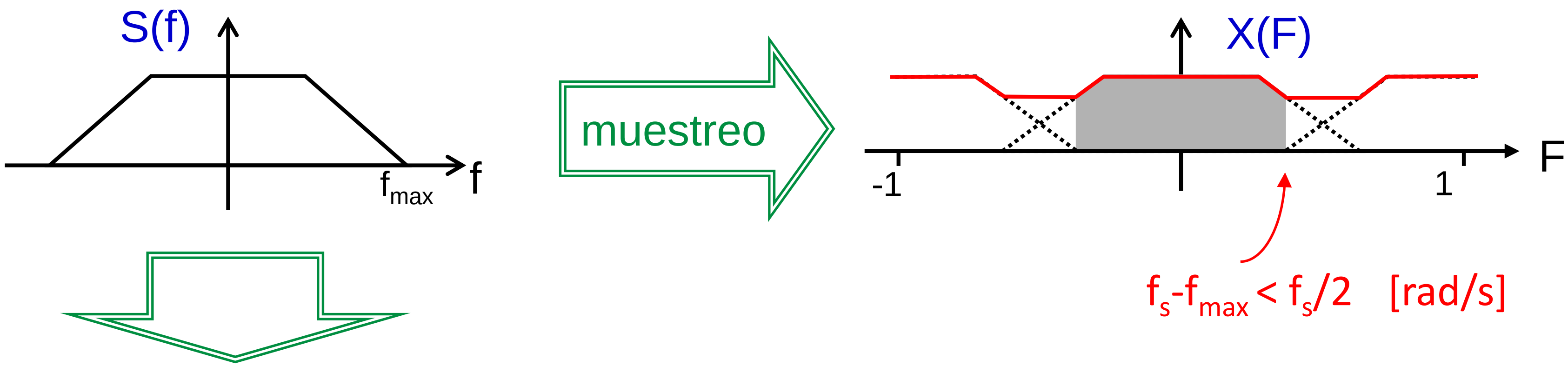


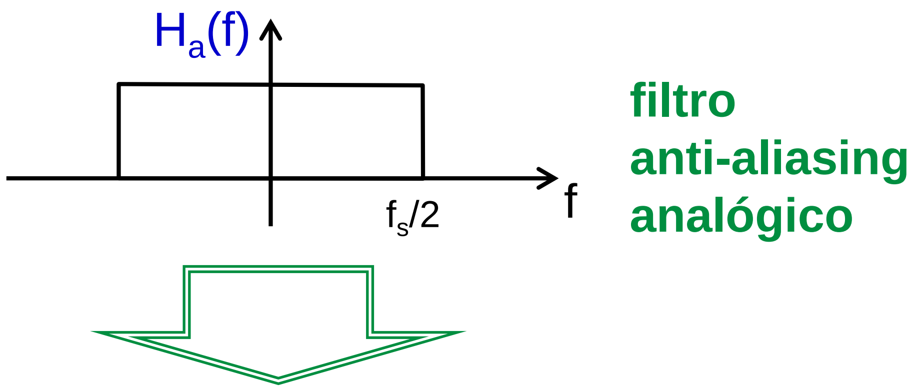
c) $f_0 = 700\ Hz$



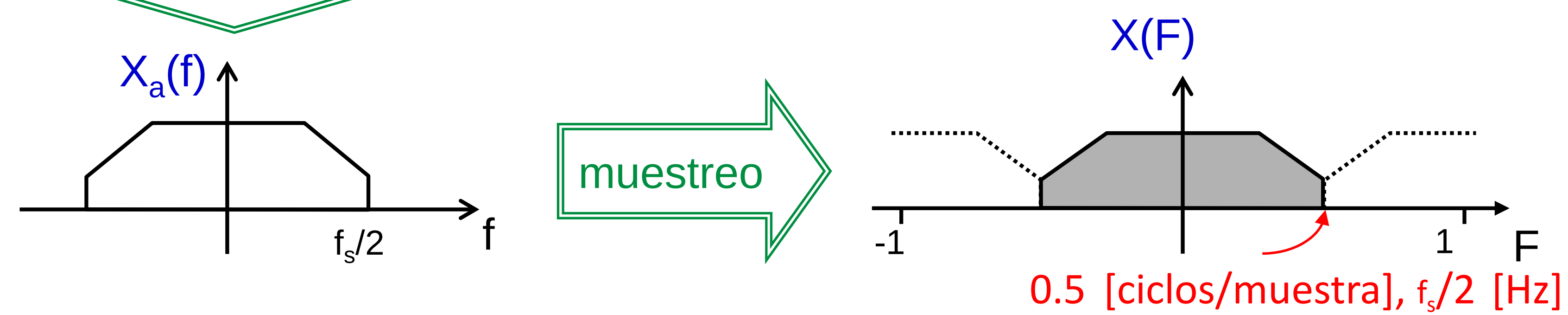
d) Hay muchos otros casos (infinitas posibilidades)…

Filtro anti-aliasing



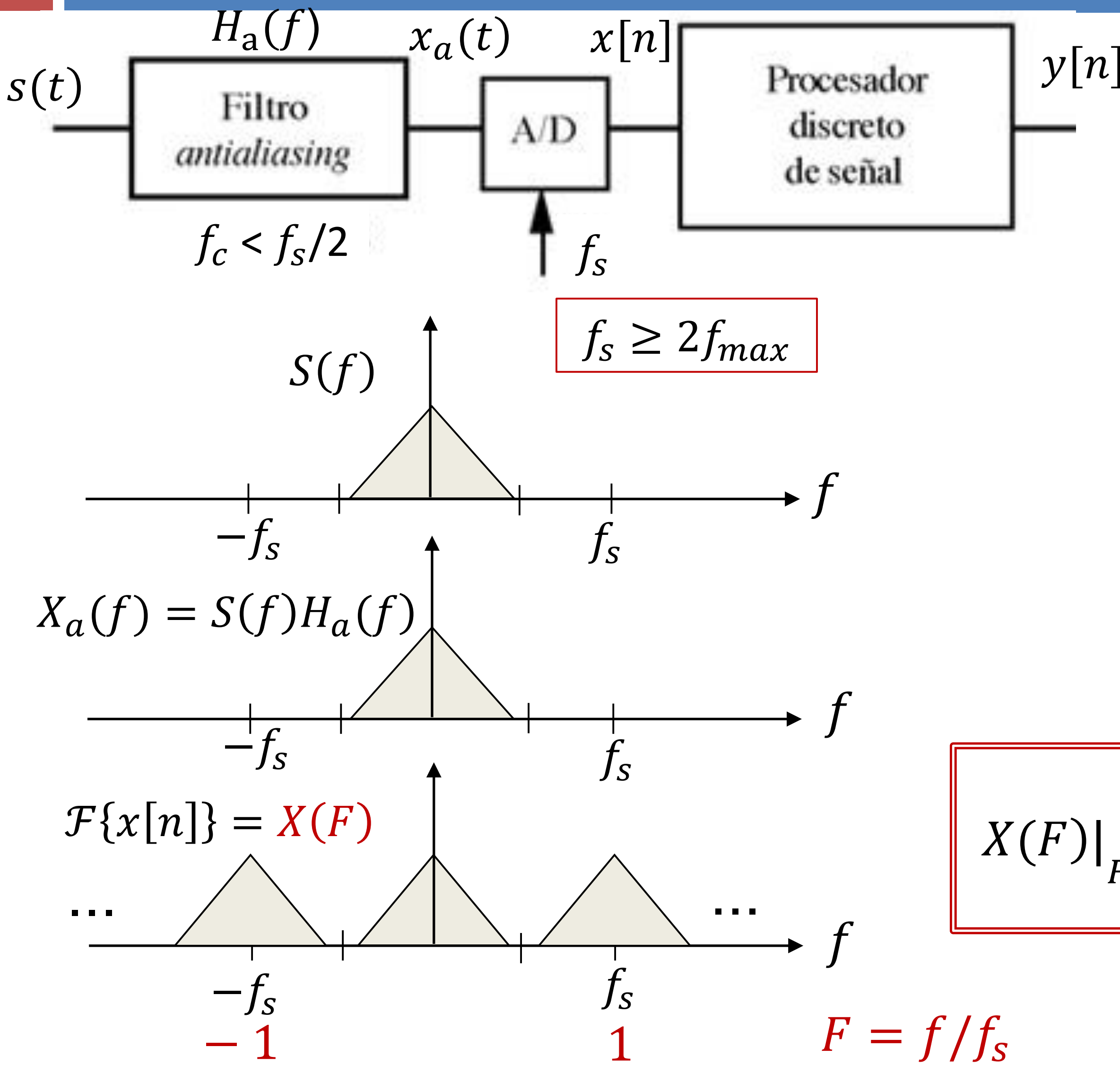


El filtro anti-aliasing minimiza la cantidad de información que se pierde cuando no se cumple el criterio de Nyquist



Resumen (se cumple criterio de Nyquist)

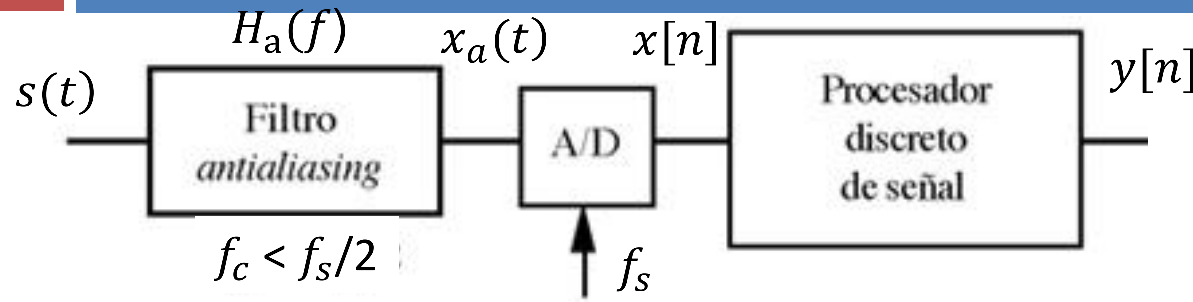
U5

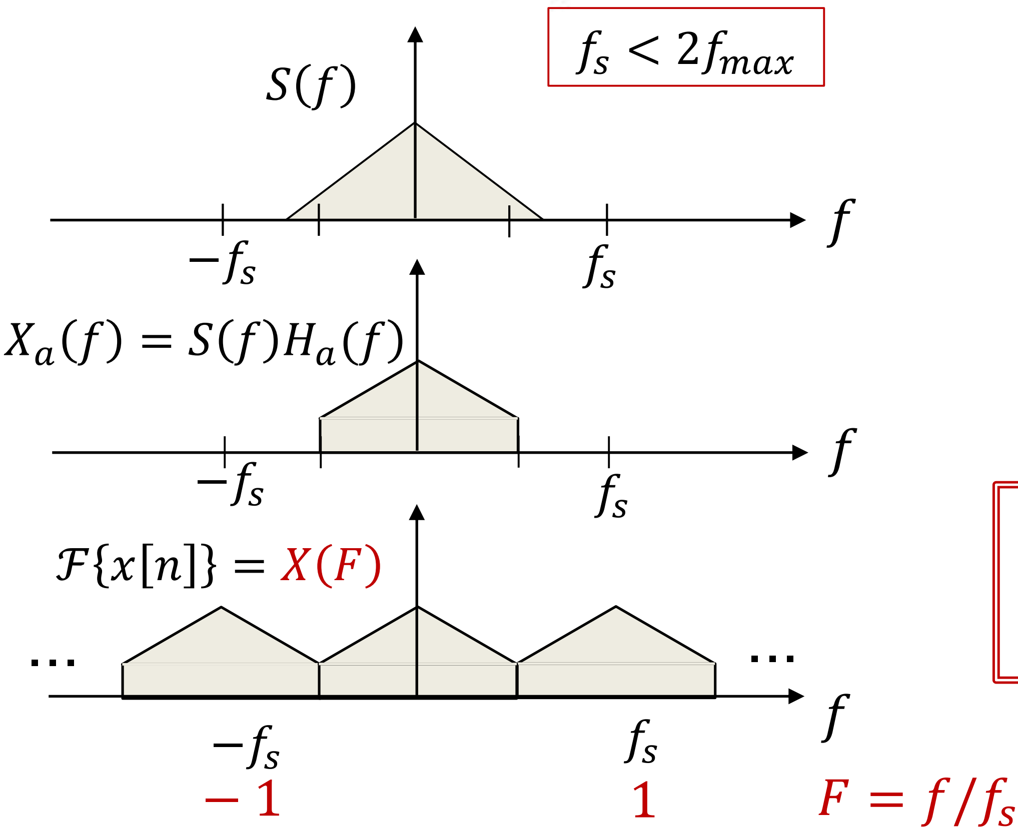


$$X(F)\big|_{F=\frac{f}{f_s}} = f_s \sum_{k=-\infty}^{\infty} X_a(f - k \cdot f_s)$$

$F = f/f_s$

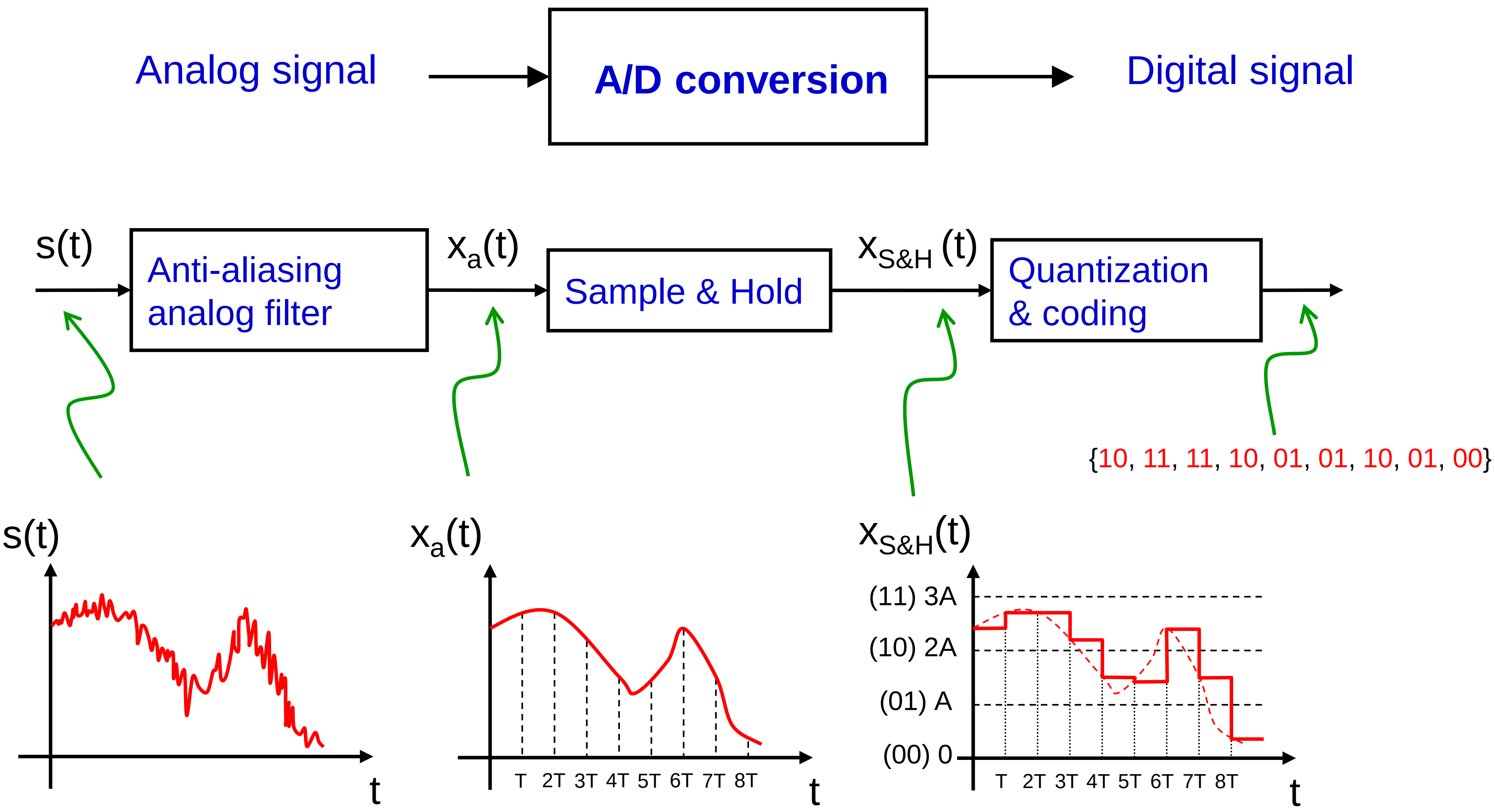
Resumen (NO se cumple criterio de Nyquist)





$$X(F)\big|_{F=\frac{f}{f_s}} = f_s \sum_{k=-\infty}^{\infty} X_a(f - k\cdot f_s)$$

Conversión A/D práctica



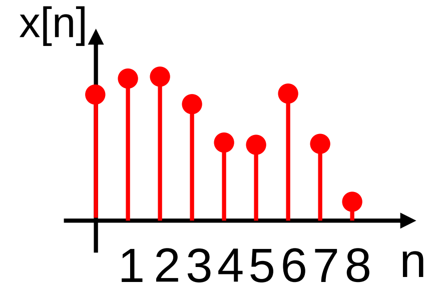
Señal filtrada paso bajo

Tema 5. Conversión A/D y D/A y cambio de la frecuencia de muestreo

1. Muestreo y conversión A/D (3/12/2019)
2. Reconstrucción y conversión D/A (10/12/2019)
3. Conversión A/D y D/A de imágenes
4. Cambio de la frecuencia de muestreo
   4.1 Diezmado (downsampling) (13/12/2019)
   4.2 Interpolación (upsampling) (17/12/2019)

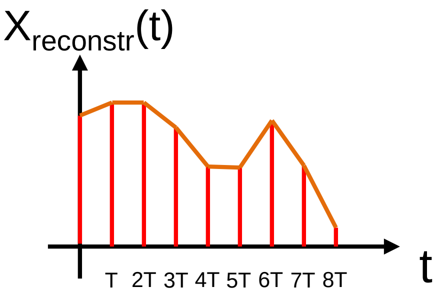
Reconstrucción (1)

Se basa en la interpolación: Usamos las muestras $x[n]$ para generar una señal analógica que aproxima la señal analógica original antes del muestreo.

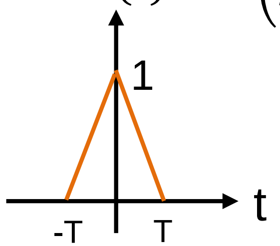


$$x_{reconstr}(t) = \sum_{n=-\infty}^{\infty} x[n]\, h(t - nT)$$

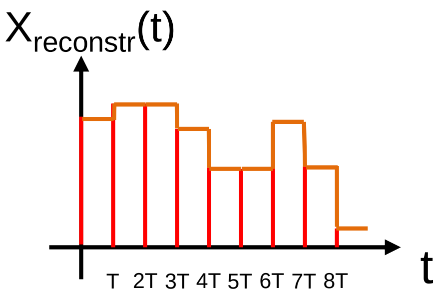
- Interpolador lineal:



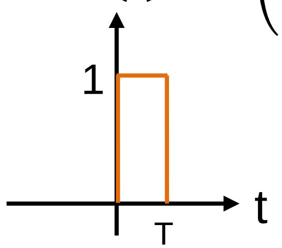
$$h(t) = \Lambda\left(\frac{t}{T}\right)$$



- Interpolador de orden cero: Zero-order-hold (ZOH)



$$h(t) = \Pi\left(\frac{t - T/2}{T}\right)$$



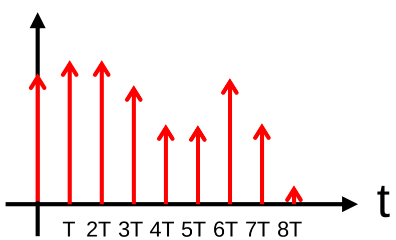
- Otros interpoladores son posibles

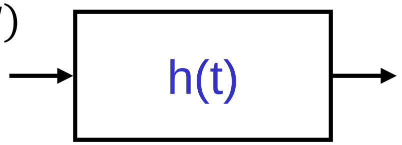
Reconstrucción (2)

$$x_{reconstr}(t) = \sum_{n=-\infty}^{\infty} x[n]\, h(t - nT) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT) * h(t)$$

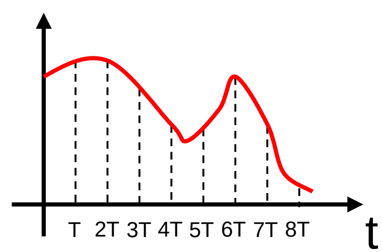
La señal analógica resultado de interpolar las muestras se puede **modelar** como la señal analógica original muestreada con deltas (muestreo ideal) y convolucionada con h(t)

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT)$$





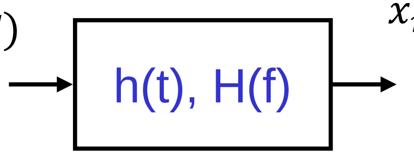
$$x_{reconstr}(t) = x_p(t) * h(t) = \sum_{n=-\infty}^{\infty} x(nT)h(t - nT)$$



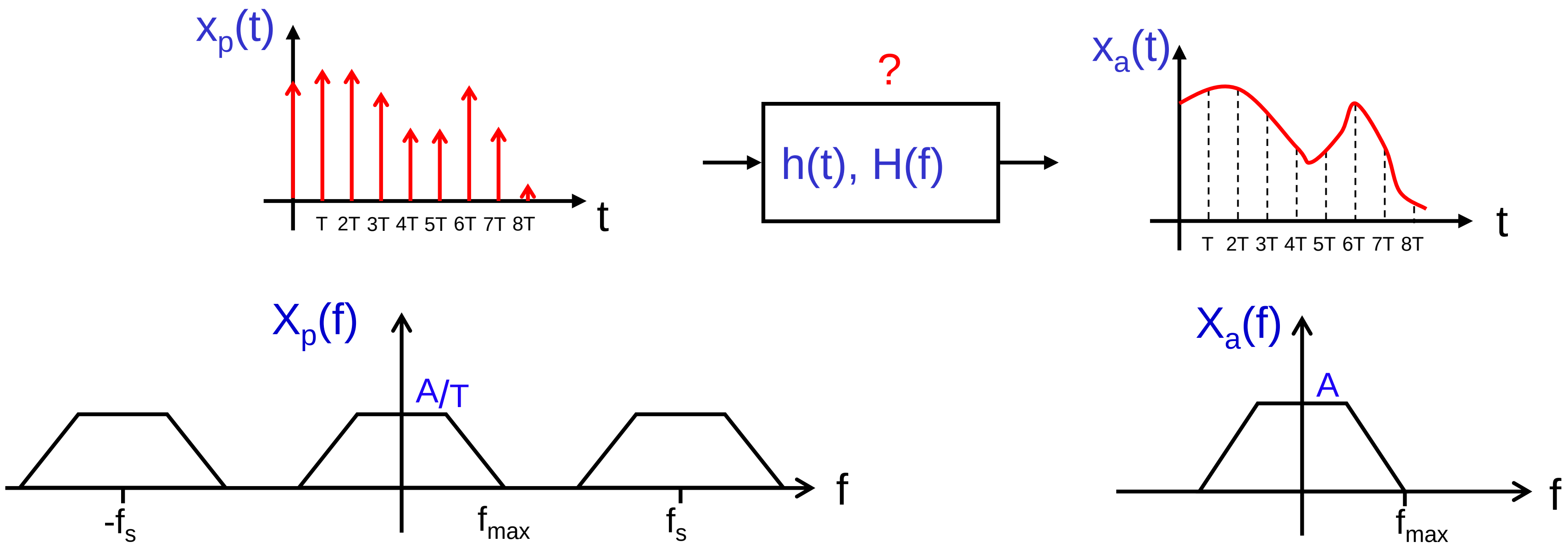
¿Cuál es el mejor filtro interpolador (interpolador ideal)?

Interpolador ideal (1)

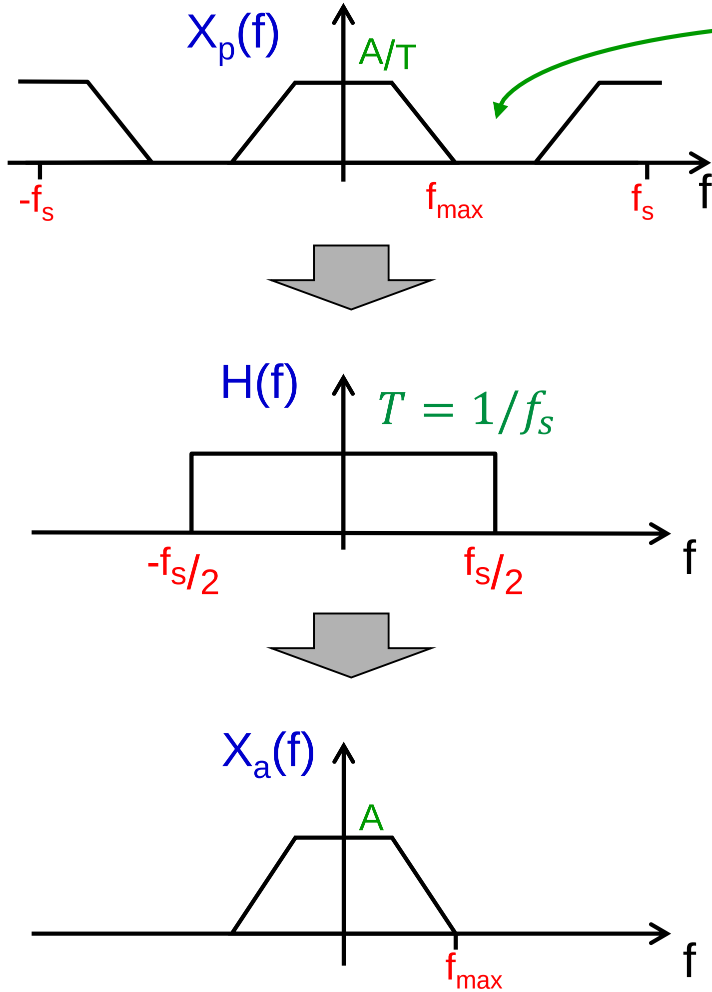
$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t-nT)$$



$$x_{reconstr}(t) = x_p(t) * h(t)$$
$$= \sum_{n=-\infty}^{\infty} x(nT)h(t-nT)$$



Interpolador ideal (2)

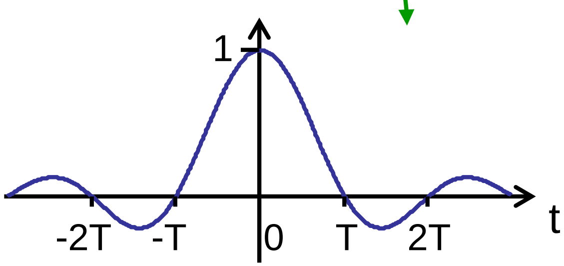


Supondremos que no hay aliasing

$$h(t) = \int_{-\infty}^{\infty} H(f) e^{j2\pi f t}\, df =$$

$$= \int_{-f_s/2}^{f_s/2} T e^{j2\pi f t}\, df = \frac{T}{j2\pi t}\; e^{j2\pi f t}\Big|_{-\frac{f_s}{2}}^{\frac{f_s}{2}}$$

$$= \frac{T}{\pi t}\sin(\pi f_s t) = \mathrm{sinc}\left(\frac{t}{T}\right)$$



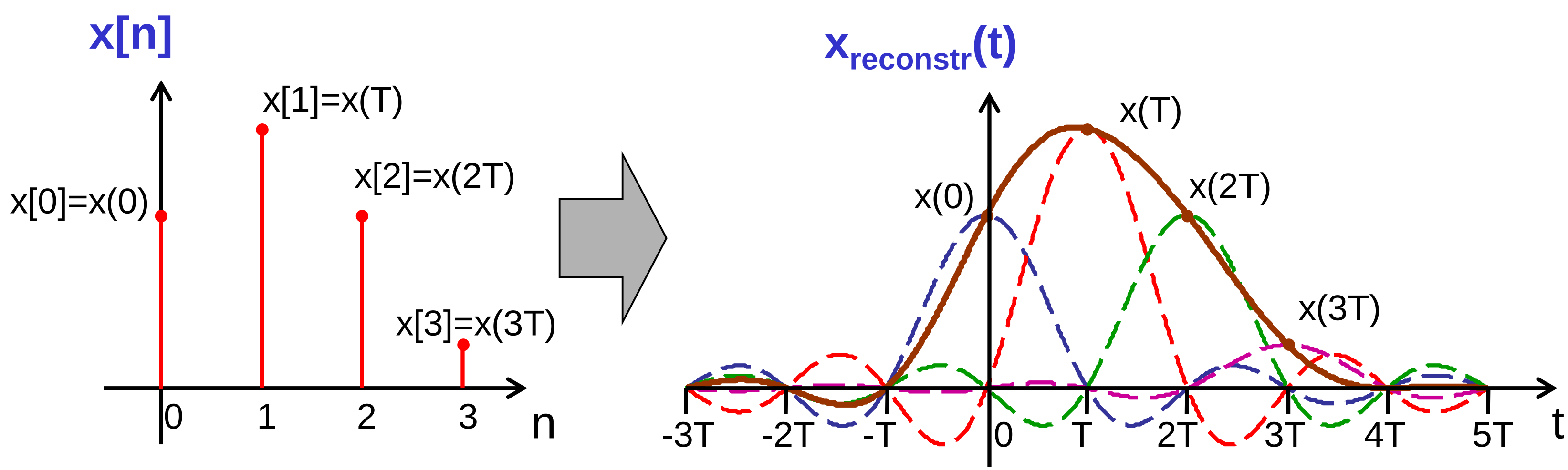
Interpolador ideal (3)

$$x_p(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t-nT)$$

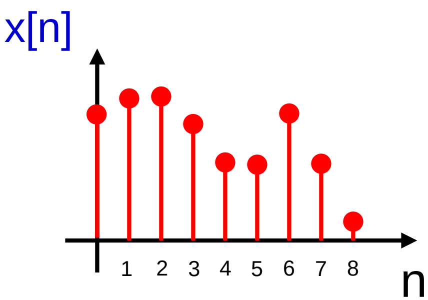
→ h(t), H(f) →

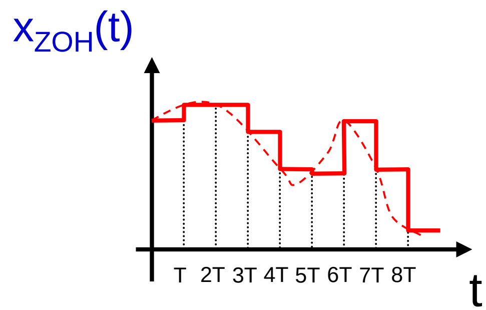
$$x_{reconstr}(t) = x_p(t) * h(t) = \sum_{n=-\infty}^{\infty} x(nT)h(t-nT)$$

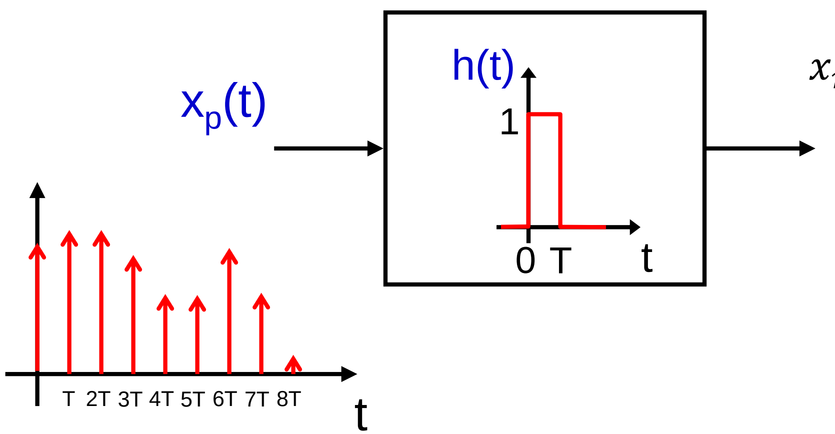
$$x_{reconstr}(t) = x_p(t) * h(t) = \sum_{n=-\infty}^{\infty} x(nT)sinc\left(\frac{t-nT}{T}\right)$$



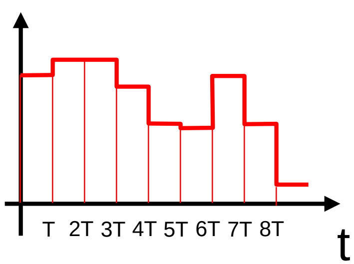
Interpolador ZOH (1)





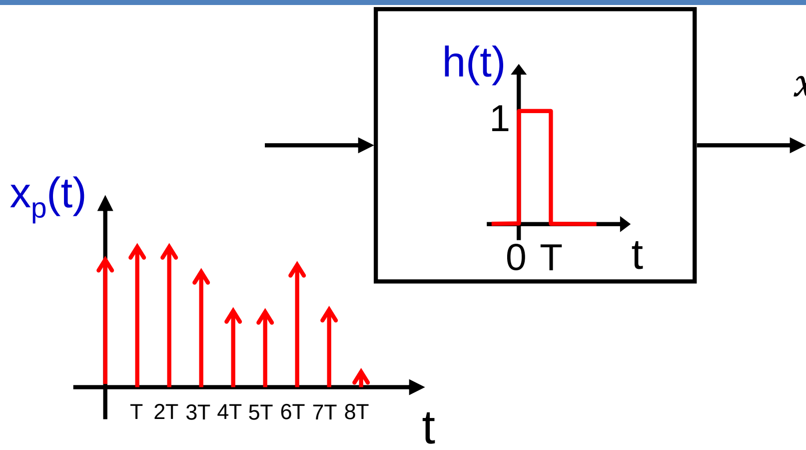


$$x_{reconstr}(t) = x_{ZOH}(t) = x_p(t) * h(t)$$
$$= \sum_{n=-\infty}^{\infty} x(nT)h(t-nT)$$

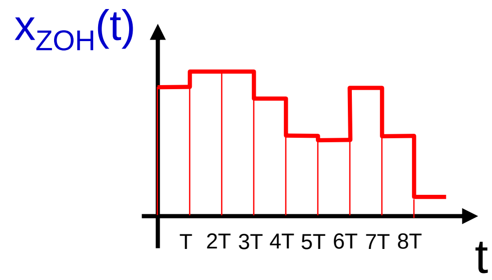


Interpolador ZOH (2)

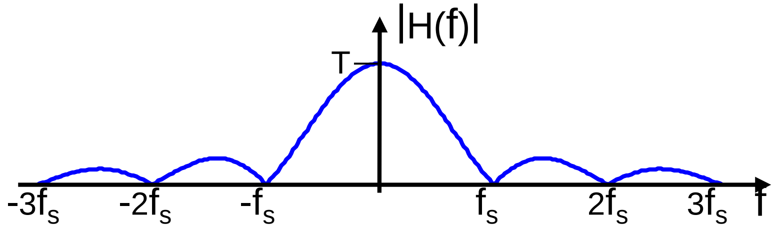
U5

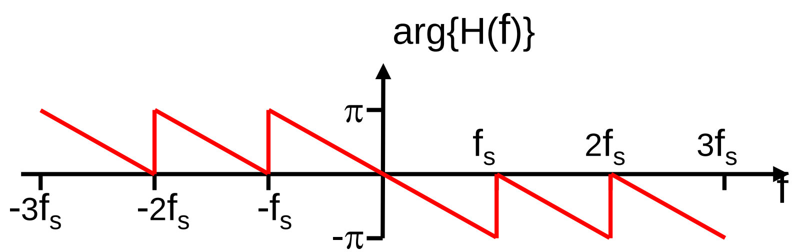


$$x_{reconstr}(t) = x_{ZOH}(t) = x_p(t) * h(t)$$
$$= \sum_{n=-\infty}^{\infty} x(nT)h(t-nT)$$



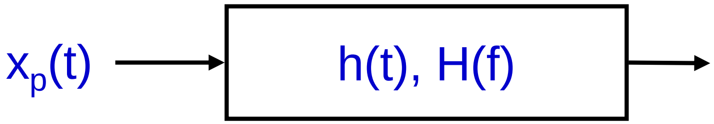
$$H(f) = \int_{-\infty}^{\infty} h(t)e^{-j2\pi ft}\,dt = \int_0^T e^{-j2\pi ft}\,dt = \frac{1-e^{-j2\pi fT}}{j2\pi f} = e^{-j\pi fT}\,\frac{\left(e^{j\pi fT}-e^{-j\pi fT}\right)}{j2\pi f}$$
$$= e^{-j\pi fT}\,\frac{sin(\pi fT)}{\pi f} = e^{-j\pi fT}\,Tsinc(fT)$$





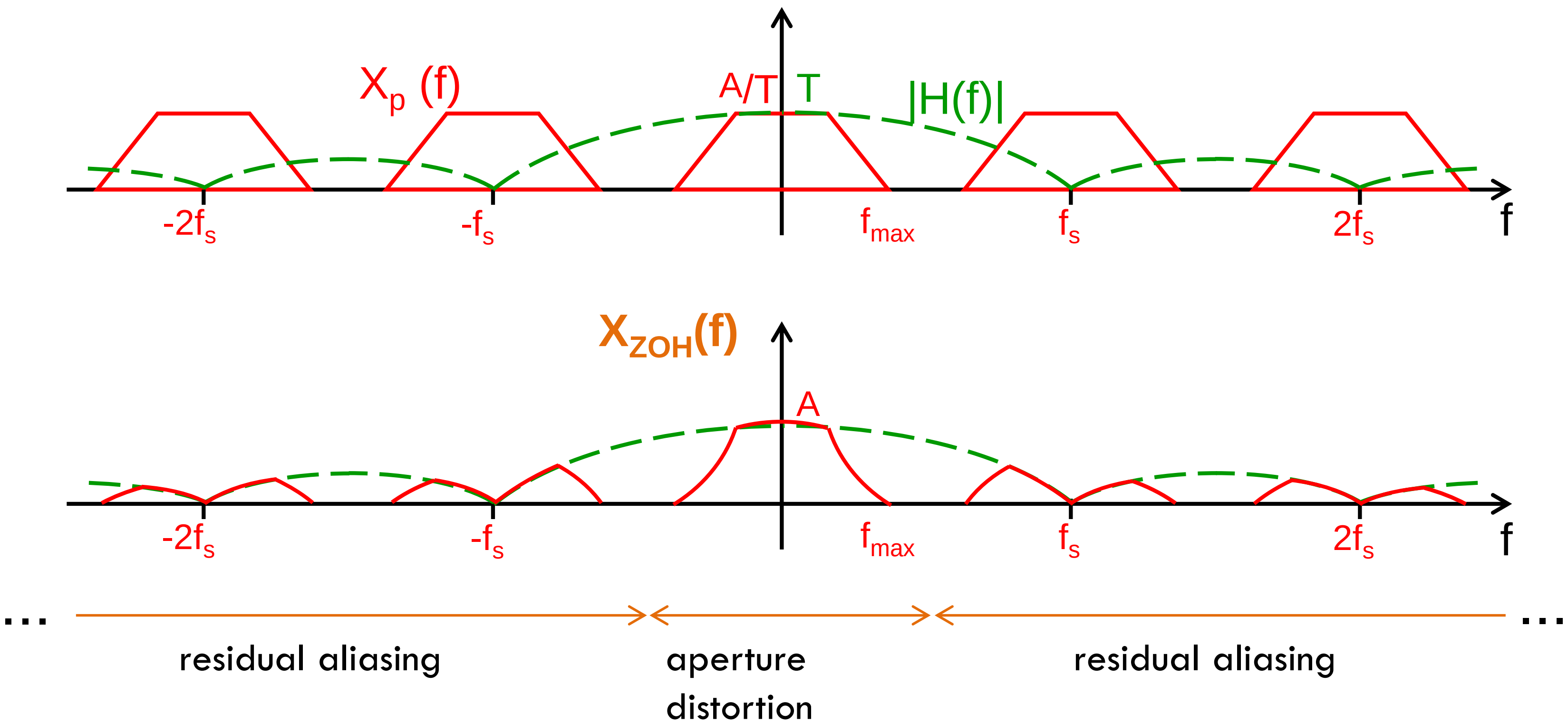
Interpolador ZOH (3)

U5



$x_{reconstr}(t) = x_{ZOH}(t) = x_p(t) * h(t)$

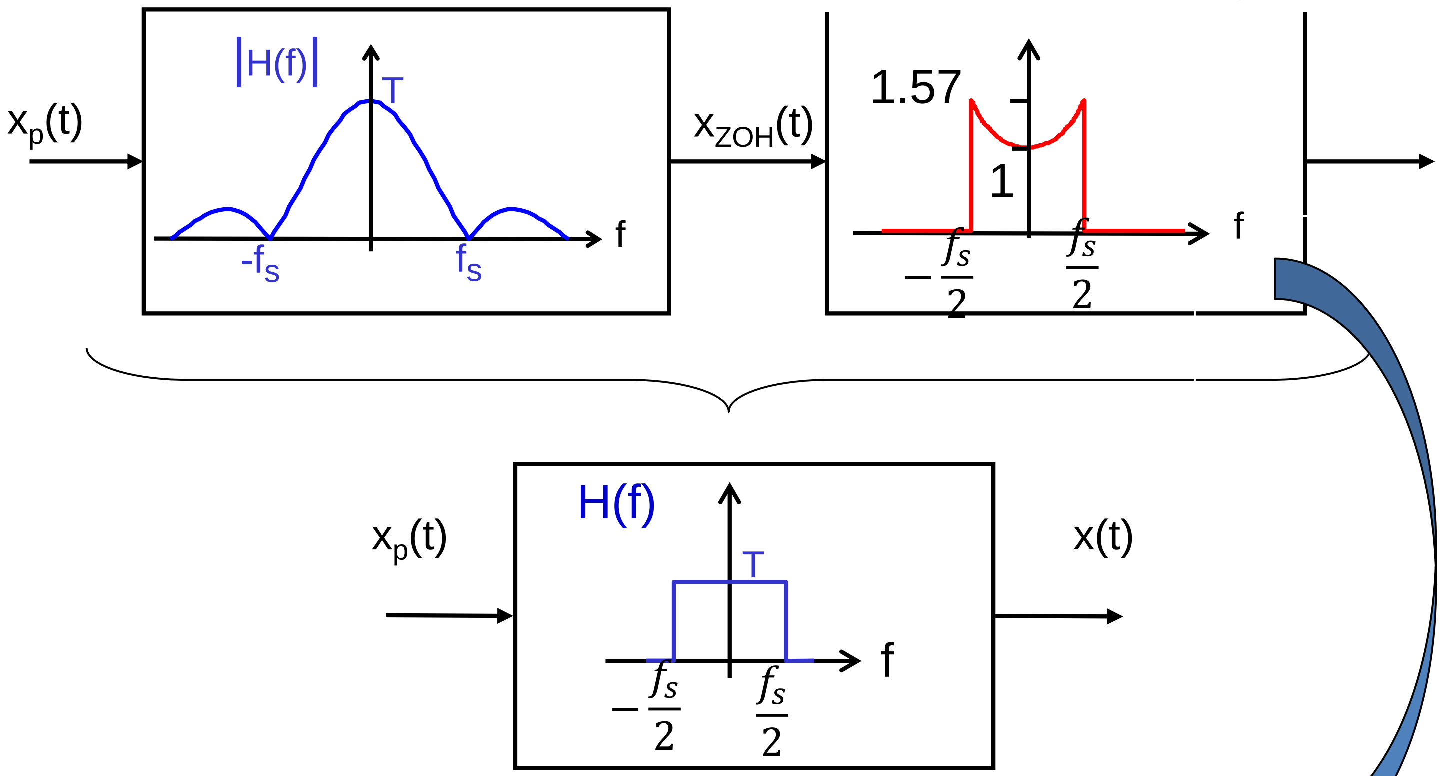
$X_{reconstr}(\omega) = X_{ZOH}(\omega) = X_p(\omega) \cdot H(\omega)$



Interpolador ZOH (4)

U5

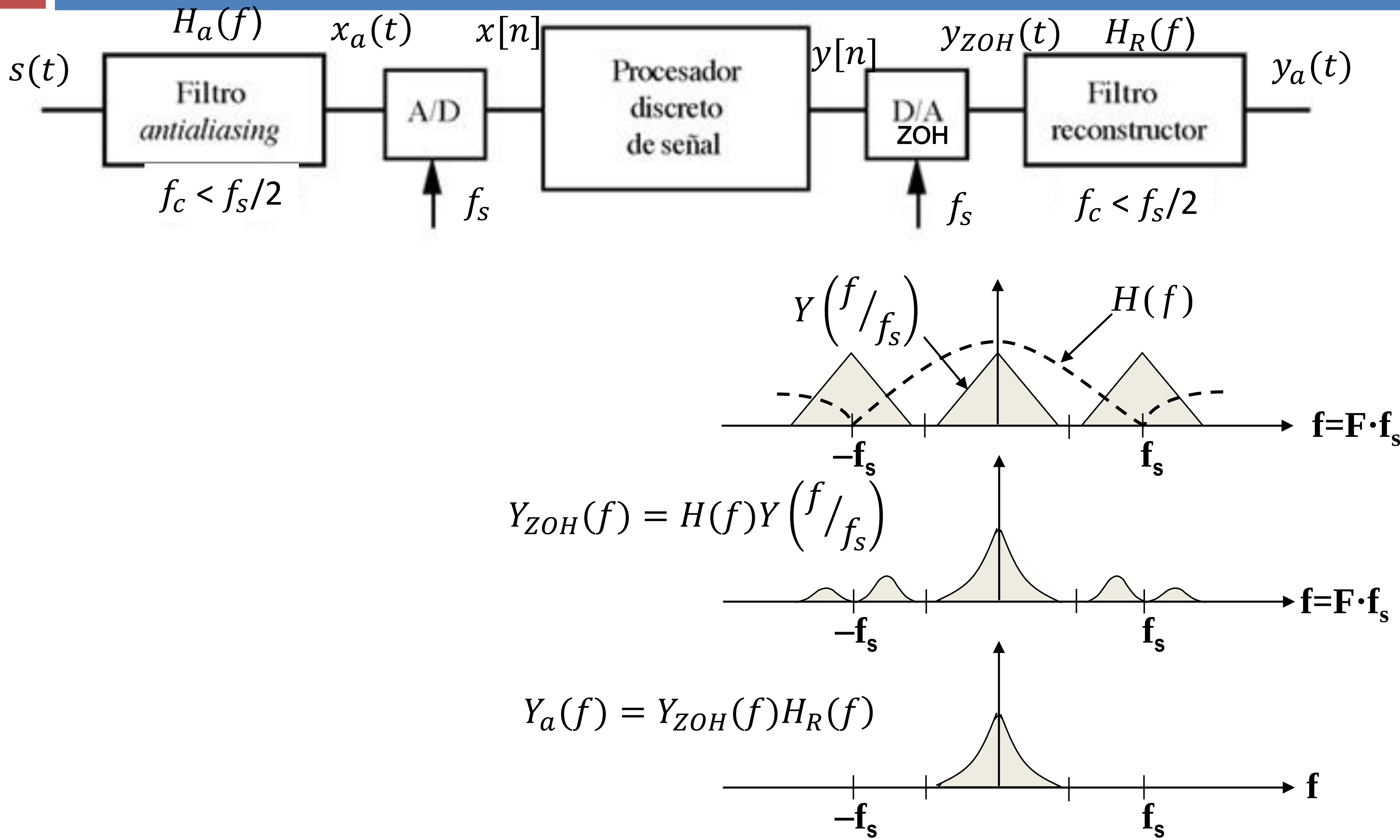
Compensation of the aperture distortion (1/sinc) and elimination of the residual aliasing



NOTE: in practice this filter is implemented in two stages:

1) *<u>Digital filter</u>* (1-periodic frequency response) with response sinc^{-1} that corrects the aperture distortion
2) *<u>Analog low-pass filter</u>* that eliminates the residual aliasing

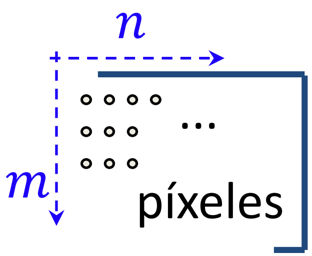
Resumen


$H_a(f)$
$x_a(t)$
$x[n]$
$s(t)$
Filtro antialiasing
A/D
Procesador discreto de señal
$y[n]$
D/A ZOH
$y_{ZOH}(t)$
$H_R(f)$
Filtro reconstructor
$y_a(t)$
$f_c < f_s/2$
f_s
f_s
$f_c < f_s/2$
$Y\left(f/f_s\right)$
$H(f)$
f=F·f$_s$
–f$_s$
f$_s$
$Y_{ZOH}(f) = H(f)Y\left(f/f_s\right)$
f=F·f$_s$
–f$_s$
f$_s$
$Y_a(f) = Y_{ZOH}(f)H_R(f)$
f
–f$_s$
f$_s$

Tema 5. Conversión A/D y D/A y cambio de la frecuencia de muestreo

1. Muestreo y conversión A/D (3/12/2019)
2. Reconstrucción y conversión D/A (10/12/2019)
3. Conversión A/D y D/A de imágenes
4. Cambio de la frecuencia de muestreo
   - 4.1 Diezmado (downsampling) (13/12/2019)
   - 4.2 Interpolación (upsampling) (17/12/2019)

Muestreo 2D



256x256 píxeles

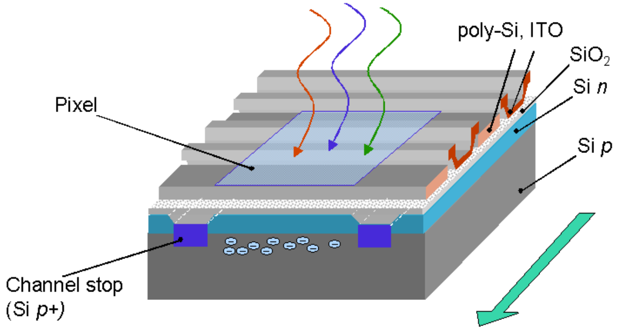
El número de píxeles determina la resolución espacial

64x64

Sensores de imagen

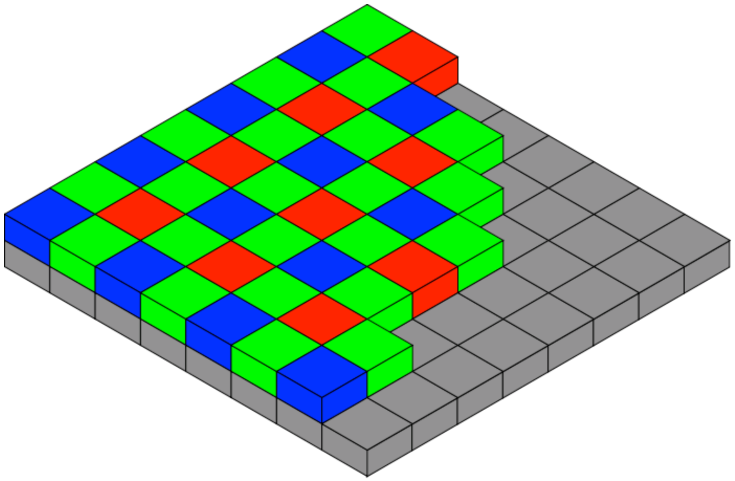
U5

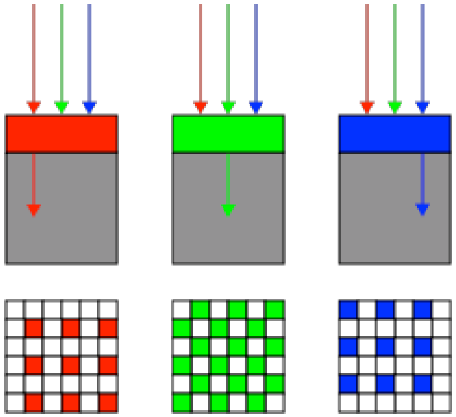
- Transforman la intensidad de luz recibida en corriente eléctrica
- Número de electrones producido proporcional a la cantidad de luz recibida
- Las dos tecnologías más importantes son:
  - **CCD** (charge-coupled device): la señal eléctrica producida por cada fotosito se envía al exterior y desde allí se amplifica y se digitaliza.
  - **CMOS** (Complementary metal-oxide-semiconductor): incorpora un amplificador de la señal eléctrica en cada detector individual (fotosito) y es común incluir el conversor digital en el propio chip. Menor consumo y mayor velocidad que CCD.



Filtro de Bayer

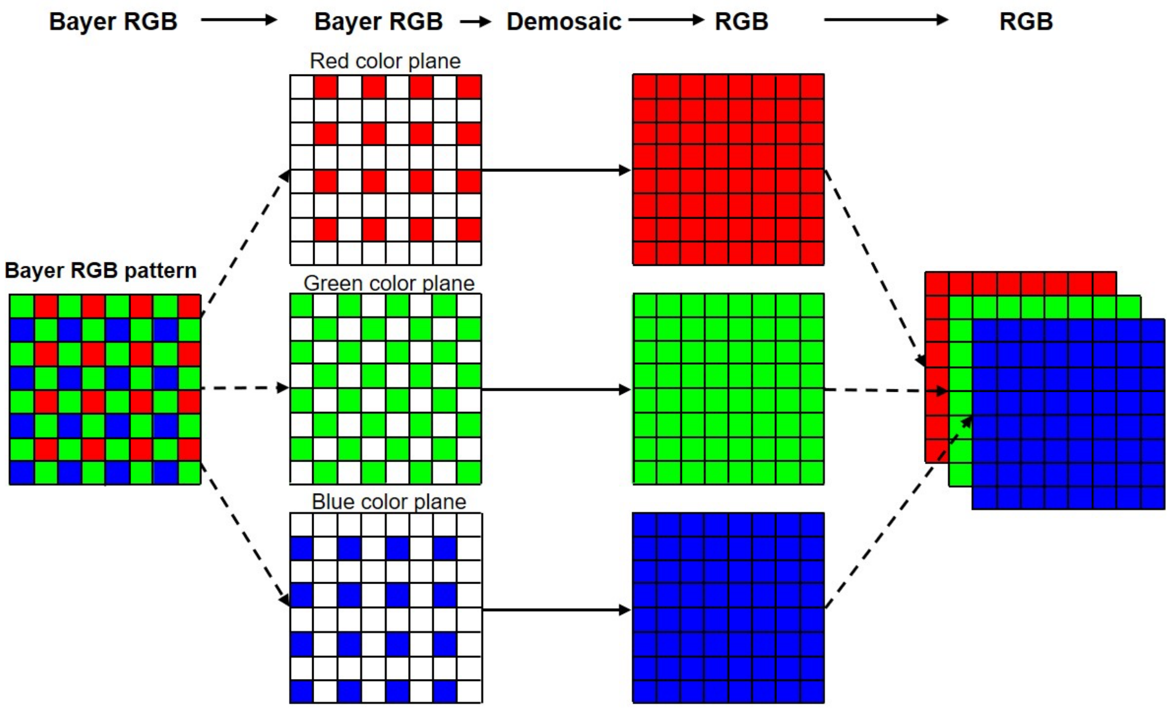
- Cada fotosito registra una componente de color: roja, azul y verde (RGB)
- Para establecer la componente a la que responde cada fotosito se utiliza un patrón o filtro de Bayer
  - Cada 4 píxeles se forma una trama con una componente roja, otra azul y dos verdes





Filtro de Bayer

- Mediante un proceso de interpolación se calcula el valor de las tres componentes en todos los píxeles de la imagen.



Source: https://theailearner.com/2018/10/28/bayer-filter/

Conversión D/A de imágenes

Para reconstruir una imagen (a partir de su versión muestreada), **el propio ojo actúa como filtro reconstructor.**

Imagen digital con diferentes resoluciones (expresadas en píxeles por dimensión)

40x40 80x80 320x320

- Si el número de píxeles es suficientemente alto la imagen se percibe como analógica => Depende del SVH, tamaño de la imagen y distancia de observación.
  Si nos alejamos, percibiremos las tres imágenes con calidad similar (seguramente baja).
- Los sistemas de alta definición aumentan el número de píxeles: el espectador puede acercarse más a la pantalla o tener pantallas más grandes.

Conversión D/A de imágenes

¿Qué # píxeles debe tener una imagen para que el SVH no sea capaz de distinguir su carácter digital?

El ángulo con el que se observan dos píxeles adyacentes ha de ser menor o igual que el límite impuesto por la agudeza visual.

- Visión normal: 30 ciclos por grado (60 píxeles por grado)
- Límite SVH: 60 ciclos por grado (120 píxeles por grado)

Una **pantalla** puede denominarse **retina** (concepto introducido por Apple a partir del iPhone4) si el # píxeles que ofrece, desde la distancia a la que tiene que ser observada, está alrededor de 60 píxeles por grado

iPhone 5S: 1136x640 píxeles. H=5.86 cm= 640 píxeles.

En en 1 grado = $\frac{2\pi}{360}$ radianes, para una distancia de observación L=30 cm=5.12H=3276.8 píxeles:

$$\frac{2\pi}{360} \backsimeq \frac{h}{L} \Rightarrow h \backsimeq 57\,píxeles$$

Formato 8K = 7680 x 4320 (WxH)

$$\frac{2\pi}{360} \backsimeq \frac{h}{L} = \frac{60\,píxeles}{L} \Rightarrow L \backsimeq 60\frac{360}{2\pi}\,píxeles = 0.8H$$

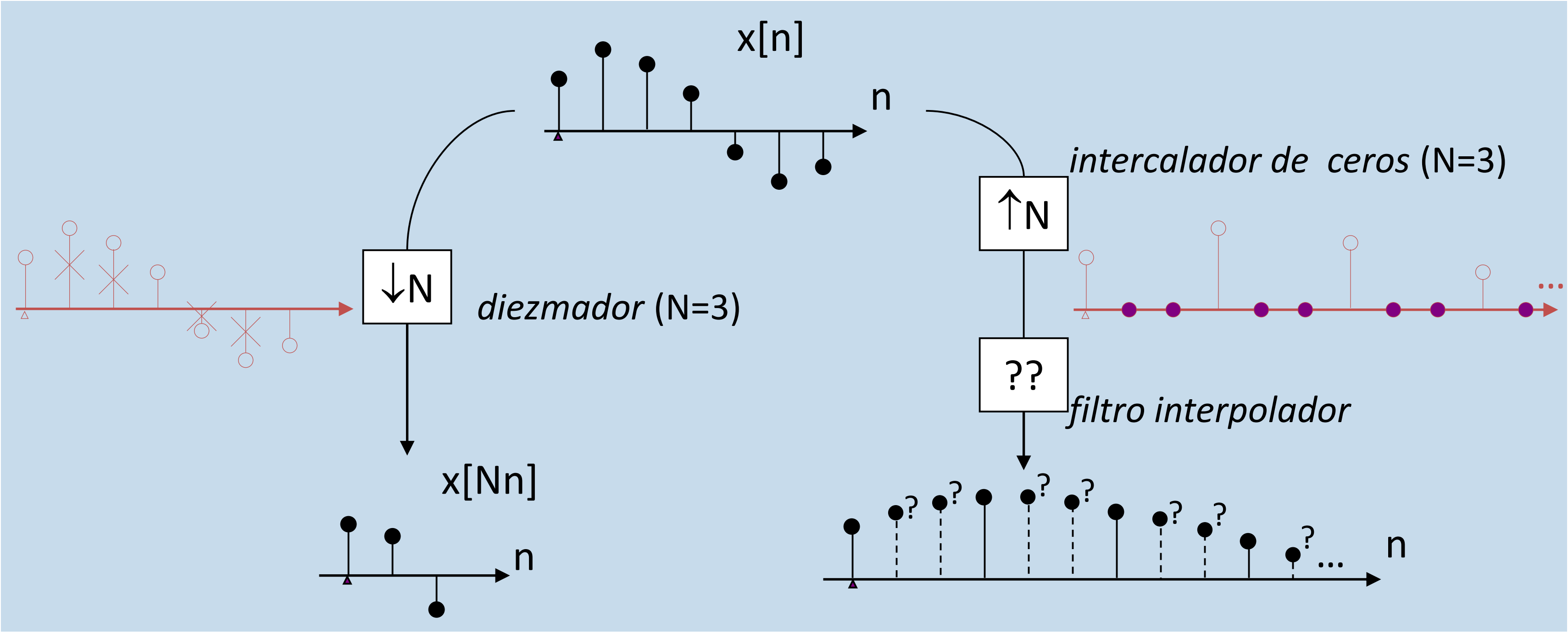
Formato TV HD = 1920 x 1080 (WxH)

$$\frac{2\pi}{360} \backsimeq \frac{h}{L} = \frac{60\,píxeles}{L} \Rightarrow L \backsimeq 60\frac{360}{2\pi}\,píxeles = 3.2H$$

Tema 5. Conversión A/D y D/A y cambio de la frecuencia de muestreo

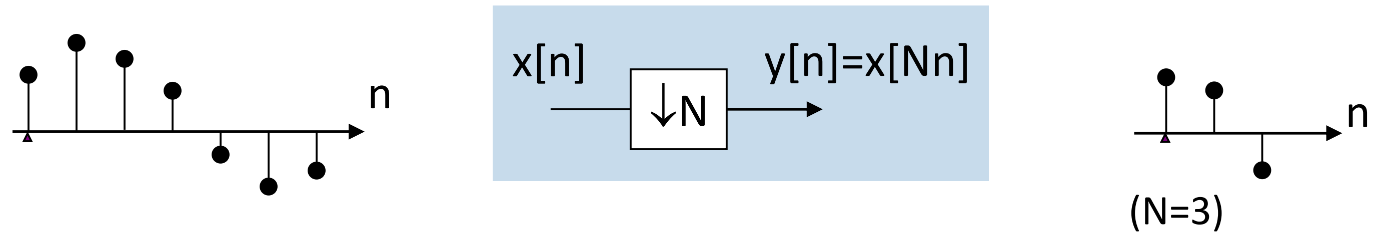
1. Muestreo y conversión A/D (3/12/2019)
2. Reconstrucción y conversión D/A (10/12/2019)
3. Conversión A/D y D/A de imágenes
4. Cambio de la frecuencia de muestreo
   4.1 Diezmado (downsampling) (13/12/2019)
   4.2 Interpolación (upsampling) (17/12/2019)

Cambio de la frecuencia de muestreo



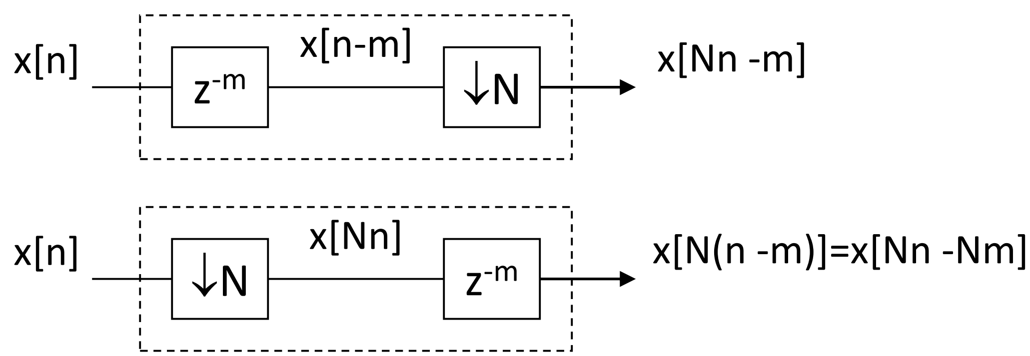
Diezmado

U5

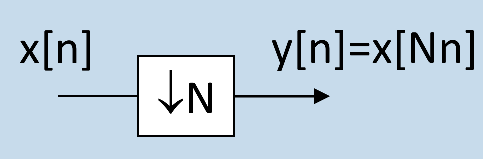


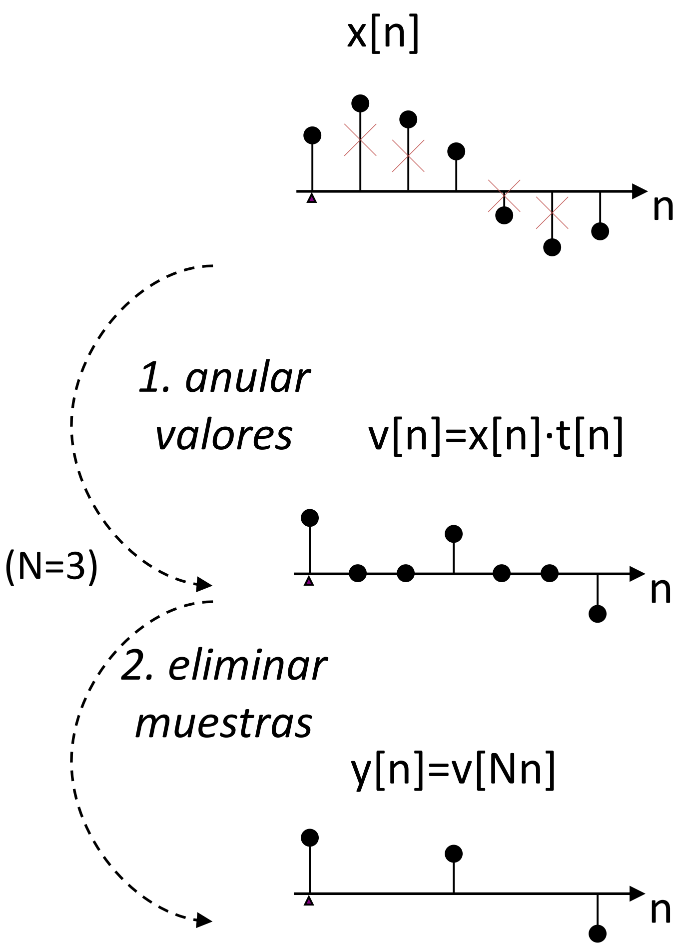
Un diezmador (N>1) es un sistema discreto

- lineal
- no causal: Nn>n si n>0.
- estable BIBO: “bounded input bounded output”.
- variante en el tiempo: los dos sistemas siguientes no son equivalentes



Descomposición teórica del diezmado

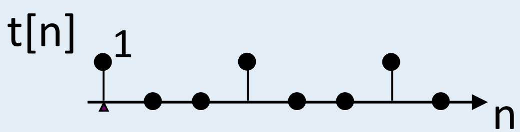




Secuencia intermedia:

$$v[n]=\begin{cases} x[n] & n=0,\pm N,\pm 2N\ldots \\ 0 & otro \quad n \end{cases} = x[n]\cdot t[n]$$

donde



$$t[n]=\sum_{r=-\infty}^{\infty}\delta[n-rN]=\frac{1}{N}\sum_{k=0}^{N-1}e^{j\frac{2\pi}{N}kn}$$

Análisis frecuencial del diezmado

U5

$$x[n] \longrightarrow \boxed{\downarrow N} \longrightarrow y[n]=x[Nn]$$

1. Transformada de la secuencia intermedia

$$V(F)=\sum_{n=-\infty}^{\infty} v[n]e^{-j2\pi Fn}=\sum_{n=-\infty}^{\infty} x[n]\Bigg(\underbrace{\frac{1}{N}\sum_{k=0}^{N-1} e^{j\frac{2\pi}{N}kn}}_{t[n]}\Bigg)e^{-j2\pi Fn}=$$

$$=\frac{1}{N}\sum_{k=0}^{N-1}\sum_{n=-\infty}^{\infty} x[n]e^{-j2\pi\left(F-\frac{k}{N}\right)n}=\frac{1}{N}\sum_{k=0}^{N-1} X\left(F-\frac{k}{N}\right)$$

réplicas espectrales cada 1/N

2. Transformada de la secuencia diezmada: y[n]=x[Nn]=v[Nn]

$$Y(F)=\sum_{n=-\infty}^{\infty} y[n]e^{-j2\pi Fn}=\sum_{n=-\infty}^{\infty} v[Nn]e^{-j2\pi Fn}=$$

$$\underset{m=Nn}{\uparrow}{=}\sum_{m=-\infty}^{\infty} v[m]e^{-j2\pi Fm/N}=V\left(\frac{F}{N}\right)=\frac{1}{N}\sum_{k=0}^{N-1} X\left(\frac{F}{N}-\frac{k}{N}\right)$$

$m=Nn$ posible, ya que $v[m]=0 \quad m\neq Nn$

expansión frecuencial xN

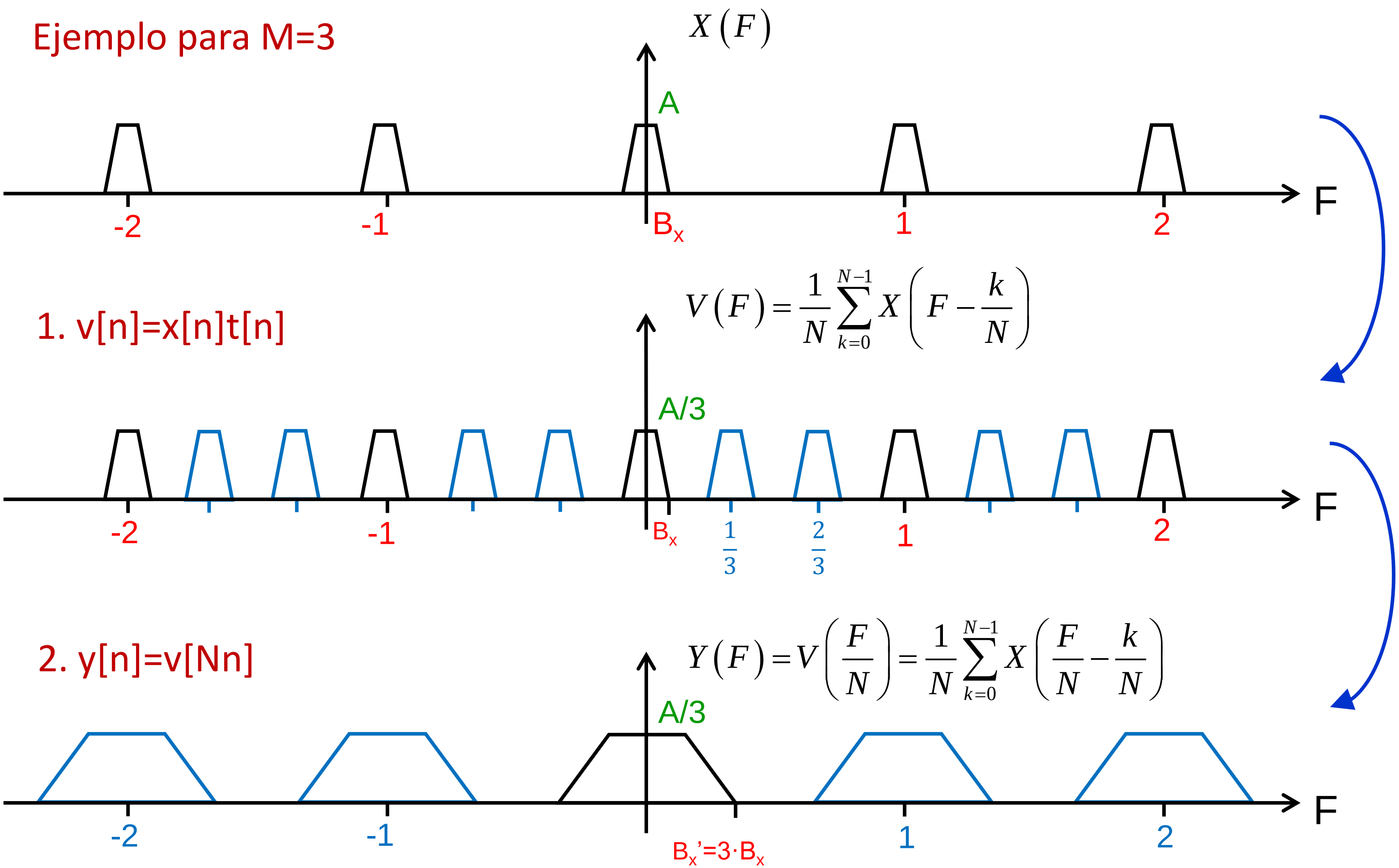
Análisis frecuencial del diezmado

U5

Ejemplo para M=3

1. v[n]=x[n]t[n]

2. y[n]=v[Nn]

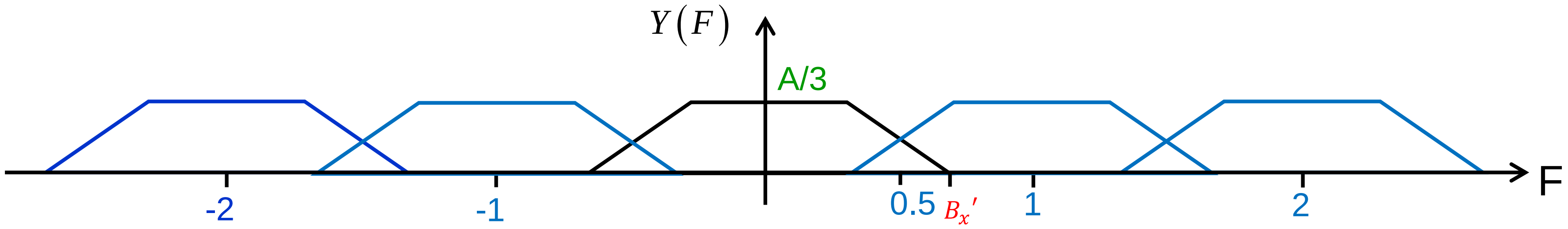


generación de N-1 nuevas réplicas con amplitud dividida por N

Expansión frecuencial por un factor N

Aliasing (1)

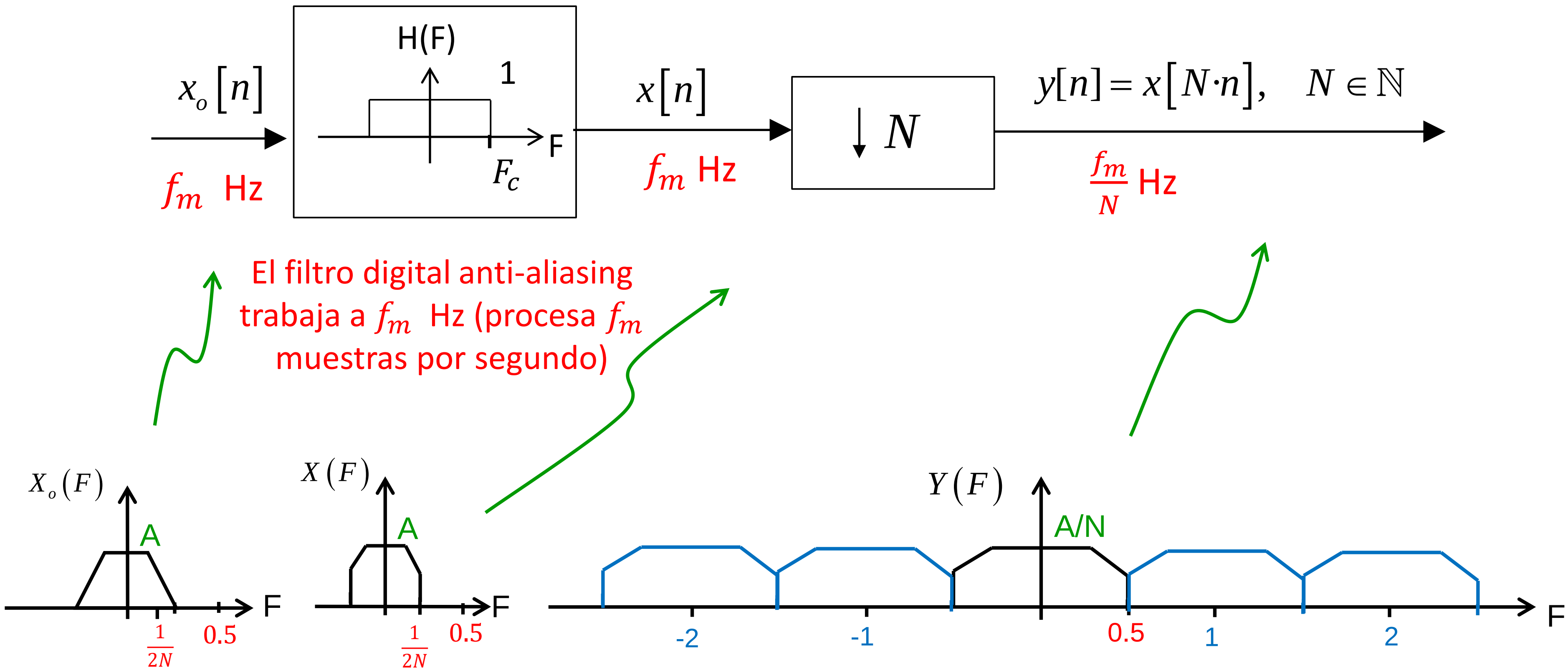
- Si $B_x{}' = N \cdot B_x > 0.5$ (i.e., si $B_x > \frac{1}{2N}$) habrá aliasing



Solución: aplicar un filtro digital anti-aliasing con frecuencia de corte $\mathrm{F}_c = \frac{1}{2\mathrm{N}}$, antes de diezmar

Aliasing (2)

- Para evitar el aliasing aplicamos un filtro digital anti-aliasing con frecuencia de corte $\mathrm{F}_c = \frac{1}{2\mathrm{N}}$:



Ejemplo (1)

Milstein plays bach, sonata #1 ,presto (mvt 4)

Original, fm = 44.1 kHz (CD)

https://www.youtube.com/watch?v=k38H25fDCb4

Diezmada por N

- N= 2, fm’ = 22 kHz
- N= 4, fm’ = 11 kHz
- N= 8, fm’ = 5.5 kHz
- N= 16, fm’ = 2.8 kHz

```
#%% Load audio
sf, data = wavfile.read('Milstein.wav')
data=data/2**15
x = data[0:10*sf,:]
print('Original sampling frequency is ',sf, 'Hz')
sound(x, sf=sf)


#%%Decimation by 2, SAMPLING FREQUENCY IS 22 kHz
y1 = x[::2,:]
fs = int(0.5*sf)
print('sampling frequency is ',fs, 'Hz')
sound(y1, sf = fs)
wavfile.write('Decimated_by_2.wav', fs, (y1*2**15).astype('int16'))
```

Ejemplo (2)

U5

Milstein plays bach, sonata #1 ,presto (mvt 4)

https://www.youtube.com/watch?v=k38H25fDCb4

Original, fm = 44.1 kHz (CD)

Diezmada:

- N= 2, fm’ = 22 kHz
- N= 4, fm’ = 11 kHz
- N= 8, fm’ = 5.5 kHz
- N= 16, fm’ = 2.8 kHz

Filtrada antes de diezmar

```
b, a = signal.iirdesign(wp = 1/2.05, ws=
1/2.0, gstop= 50, gpass=0.1, ftype='ellip')
z = np.zeros_like(x)
z[:,0] =signal.lfilter(b, a, x[:,0])
z[:,1] =signal.lfilter(b, a, x[:,1])
sound(z[::2,:], sf=fs)
wavfile.write('Filtered_and_dec_by_2.wav',
fs, (2**15*z[::2,:]).astype('int16'))
```

Ejemplo de diezmado 2D

original

```
x = plt.imread('barbara.png')
plt.figimage(x, cmap='gray')
```

Diezmada
(M=N=2)

```
y1 = x[::2, ::2]
plt.figimage(x, cmap='gray')
```

Filtrada paso bajo antes de diezmado
(M=N=2)

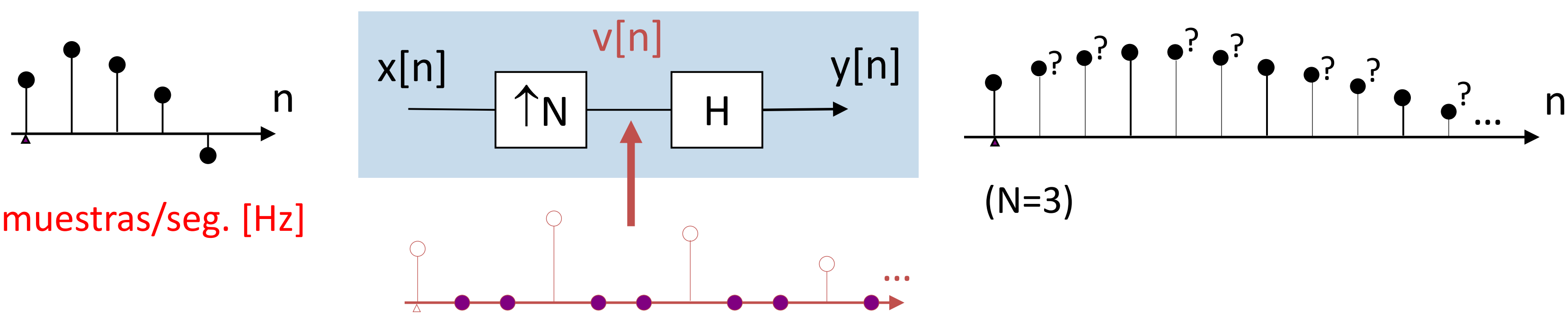
```
h = 1/9*np.ones((3,3)))
z = convolve2d(x, h, mode=‘same’)
y2 = z[::2, ::2]
```

Interpolación de secuencias discretas

- La interpolación es una operación que comprende los pasos siguientes:

  1. Se intercalan N-1 ceros entre cada dos muestras consecutivas de la secuencia original (simbolizado por ↑N)

  2. Un filtro “ interpolador” adecuado, calcula los valores de las muestras intercaladas



f_m muestras/seg. [Hz]

$f_m{}' = N \cdot f_m$ muestras/seg. [Hz]

Análisis frecuencial de la interpolación

U5

1. Intercalado de ceros
$$v[n]=\begin{cases} x\left[\dfrac{n}{N}\right], & n=mN, \quad m\in\mathbb{Z} \\ 0, & \text{otherwise} \end{cases}$$

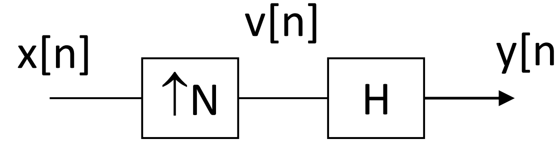
$$V(F)=\sum_{n=-\infty}^{\infty} v[n]e^{-j2\pi Fn}=\sum_{m=-\infty}^{\infty} v[mN]e^{-j2\pi FNm}=\sum_{m=-\infty}^{\infty} x[m]e^{-j2\pi FNm}=X(NF)$$

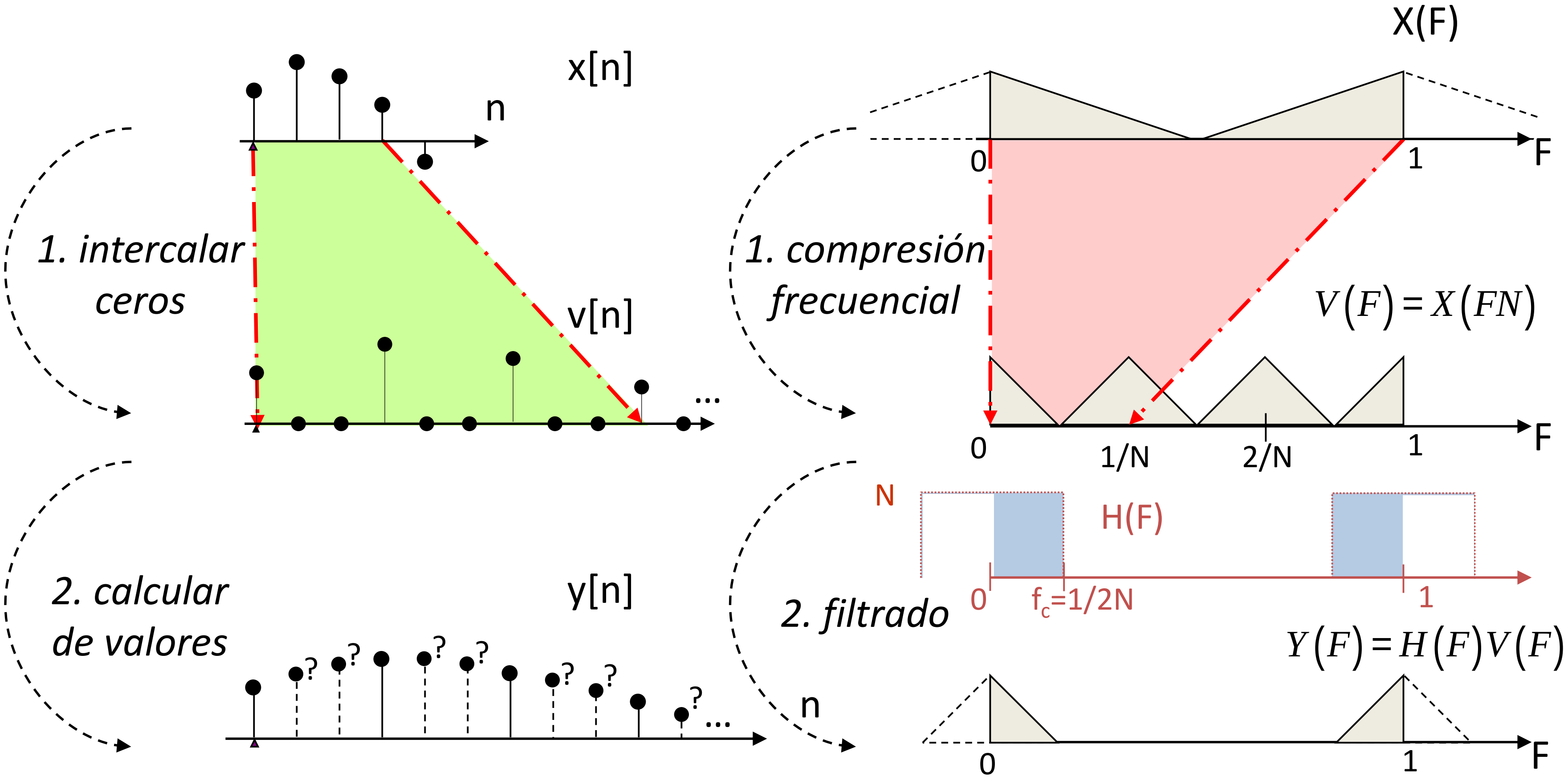
compresión frecuencial en un factor N
-- no hay aliasing !!!! --

2. Filtrado: y[n]=v[n]*h[n]

$$Y(F)=V(F)H(F)=X(NF)H(F)$$

Esquema

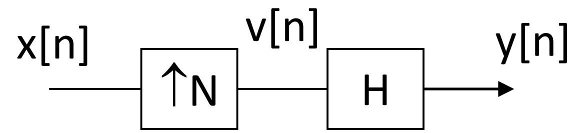


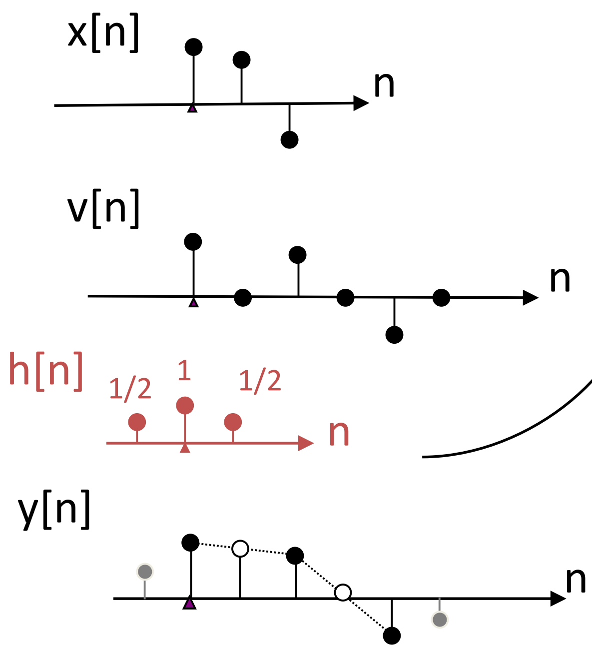


$$Y(F) = H(F)X(FN)$$

Ejemplo: interpolación lineal

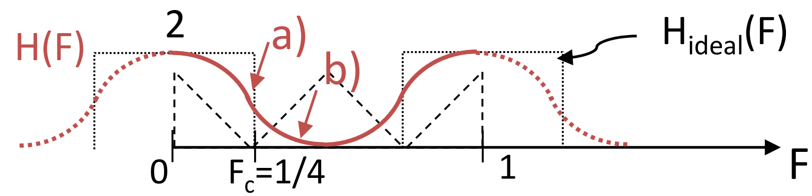
- Regla de interpolación lineal con N=2

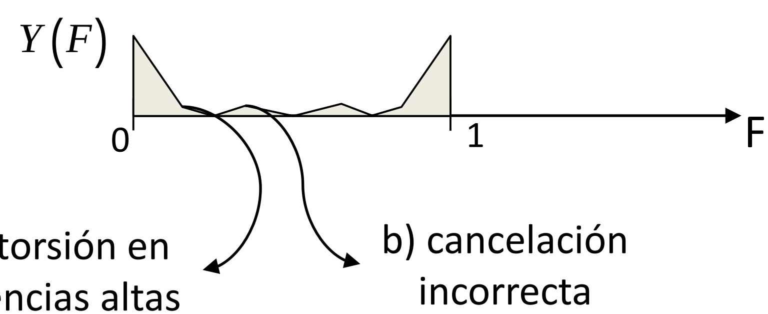




$$h[n] = \{1/2, \underline{1}, 1/2\} \quad \text{(no causal)}$$

$$H(F) = \frac{1}{2} e^{j2\pi F} + 1 + \frac{1}{2} e^{-j2\pi F} = 1 + \cos(2\pi F)$$





Ejemplo de interpolación (1)

$x[m,n]$

```
x = plt.imread('lena.bmp')
M, N = np.shape(x)
plt.figimage(x, cmap='gray')
```

1. Intercalado de ceros

```
v = np.zeros((2*M, 2*N))
v[::2, ::2] = x
plt.figimage(v, cmap='gray')
```

Ejemplo de interpolación (2)

2. Filtrado

Filtro Nearest Neighbour:
copia el valor de la muestra más cercana

$$h[m,n]=\begin{pmatrix} \underline{1} & 1 \\ 1 & 1 \end{pmatrix}$$

```
h = np.ones((2, 2))
y1 = convolve2d(v, h, mode='same')
plt.figimage(y1, cmap='gray')
```

Ejemplo de interpolación (3)

2. Filtrado

Interpolador bilineal:

$$h[m,n] = \begin{pmatrix} 0.5 \\ \underline{1} \\ 0.5 \end{pmatrix} * \begin{pmatrix} 0.5 & \underline{1} & 0.5 \end{pmatrix}$$

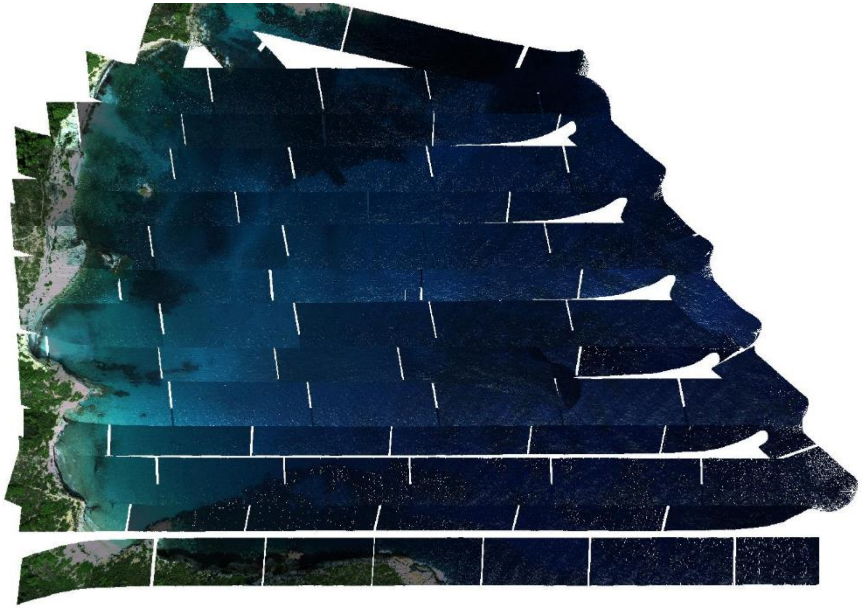
$$= \begin{pmatrix} 0.25 & 0.5 & 0.25 \\ 0.5 & \underline{1} & 0.5 \\ 0.25 & 0.5 & 0.25 \end{pmatrix}$$

```
a = np.array([0.5, 1, 0.5])
h = convolve2d(a[np.newaxis, :], a[:, np.newaxis])
y2 = convolve2d(v, h, mode='same')
plt.figimage(y2, cmap='gray')
```

Estudio del fondo marino a baja profundidad para monitorizar las colonias de posidonia

U5

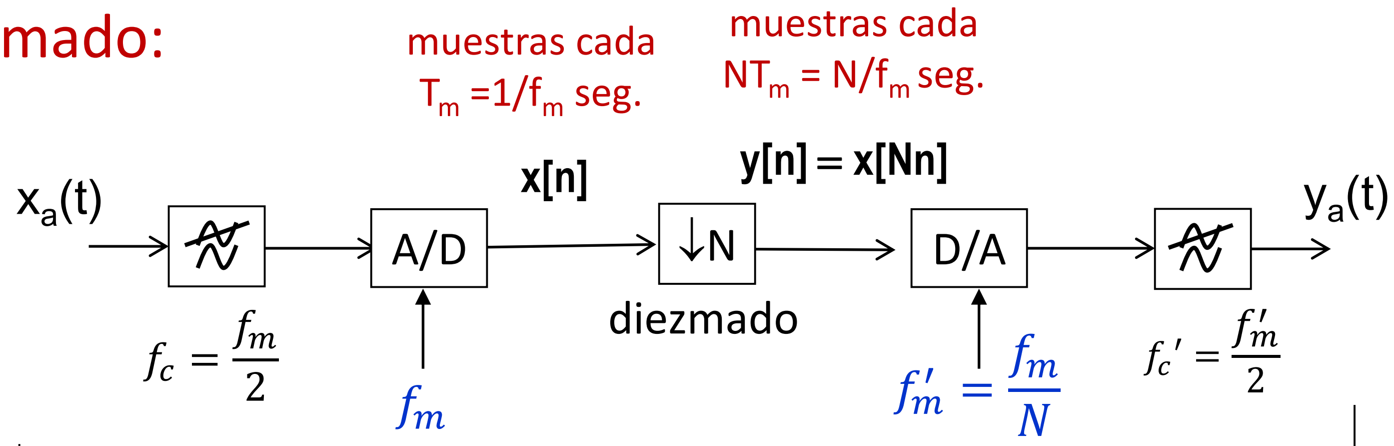
Imágenes obtenidas por un dron con sensor hyperspectral de resolución 0.2m

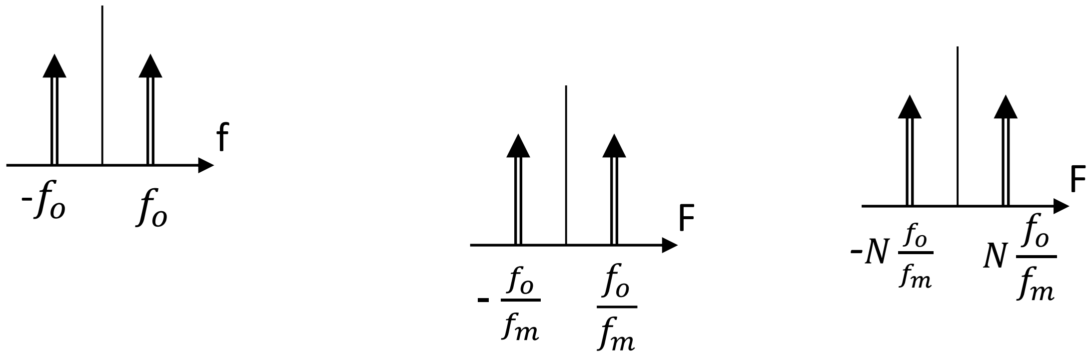


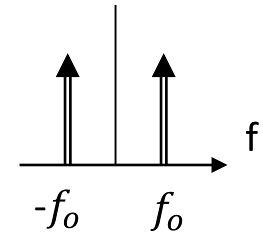
1. El primer vuelo se puede hacer sin problemas: los distintos pasos del dron se solapan correctamente
2. La segunda imagen es de otra cala para la que se han de enviar al dron los datos geográficos: faltan datos de aquellas zonas donde el dron está 'escuchando' en lugar de guardando datos.

Relación con el entorno analógico

Diezmado:







Suponiendo que $N\frac{f_o}{f_m} < 0.5$:

no se produce aliasing

Para evitar aliasing: filtro paso bajo de Fc=1/(2N) antes del diezmado (bloque ↓N)

Relación con el entorno analógico

Interpolación: