Laboratorio de Procesado Digital de Señal - 3º GITT

Práctica 8: Arquitecturas avanzadas de PDS

El objetivo de esta práctica es introducir al alumno al diseño y análisis de arquitecturas avanzadas de procesamiento digital de señales, aplicadas sobre un sistema de comunicaciones monoportadora (en inglés "Single Carrier").

La práctica se compone de dos partes. En la primera, el objetivo es interactuar con una placa de evaluación de radio definida por software (en inglés "Software Defined Radio") e implementar un módulo transmisor M-PSK capaz de transmitir muestras en tiempo real. La placa de evaluación seleccionada es el modelo ADALM-PLUTO del fabricante "Analog Devices" (podéis encontrar más información en https://wiki.analog.com/university/tools/pluto). En la segunda parte, se analizará (por medio de un simulador programado en Matlab) los distintos bloques que componen la capa física de un modem M-PSK.

Se facilitan al alumno los siguientes archivos de apoyo para el desarrollo de la práctica:

- Código completo .m de un modem M-PSK (pruebas_tx_rx_MPSK_sample_based.m)
- Código de generación de coeficientes del filtro en raíz cuadrada de coseno alzado (srrc_filter.m)
- Código de evaluación de la cadena de transmisión (Tx) y recepción (Rx) de la placa de evaluación ADALM-PLUTO (pluto_evaluate_tx_rx_chain.py).
- Código base de un transmisor M-PSK capaz de transmitir muestras en tiempo real (mpsk_tx_pluto_pds.py)

Al finalizar la práctica, suba a la plataforma web de la asignatura (Moodle) un archivo PDF, en el que se responda a los apartados de la práctica y en el que se presenten las figuras que se piden. Para los apartados que así lo requieran, se deberá entregar un archivo ".py" o ".m" con las rutinas de programación solicitadas.

Librerías y drives de control de la placa ADALM-PLUTO

Para poder desarrollar la primera parte de la práctica es necesario instalar, en los equipos personales, los drivers de comunicación y la API de control de la placa ADALM PLUTO.

El sistema ADALM PLUTO dispone de un único puerto USB a modo de interfaz de comunicación física, que a su vez hace las funciones de entrada de alimentación de la placa. El intercambio de información entre el "host" (ordenador personal) y la placa se realiza a través de un protocolo TCP/IP que se encuentra encapsulado por una librería de control de alto nivel desarrollada por el propio fabricante ("Analog Devices"), denominada "libiio".

Para permitir la comunicación IP a través de un puerto USB del "host" es necesario instalar unos drivers específicos en función del sistema operativo del que disponga el alumno instalado

en su ordenador personal. A tal efecto se recomienda seguir la instrucciones detalladas en el punto 3 del siguiente enlace https://wiki.analog.com/university/tools/pluto/users.

Es importante que el alumno siga todas las instrucciones detalladas el enlace anterior y realice las comprobaciones sugeridas.

Como ya se ha detallado, la interacción con la ADALM PLUTO se realiza a través de llamadas a funciones de la librería "libiio". Libiio está originalmente escrito en C/C++ y es el lenguaje utilizado por defecto debido principalmente a su rapidez de ejecución. No obstante, el fabricante ofrece APIs de acceso a libiio mediante el uso de otros lenguajes de programación más intuitivos y portables: Matlab, C#, Python y GNU Radio.

En este laboratorio emplearemos el lenguaje de programación Python. Para poder editar y ejecutar códigos .py se recomienda el uso de un editor interactivo (emplearemos Spyder) y un gestor de paquetes de librerías científicas (haremos uso de Anaconda).

A continuación, se detallan los enlaces de descarga de Anaconda y Spyder (aunque este último puede que venga preinstalado con la propia instalación de Anaconda):

https://www.anaconda.com/products/individual

https://www.spyder-ide.org/

Posteriormente, descargue e instale la librería libiio:

https://wiki.analog.com/resources/tools-software/linux-software/libiio

Además, descargue la carpeta de libiio de su directorio de github:

https://github.com/analogdevicesinc/libiio

Se descargará como un archivo comprimido, que deberá ser descomprimido y guardado en una ruta conocida. Haremos uso de la localización y contenido de dicho directorio en el siguiente punto.

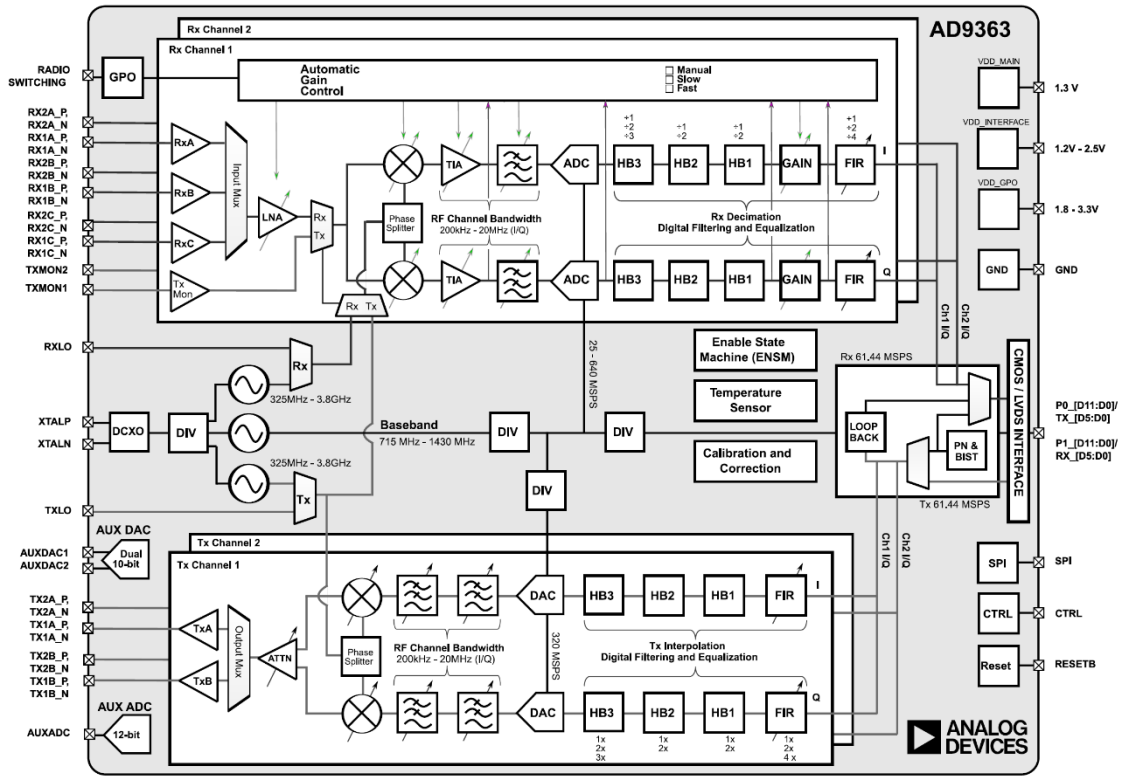
Abra un terminal de comandos y navegue hasta el interior de la carpeta que acaba de descomprimir. Acceda al directorio "bindings" y de ahí a la carpeta "python". Desde esta última ruta ejecute la siguiente sentencia: *python setup.py.cmakein install*

Ya debería haber finalizado la instalación de la API empleada para realizar llamadas a las funciones de "libiio". Para comprobarlo, acceda a Spyder y ejecute la sentencia *import iio.* Si no observa ningún error, la instalación se habrá llevado a cabo con éxito.

Por último, descargue la herramienta "IIO Oscilloscope" en el siguiente enlace https://wiki.analog.com/resources/tools-software/linux-software/iio_oscilloscope. Este programa es una potente herramienta de análisis y configuración de dispositivos hardware como la radio definida por software que se usará en algunos apartados de la práctica.

Análisis de la cadena de Tx y Rx de la placa ADALM-PLUTO

En la siguiente figura se muestra un esquema detallado de los elementos HW que componen el transceptor AD9363 que lleva integrado la ADALM-PLUTO.



Como se puede observar, la señal recibida atraviesa un demodulador I-Q cuyo oscilador local tiene un margen de variación desde 325 MHz hasta 3.8 GHz (aunque puede ser ampliamente extendido mediante una ligera modificación en el driver de control del dispositivo). La señal compleja a la salida del demodulador I-Q es filtrada paso bajo y digitalizada por sendos ADCs (convertidores analógico-digitales) de 12 bits de resolución. La frecuencia de muestreo de los ADCs puede configurarse como cualquier valor comprendido entre 25 MHz y 640 MHz.

En el dominio digital, la señal atraviesa una cadena de diezmadores (cada uno de ellos acompañado de su filtro “antialiasing”) antes de trasladar los datos recibidos al usuario. El objetivo de este pre-procesado es disminuir la tasa de datos que finalmente será analizada, y así permitir el procesado en tiempo real sin que se produzca un desbordamiento de la información recibida.La tasa de datos máxima permitida por el chip AD9163 es de 62.44 MHz, pero puede ser reducida hasta los 521 KHz (haciendo uso de mayores factores de diezmado en la cadena de preprocesado).

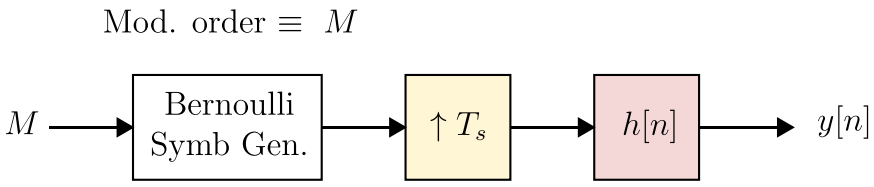
Finalmente, los datos recogidos son enviados a un FPGA de control, incluida en la propia placa ADALM-PLUTO, y posteriormente son empaquetados y trasladados a un sistema multiprocesador gobernado por un sistema operativo Linux. Sobre el sistema operativo un servidor de “libiio” controla el traspaso de datos hacia (o desde) el “host” (ordenador personal).

Para el caso de la cadena de transmisión, la ruta descrita ha de recorrerse en sentido opuesto.

Modulador M-PSK sobre la placa de evaluación ADALM-PLUTO

En este apartado se pretende implementar (en Python) un módulo transmisor M-PSK SW en tiempo real sobre la placa de evaluación ADALM-PLUTO.

A continuación, se muestra un diagrama de bloques del modulador M-PSK.



La tarea solicitada consiste en implementar un interpolador seguido de un filtro FIR. Como se trata de una implementación SW en tiempo real, el procesado se ha de realizar por bloques, y por tanto las operaciones de interpolado y filtrado también. Se recomienda el uso del algoritmo "overlap-add" u "overlap-save".

Añada el código Python desarrollado a la entrega de la práctica. En el informe, explique el procedimiento seleccionado y justifique los valores de diseño del algoritmo de procesado por bloques.

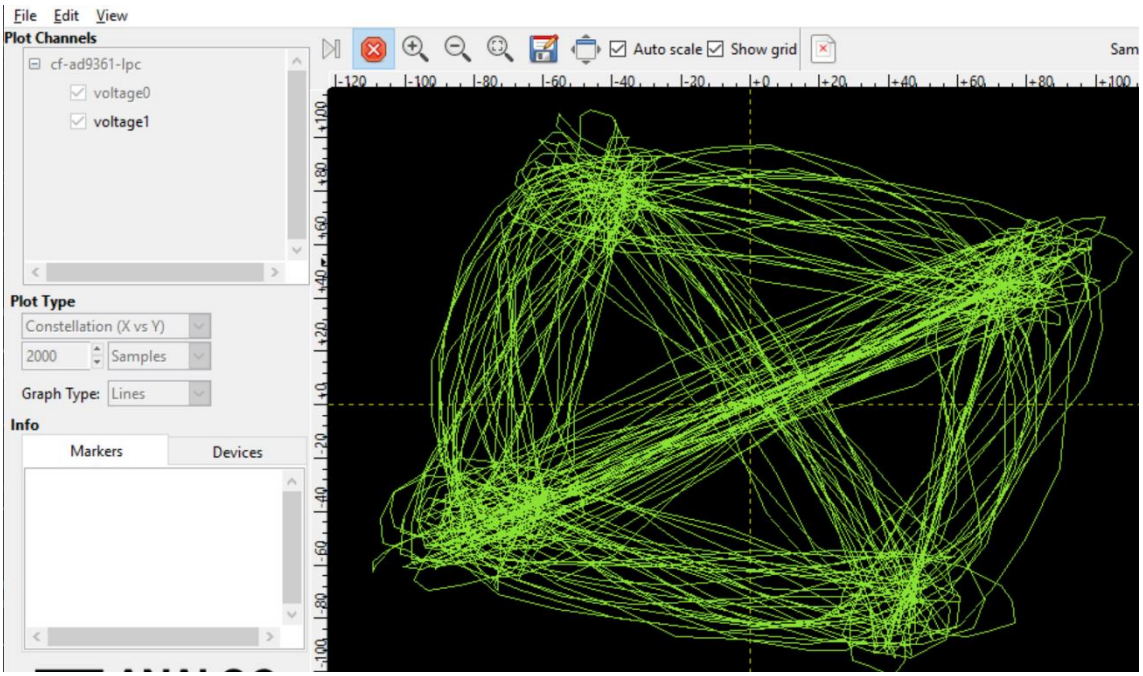
El generador de símbolos aleatorio ya está programado en el código Python que se ha ofrecido a modo de plantilla. Las únicas restricciones a tener en cuenta son:

- La longitud del bloque final que se manda al módulo transmisor es fija e igual a BL.
- El periodo de símbolo (T_s) debe ser configurable en longitud mediante un número entero.
- La respuesta al impulso ($h[n]$) del filtro reconstructor a la salida del interpolador se corresponde con los coeficientes de un filtro en raíz cuadrada de coseno alzado para evitar la interferencia entre símbolos en el proceso de transmisión. Esta subrutina ya está programada en el código base.

NOTA: Para el curso 2020/21 el código del modulador ya se ofrece programado en Python, el alumno deberá crear un modelo Matlab del mismo (el cual deberá agregar a la entrega en sustitución del código Python) y explicar los valores asignados en el informe. Se deberá comprobar que la convolución por bloques es equivalente a una convolución completa.

Finalmente, conecte el cable usb entre el "host" y a la placa de evaluación, enlace el Tx y Rx de la placa ADALM-PLUTO mediante un cable coaxial y ejecute el código Python del transmisor. Para observar la constelación transmitida, mientras el sistema está emitiendo, abra la aplicación IIO Oscilloscope, haga "click" en "File>New plot" y seleccione la visualización en forma de constelación. Es importante que preseleccione los canales de señal (recuadro superior a la izquierda) antes de hacer "click" en el botón de "play".

Si tiene problemas con la configuración de la herramienta apóyese en la captura de pantalla mostrada en la siguiente figura.



Análisis de un modem M-PSK en Matlab

En el apartado anterior ya se ha descrito la arquitectura típica empleado en un modulador M-PSK. En los siguientes puntos, se describe el modelo de canal empleado en las simulaciones y una descripción detallada de cada uno de los módulos que componen el sistema receptor.

Se recomienda al alumno, antes de continuar leyendo el guion de laboratorio, que proceda a inspeccionar visualmente el código Matlab del simulador para familiarizarse con el contenido. Se anima así mismo al alumno a ejecutar el código y observar el funcionamiento por defecto.

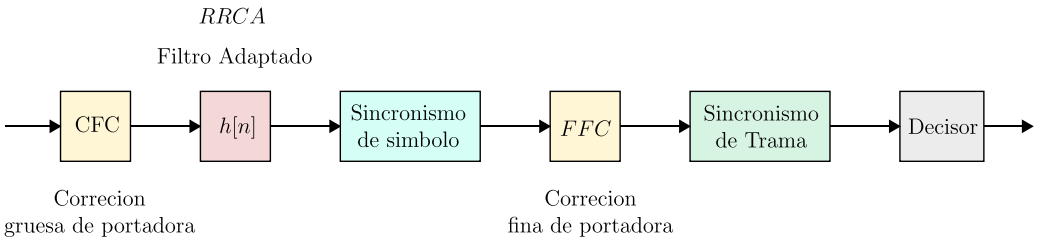
Modelo de canal

En este caso, se ha seleccionado un modelo de canal ruidoso simple sin tener en cuenta los posibles efectos derivados del multitrayecto. Es decir, únicamente se añade ruido blanco Gaussiano de acuerdo de una relación señal a ruido previamente configurada.

Se anima al alumno a que aumente y/o disminuya progresivamente el valor de la relación señal a ruido y observe su efecto sobre la señal recibida mediante la ejecución reiterada del simulador (para cada nuevo valor de relación señal a ruido que introduzca).

Módulo Receptor

La función principal de un sistema receptor de comunicaciones es recuperar con éxito la información originalmente transmitida. Para ello, el módulo receptor deberá acondicionar la señal interceptada, y corregir los efectos indeseados derivados de desajustes del propio receptor y/o imperfecciones del canal de comunicaciones. A continuación, se muestra un esquema de cada uno de los bloques que componen el sistema receptor del modem M-PSK.



Estimación gruesa de portadora

Para la compensación de la desviación gruesa de portadora se usa una técnica basada en la FFT. El algoritmo adoptado utiliza el conocimiento de la estructura de la señal recibida, y específicamente el índice de modulación de fase, M (equivalente al número de fases PSK. La señal se eleva a la potencia de M, lo que produce un tono significativo en M veces la frecuencia de desplazamiento como resultado de la estructura de la señal. A partir del modelo, ignorando el ruido, podemos observar lo siguiente:

$$r^M(k) = S^M(k) * e^{j(2\pi fokT+\theta)*M}$$

Esto desplazará la desviación en frecuencia a M veces su ubicación original y hará que s(k) sea puramente real o puramente complejo. Por lo tanto, el término $s^M(k)$ se puede ignorar y sólo quedará el exponencial o tono restante. Para estimar la posición de este tono tomaremos la FFT de $r^M(t)$ y luego se identifica el bin en la FFT con la mayor magnitud.

Dada la frecuencia de muestreo, fs, y el número de puntos de la FFT (N_{FFT}), el espaciado de frecuencias de los bins de la FFT utilizados en este algoritmo viene dado por:

$$f_{\Delta FFT} = \frac{f_s}{N_{FFT}}$$

La frecuencia de muestreo debe elegirse en función del máximo desplazamiento de frecuencia esperado. También habrá un error en la estimación del desplazamiento de frecuencia, derivado de la resolución finita de la FFT (es decir, el número fijo de tonos de la FFT). En el peor de los casos, el error en la estimación del desplazamiento de frecuencia viene dado por:

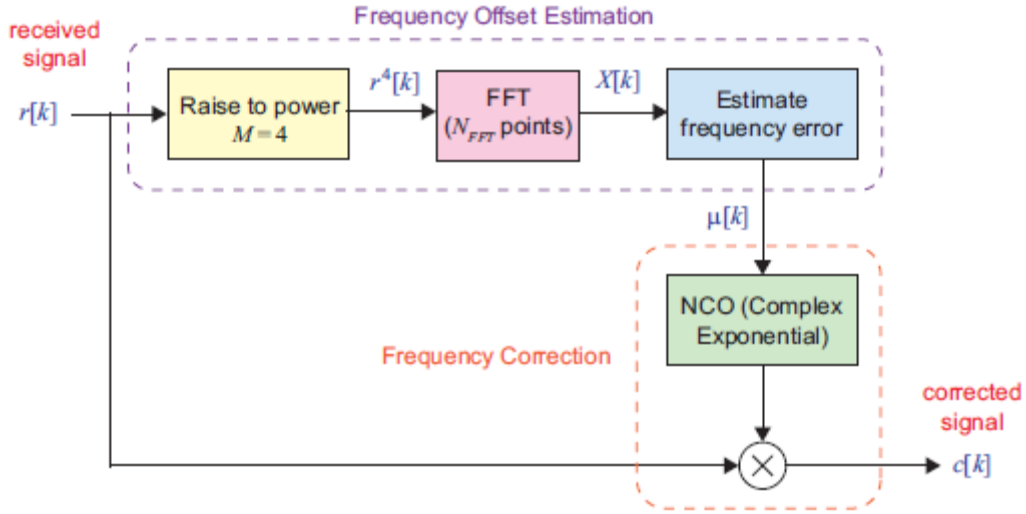
$$f_{error\_est} = \frac{f_{\Delta FFT}}{2 * M}$$

Esto se debe a que el error en la identificación del tono FFT de máxima magnitud (de la señal de M potencia) es igual a la mitad de un ancho de bin FFT, y este error se divide posteriormente por M al calcular el desplazamiento de frecuencia.

El desplazamiento de frecuencia, generado por el proceso de estimación descrito anteriormente, puede utilizarse posteriormente para realizar una corrección, de forma que el espectro de la señal se centre en aproximadamente 0 Hz.

La corrección se lleva a cabo generando un término exponencial complejo en la frecuencia estimada y multiplicándolo por la señal recibida. Se trata de una modulación de la señal en

banda base compleja. Las etapas de estimación y corrección de la frecuencia gruesa pueden resumirse en el diagrama de bloques de la siguiente figura.



a) Teniendo en cuenta el tipo de señal empleada en el programa suministrado (MPSK), diga cual es la relación mínima que debe de tener la frecuencia de muestreo (fs), con respecto a la desviación en frecuencia para poder aplicar el algoritmo anteriormente descrito. Si la desviación en frecuencia que presenta la señal no cumple esta relación, que bloque se debe de incluir antes de la estimación del desplazamiento.

b) Para mejorar el error en la estimación del desplazamiento de frecuencia, qué parámetros serían los más adecuados modificar, diga valores aproximados de dichos parámetros y explique el efecto que tiene estos en la simulación del sistema.

c) Calcule el error en la estimación del desplazamiento de frecuencia para una desviación de frecuencia esperada de 800 Hz, teniendo en cuenta los parámetros utilizados en el sistema dado.

Sincronismo de Símbolo

En un sistema de comunicaciones existen diversos factores que producen retardos sobre la señal transmitida (propagación en el canal, recepción en el front-end de RF, etc.) que generan incertidumbre sobre cuál es el momento óptimo para muestrear la señal, ya que el transmisor y el receptor no comparten la misma referencia temporal. En las siguientes figuras se muestra cómo al no muestrear en el momento óptimo se produce un error de amplitud de la señal, lo que se traduce en una degradación de la SNR. Hay que recordar también que si no se muestrea de forma óptima a la salida del filtro adaptado se produce ISI, ya que los símbolos adyacentes no se muestrean en su cruce por cero.

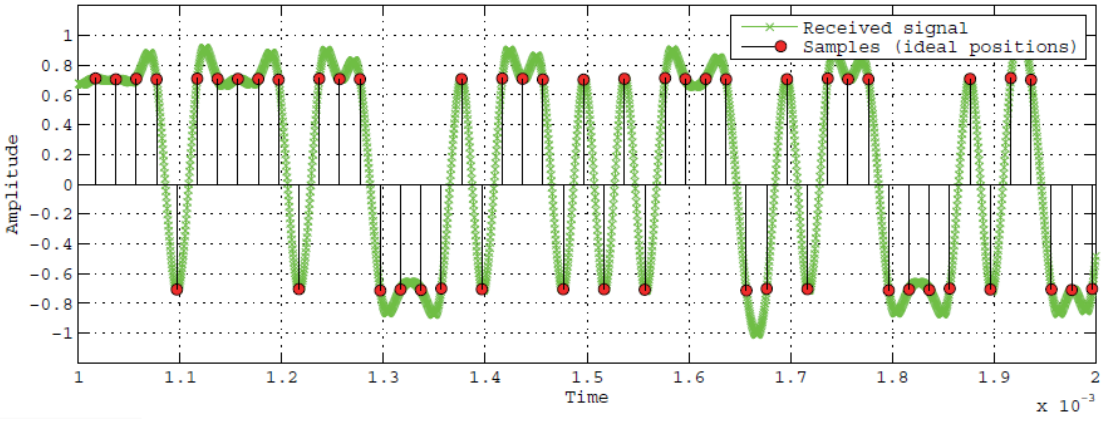


Figura 1. Ejemplo de muestreo en el instante óptimo. [1]

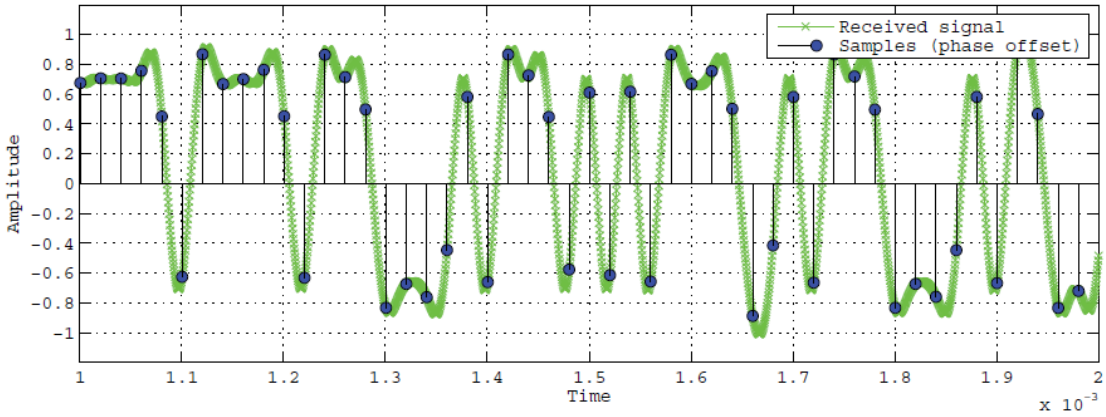


Figura 2. Ejemplo de muestreo con offset de tiempo. [1]

Una posible solución para conocer el instante óptimo de muestreo consiste en transmitir junto a la señal de datos, la señal de reloj del transmisor. Esta opción tiene varios inconvenientes, por el lado del transmisor aumenta la potencia y ancho de banda requeridos, y por el del receptor aumenta la complejidad tanto de recepción como de procesamiento. Por estas razones se suelen utilizar otros métodos basados directamente en la señal de datos.

En el simulador se ha implementado una solución basada en un PLL. Este bloque tiene múltiples aplicaciones y es parte fundamental de los sistemas de comunicaciones. Aunque se presentará de forma breve en esta práctica, se estudiará en detalle en la asignatura de Electrónica de Comunicaciones que se imparte en el Máster Universitario en Ingeniería de Telecomunicación.

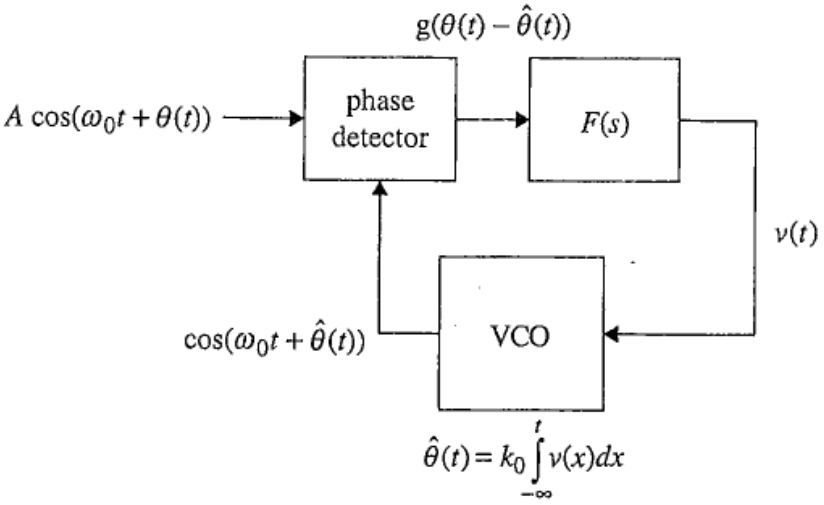


Figura 3. Estructura básica de un PLL en el dominio continuo. [2]

Un PLL, Phase Locked Loop, es un subsistema retroalimentado que busca minimizar la diferencia de fase entre la señal de entrada respecto a la señal de salida. Los bloques que lo componen son los siguientes:

- Phase detector: Obtiene la diferencia de fase de fase entre la señal de entrada y salida, multiplicada por un factor Kp, que definimos como señal de error e(t).
- Filtro de retroalimentación, F(s): Filtro que controla el funcionamiento del sistema mediante el filtrado de la señal de error. Permite controlar parámetros como la velocidad de adaptación ante cambios o el error residual una vez enganchados en fase.
- VCO: Oscilador controlado por tensión, según el voltaje a la entrada, en nuestro caso la salida del filtro genera una señal sinusoidal a una frecuencia determinada. Este elemento tiene una ganancia Ko.

El algoritmo implementado, con una estructura tipo PLL es el siguiente:

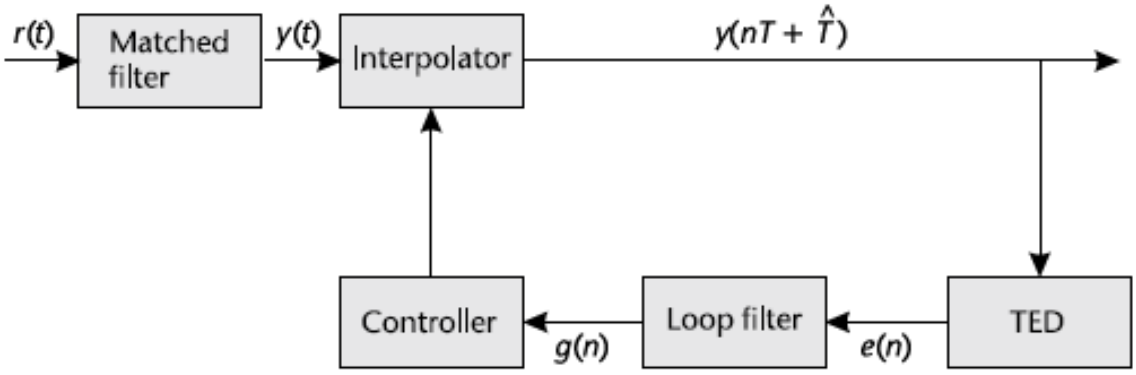


Figura 4. Esquema del sincronizador del tiempo de muestreo. [3]

La entrada de este algoritmo son las muestras a la salida del filtro adaptado, el cual tiene un factor de sobremuestreo de N muestras, esto quiere decir que por cada símbolo el ADC

obtiene N muestras. Gracias a tener varias muestras por símbolo es posible minimizar en el error de muestreo siguiendo uno de los posibles criterios que existen.

A continuación, se presenta la funcionalidad de cada bloque:

- El interpolador genera la muestra que se obtendría en el momento óptimo de muestreo (o una aproximación) a partir de las muestras a la salida del filtro adaptado. La interpolación es variable a lo largo del tiempo, ya que está controlada por un factor μ[n], que indica que fracción de retardo tiene el instante óptimo respecto al tiempo de muestro, lo cual se puede observar en la figura inferior. Se puede demostrar que el interpolador óptimo es un filtro paso bajo ideal, y por tanto un filtro no casual. Para tener una implementación causal e implementable en un sistema en tiempo real, se opta por un tipo de filtro FIR llamado Piecewise Polynomical Filter (PPF), que implementa una interpolación polinómica. El uso de estos filtros es muy extendido ya que tiene una implementación hardware eficiente llamada estructuras de Farrow.

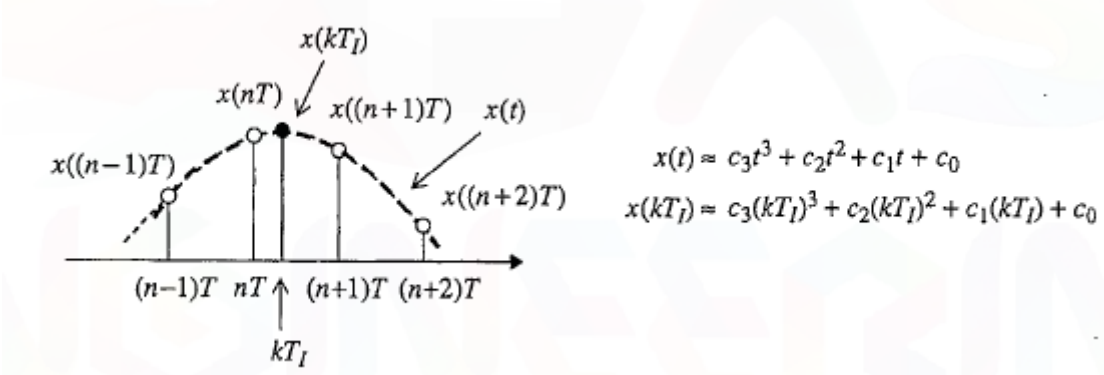


Figura 5. Interpolación cúbica con filtro PPF. [2]

Este proceso de realiza muestra a muestra, pero solo pasarán al siguiente bloque de sincronismo las salidas del interpolador correspondientes a una señal de strobe que marca el diezmado para tener una muestra por símbolo, de forma que se selecciona la muestra más cercana a la óptima. Es importante realizar el proceso para todas las muestras ya que las salidas se emplearán para estimar el error en el instante de muestreo.

- El controlador de interpolador es el bloque que a partir del cual la señal de error filtrada genera la señal de strobe, y el término fraccional de la interpolación μ(n). Se ha implementado como un contador módulo 1.

  El contador módulo 1 consiste en un contador que se incremente en un factor $\frac{1}{N}$ en cada muestra recibida, siendo N el número de muestras por símbolo. En la situación donde no existe error, tras N muestras alcanza el máximo y lanza la señal de strobe, y se reinicia el contador a 0, tal como se puede observar en la figura 6. En el caso en el que exista error en el tiempo de muestreo, se aplica un factor corrector que ralentiza o

acelera el incremento en el contador, quedando el paso como $d[n] = g[n] + \frac{1}{N}$, siendo g(n) la salida del filtro de retroalimentación.

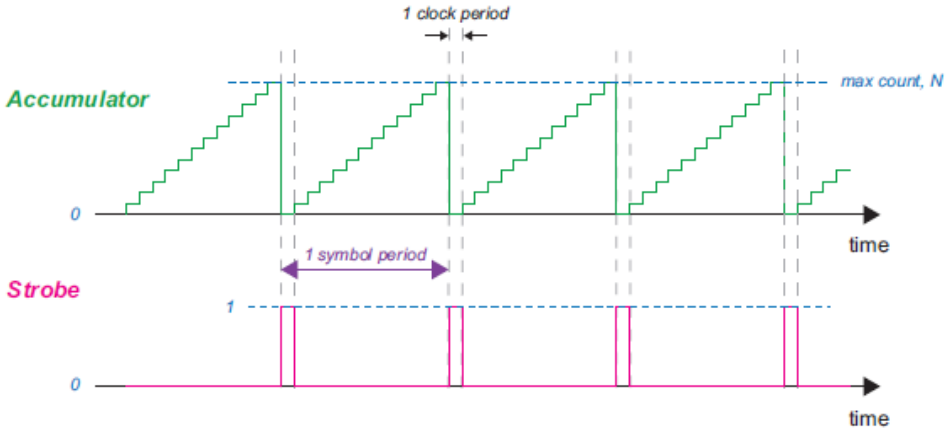


Figura 6. Funcionamiento del contador módulo 1. [1]

Finalmente, el factor μ[n] se actualiza junto a la activación de la señal de strobe, como el cociente entre el paso de actualización del contador entre el valor del contador.

El uso conjunto del interpolador y su controlador es equivalente al VCO en un PLL.

- El Timing Error Detector (TED), es el bloque que estima el error en el instante de muestreo según una métrica a partir de las salidas del interpolador, siendo equivalente al detector de error de fase en un PLL. Se han implementado dos en el simulador:
    - Zero Crossing (ZC): Este algoritmo busca los cruces por cero en el diagrama de ojo y requiere como mínimo dos muestras por símbolo. Este bloque devuelve error 0 cuando están alineadas con los cruces por cero de la salida del filtro adaptado. En la figura inferior se muestra como puntos blancos el instante óptimo de muestreo, y como negros los obtenidos al muestrear. En el caso superior se ha muestreado antes de tiempo, y en el inferior después.

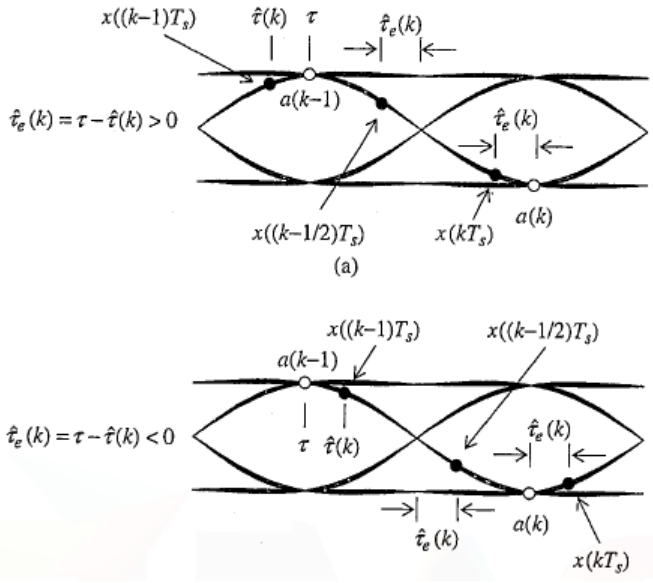


Figura 7. Ejemplo ZCTED con errores. [2]

Finalmente, la expresión del error para el ZCTED es:

$$e[n] = x\left[\left(n - \frac{1}{2}\right)T_s + \tau\right] * (\Phi[n-1] - \Phi[n]) \quad (1)$$

Siendo x[n] la salida del interpolador, Ts el tiempo de símbolo, τ el offset de tiempo, y Φ el símbolo decidido.

- Gardner: El que está activo dentro del simulador. Es similar al anterior, pero con la diferencia que el ZC necesita la decisión de los símbolos anteriores mientras que el Gardner no lo necesita. Este algoritmo fue desarrollado para modulaciones BPSK y QPSK, y como principal ventaja tiene que es rotacionalmente invariante, y por tanto es independiente de cualquier tipo de rotación sobre la constelación, siendo apta para utilizar antes la corrección fina de frecuencia.

  La salida de error del Gardner-TED es:

$$e[n] = x\left[\left(n - \frac{1}{2}\right)T_s + \tau\right] * (x[(n-1)T_s + \tau\,] - x[nT_s + \tau]) \quad (2)$$

Al igual que con el PLL, el TED introduce una ganancia Kp, que habrá que tener en cuenta en el diseño del filtro de retroalimentación.

- Loop Filter (Filtro de retroalimentación): es el mismo bloque que en un PLL. Se ha implementado un filtro de orden II con una estructura de control típica, un filtro proporcional más integrador.

  El objetivo de este bloque es controlar el comportamiento del sistema, ya que al filtrar el error se puede cambiar parámetros como la velocidad de adaptación antes cambios en el offtset de tiempo, o fijar el error residual una vez se ha enganchado el bucle de sincronismo.

  La expresión del filtro proporcional más integrador en el dominio discreto es:

$$F[z] = G_1 + \frac{G2}{1 - z^{-1}} \quad (3)$$

Siendo G1 la ganancia del término proporcional y G2 la ganancia del término integrador. La selección de estas ganancias depende de los parámetros característicos de los filtros de segundo orden, el ancho de banda de ruido equivalente $\in$ (0,1), y el factor de amortiguamiento mediante las siguientes expresiones:

$$\theta = \frac{Bloop}{N * (\tau + 0.25\tau)} \quad (4)$$

$$\Delta = 1 + 2\tau\theta + \theta^2 \quad (5)$$

$$G_1 = \frac{4\tau\theta}{N\Delta K_p} \quad (6)$$

$$G_2 = \frac{4\theta^2}{N\Delta K_p} \quad (7)$$

Donde Bloop es el ancho de banda de ruido equivalente, τ es el factor de amortiguamiento y N el número de muestras por símbolo.

Sincronismo fino de frecuencia

Mediante el sincronismo grueso de frecuencia se ha corregido el offset de frecuencia en gran medida, pero sigue existiendo una ambigüedad en función de la resolución en frecuencia del bin de la FFT, en este apartado se presentará una técnica que permita afinar más en la corrección del offset de frecuencia (idealmente se compensará el offset completamente).

Para poder implementar este algoritmo es necesario que se tenga una constelación estable, por lo cual es necesario realizar el proceso de Symbol Recovery previamente. El efecto del offset de frecuencia sobre la constelación es una rotación sobre la misma, que va girando continuamente. Si el offset es positivo, la constelación rotará en sentido antihorario, y viceversa. La velocidad de rotación dependerá del offset de frecuencia.

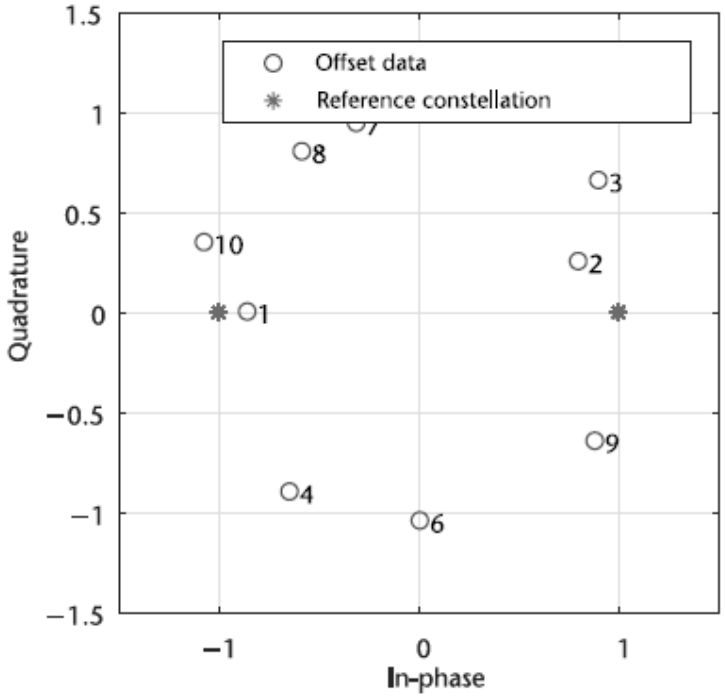


Figura 8: Ejemplo de constelación rotada debido a offset de frecuencia. [3]

El algoritmo implementado vuelve a estar basado en un PLL, como el algoritmo de Symbol Recovery, tal como se muestra en la siguiente figura:

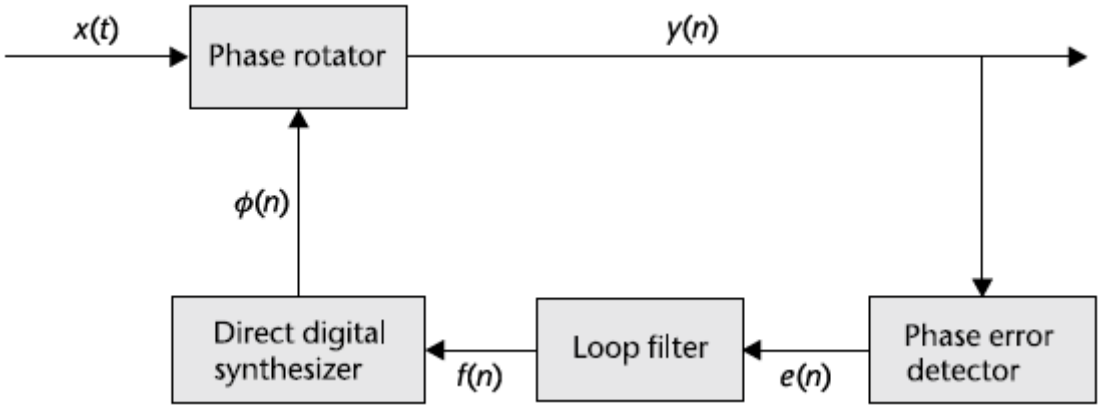


Figura 9: Algoritmo FFC. [3]

Se procede a explicar cada bloque del subsistema:

- Phase Error Detector (PED): Su objetivo es detectar la diferencia de fase entre la constelación recibida y la constelación ideal. Este bloque es el equivalente al Phase Detector en un PLL. La salida del bloque es una señal de error que es independiente de la amplitud de los símbolos recibidos, pero fuerza a que la parte real e imaginaria sean iguales:

$$e[n] = sign\big(Re(y[n]) * Im(y[n])\big) - sign\big(Im(y[n]) * Re(y[n])\big) \ (8)$$

- Loop filter: Se diseña igual que en el apartado de Symbol Recovery, cambiando el término de muestras por símbolo por el número de bits por símbolo de la constelación. Existe una relación entre el ancho de banda y la máxima corrección de frecuencia que es capaz de corregir el subsistema. La expresión se ha obtenido tras un estudio linealizado sobre el funcionamiento del bucle:

$$\Delta_{f,pull} \sim 2\pi\sqrt{2\tau} * Bloop \ (9)$$

  Siendo $\Delta_{f,pull}$ la máxima frecuencia normaliza que se puede corregir.

- DDS, Direct Digital Synthesizer: Sustituye al VCO en un PLL clásico. Se encarga de generar una exponencial compleja cuya fase es la salida acumulada en el integrador del loop filter cambiada de signo.
- Phase rotator: Aplica la corrección de desfase al símbolo entrante. Se multiplica el símbolo por la exponencial compleja del DDS, a la cual se le ha cambiado el signo de la fase de tal forma que se anule el error de fase al multiplicar.

$$y[n] = x[n] * \ e^{-j\Phi} \ (10)$$

Cuestiones (relativas al sincronismo de símbolo):

1) Si en el algoritmo FFC, se ha diseñado el loop filter con un factor de amortiguamiento τ de 1 (amortiguado críticamente), seleccione y justifique el parámetro del FFC para que sea capaz de corregir un offset en frecuencia de 2 KHz.
2) A partir del diseño anterior, compruebe experimentalmente la máxima desviación en frecuencia que es capaz de corregir el bloque. Justifique los resultados.

Sincronismo de Trama

El aspecto final de la sincronización es la sincronización de trama. Dado que el sistema es realista, el inicio de un cuadro aún será desconocido, lo que nos lleva a realizar un corrección o estimación adicional.

La estructura de la trama que se utilizará en este protocolo de comunicaciones contendrá un preámbulo dentro del encabezado para facilitar la sincronización a nivel de la trama. El diseño de la secuencia del preámbulo es importante: el preámbulo debe tener propiedades de correlación favorables para que sea fácilmente identificable.

Las secuencias adecuadas para un preámbulo de trama tienen una función de autocorrelación que produce un máximo claro cuando se alinean, pero un resultado muy bajo cuando las secuencias no están alineadas, como se muestra en la Figura 1.

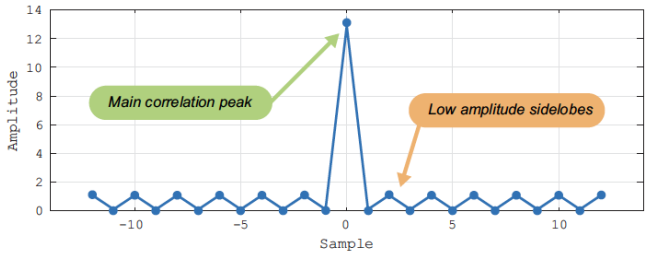


Figura 1 – Autocorrelación.

La sincronización de tramas se logrará monitoreando continuamente la correlación cruzada entre el preámbulo y los datos recibidos, definiendo un valor umbral, es decir, cuando los valores estén por encima de este umbral indicarán la detección de la secuencia del preámbulo (Figura 2).

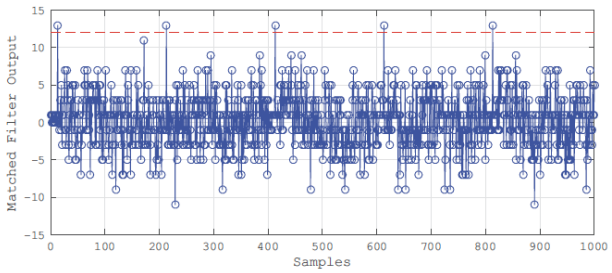


Figura 2 - Representación del umbral (en rojo)

- ¿Si ahora queremos implementar la correlación a través de un filtro FIR, qué coeficientes debemos utilizar?
- ¿Cuál es la potencia de ruido que se obtendría en la salida del modulador?

De-rotación de la Modulación

La constelación puede llegar rotada en algunas de las posibles posiciones de convergencia. Para corregirlo, se realizan mediciones de error de fase para determinar la rotación de la constelación en comparación con la posición deseada. Para esto, debemos multiplicar, término a término, el preámbulo recibido con el preámbulo esperado y obtener el promedio de las diferencias de fase. Una vez que tengamos la rotación, podemos corregir la fase.