

# PROYECTO FINAL PROCOM

Transceptor con corrección de efectos de canal, reflex y  
swap de polarización

(Avances Diciembre 2023)

# Integrantes:

- Carreño Marín, Sebastián
- Fernández, Andrés
- Oliva, Agustina
- Soto, Luis Franco
- Taborda, Andrea
- Tarnoski, Santiago

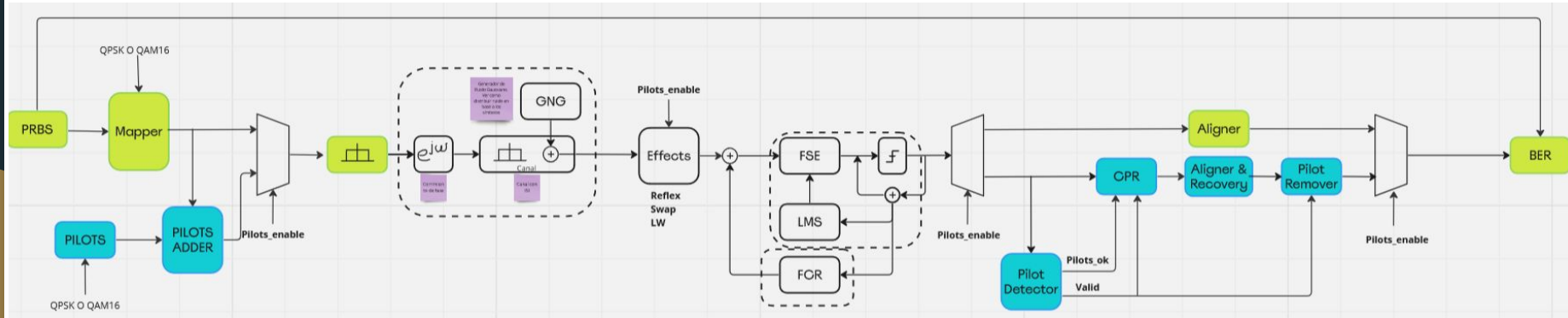
# Concepto general

El sistema cuenta con la posibilidad de dos entradas elegibles por mux. Por un lado, la PRBS con modulación QPSK y, por el otro, PRBS con modulación QAM16 con símbolos PILOTS intercalados cada una determinada cantidad de bits.

En ambos casos, al canal se le agrega un corrimiento de fase y ruido Gaussiano. También es posible añadir otros efectos, como reflex y swap. Seguido a esto, el desfase se corrige mediante un módulo FCR, el cual consiste en un PLL, y los símbolos se recuperan con el uso de un FSE en conjunto de un LMS, los cuales conforman un filtro adaptivo.

Finalmente, con un segundo mux, se selecciona el tipo de salida en base a la entrada elegida. En el caso de entrada con modulación QPSK, la salida pasa solo por la BER. Para el caso de modulación QAM16 con símbolos PILOT, es necesario agregar un Aligner, un Pilot Remover que trabaja en conjunto con un detector de Pilot, para concluir con el módulo BER.

# Diagrama en bloques del sistema completo



# División de Tareas

Integrante	Tarea
Andrés	Generación pilotos, Pilot detector y Pilot remover
Santiago	Effects block, CPR, Aligner
Sebastián	Generador de desfasaje
Franco	Filtro y generación de ruido
Andrea	Corrección de desfasaje
Agustina	Filtro adaptivo

## Generación de desfase

$$signal_{PhaseOff} = signal \cdot 1e^{j\omega t}$$

$$signal_{PhaseOff} = \{Re[signal] + jIm[signal]\} \cdot \{\cos(\omega t) + j\sin(\omega t)\}$$

$$signal_{PhaseOff} = \{Re[signal] \cdot \cos(\omega t) - Im[signal] \cdot \sin(\omega t)\} \\ + j\{Re[signal] \cdot \sin(\omega t) + Im[signal] \cdot \cos(\omega t)\}$$

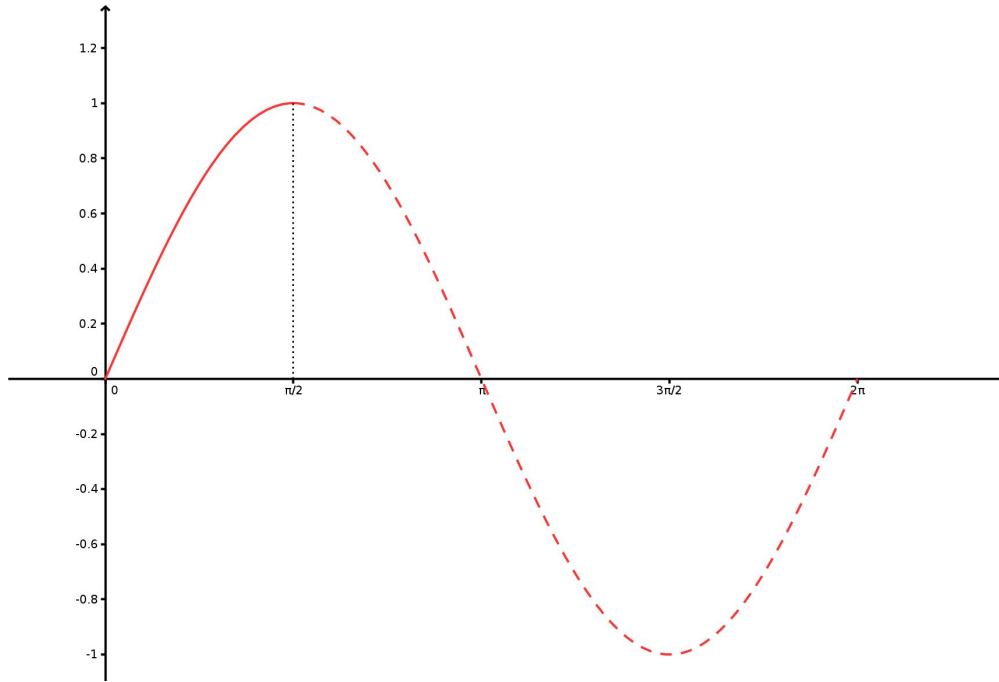
## Generación de desfase

$$signal_{PhaseOff} = signal \cdot 1e^{j\omega t}$$

$$signal_{PhaseOff} = \{Re[signal] + jIm[signal]\} \cdot \{\cos(\omega t) + j\sin(\omega t)\}$$

$$signal_{PhaseOff} = \{Re[signal] \cos(\omega t) - Im[signal] \sin(\omega t) + j\{Re[signal] \sin(\omega t) + Im[signal] \cos(\omega t)\}$$

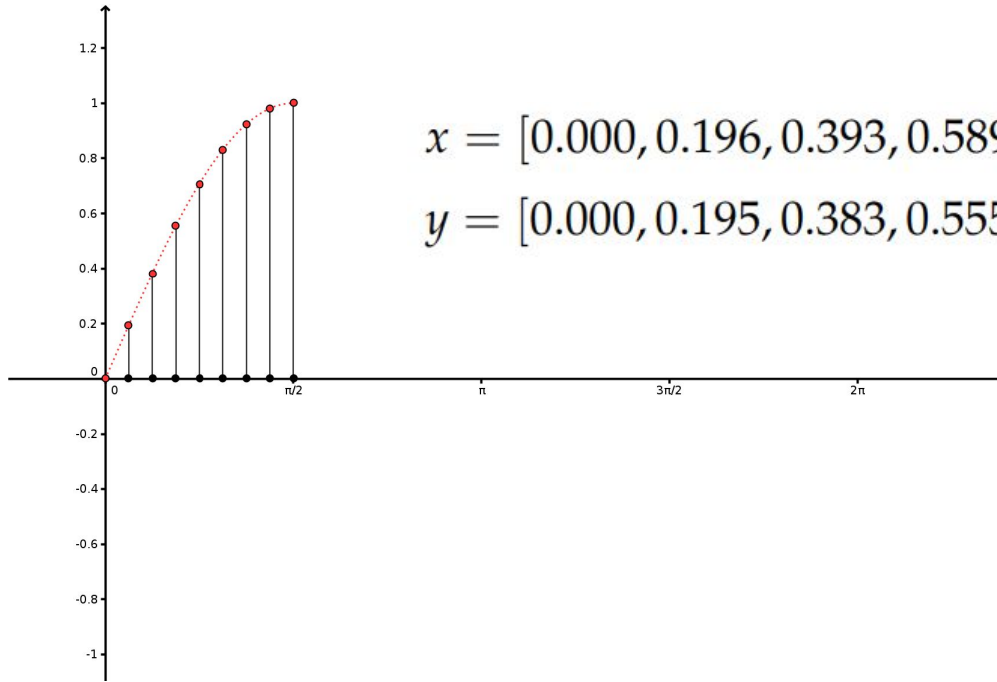
## Generación de desfase: obtención del $\sin(x)$



Para optimizar, basta con tener almacenados en memoria los valores de  $\frac{1}{4}$  de la señal senoidal.



## Generación de desfase: obtención del $\sin(x)$



$$x = [0.000, 0.196, 0.393, 0.589, 0.785, 0.982, 0.178, 1.374, 1.571]$$

$$y = [0.000, 0.195, 0.383, 0.555, 0.707, 0.831, 0.924, 0.981, 1.000]$$

# Generación de desfase: obtención del $\sin(x)$

## Pseudocódigo

```
if(x0 <= pi/2){
    Busca el índice 'k' del elemento x[k], para el que x[k] sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...kmax-1,kmax

    senX = y[k];

} else if(x0 <= pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+pi/2 sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...kmax-1,kmax

    senX = y[kmax - k];

} else if(x0 <= (3/2)*pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+pi sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...kmax-1,kmax

    senX = -y[k];

} else if(x0 <= 2*pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+(3/2)*pi sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...kmax-1,kmax

    senX = -y[kmax - k];

}
```

# Generación de desfase: obtención del $\cos(x)$ ( $\sin(x+\pi/2)$ )

## Pseudocódigo

```
x0 = x0 + pi/2;
if(x0 <= pi/2){
    Busca el índice 'k' del elemento x[k], para el que x[k] sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...,kmax-1,kmax

    cosX = y[k];

} else if(x0 <= pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+pi/2 sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...,kmax-1,kmax

    cosX = y[kmax - k];

} else if(x0 <= (3/2)*pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+pi sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...,kmax-1,kmax

    cosX = -y[k];

} else if(x0 <= 2*pi){
    Busca el índice 'k' del elemento x[k], para el que x[k]+(3/2)*pi sea cercano a x0.
    La variable 'k' se desplaza k=0,1,...,kmax-1,kmax

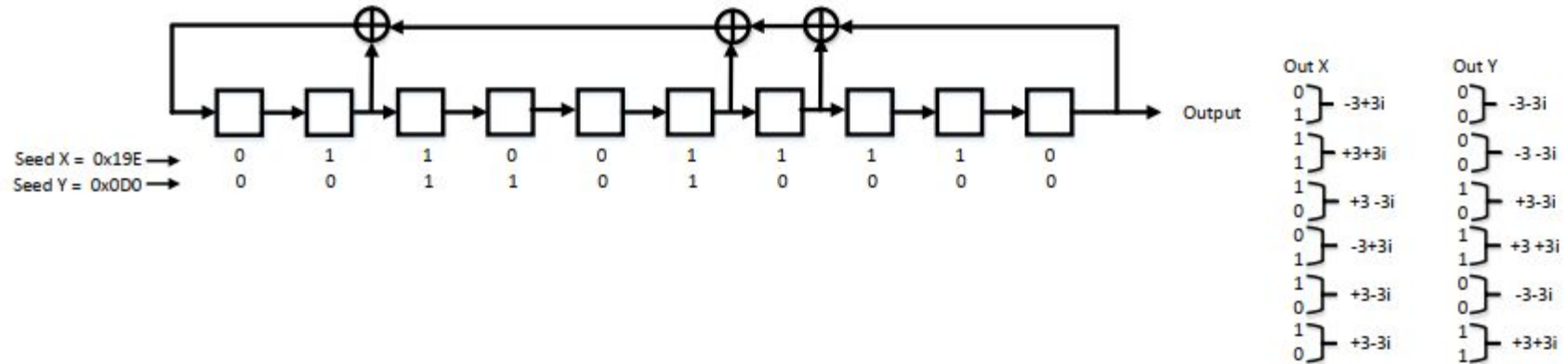
    cosX = -y[kmax - k];

}
```

# Generación de Símbolos Pilotos

**OIF-400ZR-01.0**

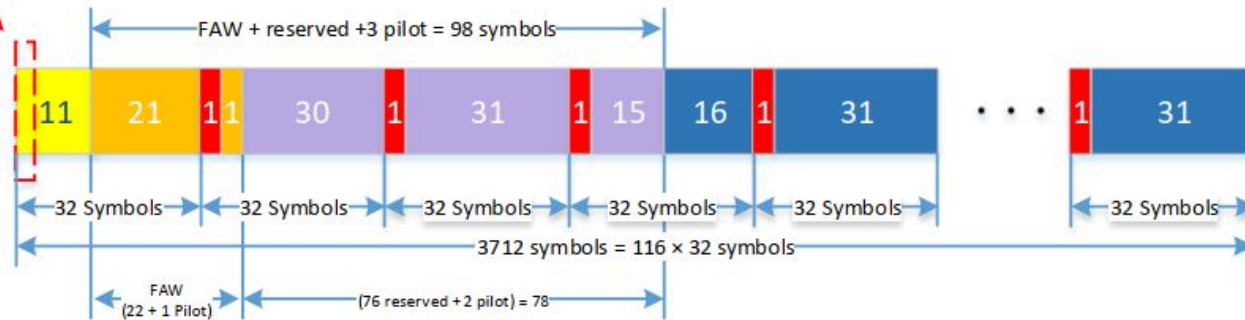
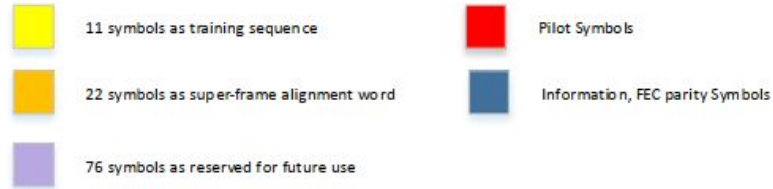
Generator polynomial	Seed X	Seed Y
$x^{10} + x^8 + x^4 + x^3 + 1$	0x19E	0x0D0



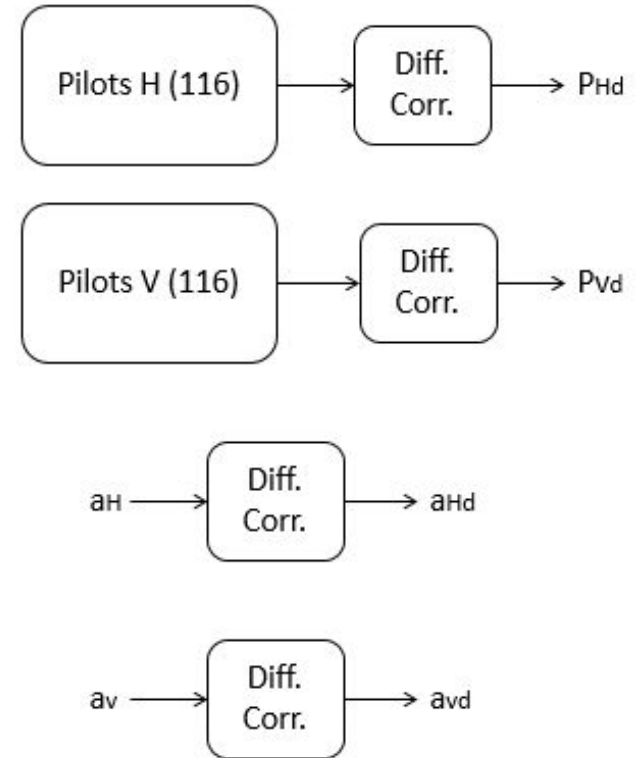
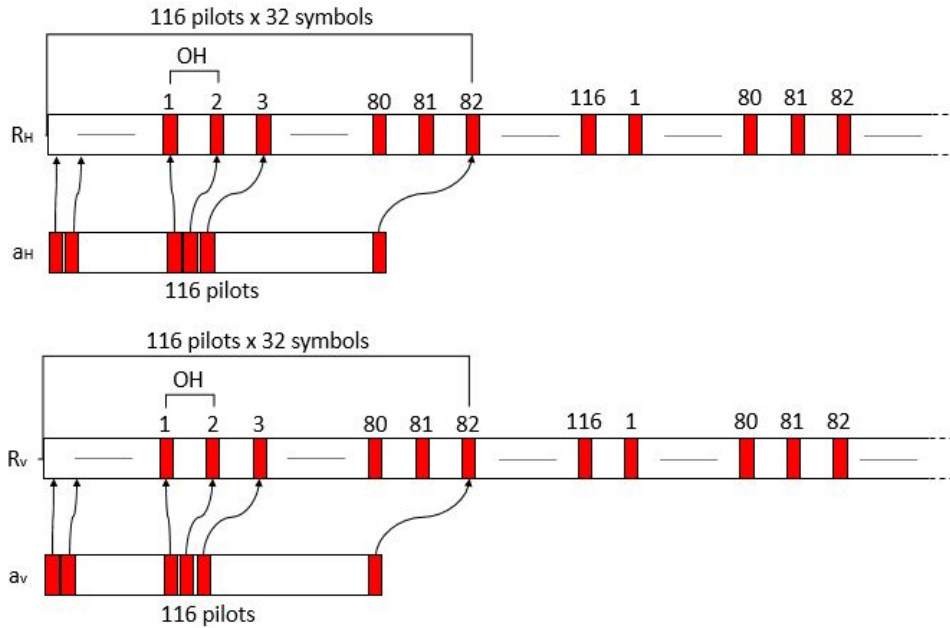
# Generación de Símbolos Pilotos

Since 1<sup>st</sup> symbol is known QPSK symbol it can be processed as a Pilot









Seeds for pilot PRBS selected so that this is a sequence

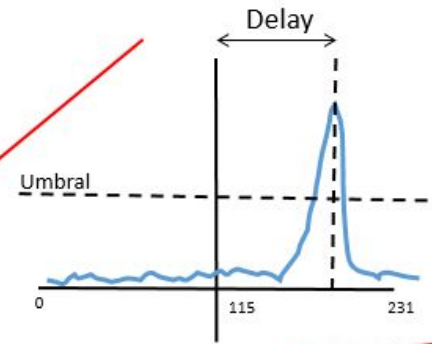


# Pilot Detector



# Pilot Detector

XCORR( Pxd , axd )	Pol H	Pol V
1		
2		
⋮	⋮	⋮
20		
⋮	⋮	⋮
32 (OH)		



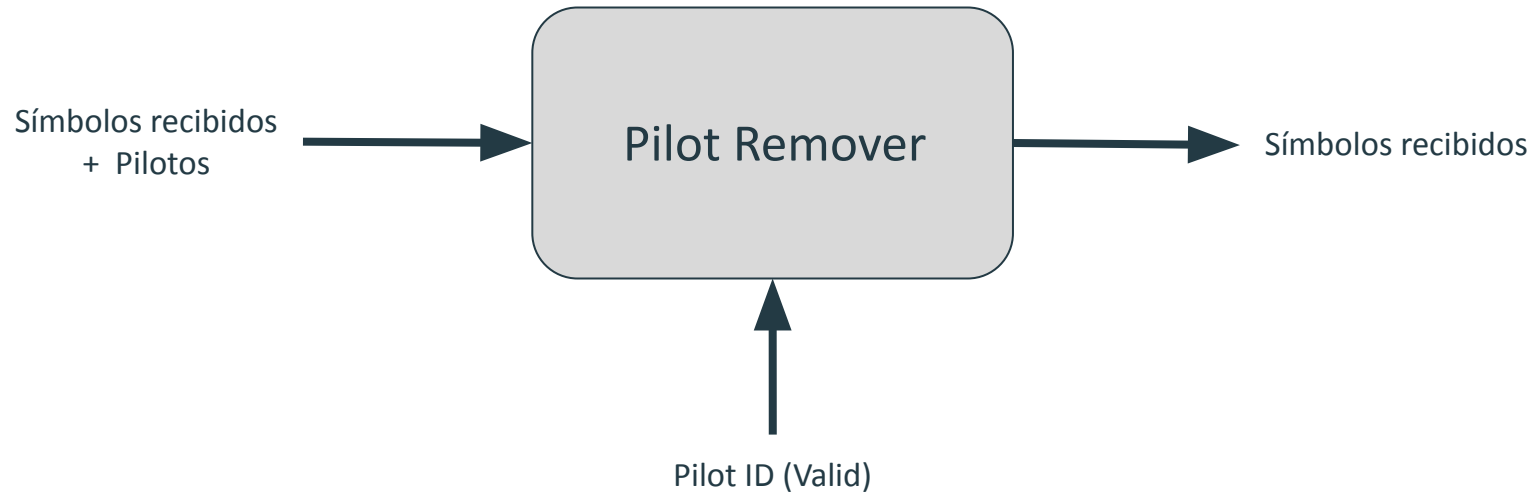
# Pilot Detector

	P <sub>HI</sub>	P <sub>HQ</sub>	P <sub>VI</sub>	P <sub>VQ</sub>
R <sub>HI</sub>	(+) (-) (+)		(+)	
R <sub>HQ</sub>		(+) (+) (+)		(+)
R <sub>VI</sub>	(+) (+)		(+) (-)	
R <sub>VQ</sub>		(+) (+)		(+) (-)

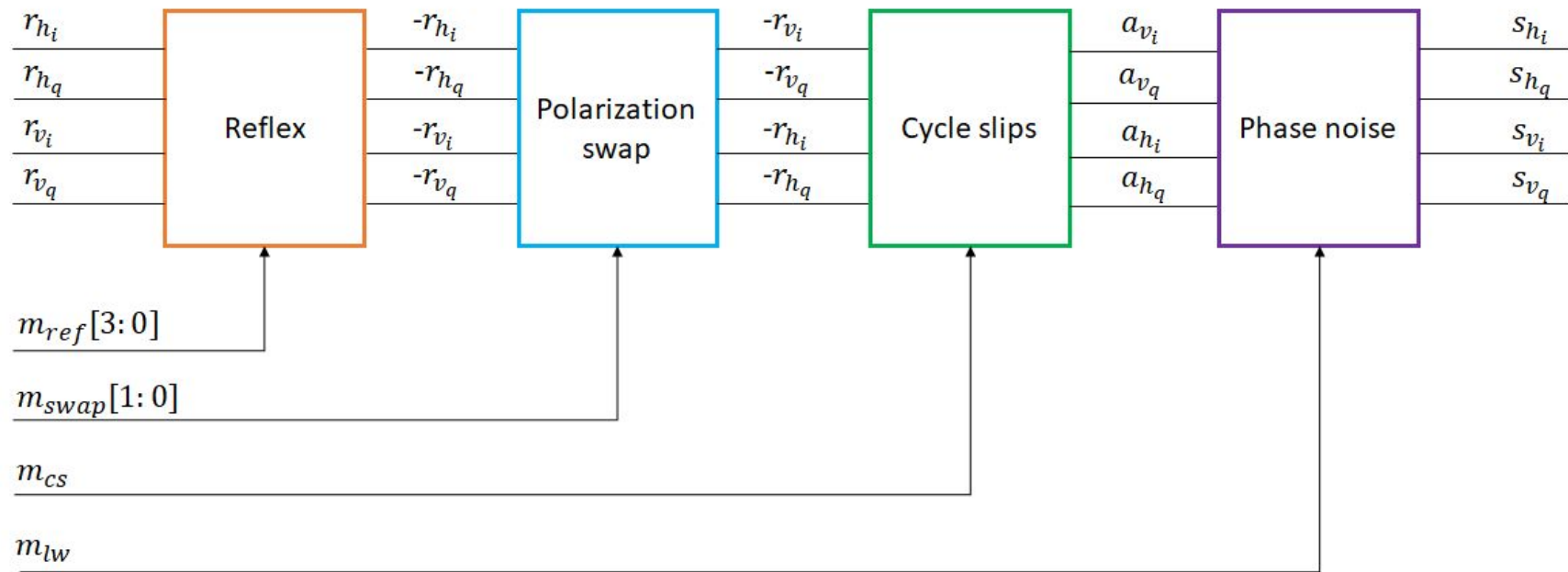
- OK
- Reflex
- Swap
- Error



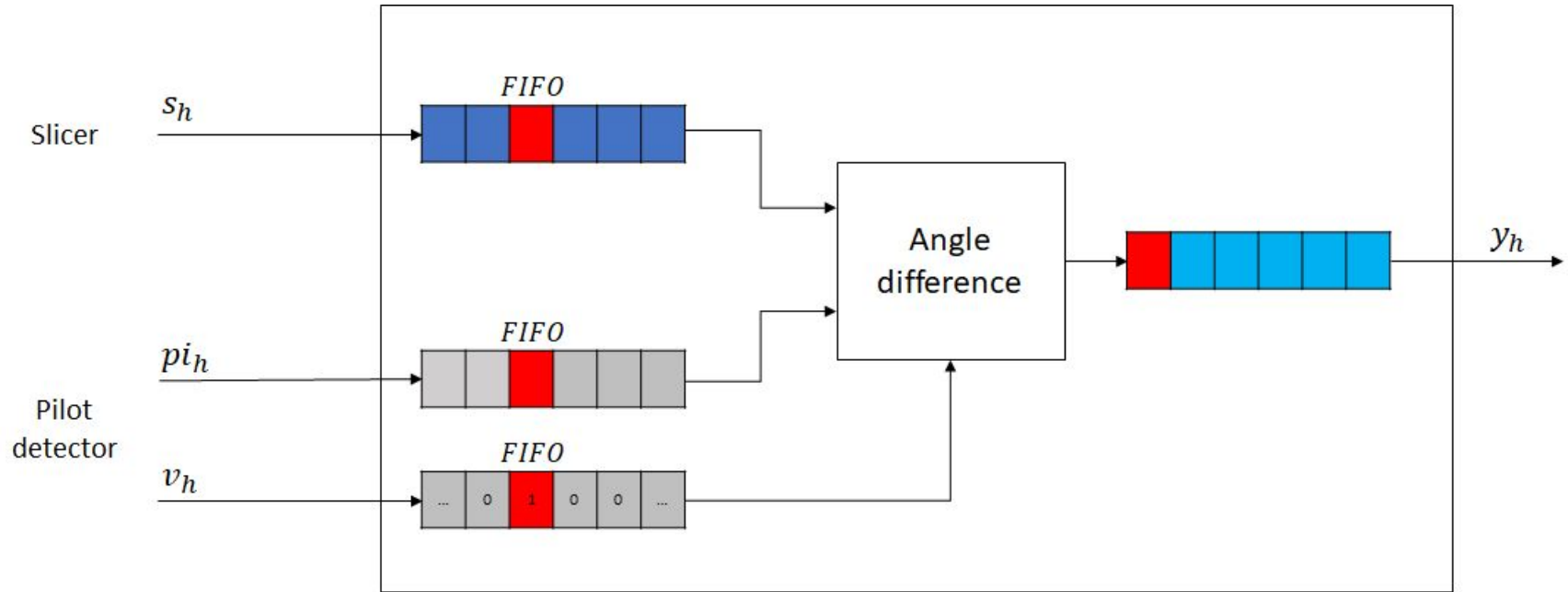
# Pilot Remover



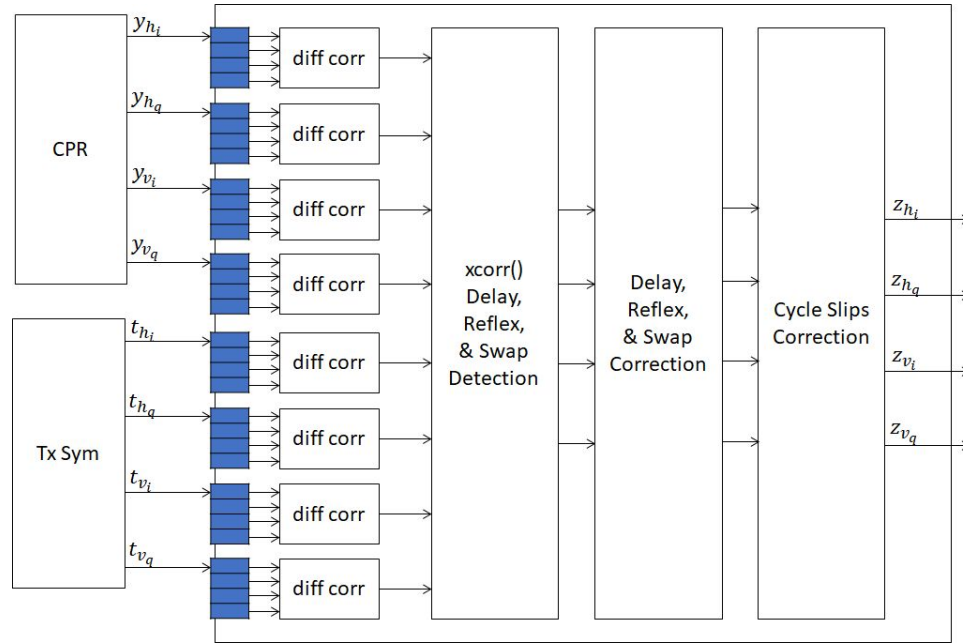
# Effects generator



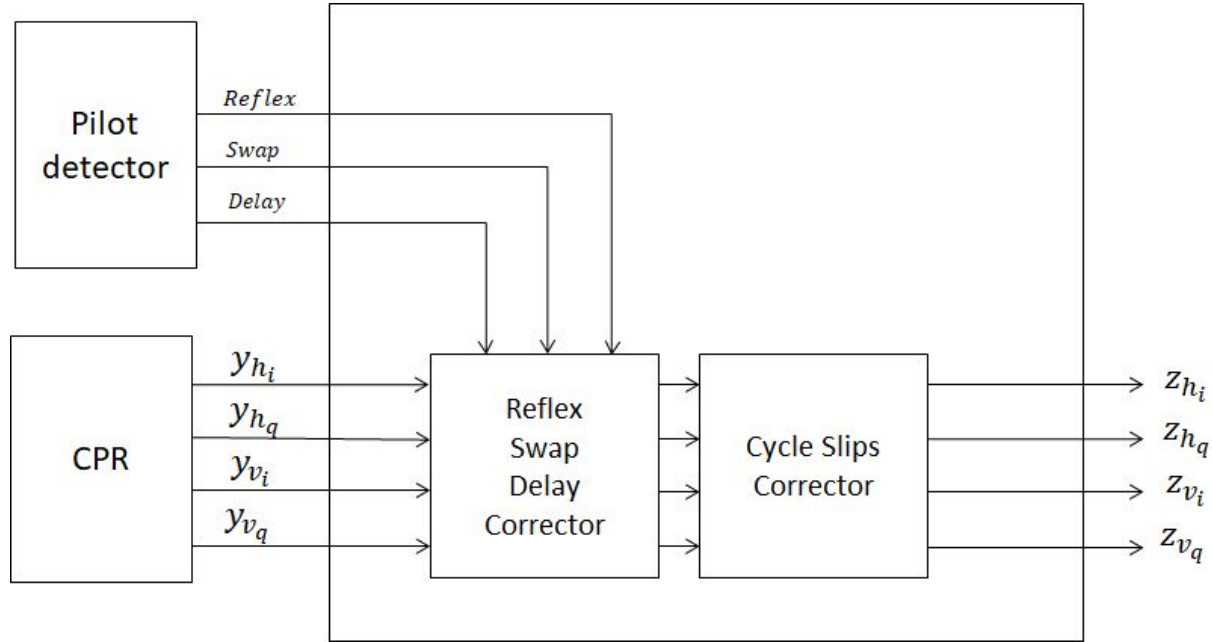
# Carrier Phase Recovery (pilot assisted)



# Actual Aligner and effects recovery



# Upgraded Aligner and effects recovery

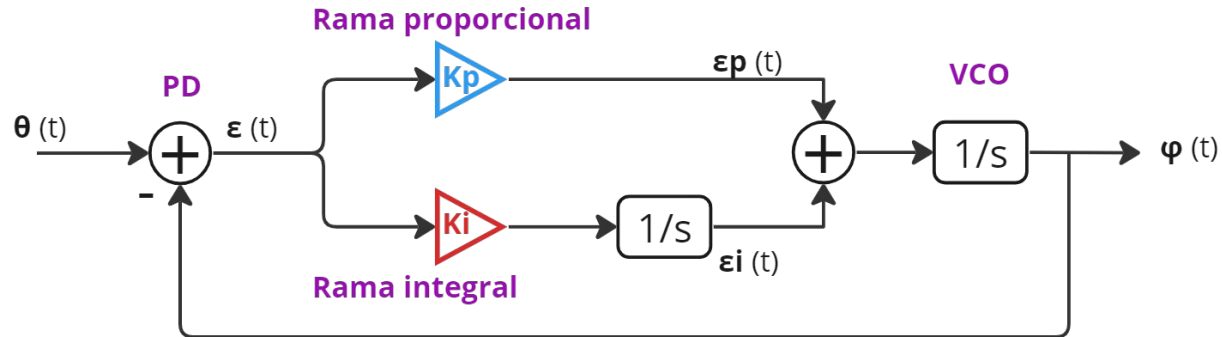


# PLL

## PLL1

Sistema de Control P: Problema: no sirve en nuestro caso, ya que no sigue entrada rampa

□ Sistema de Control PI



*Tiempo continuo*

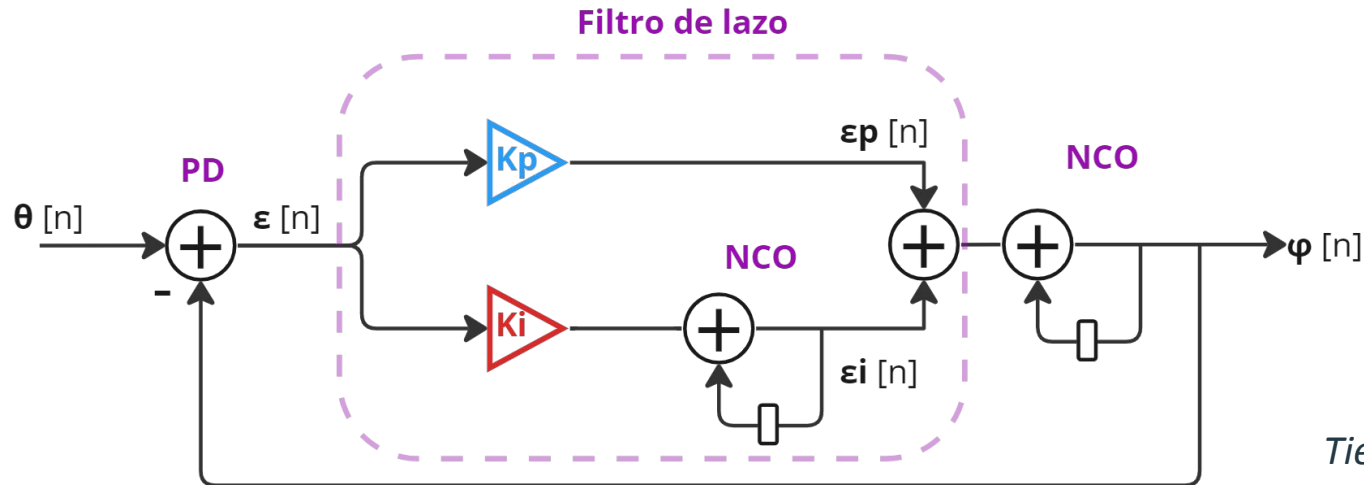
## Integrador

$$H(z) = \frac{1}{1 - Z^{-1}} = \frac{Y(z)}{X(z)}$$

$$H(z) = Y(z) - Y(z) \cdot Z^{-1} = X(z)$$

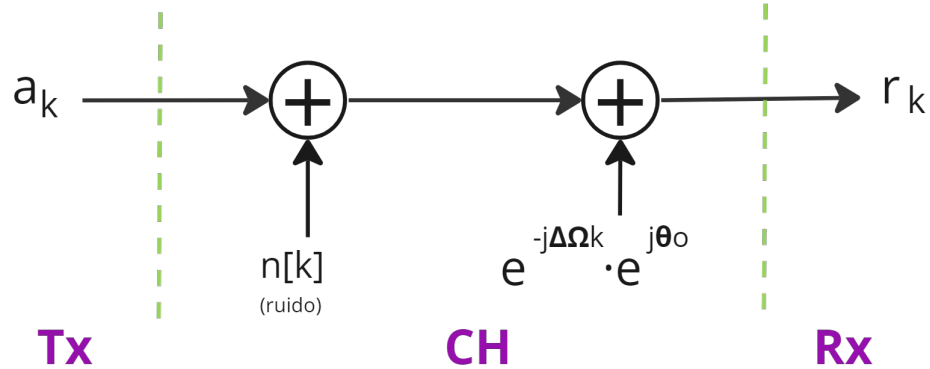
$$\Rightarrow Y(z) = X(z) + Y(z) \cdot Z^{-1}$$

$$y[n] = x[n] + y[n - 1]$$

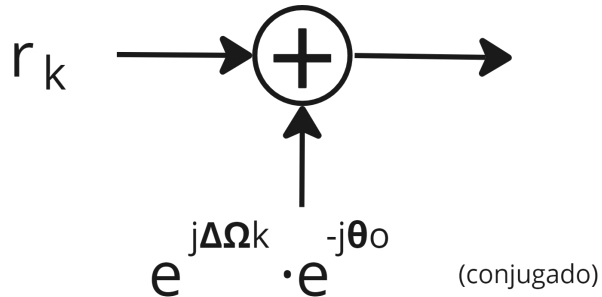


*Tiempo discreto*

## Modelado



Si se conocen  $\Delta\Omega$  y  $\theta_0$

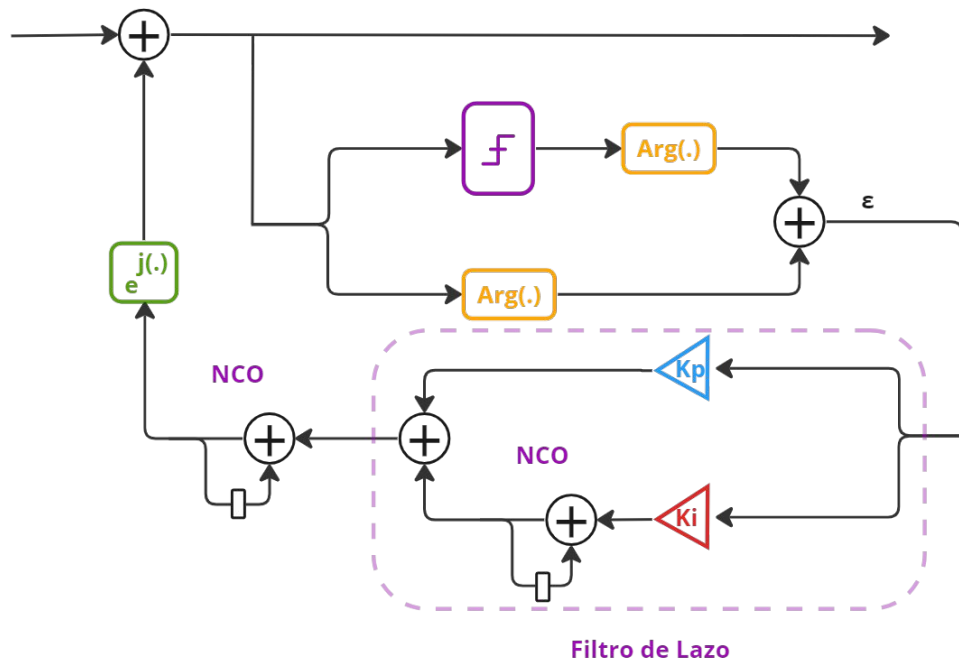


Si no se conocen  $\Delta\Omega$  y  $\theta_0$   
se deben estimar

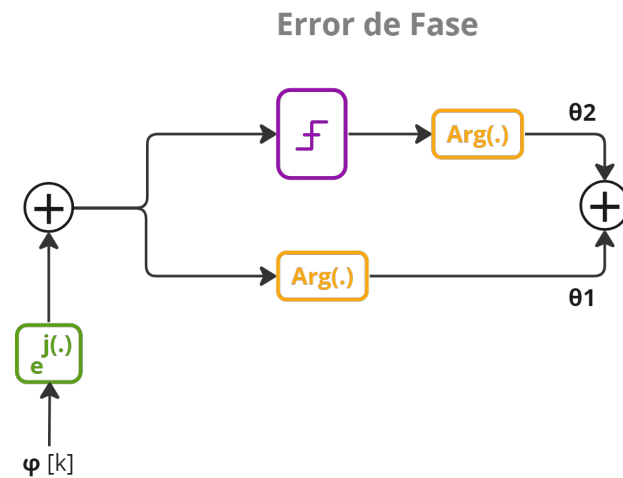
$$\Rightarrow \widetilde{\Delta\Omega} \quad \widetilde{\theta_0}$$



# PLL2

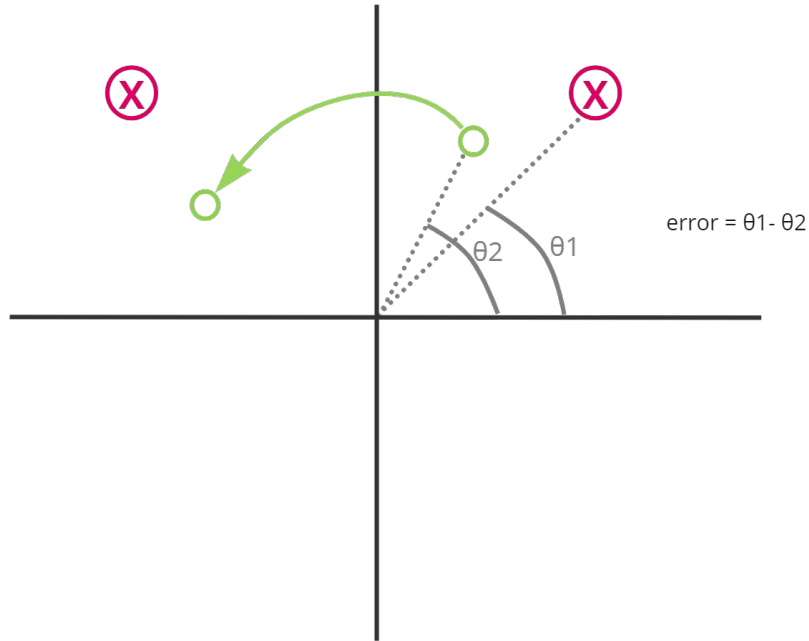


## Find Carry Recovery - FCR



Tener en cuenta:

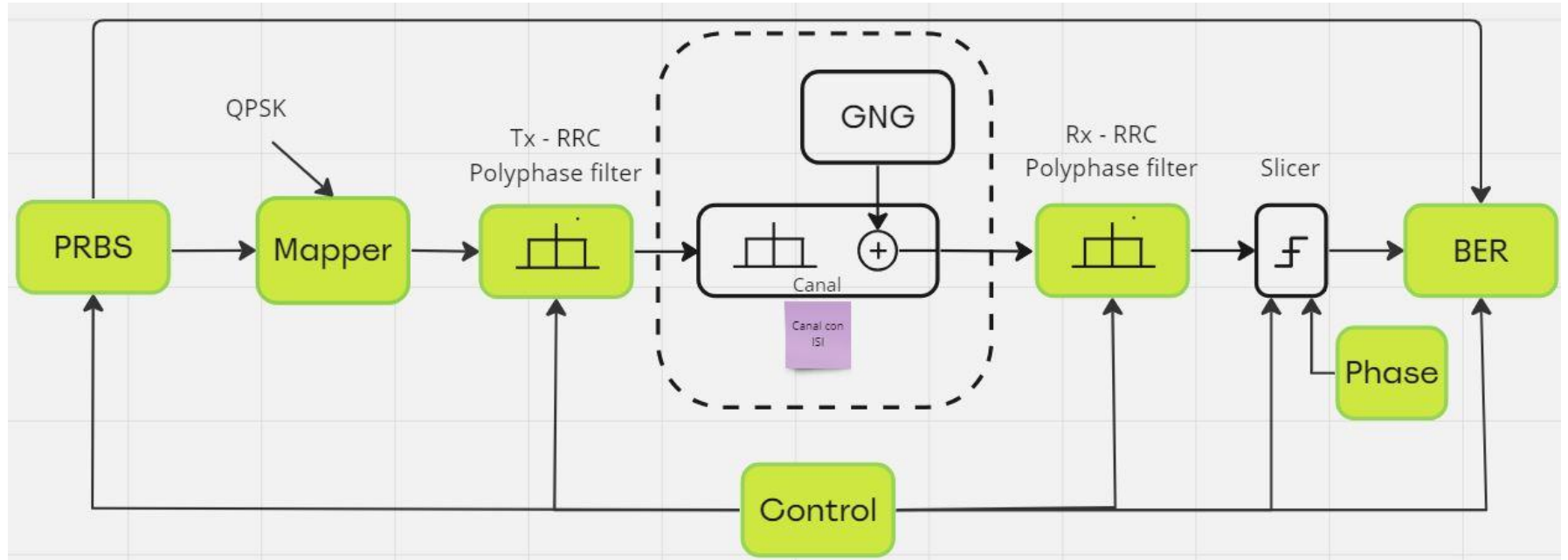
No se detectan desfases múltiplos de  $90^\circ$



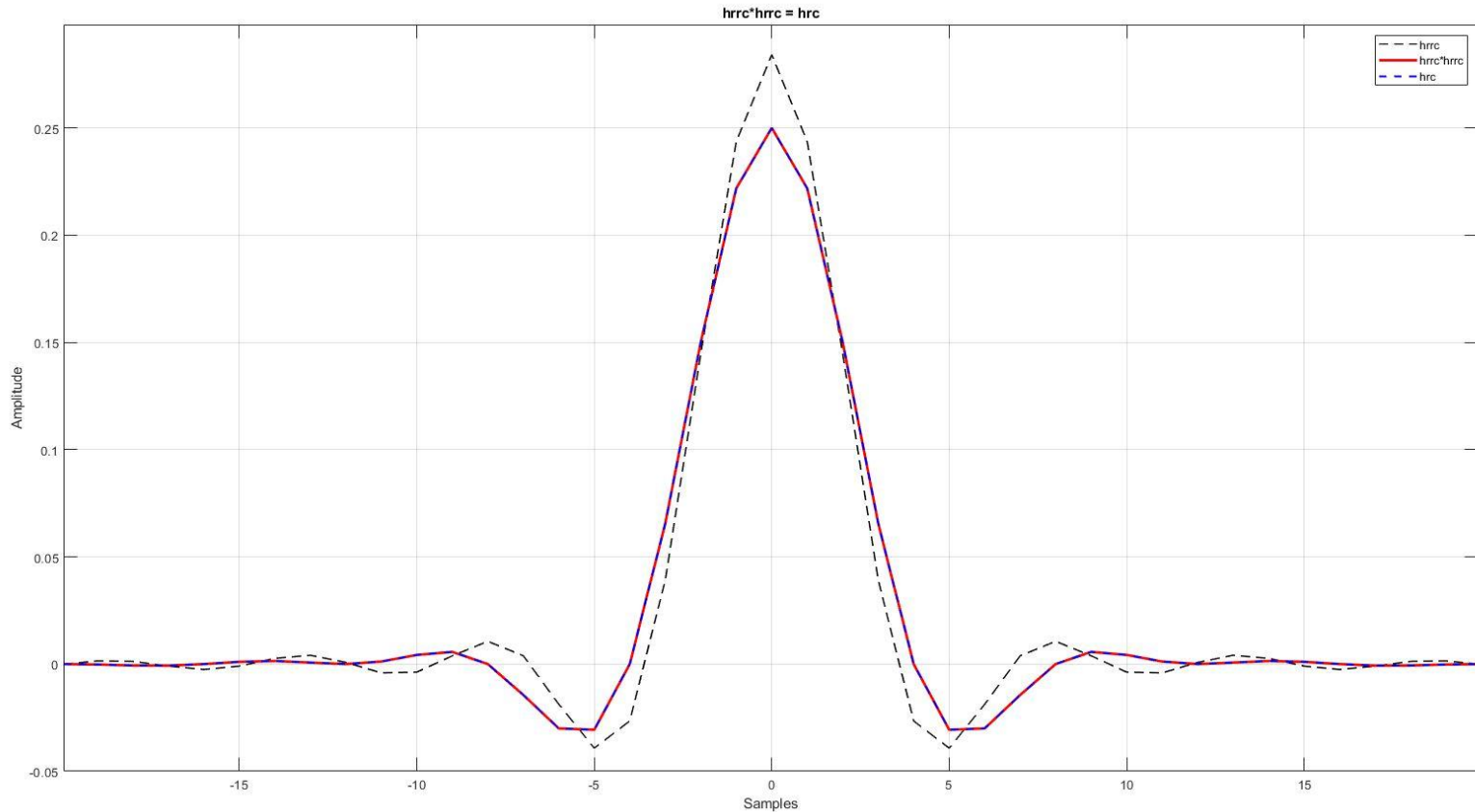
Cuando hay mucho ruido, se tiene un sesgo en el error (acumulación)

Solución: símbolo piloto, valor conocido

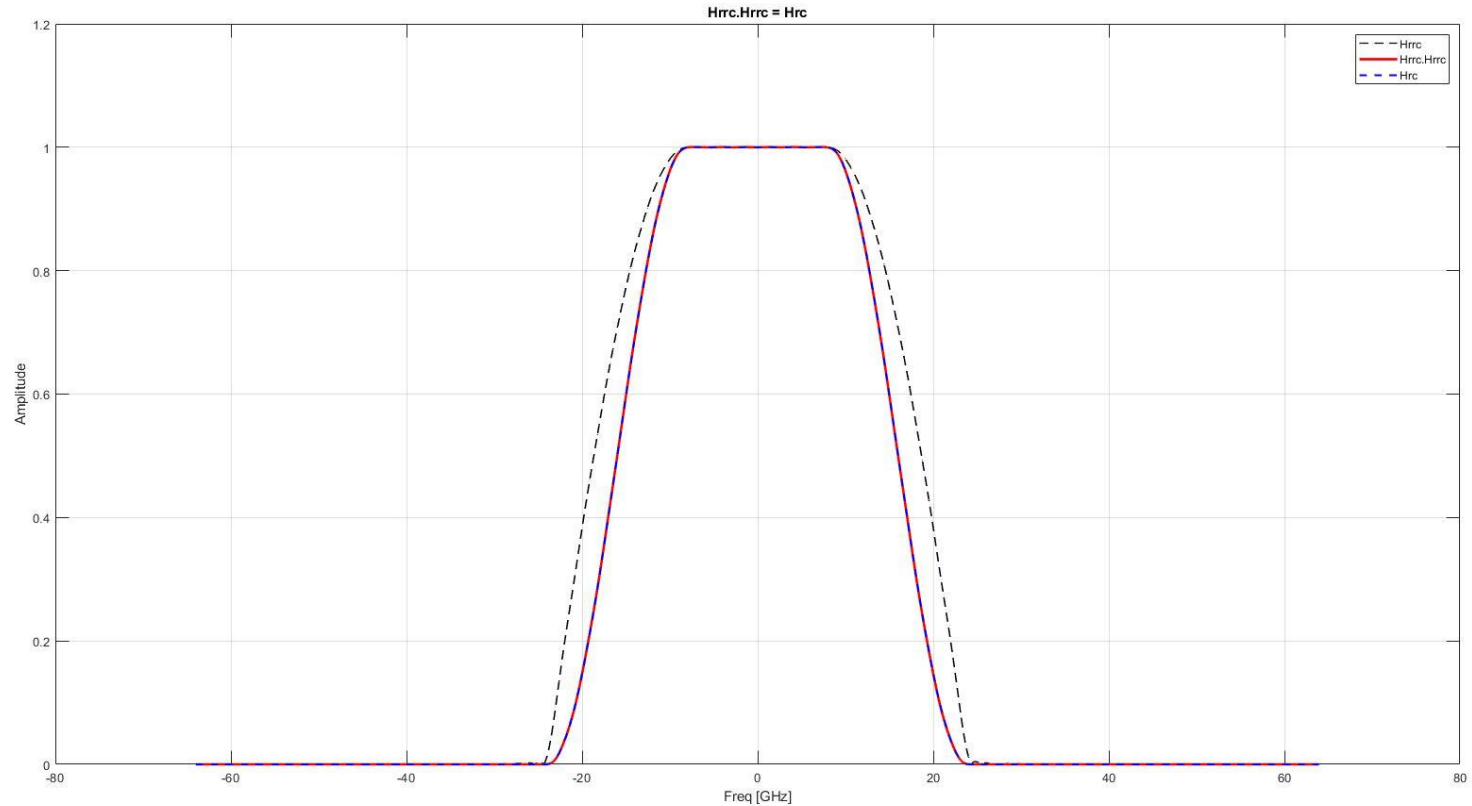
# Modelo de aproximación: Python



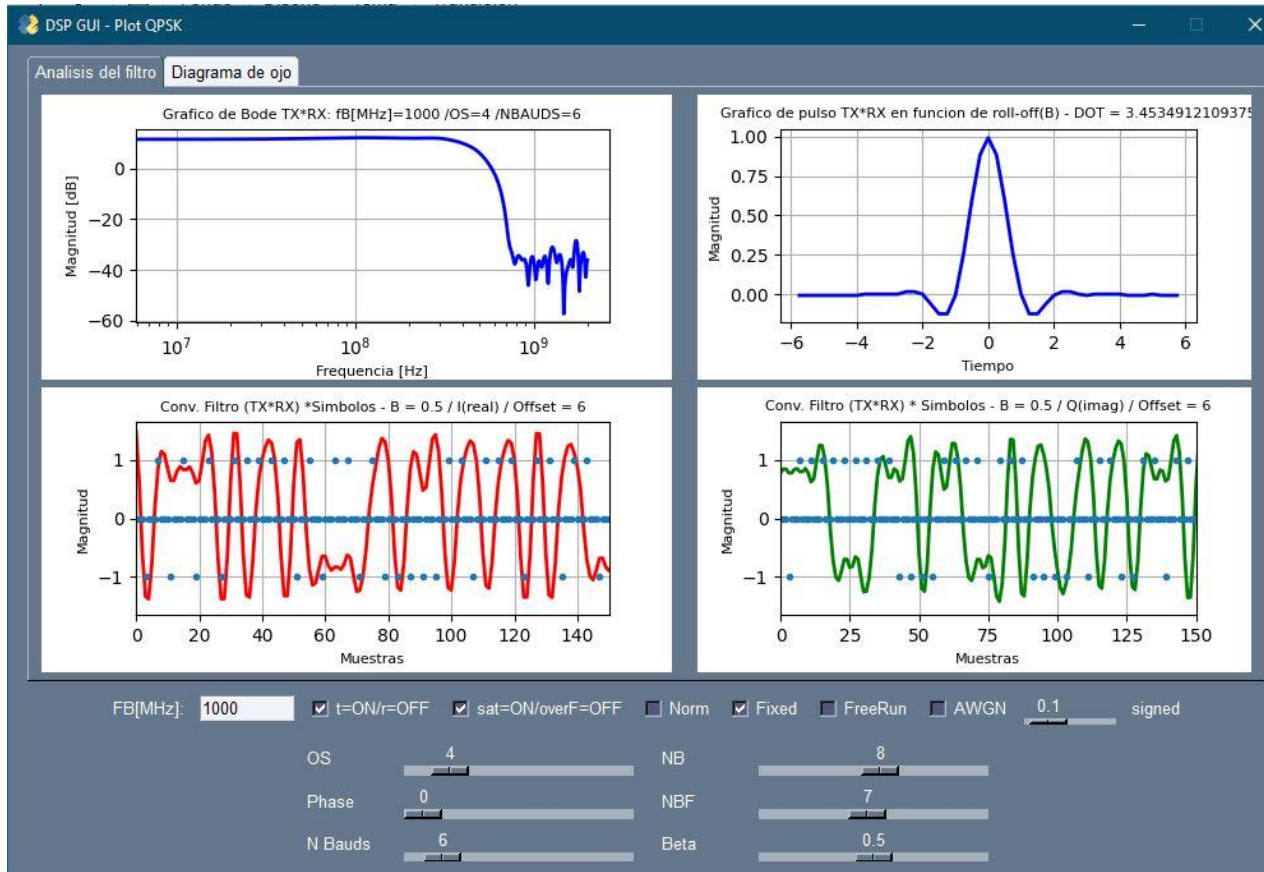
# Respuesta al impulso RC, RRC y RRC\*RRC



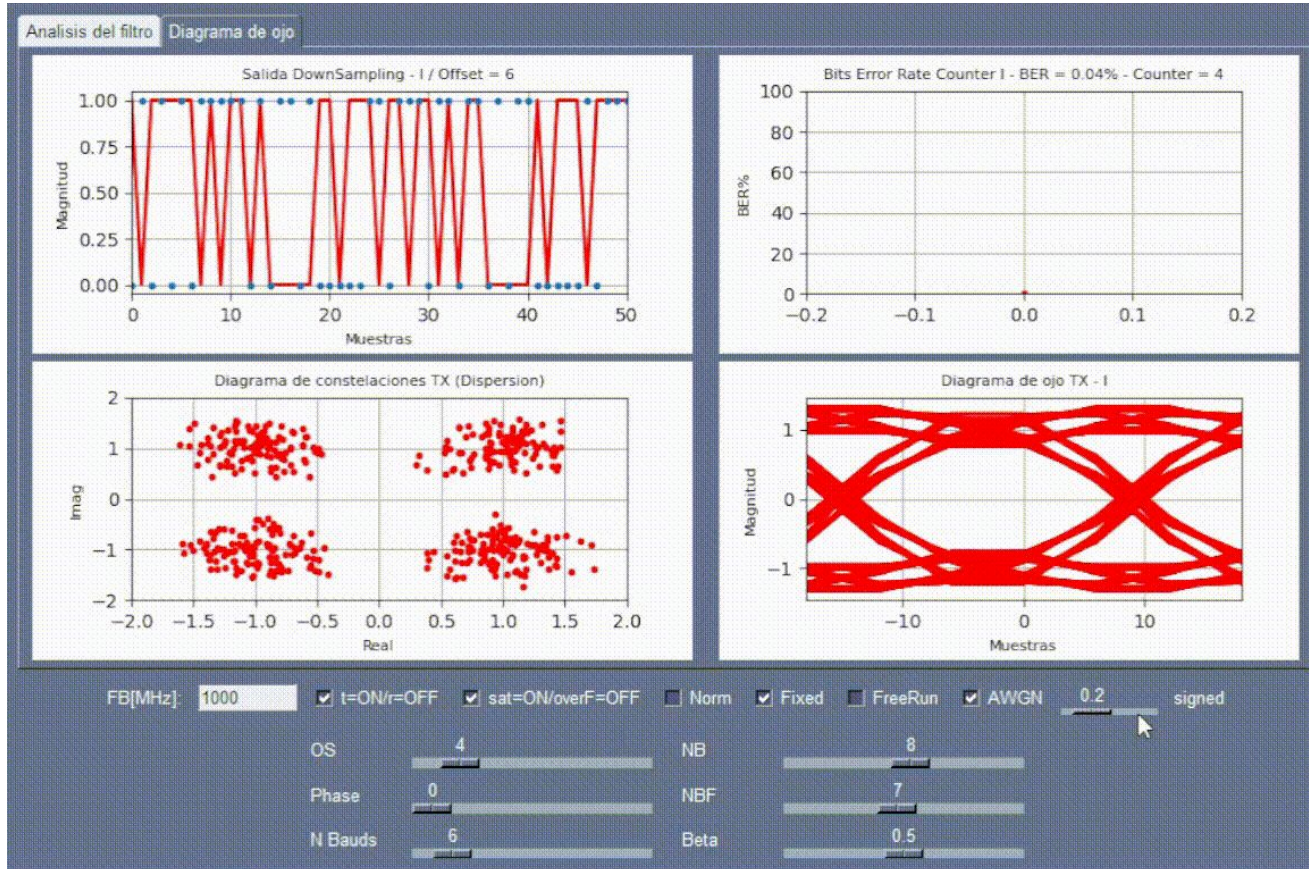
# Respuesta en frecuencia RC, RRC y RRC\*RRC



# Interfaz gráfica de usuario: Python



# Interfaz gráfica de usuario: Python



# “To do list”