Eduardo Nicolás Baciredo

Tutores: Ing. Adrián Carlotto
Ing. José Juárez

Receptor Portátil para Transmisiones del Sistema DCS

La Plata – Argentina

Agradecimientos

Aprovecho este espacio para agradecerles a todas las personas que estuvieron a mi lado ayudándome y guiándome en este camino que fue el estudio de esta profesión.

En primer lugar me gustaría agradecer a mi familia por estar conmigo en cada instante de este proceso. Quisiera agradecer principalmente a mis padres Ana y Eduardo por haberme impulsado a cumplir este sueño de ser profesional y a mi hermana Julieta por haber estado siempre a mi lado en el día a día.

En segundo lugar, pero no menos importante a los Ingenieros Adrián Carlotto y José Juárez por haberme ayudado en todo el proceso que fue desarrollar este trabajo, brindándome sus conocimientos y experiencias.

Finalmente me gustaría agradecerles a mis amigos por haber compartido conmigo estos años apoyando y ayudándome continuamente.

Resumen

El presente trabajo aborda todos los aspectos del desarrollo de un receptor para los sistemas de recolección de datos medioambientales DCS (Data Collection System), SCD (Satélite de Coleta de Dados) y ARGOS hasta concluir con su final implementación en MATLAB. Estos sistemas se basan en un conjunto de plataformas transmisoras ubicadas en sitios remotos, que miden parámetros en su entorno y un conjunto de satélites que recolectan esos datos para luego ser retransmitidos a la estación terrena donde son procesados.

El propósito de este trabajo es el de implementar un receptor de fácil utilización que permita corroborar el correcto funcionamiento de las plataformas terrestres durante su instalación.

Para este fin se planteó la solución de utilizar como etapa de RF un "dongle" (generalmente usado para televisión digital abierta) y realizar el procesamiento digital en una computadora. Con lo cual se tiene un esquema de funcionamiento del receptor de SDR o radio definida por software.

En la primera instancia del trabajo, como introducción, se describen los sistemas de DCS, SCD y ARGOS y seguido de esto, el hardware utilizado y su forma de uso.

Terminada la parte introductoria se describe el desarrollo de la etapa de procesamiento digital, compuesto por las etapas de: Detección, Estimación de parámetros, PLL, Filtro adaptado, Sincronismo de bit y Sincronismo de trama.

A su vez, se implementaron dos interfaces graficas que muestren el funcionamiento del receptor en varios puntos intermedios del proceso de recepción, así como también, una interfaz grafica del receptor terminado para el usuario final.

Finalmente para concluir el trabajo se muestran las mediciones de desempeño finales y las conclusiones de este desarrollo.

Índice

1.Introducción

En este trabajo se desarrollará un receptor en tierra para los sistemas de recolección de datos medioambientales DCS (Data Collection System), SCD (Satélite de Coleta de Dados) y ARGOS por lo cual en primera instancia se describen dichos sistemas.

1.1. Descripción de los Sistemas

Los sistemas DCS, SCD y ARGOS están formados por un conjunto de plataformas que sensan parámetros medioambientales de su entorno, un grupo de satélites que recolectan la información generada por las plataformas y estaciones terrenas que procesan dicha información.

Las plataformas pueden ser, estaciones terrenas encargadas de mediciones ambientales de una determinada región, boyas en mares y océanos, dispositivos transmisores en animales, etc. Estas plataformas se encargan que una vez que los datos son generados los mismos sean transmitidos al satélite, un esquema se muestra en la Imagen1.

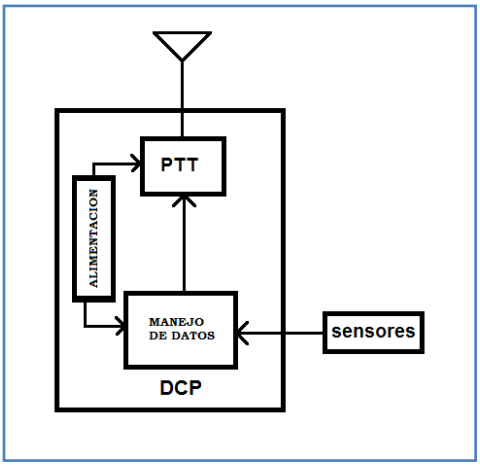

Imagen 1.1: Esquema de plataforma transmisora

Como puede verse en la figura, las plataformas (llamadas DCP: Data Collection Platform) cuentan con sensores encargados de obtener los datos a transmitir, unidad de almacenamiento y manejo de datos, alimentación y los terminales transmisores (PTT, del inglés Platform Transmitter Terminal).

Todas las plataformas poseen un periodo de repetición distinto en el cual pueden transmitir. Esta implementación tiene la función de minimizar las colisiones entre mensajes, evitando así que dos estaciones se interrumpan continuamente y que de esta forma se pierda alguna transmisión. El período de transmisión se encuentra preestablecido en las estaciones y tiene un valor de entre 45 y 200 segundos dependiendo de la aplicación. Otras características de las estaciones son su velocidad de transmisión, de 400bps para los tres sistemas y su frecuencia de funcionamiento, mostradas en la Tabla1.

Sistema	Frecuencia
DCS	401.55MHz
SCD	401.62MHz
ARGOS	401.65MHz

Tabla 1.1: Frecuencia de funcionamiento

Los satélites que usan estos sistemas son de órbita baja, es decir, estos recorren la superficie de la tierra y no poseen una posición fija. La función de los satélites es la de recolectar la información de todas las plataformas terrenas e ir almacenándola y cuando pasa por la estación terrena transmitirla para su procesamiento.

Las Estaciones terrenas realizan la recepción, procesamiento, catalogación, almacenamiento de los datos y la distribución al usuario de ciencia. También se encargan de realizar el seguimiento, telemetría y control de los satélites.

Imagen 1.2: Satélite SAC-D (Fuente: sitio web CONAE).

En la imagen se muestra el satélite Argentino SAC-D (actualmente fuera de funcionamiento) utilizado por el sistema DCS.

1.2. Motivación del trabajo

La motivación que impulsó el desarrollo de este trabajo fue la necesidad concreta de tener un receptor portátil que verifique el funcionamiento de las plataformas en el momento de su instalación.

La motivación que impulsó el desarrollo de este trabajo fue al momento de instalar una plataforma es necesario esperar que el satélite pase por el lugar donde la misma fue instalada, recolecte los datos, transmita estos a la estación terrena, que esta procese los datos y los ponga en línea, para poder verificar su funcionamiento. Por estos retrasos es que surge la necesidad de tener un receptor portátil que verifique al momento de la instalación el funcionamiento de las plataformas.

La solución que se plantea entonces es implementar un receptor utilizando una computadora y un "dongle de televisión digital abierta"

como etapa de RF. Esto significa que se adopta un esquema de recepción de SDR o radio definida por software.

1.3. Descripción de la Señal DCS

Para analizar la señal utilizada en el sistema DCS, primero dividimos la misma en dos partes. La primera parte está formada por los primeros 160ms de la señal en los cuales lo único que se transmite es la portadora, sin ningún tipo de modulación, es decir, se transmite una señal sinusoidal de frecuencia correspondiente al sistema (ver Tabla 1.1). La segunda parte de la señal es la transmisión que continua después de esos 160ms. En este instante de la transmisión es cuando la modulación comienza. El sistema de modulación utilizado es PSK binario. La fase de los símbolos varía ±1,1rad, es decir:

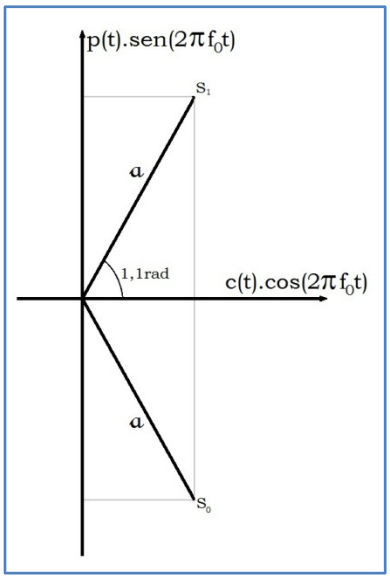

Imagen 1.3: Constelación

Siendo S_1 el símbolo correspondiente a "1" en binario, S_0 el símbolo correspondiente a "0", "a" es la amplitud de la señal, f_0 es la frecuencia de

la portadora, $p(t)$ es el pulso Manchester y $c(t)$ es un cajón que indica la duración del pulso. Es decir:

$$c(t) = \Pi\left(\frac{t - \frac{T_b}{2}}{T_b}\right) \quad \textbf{1.1}$$

$$\boldsymbol{p(t)} = \Pi\left(\frac{t - \frac{T_b}{4}}{\frac{T_b}{2}}\right) - \Pi\left(\frac{t - \frac{3T_b}{4}}{\frac{T_b}{2}}\right) \quad \textbf{1.2}$$

Donde Π es la función "Cajon", una función que vale 1 entre ½ y -½.

De constelación de la imagen1.3 puede verse que no importa que símbolo, siempre se transmite una porción de la potencia como portadora residual de amplitud $P_{res} = a.\cos(1{,}1rad)$. Esto es deseable ya que de esta forma se mejora la sincronización de portadora en condiciones de SNR (Signal to Noise Ratio) muy desfavorables.

La forma de pulso utilizada, Manchester (biphase-L), es:

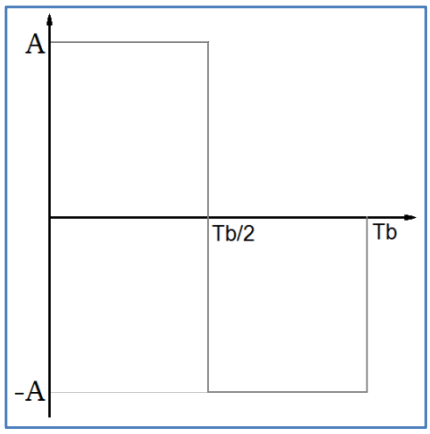

Imagen 1.4: Pulso Manchester

Donde Tb es 2,5ms ya que la tasa de bits es de 400bps.

Para comenzar el análisis de la densidad espectral de potencia de la señal, primero se expresa la señal recibida en función de su envolvente compleja, es decir:

$$Y(t) = Re\{X(t)e^{j2\pi f_0 t}\} \quad \textbf{1.3}$$

Donde $Y(t)$ es la señal recibida, $X(t)$ es la señal en banda base original y f_0 la frecuencia de la modulación.

Por lo cual podría decirse que existe un:

$$Z(t) = X(t)e^{j2\pi f_0 t} \quad \textbf{1.4}$$

Esto nos permite expresar la densidad espectral de potencia de $Y(t)$ de la siguiente manera:

$$S_{YY}(f) = \frac{1}{4}\left(S_{ZZ}(f) + S_{ZZ}(-f)\right) = \frac{1}{4}\left(S_{XX}(f - f_0) + S_{XX}(-f + f_0)\right) \quad \textbf{1.5}$$

Siendo $S_{YY}(f), S_{XX}(f)$ y $S_{ZZ}(f)$ las densidades espectrales de potencia de $Y(t)$, $X(t)$ y $Z(t)$ respectivamente. Para el análisis de la densidad espectral de potencia de la señal en banda base, es decir, $S_{XX}(f)$ se va a separar la señal en sus dos componentes, lo cual es posible debido a que $p(t)$ y $c(t)$ son ortogonales. Se va a analiza primero el espectro generado por los pulsos (componente en cuadratura) y luego el generado por la portadora residual.

La densidad espectral de la componente en cuadratura, se va a analizar utilizando el cálculo de espectro para señales PAM (Pulse Amplitude Modulation) [Ref 1.1]. Por lo cual se la puede expresar de la siguiente forma:

$$S_{XQ}(f) = \frac{|P(f)|^2}{T_b} S_{AA}\left(e^{-j2\pi f T_b}\right) \quad \textbf{1.6}$$

Donde $P(f)$ es la transformada de Fourier del pulso $p(t)$, $S_{AA}\left(e^{-j2\pi fT}\right)$ es la transformada de Fourier de la función de autocorrelación de la secuencia de amplitudes posibles (de valores A y –A) y T_b es el tiempo de bit. A su vez de la constelación del sistema se puede saber que $A = \pm a.sen(\theta)$, siendo θ la variación de fase de los símbolos (1,1 rad).

Para calcular la función de autocorrelación R_{AA}, se debe calcular

$$R_{AA}[k] = E\{A_{N+k}A_N^*\} = \begin{cases} E\{|A_N|^2\} & Para\ k = 0 \\ E\{A_{N+k}\}E\{A_N^*\} & Para\ k \neq 0 \end{cases} \quad \textbf{1.7}$$

Donde A_N es la secuencia de amplitudes.

Suponiendo ambos símbolos equiprobables, se tiene:

$$E\{A_{N+k}\} = E\{A_N^*\} = \frac{A}{2} - \frac{A}{2} = 0 \quad \textbf{1.8}$$

$$E\{|A_N|^2\} = \frac{A^2}{2} + \frac{A^2}{2} = A^2 = a^2 sen^2(\theta) \quad \textbf{1.9}$$

Quedando:

$$R_{AA}[k] = \delta[k]a^2 sen^2(\theta) \quad \textbf{1.10}$$

Finalmente si se calcula la transformada de Fourier resulta:

$$S_{AA}\left(e^{-j2\pi fT}\right) = a^2 sen^2(\theta) \quad \textbf{1.11}$$

Una vez calculado esto, se debe calcular la transformada de Fourier del pulso Manchester $p(t)$, (ecuación 1.2).

La transformada de Fourier de esta función resulta:

$$P(f) = \frac{T_b}{2} Sinc\left(\frac{T_b}{2} f\right) e^{-j\pi f \frac{T_b}{2}} - \frac{T_b}{2} Sinc\left(\frac{T_b}{2} f\right) e^{-j\pi f \frac{3T_b}{2}} \quad \textbf{1.12}$$

Ahora bien, si se reemplazan los valores calculados para $S_{AA}\left(e^{-j\,2\pi fT}\right)$ y $P(f)$ en la expresión de $S_{XQ}(f)$ (ecuación 1.6) se obtiene:

$$S_{XQ}(f) = \frac{\left|\frac{T_b}{2} Sinc\left(\frac{T_b}{2} f\right)\left(e^{-j\pi f \frac{T_b}{2}} - e^{-j\pi f \frac{3T_b}{2}}\right)\right|^2}{T_b} a^2 sen^2(\theta) \quad \textbf{1.13}$$

Reagrupando los términos y simplificando, se llega a:

$$S_{XQ}(f) = T_b Sinc^2\left(\frac{T_b}{2} f\right) sen^2\left(\frac{T_b}{2}\pi f\right) a^2 sen^2(\theta) \quad \textbf{1.14}$$

A esta expresión es necesario sumarle la componente de la portadora, con lo cual resulta:

$$S_{XX}(f) = T_b Sinc^2\left(\frac{T_b}{2} f\right) sen^2\left(\frac{T_b}{2}\pi f\right) a^2 sen^2(\theta) + a^2 sen^2(\theta)\delta(f) \quad \textbf{1.15}$$

Reemplazando $S_{XX}(f)$ en $S_{YY}(f)$ (ecuación 1.5):

$$S_{YY}(f) = \frac{1}{4}\left(S_{XX}(f - f_0) + S_{XX}(-f + f_0)\right) \quad \textbf{1.16}$$

Se puede despejar el valor de $S_{YY}(f)$:

$$\begin{aligned} S_{YY}(f) = \frac{1}{4}\Bigg(& T_b Sinc^2\left(\frac{T_b}{2}(f - f_0)\right) sen^2\left(\frac{T_b}{2}\pi(f - f_0)\right) a^2 sen^2(\theta) + a^2 sen^2(\theta)\delta(f - f_0) \\ & + T_b Sinc^2\left(\frac{T_b}{2}(-f + f_0)\right) sen^2\left(\frac{T_b}{2}\pi(-f + f_0)\right) a^2 sen^2(\theta) + a^2 sen^2(\theta)\delta(-f + f_0)\Bigg) \end{aligned} \quad \textbf{1.17}$$

Si se grafica esta densidad espectral de potencia resultante se obtiene:

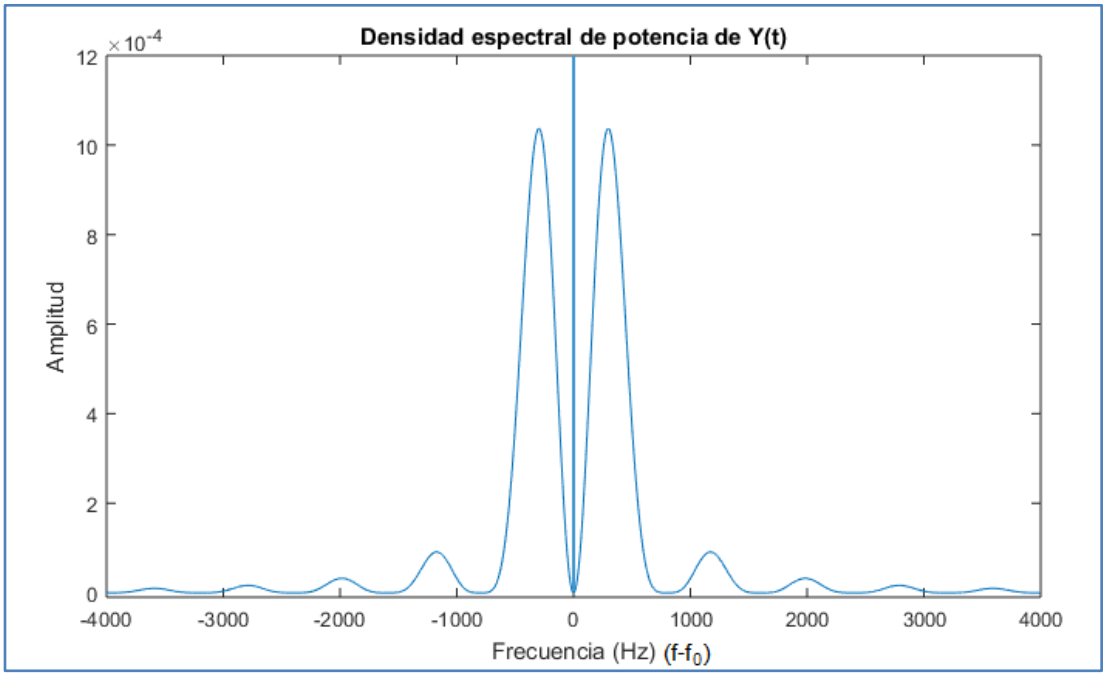

Imagen 1.5: Densidad espectral de la señal transmitida

Es necesario explicar que la frecuencia graficada como 0Hz corresponde a la frecuencia de la portadora en el espectro real, esto

Otro dato que se puede concluir del gráfico es que el ancho de banda que contiene la mayor parte de la potencia es de 1600Hz.

1.4. Estructura del mensaje

A continuación se muestra la estructura del mensaje:

160 ms De portadora	Sincronismo de bit	Trama	inicio	N° de grupos	DCP ID	Datos	Checksum
	15	8	1	4	20	32 - 256	8

Tabla 1.2: Estructura de mensaje

Esta estructura de mensaje consta de un preámbulo cuya finalidad es que el receptor pueda sincronizarse (en portadora, bits y trama).Comienza con 160ms en los cuales lo único que se transmite es la portadora de la transmisión. Una vez que la modulación comienza, lo primero que se transmite son 15 unos para que el receptor pueda encontrar el reloj de bit se sincronice. A continuación se transmite la trama, que consta de 8 bits de valor conocido (00010111). Esta trama cumple tres funciones, primero sirve como punto de referencia para saber en qué lugar del mensaje se encuentra. Segundo, cumple la función de resolver una ambigüedad de π en la fase que el sincronismo de portadora no puede resolver. Tercero como verificador de la transmisión que se está recibiendo, es decir, si la trama que se recibe es distinta a la mencionada el mensaje será descartado.

Luego, se transmite un uno para indicarle al receptor que a partir de ese punto comienza la transmisión de los datos relevantes del mensaje.

Los siguientes 4 bits corresponden al campo de largo de mensaje, este campo indica cuantos paquetes de 32 bits posee el campo de datos propiamente dicho.

La forma en la cual se codifican estos 4 bits es la siguiente:

Bits de N	N° de paquetes	N° bytes
0000	1	4
0011	2	8
0101	3	12
0110	4	16
1001	5	20
1010	6	24
1100	7	28
1111	8	32

Tabla 1.3: Codificación de N° de grupos

Seguido de esto se transmite la identificación de plataforma, este campo consta de 20 bits. Este bloque también implementa un código de corrección de errores simples.

El campo siguiente es el de datos. Este campo consta de un máximo de 256 bits y un mínimo de 32 bits para transmitir información.

Finalmente el último campo que se transmite es el Checksum, este campo sirve para verificar la transmisión y costa de 8 bits.

Dado que el campo de datos es variable se tiene una duración mínima del mensaje de 380ms cuando se transmiten los 88 bits del mensaje más corto y una duración máxima de 940ms cuando se transmite el mensaje más largo, de 312 bits.

1.5. SDR (Software Defined Radio)

Radio definida por software o SDR es una forma de implementar un sistema de comunicaciones en el cual varios de los componentes, típicamente implementados en hardware (mezcladores, filtros, moduladores, demoduladores, etc.) son implementados en software utilizando una computadora.

Esta forma de implementación de sistemas de comunicación, se ha popularizado en el último tiempo debido a que en la actualidad, el avance tecnológico ha permitido el surgimiento de dispositivos de adquisición de datos de muy elevadas tasas de muestreo. Permitiendo que las radios definidas por software puedan alcanzar frecuencias de funcionamiento muy elevadas.

Un sistema típico de SDR, consta de una placa de adquisición o conversor analógico-digital que se encarga de muestrear la señal y una computadora encargada de procesar dichas muestras.

Esquemáticamente:

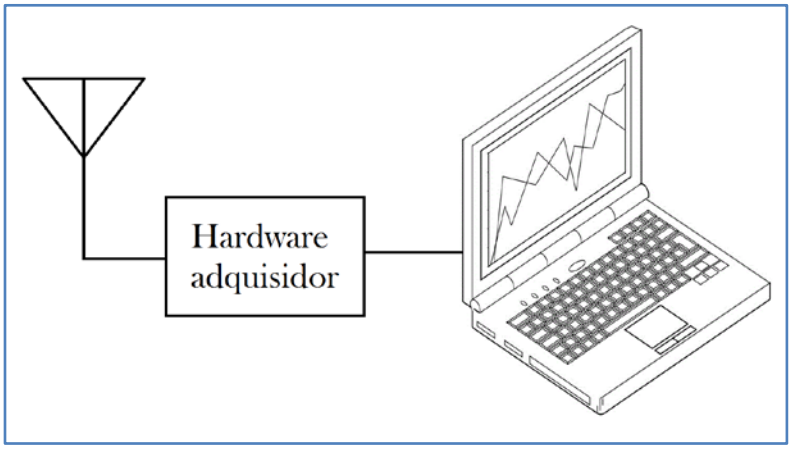

Imagen 1.6: Esquema de receptor SDR

Por lo cual surge el problema de seleccionar correctamente el hardware adquisidor.

2.Descripción de Hardware

Como resulta en el capitulo anteriormente analizado, la modulación implementada en el sistema DCS, es digital, ello quiere decir que lo que se desea recibir es una secuencia de unos y ceros que contienen la información. A su vez la idea del trabajo es realizar un receptor descripto por software o SDR (Software Defined Radio) lo que implica que la totalidad del desarrollo del mismo será mediante software.

Esto nos lleva tener que considerar la forma en que será implementada la etapa de RF, optándose en este caso por la utilización de un dispositivo Front-end de RF, que se desarrollará en el presente capítulo.

Un dispositivo Front-end de RF es el conjunto de circuitos utilizados en receptores de radio frecuencia, que procesan la señal desde que es recibida por la antena hasta que es convertida a frecuencia intermedia. Dicho dispositivo esta generalmente integrado por:

- Adaptadores de impedancia
- Filtros pasa banda
- Amplificadores de RF de bajo ruido
- Osciladores locales
- Mezcladores

En el mercado existen variedad de estos dispositivos, por lo tanto a continuación analizaremos cuál de estas opciones se adecua más a los fines perseguidos.

2.1. Selección de Front-End de RF

Pese a que existe bastante diversidad de estos dispositivos Front-End de RF, la gran mayoría presenta una forma constructiva similar. Se basan en dos integrados, uno cumpliendo la función de sintonizador y otro encargado del manejo de muestras. Esquemáticamente resulta:

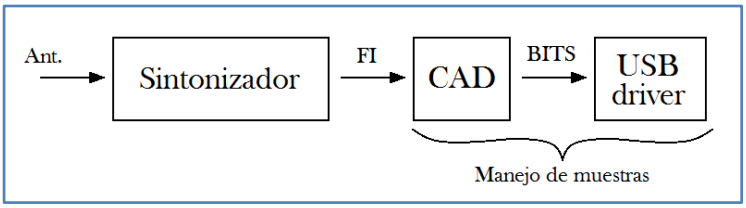

Imagen 2.1: Esquema Front-End de RF

En el esquema puede verse, donde se tiene la frecuencia intermedia y donde se encuentran los bits.

El integrado que se encarga del manejo de muestras es común para todos los dispositivos, se encuentra patentado y no se dispone de información más que la necesaria para su funcionamiento.

En cuanto al primer integrado, encargado de la sintonización se tienen diversas opciones, los dispositivos más comunes en el mercado utilizan los siguientes circuitos sintonizadores:

- Elonics E4000
- Rafael Micro R820T
- Rafael Micro R828D
- Fitipower FC0013
- Fitipower FC0012
- FCI FC2580

De los integrados mencionados, a excepción del FCI FC2580, todos califican en primera instancia para el trabajo, debido a que pueden manejar señales del orden de los 400MHz (frecuencia a la cual operan los sistemas).

A continuación se muestran los rangos:

Sintonizador	Rango de frecuencias	
	Mínima	Máxima
Elonics E4000	52 MHz	2200 MHz
Rafael Micro R820T	24 MHz	1766 MHz
Rafael Micro R828D	24 MHz	1766 MHz
Fitipower FC0013	22 MHz	1100 MHz
Fitipower FC0012	22 MHz	948.6 MHz
FCI FC2580	146 MHz	924 MHz*

Tabla 2.1: Rango de frecuencia de los sintonizadores

Es necesario aclarar, que si bien el FCI FC2580 parece contener la frecuencia de 400 MHz, presenta una banda prohibida entre 438 MHz y 924 MHz.

De los circuitos sintonizadores mencionados, los más populares son el "Elonics E4000" y el "Rafael Micro R820T" debido a que existe gran variedad de dispositivos Front-end que los usan.

Elonics E4000

Imagen 2.2: Circuito Elonics E4000

El Elonics E4000, es el circuito sintonizador más frecuentemente utilizado en aplicaciones de radio particulares. [Ref 2.1]

Sus características principales son las siguientes:

Característica	Valor
Potencia consumida	118mW
Figura de ruido	<4dB
Tensión de alimentación	1.5V
Potencia de entrada Max.	+10dBm
Ruido de fase @10kHz	-80dBc/Hz

Tabla 2.2: Características principales del circuito Elonics E4000

Pese a ser un circuito muy popular y que todavía pueden conseguirse unidades, es un dispositivo que recientemente ha sido descontinuado.

Rafael Micro R820T

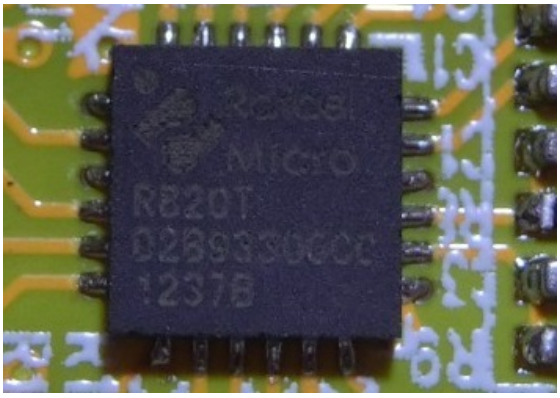

Imagen 2.3: Circuito Rafael Micro R820T

El Rafael Micro R820T, es actualmente el circuito más económico y de mejor rendimiento en el mercado. [Ref 2.2]

Sus características principales son las siguientes:

Característica	Valor
Corriente consumida	<178mA
Figura de ruido	3.5dB
Tensión de alimentación	3.3V
Potencia de entrada Max.	+10dBm
Ruido de fase @10kHz	-98dBc/Hz
Rechazo de frecuencia imagen	65dBc

Imagen 2.4: Características principales del circuito Rafael Micro R820T

En conclusión, de los dispositivos disponibles, el Elonics E4000 presenta gran desempeño en frecuencias bajas, no así para altas frecuencias, en las que decae su rendimiento. Presenta además la gran desventaja de tener poca disponibilidad y elevado costo.

Los dispositivos basados en los circuitos Rafael Micro R828D, Fitipower FC0013, Fitipower FC0012, FCI FC2580 son dispositivos muy raros, y por lo tanto, caros y de poca disponibilidad.

Como puede verse el mayor inconveniente presentado a la hora de la selección del Front-end de RF, es la disponibilidad y el precio, más aún, con las dificultades que conlleva acceder a dispositivos importados.

En este sentido, las ventajas más destacadas del circuito Rafael Micro R820T lo convierten en la mejor opción, ya que, los front-end's con éste circuito, son más económicos y más variados en cuanto a marcas que lo fabrican, ello mejora su disponibilidad. Se debe adicionar a estas ventajas, que el desempeño de este circuito, es muy bueno.

Otra gran ventaja de este dispositivo es que el soporte en MATLAB ha sido ampliamente probado.

Para concluir, por las bondades de este circuito y principalmente por sus ventajas prácticas, de disponibilidad y precio, se eligió para éste trabajo el circuito Rafael Micro R820T, y dentro de de los posibles dispositivos que lo utilizan se optó por usar un “dongle” de televisión digital abierta genérico. A continuación se muestra una foto del mismo:

Imagen 2.5: Dongle de TDA

2.2. Descripción del Front-end de RF

Como se mencionó previamente, el integrado encargado de la digitalización de la señal (RTL2832U) se encuentra sujeto a una licencia, pero sin embargo, se puede describir el funcionamiento del R820T que representa la etapa de sintonización. El esquema de funcionamiento de dicho integrado puede verse en la imagen2.3. Este esquema de funcionamiento del R820T puede encontrarse en la hoja de datos del mismo [Ref 2.2]. La información que se puede obtener del esquema anterior es, que cuenta con un LNA (low noise amplifier) integrado de entrada encargado de realizar la primera amplificación, ya que los niveles de la señal en esta instancia son muy bajos. A continuación, la señal es filtrada y mezclada para trasladarla a frecuencia intermedia. En última instancia, la señal resultante, es nuevamente filtrada y amplificada por un amplificador de ganancia variable, que es configurable por el usuario. El conjunto de bloques de la mitad inferior del esquema se encargan de generar una señal de referencia lo más estable posible utilizando un

oscilador a cristal. Esta señal generada es la que se mezcla con la señal de entrada para hacer el pasaje a frecuencia intermedia.

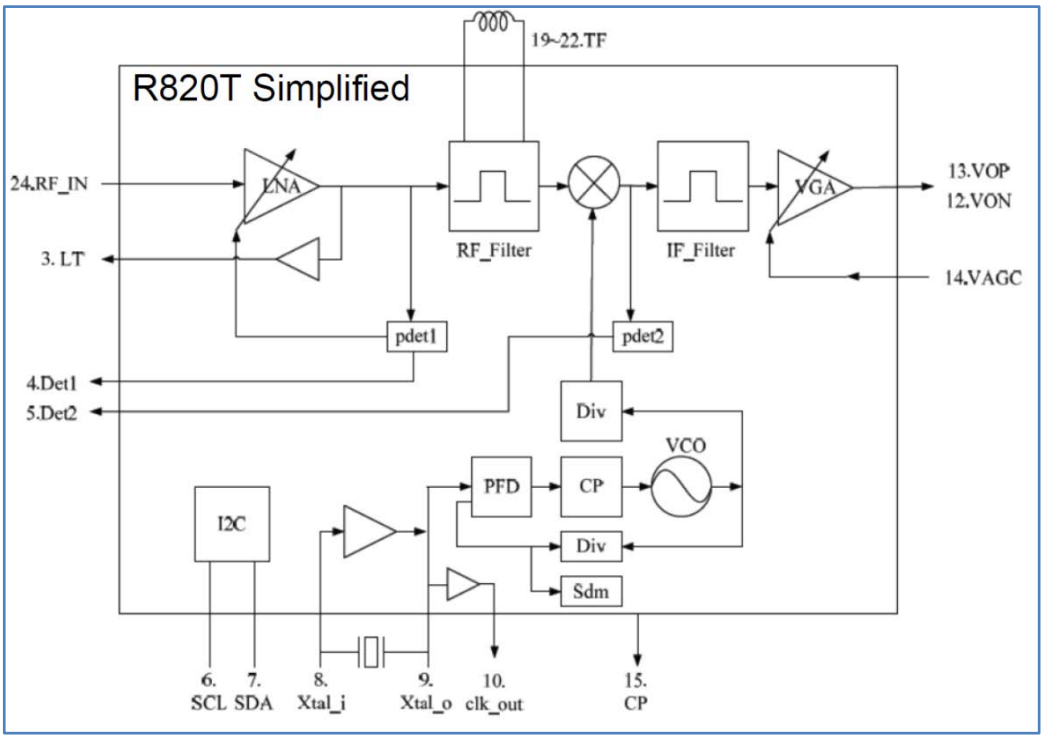

Imagen 2.6: Esquema de funcionamiento del R820T

Para entender mejor el funcionamiento total del dongle se muestra un diagrama simplificado que integre ambos circuitos, (imagen2.4). Puede apreciarse en este esquema como el R820T se comunica con el RTL2832U, (conformado por el ADC y el Driver). Si bien no se tiene la hoja de datos sobre este último integrado, se sabe que en el mismo se lleva a cabo el procesamiento digital de la señal y la comunicación con la PC, la cual permite, a su vez, controlar el R820T. La salida que se obtiene del RTL2832U son muestras de 8 bits en fase y cuadratura (I y Q).

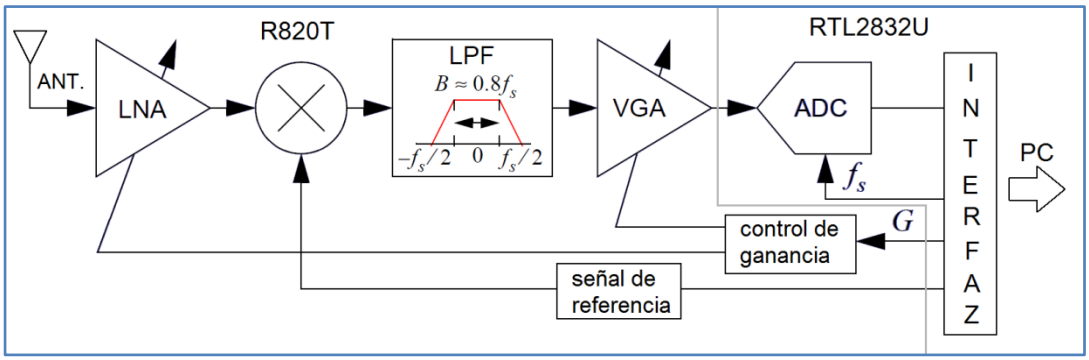

Imagen 2.7: Esquema simplificado de funcionamiento del Dongle

2.3. Comunicación con MATLAB

La utilización del Front-end de RF en una PC con un sistema operativo Windows, requiere la instalación de un set de drivers. Estos drivers, así como las funciones implementadas en MATLAB se instalan de forma automática con el paquete de soporte que trae MATLAB.

El paquete de soporte de MATLAB, crea un objeto de sistema el cual permite el control del dispositivo y todos sus parámetros. Este objeto de sistema actúa como fuente de las muestras.

En este punto es necesario aclarar que el objeto de sistema puede ser usado como una función o como un bloque en "Simulink". La forma de implementación utilizando la función creada no se realiza sobre un esquema de proceso de tiempo real ya que MATLAB no incorpora funcionalidades de tiempo real para sus códigos de forma intrínseca. En cambio la otra opción utilizando "Simulink" puede ser usada en conjunto con "Real Time Workshop" que es un entorno que posee MATLAB para asegurar la funcionalidad de tiempo real. El inconveniente con este entorno es que el mismo se basa en programación por diagrama de bloques, y de esta forma no se posee el total control sobre todos los parámetros del programa. Teniendo en cuenta esto, se optó por utilizar la función en lugar del diagrama de bloques, ya que existe un método que

permite asegurar si el proceso pierde muestras durante su ejecución creando así un resultado equivalente al de tiempo real.

Como se dijo, este objeto de sistema "comm.SDRRTLReceiver" posee todos los parámetros configurables del receptor. A continuación se muestran cuales son dichos parámetros:

Parámetros	
RadioAddress	Dirección del USB en el cual está conectado el dispositivo. La dirección por defecto es '0'. Mediante la función sdrinfo se puede conocer los dispositivos conectados a la computadora
CenterFrequency	Frecuencia central deseada especificada como valor de doble precisión no negativo. El valor por defecto es 102.5 MHz
EnableTunerAGC	Habilita el control automático de ganancia
TunerGain	Ganancia del dispositivo especificada como valor de doble precisión. El valor por defecto es '0'. Esta característica aparece únicamente cuando EnableTunerAGC es falso.
SampleRate	Tasa de muestreo especificada como valor de doble precisión positivo. El valor por defecto es 250kHz y el valor máximo es 3.2MHz
OutputDataType	Especifica el tipo de dato de salida como doble precisión, simple precisión o entero de 16 bits.
SamplesPerFrame	Numero de muestras por frame. Por defecto 1024
FrequencyCorrection	Corrección de frecuencia en ppm

Tabla 2.3: Parámetros configurables del receptor

Finalmente la forma de implementar esta función es creando el objeto "radio" donde se especifican todos estos parámetros. A continuación se muestra la creación del objeto:

```
% Ejecución del objeto radio

radio = comm.SDRRTLReceiver('0', 'CenterFrequency', 91.8e6, 'SampleRate',
1024000, 'SamplesPerFrame', 2048, 'EnableTunerAGC', true,
'OutputDataType', 'double')
```

Una vez definidos todos los parámetros del receptor, se debe utilizar el método “Step” para pedir muestras. A continuación se muestra como se utiliza este método:

```
% Petición de datos por método step
datos = step(radio);
```

De esta forma en la variable “datos” se obtiene un set de muestras de largo especificado en la definición del objeto.

Existe otra forma de ejecutar este método, la cual permite obtener parámetros adicionales de desempeño del receptor.

```
% Petición de datos y parámetros por método step
[rx,LEN,LOST,LATE] = step(h);
```

Donde “LEN” es el número de muestras obtenidas, “LOST” es la cantidad de muestras perdidas (este parámetro siempre es igual a un número entero de veces el largo del set de muestras) y “LATE” es el retardo en el procesamiento digital medido en cantidad de sets de muestras.

3. Introducción al desarrollo del receptor

En este capítulo se trataran todos los procesos necesarios para lograr la recepción de datos luego de la etapa de RF.

Al comenzar la recepción debido a que el receptor no tiene forma de conocer si se está transmitiendo o no la señal que debe recibir, o cuando comenzará a trasmitirse la misma, es necesario que la primera tarea a realizar sea la de DETECCIÓN.

Una vez que se detecta que la transmisión está presente, el receptor debe comenzar a estudiar la señal con el fin de conocer los parámetros de la misma para lograr la recepción, por lo cual inicia la etapa de ESTIMACIÓN de parámetros.

Para lograr que la portadora se sincronice se utiliza un LAZO DE ENGANCHE DE FASE que como la señal que se desea recibir esta modulada en fase además realiza la demodulación. En adición a esto, el lazo compensa las posibles variaciones en frecuencia que se podrían presentar por la digitalización de la señal o por posibles errores en los osciladores.

Luego de que la señal es demodulada, lo que se tiene es una secuencia de símbolos. A su vez esta secuencia se encuentra inmersa en ruido, por lo cual se debe procesar la misma para optimizar el instante en el cual se toma la decisión sobre si lo que se recibió es un uno o un cero. Dicho procesamiento es óptimo cuando se utiliza el FILTRO ADAPTADO.

Una vez que la señal sale del filtro adaptado lo que se obtiene es una señal que en determinados instantes presenta la máxima distancia con el umbral de decisión. Para poder determinar cuáles son esos instantes y de esta forma lograr el mejor desempeño en la decisión se realiza el SINCRONISMO DE BIT.

Como el sistema utilizado es PSK binario, la transmisión presenta una incertidumbre de fase de π (rad), es decir, que se confunden los símbolos entre sí, pudiendo dar como resultado la secuencia inversa. Para solucionar esto, se recurre al SINCRONISMO DE TRAMA. A su vez como se conoce la ubicación de la trama dentro del mensaje, la misma se utiliza para conocer en que instante de la transmisión se encuentra.

4.Detección

Para analizar la detección de la señal en este receptor se parte de que cuando la transmisión comienza existen 160 ms en los cuales se transmite únicamente la portadora de la señal. Esta portadora es una señal sinusoidal de la cual no se conocen los parámetros (amplitud, frecuencia, fase inicia).

A su vez esta señal se encuentra inmersa en ruido que se modela como ruido blanco, aditivo y gaussiano (AWGN).

Para resolver esto, se utiliza un test de hipótesis, donde las hipótesis bajo análisis son.

$$H_0: x[n] = w[n] \quad \textbf{4.1}$$

$$H_1: x[n] = s[n] + w[n] \quad \textbf{4.2}$$

Como se puede apreciar la hipótesis nula contempla el caso en el cual la señal de entrada es únicamente el ruido blanco, es decir, que no hay transmisión presente. Por otro lado, la hipótesis primaria, considera que se está recibiendo una señal sinusoidal (portadora) inmersa en ruido blanco.

Como s[n] es una señal sinusoidal queda:

$$H_1: x[n] = Acos(2\pi f_0 n + \theta) + w[n] \quad \textbf{4.3}$$

Donde w[n] es el ruido blando aditivo y gaussiano que tiene media nula y varianza σ^2, con n=0, 1, 2…, N-1, siendo N el número de muestras, f_0 la frecuencia de portadora y θ la fase inicial.

Como se modela al ruido como blanco aditivo y gaussiano se tiene distribución normal de media nula y varianza σ^2 ($N(0,\sigma^2)$) y a su vez la señal sinusoidal es deterministica, la distribución de probabilidad de cada hipótesis es:

$$fdp(x;H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x[n])^2}{2\sigma^2}} \quad \textbf{4.4}$$

$$fdp(x;H_1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\left(x[n]-Acos\,(2\pi f_0 n+\theta)\right)^2}{2\sigma^2}} \quad \textbf{4.5}$$

Como lo que se busca en la detección es que la hipótesis primaria sea verdadera, es decir, que exista señal, se plantea

$$\frac{p\left(x;\hat{A},\hat{\theta},\hat{f_0},H_1\right)}{p(x;H_0)} > \gamma \quad \textbf{4.6}$$

Esto significa que la hipótesis H_1 será cierta cuando la probabilidad de que ocurra sea γ veces mayor a la probabilidad de que ocurra la hipótesis H_0.

Entonces lo que se plantea a continuación es buscar para qué frecuencia es cierta dicha relación. Para ello se busca primero la frecuencia que maximice la probabilidad de la hipótesis primaria.

O lo que es equivalente.

$$\frac{max_{f_0}\left(p\left(x;\hat{A},\hat{\theta},f_0,H_1\right)\right)}{p(x;H_0)} > \gamma \quad \textbf{4.7}$$

Como la distribución de probabilidad de H_0 no depende de la frecuencia se puede escribir.

$$max_{f_0}\left(\frac{p\left(x;\hat{A},\hat{\theta},f_0,H_1\right)}{p(x;H_0)}\right) > \gamma \quad \textbf{4.8}$$

Ahora, como lo que se busca es maximizar esto, para simplificar los cálculos se aplica él logaritmo natural a ambos lados que como es una

función monótonamente creciente no modifica los máximos de los parámetros en los cuales se maximiza la función de la función.

$$max_{f_0}\left(ln\left(\frac{p(x;\hat{A},\hat{\theta},f_0,H_1)}{p(x;H_0)}\right)\right) > ln(\gamma) \quad \textbf{4.9}$$

Finalmente si se conoce la frecuencia a la cual ocurre el máximo se puede deducir que:

$$ln\left(\frac{p(x;\hat{A},\hat{\theta},f_0,H_1)}{p(x;H_0)}\right) = \frac{I(f_0)}{\sigma^2} \quad \textbf{4.10}$$

Reemplazando:

$$I(f_0) > \sigma^2 ln(\gamma) \quad \textbf{4.11}$$

Con lo cual queda:

$$max_{f_0}\big(I(f_0)\big) > \sigma^2 ln(\gamma) = \gamma' \quad \textbf{4.12}$$

Siendo $I(f_0)$:

$$I(f_0) = \frac{1}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j\,2\pi f_0 n}\right|^2 \quad \textbf{4.13}$$

Se puede observar que $I(f_0)$ es el modulo al cuadrado de la transformada discreta de Fourier (TDF) de $x[n]$. Lo cual significa que a partir de este punto no se tiene una representación continua para f_0, por eso a partir de este punto se denomina k_0 a la frecuencia representable que corresponde a f_0 en el eje discreto. Para el cálculo de la misma se utilizará el algoritmo de la Transformada Rápida de Fourier (FFT).

Todo esto significa que se debe buscar el máximo de $I(k_0)$ y compararlo con un umbral γ'. Si el máximo de $I(k_0)$ supera el umbral se tiene que la hipótesis H_1 es verdadera y la frecuencia a la cual ocurre dicho máximo será la estimación de la frecuencia de la portadora.

El umbral se lo puede determinar mediante la probabilidad de falsa alarma (PFA) deseada. La probabilidad de falsa alarma es la probabilidad de que se detecte una transmisión cuando en realidad no exista ninguna.

Esta probabilidad se encuentra ligada al umbral de detección de la siguiente manera [Ref. 4.1]

$$PFA = e^{-\frac{\gamma'}{\sigma^2}} \qquad \textbf{4.14}$$

Con lo cual el proceso de detección queda de la siguiente forma:

- Obtener las muestras
- Calcular el módulo cuadrado de la FFT
- Buscar el máximo
- Comparar con el umbral

A continuación se muestra una simulación de la detección para una SNR de -2dB. El código utilizado se encuentra en el [Anexo 1].

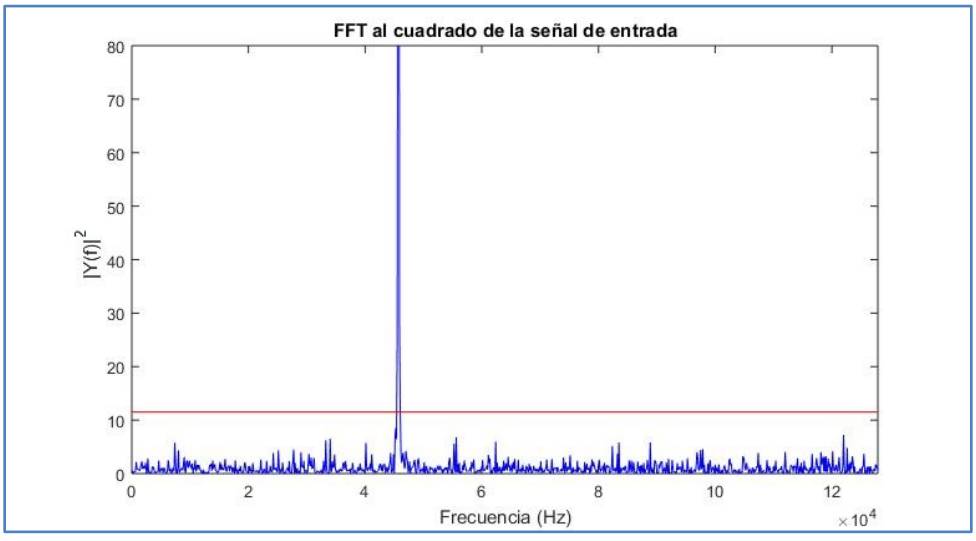

Imagen 4.1: Simulación de una detección

Este proceso de detección se implementa dos veces seguidas, es decir, si se toma un set de muestras y en el mismo la señal es detectada, se debe tomar el siguiente set de muestras y verificar que la señal sea nuevamente detectada. Y si esto se verifica, sobre este último set de muestras es sobre el cual se ejecutan los algoritmos de estimación.

Esta forma de implementación resuelve dos problemas potenciales. Primero que nada la posible detección de una señal que no existe debida a algún error esporádico. Y segundo, que si bien la señal fue detectada en el primer set de muestras, no se sabe si es que la señal existió en todas esas muestras o apareció en algún instante en medio de la toma de las mismas. Esto es importante de tener en cuenta ya que si se aplicara el algoritmo de estimación sobre el primer set de muestras se estaría cometiendo un error considerable en los valores obtenidos.

5.Estimación

Una vez que se detecta la presencia de una señal transmitida, es necesario estimar la frecuencia y la amplitud de la señal recibida para poder establecer la correcta recepción.

5.1. Frecuencia de portadora

Como ya se mencionó previamente, la frecuencia se estima al momento de la detección. Una vez que se calcula el máximo del módulo al cuadrado de la FFT, se puede en el mismo paso obtener la frecuencia a la cual ocurre dicho máximo.

Como la cantidad de muestras que se utilizan para el cálculo de la FFT es finita, el valor de frecuencia obtenido tendrá cierto error.

Si se muestrea a una frecuencia de muestreo f_m y a su vez se toman N muestras en total, la resolución en frecuencia que se tiene es de:

$$\Delta f = \frac{f_m}{N} \qquad \textbf{5.1}$$

Lo cual significa que el error máximo que se puede cometer es cuando la frecuencia cae a la mitad de dicho intervalo, es decir:

$$Err_{max} = \frac{f_m}{2N} \qquad \textbf{5.2}$$

Tomando un total de 4096 muestras (N), se ve que el error en frecuencia crece a medida que aumento la frecuencia de muestreo f_m. Como se muestra en la siguiente tabla (Tabla 5.1):

Error	f_m
3,9 Hz	32 kHz
7,8 Hz	64 kHz
15,6 Hz	128 kHz
31,2 Hz	256 kHz

Tabla 5.1: Error de frecuencia en función de la frecuencia de muestreo

Estos errores de frecuencia son las magnitudes de los escalones de frecuencia que debe poder rechazar el PLL.

Seleccionar una frecuencia de muestreo chica, manteniendo el número de muestras constante produce que el set de muestras tarde más tiempo en generarse.

Ahora evaluando desde la otra perspectiva, se toma una frecuencia de muestreo (f_m) de 256 kHz y calcula para distintos números de muestras

Error	N
125 Hz	1024
62,5 Hz	2048
31,2 Hz	4096
15,6 Hz	8192

Tabla 5.2: Error de frecuencia para distintos números de muestras

Seleccionar un número de muestras reducido trae aparejado baja precisión a la hora de estimar la amplitud de la señal, lo cual puede provocar errores más severos.

Para concluir, intentado compensar ambos efectos se seleccionó como frecuencia de muestreo f_m=256 kHz y número de muestras N=4096.

5.2. Amplitud

El estimador de máxima verosimilitud para cundo el ruido del canal puede modelarse como blanco aditivo gaussiano es [Ref. 5.2]

$$\hat{A} = \frac{2}{N}\left|\sum_{n=0}^{N-1} x[n]e^{-j2\pi k_0 n}\right| \qquad \textbf{5.3}$$

Que como puede apreciarse es el módulo de dos veces la FFT evaluada en el valor estimado def_0, es decir k_0. Reescribiendo:

$$\hat{A} = \frac{2}{N}FFT(k_0) \qquad \textbf{5.4}$$

Para demostrar esto se comienza por simular una señal sinusoidal (o una portadora) de amplitud A y frecuencia f_0. Si se calcula la FFT se obtiene:

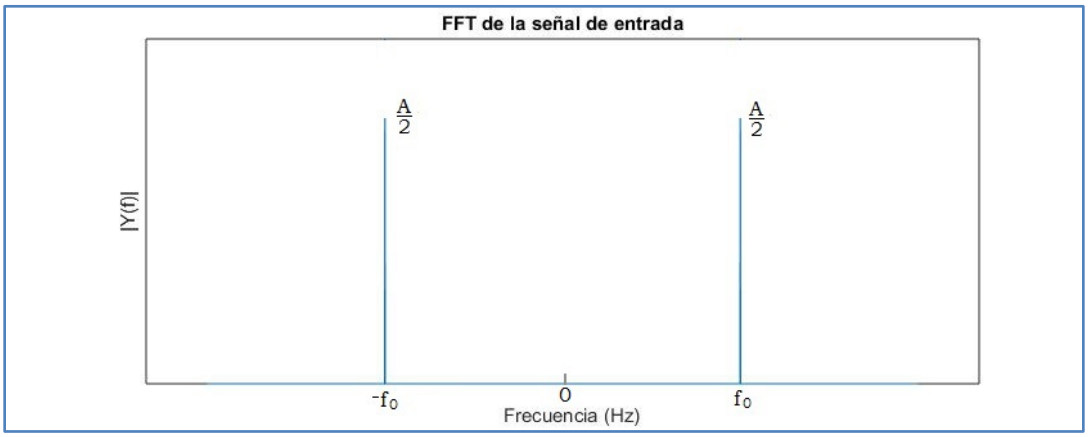

Imagen 5.1: FFT de señal sinusoidal sin ruido

Para esta simulación se seleccionó la frecuencia de muestreo de tal forma que sea múltiplo de la frecuencia de la seña, por eso en el gráfico aparecen solo dos muestras de valor A/2 en ±f_0.

Tomando solo las frecuencias positivas se obtiene:

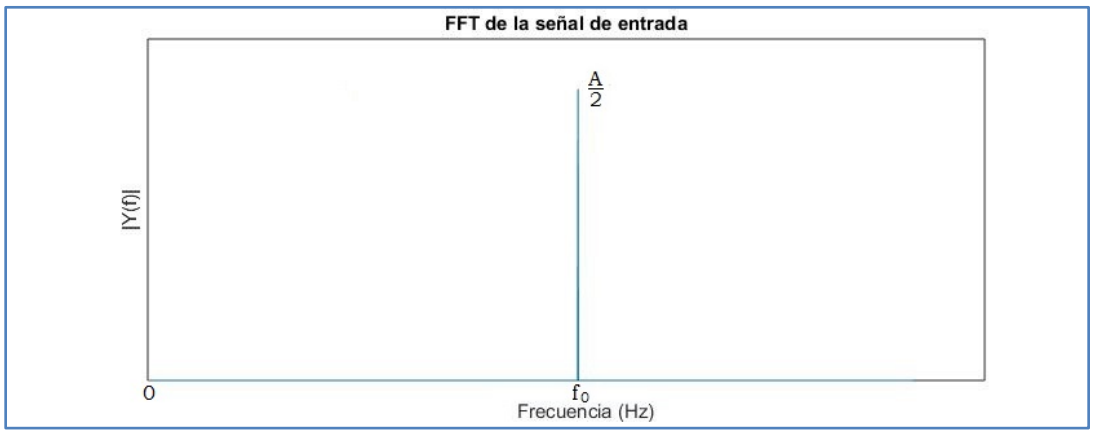

Imagen 5.2: FFT de señal sinusoidal sin ruido para frecuencias positivas

Si ahora se aplica la relación de Párseval para la transformada discreta de Fourier.

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{1}{N}\sum_{k=0}^{N-1}|X[k]|^2 \qquad \textbf{5.5}$$

Donde $x[n]$ son las muestras de la señal, $X[k]$ es la transformada de Fourier de la misma y N es la cantidad de muestras.

Esta relación significa que la energía de la señal es propia de la señal y no importa en qué dominio se calcule (tiempo o frecuencia).

Como solo se toman los valores positivos de frecuencia, es decir, solo el valor en f_0 en el gráfico de la FFT, se estaría tomando la mitad de la energía. Si se considera esto resulta:

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1}|X[k]|^2$$

Calculando el primer término de la expresión se obtiene:

$$\sum_{n=0}^{N-1}|x[n]|^2 = \frac{A^2}{2}N \qquad \textbf{5.6}$$

Reemplazando en la expresión previa y despejando:

$$\frac{A^2}{2}N = \frac{2}{N}\sum_{k=0}^{N-1}|X[k]|^2 \qquad \textbf{5.7}$$

$$A^2 = \frac{4}{N^2}\sum_{k=0}^{N-1}|X[k]|^2 \qquad \textbf{5.8}$$

$$A = \sqrt{\frac{4}{N^2}\sum_{k=0}^{N-1}|X[k]|^2} \qquad \textbf{5.9}$$

Ahora, si se analiza el término: $\sum_{k=0}^{N-1}|X[k]|^2$ se puede ver que es la suma del módulo cuadrado de cada componente en frecuencia de la señal recibida. Como para el caso bajo análisis la única componente en frecuencia es f_0 se puede decir:

$$\sum_{k=0}^{N-1}|X[k]|^2 = |FFT(f_0)|^2 \qquad \textbf{5.10}$$

Con lo cual resulta:

$$A = \frac{2}{N}FFT(f_0) \qquad \textbf{5.11}$$

Que como puede verse es el estimador de máxima verosimilitud.

Este estimador es válido bajo la gran suposición de que la frecuencia de la señal sea una de las frecuencias representables en la FFT, es decir, como el eje de frecuencias se encuentra discretizado, solo ciertos valores de serán representados y es muy improbable que la frecuencia verdadera sea un de estos valores.

Dichos valores, tomando k= 0, 1, 2… (N/2)-1, serán:

$$f = k\frac{f_m}{N} \qquad \textbf{5.12}$$

Siendo f_m la frecuencia de muestreo y N el número total de muestras usados para calcular la FFT.

Si la señal recibida tiene un valor de frecuencia tal que cae entre medio de dos de los valores representables (a la separación entre dos valores representables se la llama bin), el espectro en frecuencia de la señal ya no es un único valor centrado en f_0, sino que la potencia se esparce en las inmediaciones de f_0. Cabe destacar que el peor caso es cuando la señal cae justo en el medio de un bin. En dicho caso resulta:

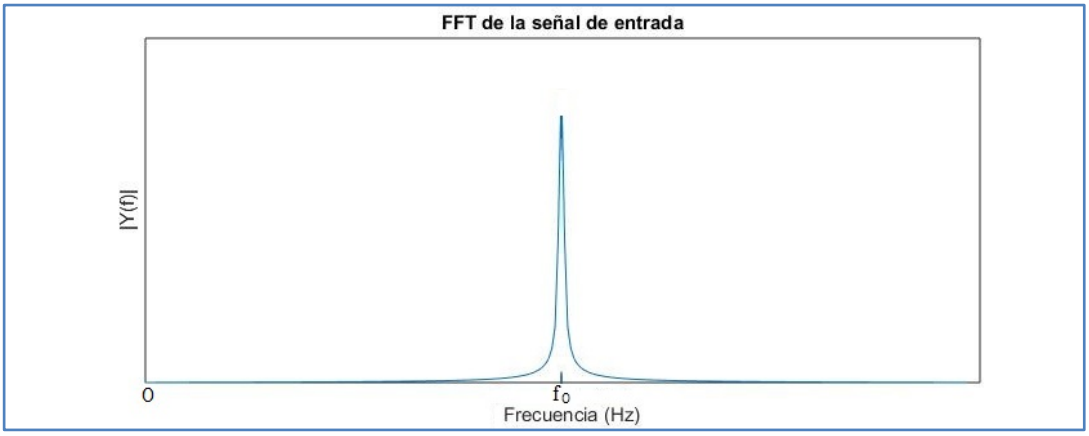

Imagen 5.3: FFT de un seno, en el cual su frecuencia cae en medio de un bin

Por lo cual se plantea una mejora para el estimador, basada en el espectro resultante de la señal.

La mejora que se propone es, una vez encontrada la frecuencia a la cual se da el máximo, realizar la estimación con su valor y los valores más próximos a ambos lados, con el fin de contemplar todas las componentes de un posible seno "esparcido".

Llamando Sxx[k] al modulo cuadrado de la FFT y k_0 al valor de la FFT que corresponde a f_0 en el espectro discreto, se llega a la siguiente expresión para la estimación de la amplitud.

$$\hat{A} = \frac{2}{N}\sqrt{Sxx[k_0 - 1] + Sxx[k_0] + Sxx[k_0 + 1]} \qquad \textbf{5.13}$$

Este estimador se puede mejorar si se suman más términos a cada lado del valor en f_0. Resultando

$$\hat{A} = \frac{2}{N}\sqrt{\sum_{i=-K}^{K} Sxx[k_0 + i]} \qquad \textbf{5.14}$$

Para poder observar esta mejora se planteó una simulación en la cual se calcula el error cometido en la estimación en función de la cantidad de términos que se sumó. Para eso se utilizó un seno sin ruido cuya frecuencia estuviese en medio de un bin. El código de la simulación se encuentra en [Anexo 2].

Como puede apreciarse en la imagen 5.2, para cuando se suman diez muestras de cada lado del máximo, se comete un error de aproximadamente un %1, lo cual consideramos aceptable.

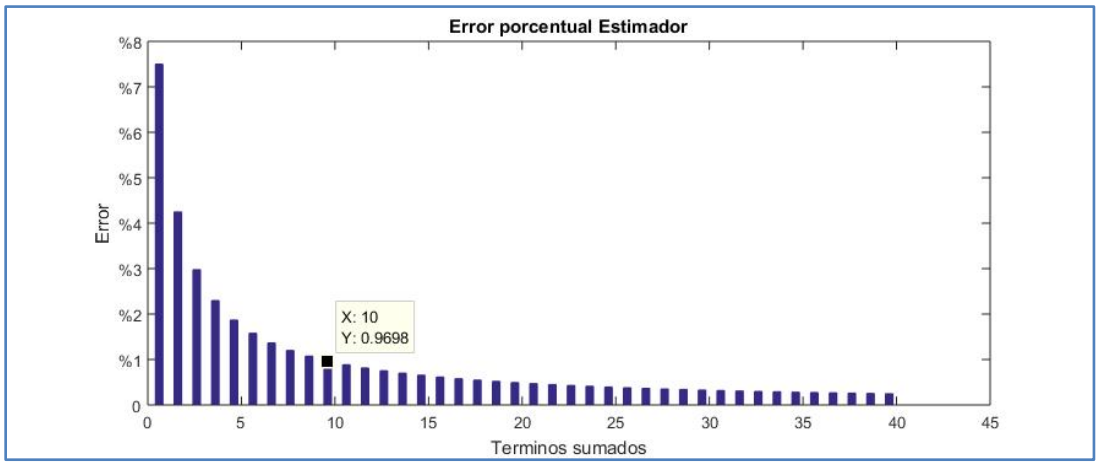

Imagen 5.4: Error porcentual en función de la cantidad de términos del estimador

Una vez especificado esto, se procede a analizar el error cometido por el estimador cuando la frecuencia cae en distintos valores dentro del bin. El código de la simulación se encuentra en [Anexo 3].

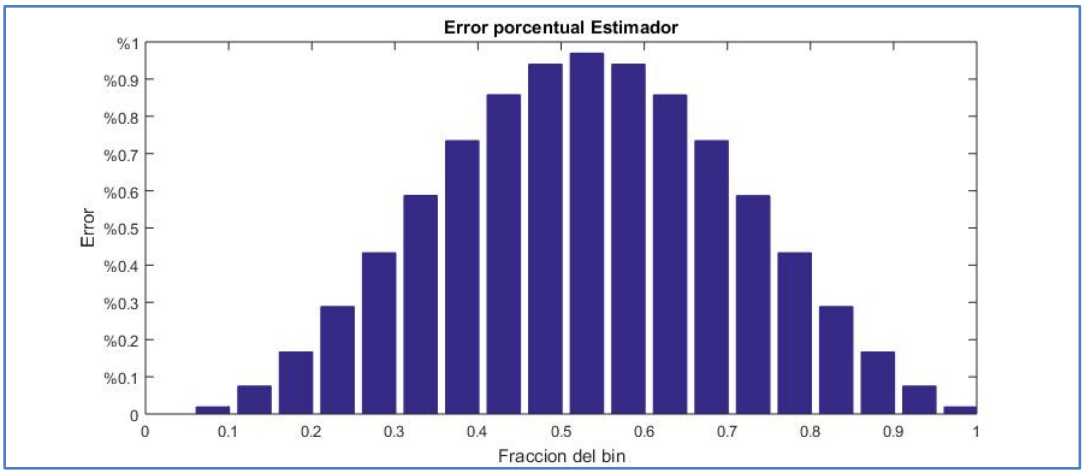

Imagen 5.5: Error porcentual en función de la ubicación de la frecuencia dentro del bin

Como se detalló anteriormente, el peor caso se da cuando la frecuencia recibida cae dentro de la mitad del bin.

Si a ahora se añade ruido a la señal simulada, se puede estudiar como esto afecta el error cometido por el estimador realizando un histograma de las estimaciones para distintas relaciones señal a ruido y siempre considerando una señal cuya frecuencia cae en el centro del bin.

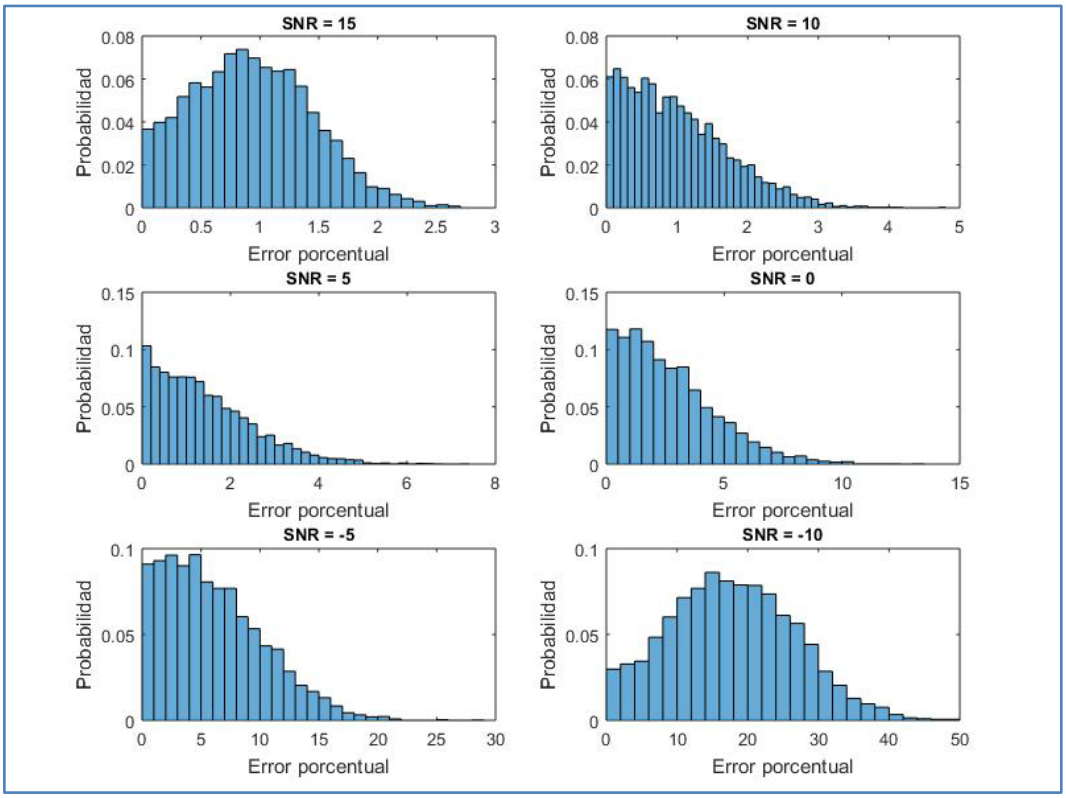

Imagen 5.6: Probabilidad para los errores porcentuales en función de la SNR de entrada

Como puede verse la probabilidad de cometer un error grande aumenta considerablemente cuando baja la relación señal a ruido hasta llegar a una media de 18% para una SNR de -10dB. Si bien es un factor a tener en cuenta, esto no es de gran importancia para este caso debido a que el receptor está pensado para ser utilizado con relaciones señal a ruido favorables ya que las medidas que realizará serán cercanas al transmisor.

6.Lazo de enganche de fase

Un lazo de enganche de fase o PLL (por sus siglas en ingles, Phase locked loop) es un sistema no lineal que permite estimar la fase de una señal inmersa en ruido.

Los usos más relevantes del mismo son:

- Demodulación de señales de FM y PM
- Filtros de seguimiento
- Síntesis de frecuencia

Cabe destacar que el análisis de este elemento será realizado en primera instancia mediante un modelo continuo y luego se discretizará dicho modelo para que pueda ser aplicado al receptor que es implementado digitalmente.

La modulación que usa el sistema DCS es BPSK coherente. Lo cual quiere decir que el receptor utiliza el conocimiento de la fase de la portadora para la demodulación de la señal.

Este lazo además de demodular la señal, sigue potenciales variaciones de frecuencia.

El lazo de enganche de fase se encuentra conformado por tres partes: el detector de fase, el filtro de lazo y el VCO (Oscilador controlado por tensión). A continuación se muestra un esquema simplificado del mismo:

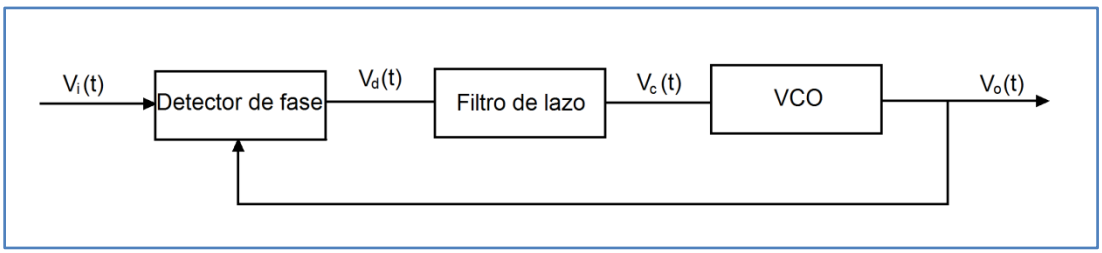

Imagen 6.1: Esquema del PLL

6.1. Componentes

6.1.1. Detector de fase

El detector de fase es un circuito que compara la fase de dos señales, una de entrada y otra generada internamente siendo esta la salida del VCO. La salida del detector de fase es una tensión continua cuya amplitud y signo dependen del módulo y sentido de la diferencia de fase entre las señales.

Los tipos de detectores de fase se pueden dividir en dos grandes grupos, los circuitos multiplicadores y los circuitos secuenciales.

Los circuitos multiplicadores trabajan, calculando el producto entre la señal de entrada y la señal generada por el VCO y a la señal obtenida se le calcula el valor medio para obtener la tensión de error.

Los circuitos secuenciales son dispositivos que poseen memoria y operan con los cruces por cero de las señales de entrada y de referencia ignorando otras características de la forma de onda de las señales.

6.1.2. Filtro de lazo

Como su nombre lo indica, es un filtro, por lo cual para su diseño se implementan técnicas normales del diseño frecuencial, como son el análisis cero-polar y su transferencia con la transformada de Laplace.

El filtro de lazo es un circuito muy importante en el lazo de enganche de fase, ya que el mismo determina las posibles respuestas que el lazo tenga, frente a distintos tipos de diferencias de fase de entrada y también asegura la estabilidad del sistema

6.1.3. VCO

El oscilador controlado por tensión, o VCO por sus siglas en ingles (Voltage controlled oscillator), es un circuito que en su salida presenta una señal cuya frecuencia es proporcional a su tensión de entrada. Típicamente generan una señal sinusoidal de salida.

Las características más importantes del VCO son su frecuencia de oscilación natural, que es la frecuencia a la cual oscila cuando su tensión de entrada es nula y su constante de oscilación que es una relación entre la tensión que se le aplica a la entrada y la variación de frecuencia que se obtiene.

6.2. Análisis lineal de comportamiento

Se comienza analizando el comportamiento del detector de fase suponiendo que el mismo es un multiplicador.

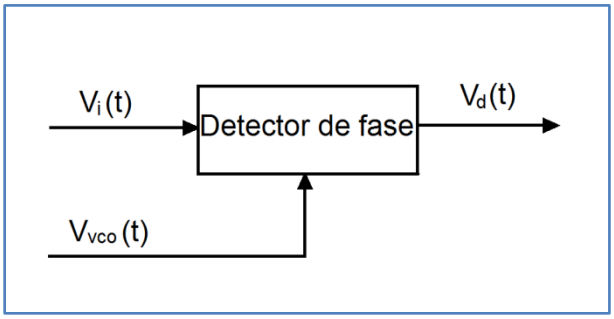

Imagen 6.2: Esquema del detector de fase

Como puede verse las señales que entran al detector son la señal de entrada al lazo y la señal de referencia generado por el VCO.

$$V_i(t) = V_i sen(\omega_i t + \theta_i) \quad \textbf{6.1}$$

$$V_{vco}(t) = V_{vco}\, cos(\omega_{vco}\, t + \theta_{vco}) \quad \textbf{6.2}$$

Como el detector es un multiplicador, a la salida se obtiene:

$$V_d(t) = V_i(t).V_{vco}(t) \quad \textbf{6.3}$$

$$V_d(t) = V_i.V_{vco}.sen(\omega_i t + \theta_i).cos(\omega_{vco} t + \theta_{vco}) \quad \textbf{6.4}$$

Reescribiendo:

$$V_d(t) = \frac{1}{2}.V_i.V_{vco}.\left[sen\big((\omega_i - \omega_{vco})t + (\theta_i - \theta_{vco})\big).sen\big((\omega_i + \omega_{vco})t + (\theta_i + \theta_{vco})\big)\right] \quad \textbf{6.5}$$

Pero además se sabe que cuando el dispositivo está enganchado, es decir, que se encuentra siguiendo la señal de entrada manteniendo una diferencia de fase constante, se tiene $\omega_i = \omega_{vco}$. Por lo cual resulta

$$V_d(t) = \frac{1}{2}.V_i.V_{vco}.[sen(\theta_i - \theta_{vco}).sen(2\omega_{vco} + \theta_i + \theta_{vco})] \quad \textbf{6.6}$$

Como puede apreciarse se obtienen dos componentes, una dependiendo de la diferencia de fase de las señales de entrada y otra del doble de la frecuencia, la cual debe ser filtrada, para obtener:

$$V_d(t) = \frac{1}{2}.V_i.V_{vco}.sen(\theta_i - \theta_{vco}) \quad \textbf{6.7}$$

Esta expresión se simplifica considerando $K_{DF} = {}^1\!/_2.V_i.V_{vco}$ y como para diferencias de fase pequeñas se puede aproximar $sen(\theta_i - \theta_{vco}) = \theta_i - \theta_{vco}$ queda:

$$V_d(t) = K_{DF}.(\theta_i - \theta_{vco}) \quad \textbf{6.8}$$

Como puede verse, la tensión de salida del detector de fase solo tiene en consideración las fases de las señales de entrada, por lo cual en análisis posteriores solo se utilizara la fase y no las señales completas.

Es necesario remarcar que K_{DF} depende de los niveles de V_i y de V_{vco} por lo cual se debe normalizar para desafectar las variaciones que pueda presentar la entrada V_i, que podría ocasionar errores de frecuencia en el VCO.

Como logra apreciarse en el esquema simplificado del PLL (imagen 6.1), la tensión de error de fase (salida del detector de fase) entra al filtro de lazo, el cual entrega a su salida la tensión de control que será la que ingresa al VCO.

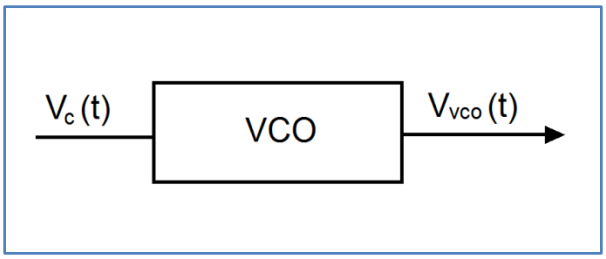

Imagen 6.3: Esquema VCO

Como puede verse al VCO ingresa la señal $V_c(t)$ y sale $V_{vco}(t)$, donde se sabe que:

$$V_{vco}(t) = V_{vco}\, cos(\omega_{vco}\, t + \theta_{vco}) \qquad \textbf{6.9}$$

Pero a su vez:

$$\omega_{vco} = \omega_o + K_{vco}\, V_c(t) \qquad \textbf{6.10}$$

Siendo ω_o la frecuencia de oscilación natural del VCO y K_{vco} la constante del VCO.

Esta es la principal característica de un VCO, que la frecuencia de su señal de salida dependa de la tensión de entrada.

Reacomodando esta expresión se llega a:

$$V_{vco}(t) = V_{vco}\, cos\big((\omega_o + \omega_{vco} - \omega_o)t + \theta_{vco}\big) \qquad \textbf{6.11}$$

Si se denomina $\theta_o(t) = (\omega_{vco} - \omega_o)t + \theta_{vco}$

$$V_{vco}(t) = V_{vco}\cos\left(\omega_o t + \theta_o(t)\right) \qquad \textbf{6.12}$$

Donde

$$\frac{d\theta_o(t)}{dt} = \omega_{vco} - \omega_o = K_{vco}V_c(t) \quad \Longrightarrow \quad \theta_o(t) = K_{vco}\int_{-\infty}^{t} V_c(t) \qquad \textbf{6.13}$$

Como solo se consideran las fases de las señales que entran al detector de fase puede verse que la salida del VCO, $\theta_o(t)$, es la integral de $V_c(t)$.

Lo último que se debe analizar es el filtro de lazo, el cual mediante su función de transferencia determina la salida mediante una convolución de la siguiente forma:

$$V_c(t) = V_d(t) * f(t) \qquad \textbf{6.14}$$

Sintetizando, las ecuaciones que describen el comportamiento del PLL son.

$$\theta_o(t) = K_{vco}\int_{-\infty}^{t} V_c(t) \qquad V_d(t) = K_{DF}.(\theta_i - \theta_o) \qquad V_c(t) = V_d(t) * f(t) \qquad \textbf{6.15}$$

Pasando ahora al análisis mediante la transformada de Laplace el esquema queda:

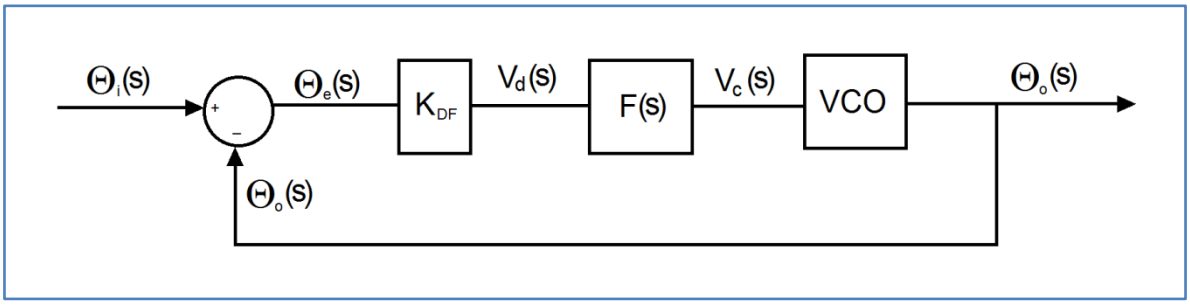

Imagen 6.4: PLL en el plano de Laplace

Siendo K_{DF} la constante del detector de fase, F(s) la transferencia del filtro de lazo y $\Theta_e(s)$ la diferencia entre las fases ($\Theta_e(s) = \Theta_i(s) - \Theta_o(s)$).

Las ecuaciones luego de aplicar la transformada de Laplace quedan:

$$\Theta_o(s) = \frac{K_{vco}}{s} V_c(s) \qquad V_d(s) = K_{DF}.\left(\Theta_i(s) - \Theta_o(s)\right) \qquad F(s) = \frac{V_c(s)}{V_d(s)} \tag{6.16}$$

Con lo cual la transferencia del lazo resulta

$$T(s) = \frac{\Theta_o(s)}{\Theta_i(s)} = \frac{K_{DF}.K_{vco}.F(s)}{s + K_{DF}.K_{vco}.F(s)} \tag{6.17}$$

Esta transferencia es válida para cuando el error de fase sea muy chico, es decir cuando el PLL está enganchado.

Una vez caracterizado de forma genérica el PLL es necesario continuar con el diseño del mismo y especificar sus parámetros.

Como se vio en el capítulo de estimación de frecuencia, se tienen valores discretos para representar la frecuencia, esto quiere decir que la estimación puede pasar de un valor a otro en un instante dependiendo a cual este más próxima la frecuencia de la señal. Es decir que la estimación puede tener errores en escalón. Por este motivo es que se diseña el filtro de lazo de tal forma que el PLL presente error de estado estacionario nulo frente a escalones de frecuencia.

Para los valores de frecuencia de muestreo y número de muestras que fueron establecidos en el capítulo anterior el error de frecuencia posible es de 31,2Hz. Esta es la amplitud de los escalones en frecuencia que el sistema tiene que poder seguir y continuar enganchado.

Para evaluar el error de estado estacionario se utiliza el teorema del valor final que para el sistema que se analiza queda de la forma:

$$E_{ss} = \lim_{s \to 0} \frac{s.\Theta_i(s)}{1 + K_{DF}K_{vco}\dfrac{F(s)}{s}} \tag{6.18}$$

Como el sistema se analiza para las fases de las señales, un escalón en frecuencia es equivalente a una rampa de fase. Por lo tanto $\Theta_i(s)$ será:

$$\Theta_i(s) = \frac{\Delta\omega}{s^2} \tag{6.19}$$

Siendo $\Delta\omega$ la amplitud del escalón de frecuencia pasado a radianes por segundo.

Reemplazando y simplificando:

$$E_{ss} = \lim_{s\to 0} \frac{\frac{\Delta\omega}{s}}{1 + K_{DF}K_{vco}\frac{F(s)}{s}} \tag{6.20}$$

$$E_{ss} = \lim_{s\to 0} \frac{\Delta\omega}{s + K_{DF}K_{vco}F(s)} \tag{6.21}$$

Este valor será nulo cuando $F(s)$ tenga un polo en el origen. Por ende quedaría un filtro con la siguiente expresión:

$$F(s) = \frac{1}{s} \tag{6.22}$$

Analizando la estabilidad mediante el diagrama del lugar de raíces, se obtiene.

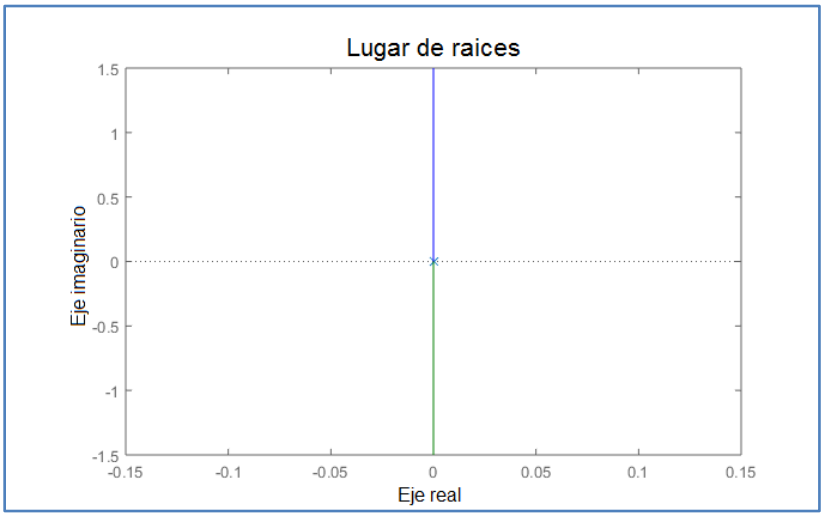

Imagen 6.5: Lugar de raíces cuando F(S) tiene solo un polo en el origen (inestable)

Como se ve, el lugar de raíces se encuentra sobre el eje imaginario. Esto no es para nada aconsejable, debido a que cualquier mínima perturbación podría correr los polos al semiplano derecho e desestabilizar el sistema.

Para mejorar esto es necesario agregar un cero en el semiplano izquierdo con el fin de que las posibles perturbaciones generen desplazamientos en el semiplano izquierdo y no desestabilicen el sistema.

Por lo cual se propone implementar un filtro de la forma.

$$F(s) = \frac{1 + \frac{s}{a}}{s} \qquad \textbf{6.23}$$

Analizando la estabilidad mediante el diagrama del lugar de raíces se obtiene.

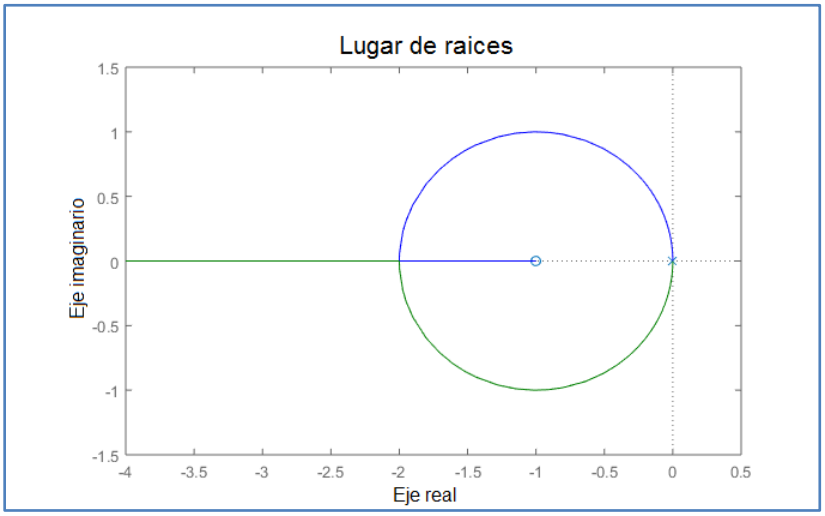

Imagen 6.6: Lugar de raíces cuando F(S) tiene un cero en a y un polo en el origen

Como puede apreciarse el lugar de raíces se encuentra en su totalidad en el semiplano izquierdo lo cual asegura la estabilidad del sistema.

A continuación se procede a analizar el comportamiento de la transferencia resultante.

Si se reemplaza el filtro propuesto en la transferencia antes calculada resulta.

$$T(s) = \frac{K_{DF}.K_{vco}.\frac{1+\frac{s}{a}}{s}}{s + K_{DF}.K_{vco}.\frac{1+\frac{s}{a}}{s}} \quad \textbf{6.24}$$

$$T(s) = \frac{K_{DF}.K_{vco}.1+\frac{s}{a}}{s^2 + K_{DF}.K_{vco}.1+\frac{s}{a}} \quad \textbf{6.25}$$

$$T(s) = \frac{K_{DF}.K_{vco} + \frac{K_{DF}.K_{vco}.s}{a}}{s^2 + K_{DF}.K_{vco} + \frac{K_{DF}.K_{vco}.s}{a}} \quad \textbf{6.26}$$

Como puede apreciarse la respuesta del sistema corresponde a una respuesta del tipo sub-amortiguada, es decir de la forma:

$$H(s) = \frac{2\xi\omega_n s + \omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2} \quad \textbf{6.27}$$

Donde

$$K_{DF}.K_{vco} = \omega_n^2 \qquad a = \frac{\omega_n}{2\xi} \quad \textbf{6.28 y 6.29}$$

Por lo cual es necesario determinar los valores de ω_n y ξ. Para analizar esto se utilizan los parámetros del ancho de banda equivalente de ruido (B_n), ancho de banda de 3dB (B_{3dB}) y tiempo de establecimiento (t_e), siendo este ultimo el tiempo que transcurre desde que la salida del sistema comienza evolucionar hasta que la respuesta se estabiliza en torno al %2 del valor de régimen estacionario. Los parámetros se encuentran determinados en función de ω_n y ξ, de la siguiente forma.

$$B_n = \frac{\omega_n}{2}\left(\xi + \frac{1}{4\xi}\right) \quad \textbf{6.30}$$

$$B_{3dB} = B_n\sqrt{1 + 2\xi^2 + \sqrt{(1 + 2\xi^2)^2 + 1}} \quad \textbf{6.31}$$

$$t_e = \frac{4}{\xi\omega_n} \quad \textbf{6.32}$$

Puede verse que existe una relación de compromiso entre estos parámetros en función de ξ, es decir, para valores grandes de ξ se logran tiempos de establecimiento reducidos, pero esto aumenta el ancho de banda de ruido, a su vez, reducir el ancho de banda equivalente de ruido resulta en un menor margen de enganche para el PLL. Si se estudia esto se puede llegar a la conclusión de que el valor usual de ξ que resuelve este compromiso es $\xi = \frac{\sqrt{2}}{2}$.

Una vez establecido ξ se debe elegir el valor de ω_n. Una limitación para elegir de ω_n es que solo se disponen de 160ms para lograr el enganche de fase. Esto quiere decir:

$$t_e < 160ms \qquad \text{quedando} \qquad \omega_n > \frac{4}{\xi.160ms} = 35.4\frac{rad}{s}$$

Para tener una visión un poco más clara de esto, se calculan varios valores.

$\omega_n\,[^{rad}/_s]$	t_e [s]
40	141,4
60	94,3
80	70,7
100	56,6
120	47,1
160	35,4
200	28,3

Tabla 6.1: te en función de ωn

En la práctica se suele pedir que luego de alcanzado el punto de establecimiento, este se mantenga por una cierta cantidad de tiempo como verificación. Un tiempo acorde para esta verificación podría ser un 25% del tiempo máximo para la sincronización, es decir, 40 ms. Lo cual resultaría en un tiempo de establecimiento de 120ms.

Otro factor que hay que considerar es el hecho de que probablemente no se detecte la señal exactamente en su comienzo por lo cual es necesario tener un margen de tiempo. Se va a optar por un margen grande ya que el mismo debe compensar el efecto generado cuando la modulación en fase comienza descripto a continuación. Si se elige $\omega_n = 125{,}7\frac{rad}{s}$ se obtiene

$$t_e = 45ms \qquad B_n = 66{,}7\frac{rad}{s} \qquad B_{3dB} = 137{,}2\frac{rad}{s}$$

Lo cual deja un margen de 75ms.

Cuando la modulación comienza, si bien la potencia de la señal no disminuye, lo que si disminuye es la potencia utilizada en transmitir la portadora, en un 55%. Este efecto se considera mediante una disminución en la constante del detector en la misma magnitud. Por lo cual calculando nuevamente considerando esto se obtiene:

$$\omega_n = 84{,}3\frac{rad}{s} \qquad B_n = 44{,}7\frac{rad}{s} \qquad B_{3dB} = 92\frac{rad}{s}$$

Como puede apreciarse, el efecto que se genera cuando comienza la modulación es que tanto el ancho de banda equivalente de ruido como el ancho de banda de 3dB, se reducen. Este efecto es conveniente ya que significa que una vez que el PLL se engancha, su ancho de banda disminuye lo cual lleva a que el sistema sea más inmune a perturbaciones y a los cambios de fase generados por la modulación.

6.3. Implementación digital del PLL

En este punto es necesario aclarar que aunque todo el marco teórico desarrollado sobre PLL en los capítulos anteriores es completamente válido, no es prácticamente aplicable al receptor, ya que el mismo se encuentra implementado de forma digital.

Por lo cual es necesario pasar todo el desarrollo del PLL al dominio digital. Para lograr esto, es necesario discretizar la función de transferencia del filtro de lazo e implementar el VCO como un NCO (Numerically controlled oscillator).

6.3.1. Filtro de lazo digital

Para la discretización de la función de transferencia del filtro de lazo se utiliza la transformación bilineal. La cual tiene la siguiente expresión:

$$S = \frac{2}{T_m}\frac{(Z-1)}{(Z+1)} \tag{6.33}$$

Donde T_m es el período de muestreo y Z la nueva variable en el dominio frecuencial.

Reemplazando en la transferencia del filtro de lazo

$$F(Z) = \frac{1 + \frac{\frac{2}{T_m}\frac{(Z-1)}{(Z+1)}}{a}}{\frac{2}{T_m}\frac{(Z-1)}{(Z+1)}} \tag{6.34}$$

Simplificando:

$$F(Z) = \frac{aT_m + 2 + Z^{-1}(aT_m - 2)}{2a - Z^{-1}2a} \tag{6.35}$$

Llamando $X(Z)$ a la transformada de la señal de entrada al filtro e $Y(Z)$ a la transformada de la señal de salida.

$$F(Z) = \frac{Y(Z)}{X(Z)} \qquad \textbf{6.36}$$

Esto permite reescribir la transferencia como una ecuación entre la entrada y la salida para posteriormente poder hallar la ecuación en diferencias.

$$2aY(Z) - 2aY(Z)Z^{-1} = (aT_m + 2)X(Z) + (aT_m - 2)X(Z)Z^{-1} \qquad \textbf{6.37}$$

Finalmente anti-transformado esta ecuación se llega a la ecuación en diferencias que se usa para implementar el filtro de forma digital:

$$2a.y[n] - 2a.y[n-1] = (aT_m + 2)x[n] + (aT_m - 2)x[n-1] \qquad \textbf{6.38}$$

$$y[n] = \frac{(aT_m + 2)}{2a}x[n] + \frac{(aT_m - 2)}{2a}x[n-1] + y[n-1] \qquad \textbf{6.39}$$

Expresado esquemáticamente resulta.

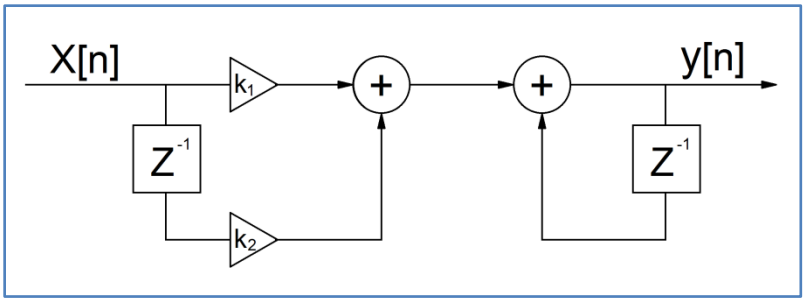

Imagen 6.7: Esquema del Filtro de lazo

Como puede verse para esta implementación solo se necesitan dos ganancias, dos retardos y dos sumadores.

Las ganancias k_1 y k_2 son:

$$k_1 = \frac{(aT_m + 2)}{2a} \qquad k_2 = \frac{(aT_m - 2)}{2a} \qquad \textbf{6.40 y 6.41}$$

6.3.2. NCO

Para desarrollar el comportamiento del NCO se parte de la transferencia y la respuesta del VCO.

Como se vio en el análisis lineal del PLL se tiene que la frecuencia de la señal generada por el VCO depende de la tensión de entrada, por lo cual se obtiene una señal de la forma.

$$V_{vco}(t) = sen\left(2\pi f_{vco}\, t + K_{vco} \int_{-\infty}^{t} V_c(t)dt\right) \qquad \textbf{6.41}$$

Para digitalizar el VCO primero se procede por discretizar la fase de dicha señal de salida, haciendo $t = nT_m$.

$$\vartheta(t) = 2\pi f_{vco}\, t + K_{vco} \int_{-\infty}^{t} V_c(t)dt \qquad \textbf{6.42}$$

$$\vartheta[n] = 2\pi f_{nco}\, nT_m + K_{nco} \sum_{k=-\infty}^{(n-1)} V_c[kT_m]\, T_m \qquad \textbf{6.43}$$

Siendo f_{nco} la frecuencia de oscilación natural del NCO, T_m el periodo de muestreo y K_{nco} la constante del NCO.

Según esta discretización se necesitarían infinitas muestras lo cual es imposible, por ello, se plantea implementar el cálculo de forma recursiva.

Primero se calcula $\vartheta[n-1]$

$$\vartheta[n-1] = 2\pi f_{nco}\,(n-1)T_m + K_{nco}\sum_{k=-\infty}^{n-2} V_c[kT_m]\,T_m \qquad \textbf{6.44}$$

Luego se reescribe $\vartheta[n]$, sumando fuera de la sumatoria el ultimo termino de la misma.

$$\vartheta[n] = 2\pi f_{nco}\,nT_m + K_{nco}\,V_c[(n-1)T_m]T_m + K_{nco}\sum_{k=-\infty}^{(n-2)} V_c[kT_m]\,T_m \qquad \textbf{6.45}$$

Finalmente restando $\vartheta[n] - \vartheta[n-1]$ se obtiene:

$$\vartheta[n] - \vartheta[n-1] = 2\pi f_{nco}\,(n+1-n)T_m + K_{nco}\,V_c[(n-1)T_m]T_m \qquad \textbf{6.46}$$

$$\vartheta[n] - \vartheta[n-1] = 2\pi f_{nco}\,T_m + K_{nco}\,V_c[(n-1)T_m]T_m \qquad \textbf{6.47}$$

Resultando:

$$\vartheta[n] = \vartheta[n-1] + (2\pi f_{nco} + K_{nco}\,V_c[(n-1)T_m])T_m \qquad \textbf{6.48}$$

Como puede apreciarse, se llega a una ecuación recursiva para calcular la fase del NCO. A continuación con esta fase se debe calcular el valor del seno de la misma y esta será la señal de salida del NCO.

La señal generada a la salida del NCO es entonces:

$$V_{vco}[n] = A.sen(\vartheta[n]) \qquad \textbf{6.49}$$

Es necesario remarcar que la amplitud de la señal generada es unitaria ya que las señales que entran al detector de fase deben estar normalizadas.

Representando esquemáticamente:

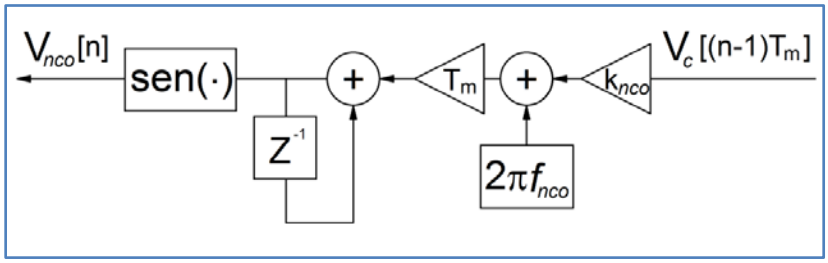

Imagen 6.8: Esquema NCO

6.3.3. Implementación y pruebas

Para resumir, lo que se debe implementar es el siguiente set de ecuaciones:

F.L.: $V_c[n] = V_c[n-1] + k_1 x[n] + k_2 x[n-1]$

NCO: $\vartheta[n] = \vartheta[n-1] + (2\pi f_{nco} + K_{nco} V_c[(n-1)T_m])T_m$

$V_{vco}[n] = A.sen(\vartheta[n])$

D.F: $V_d[n] = x[n].V_{vco}[n]$

El esquema completo de la implementación digital resulta.

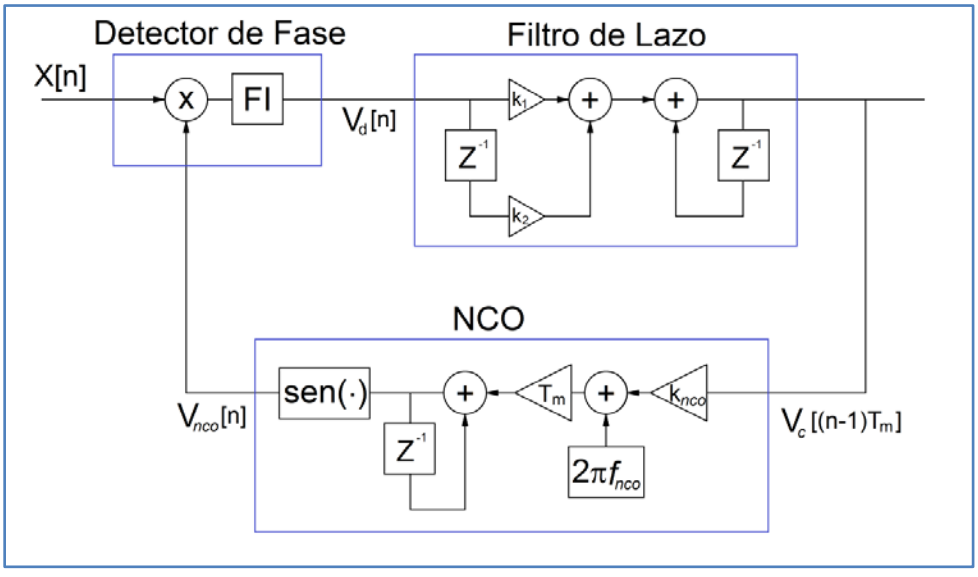

Imagen 6.9: Esquema completo del PLL

Donde las constantes valen:

$$k_1 = \frac{(aT_m+2)}{2a} = 0.01125 \qquad k_2 = \frac{(aT_m-2)}{2a} = -0.01125$$

$$K_{DF} = {}^{1}/_{2}.V_i.V_{vco} = 0.5 \qquad K_{nco} = \frac{\omega_n^2}{K_{DF}} = 31601 \qquad T_m = 1/256000$$

Finalmente se simula para poder ver cómo se comporta el error de fase. Es necesario aclarar que solo se simula el bloque del PLL sin las etapas de detección y estimación y además que la señal de error de fase es filtrada por el filtro adaptado, el cual se verá en el próximo capítulo. La simulación consta de aplicar un escalón de frecuencia de amplitud 10Hz y observar la salida del PLL, en ausencia de ruido. El código utilizado para estas simulaciones se encuentra en el [Anexo 4]

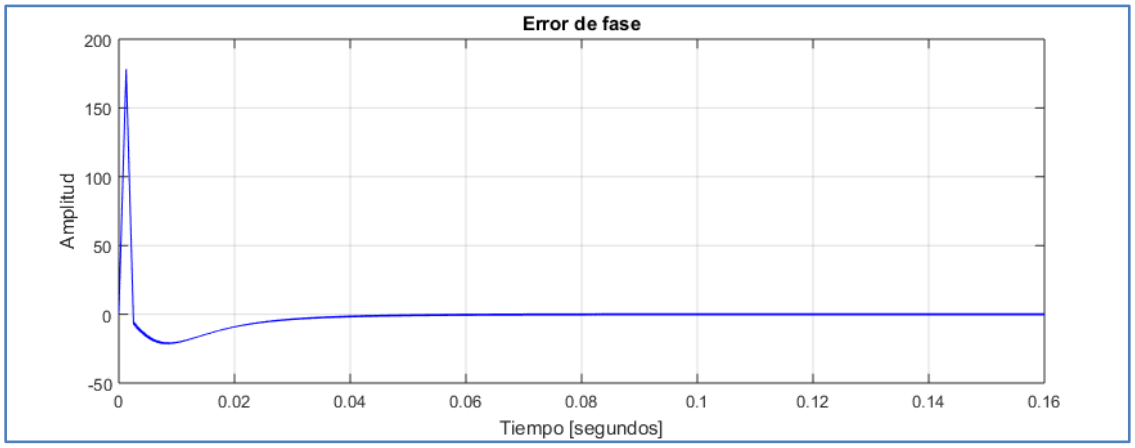

Imagen 6.10: Error de fase sin ruido

Como puede observarse el PLL intenta llevar rápidamente el error de fase a cero y el tiempo de establecimiento es aproximadamente el calculado de 40ms.

Si ahora se le suma ruido a la señal de entrada, se puede estudiar cómo se comporta el error de fase para distintas relaciones señal a ruido.

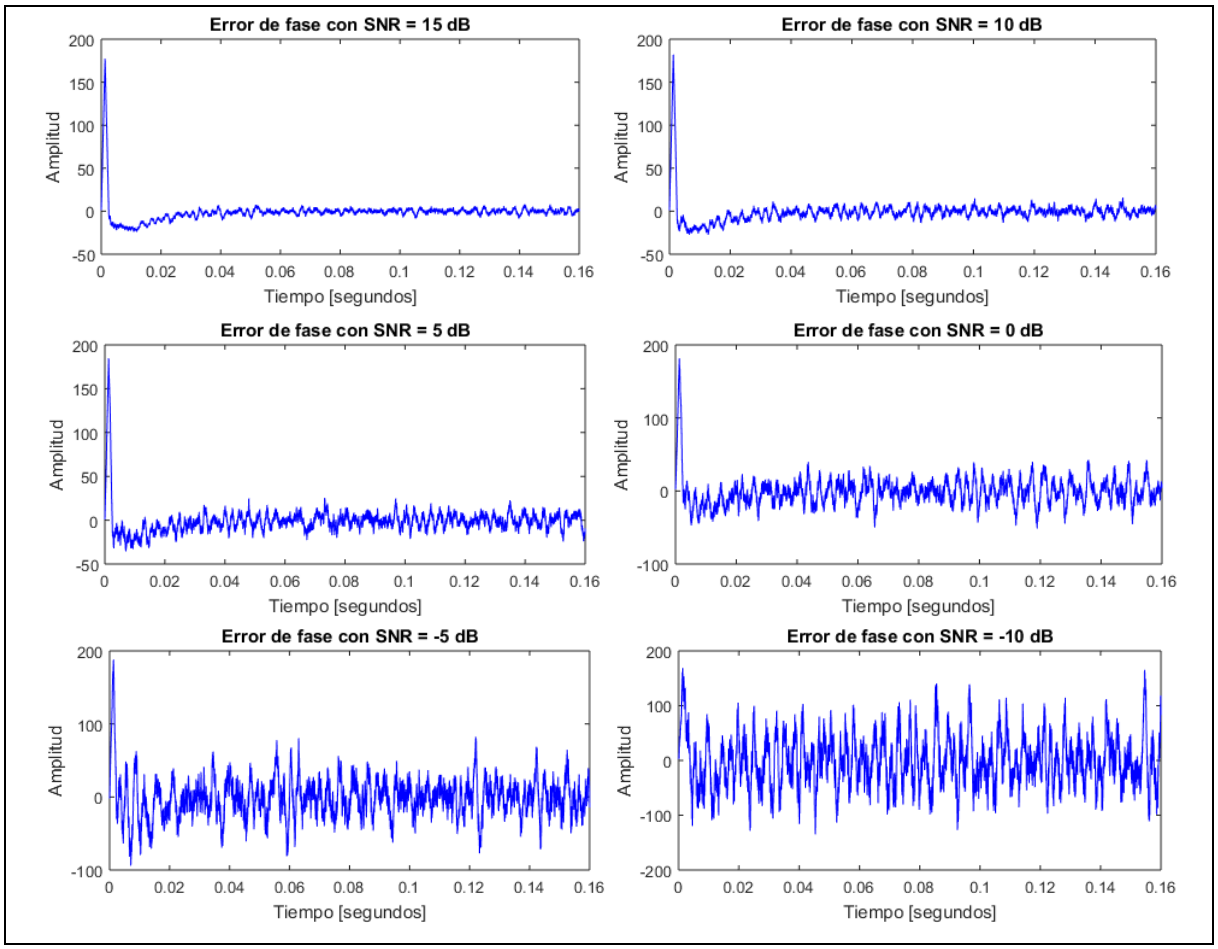

Imagen 6.11: Error de fase para distintas SNR

Como puede observarse que para el caso de bajas SNR ya no se puede decir que el valor del error de fase tiende a cero, lo que sí se puede apreciar es que el error de fase queda acotado en una banda de valores centrados en cero, mostrando que el PLL sigue enganchado. Por lo cual se plantea que para detectar, cuando el PLL está enganchado, se debe ver si el error de fase queda acotado por un umbral.

Como se espera que el receptor tenga relaciones señal a ruido de entrada bastante favorables, se fija el umbral en 150, que es el umbral que acota el error de fase cuando la relación señal a ruido es de -5dB. Para evitar posibles errores, se va comparar el módulo del error de fase con el umbral durante un lapso de 40 ms antes de decir si el PLL se encuentra enganchado.

Luego, se simula cómo se comporta el PLL en conjunto con las etapas desarrollados anteriormente, es decir, los bloques de detección, estimación de amplitud y estimación de frecuencia. Se debe aclarar que tanto para esta simulación como para las anteriores la fase inicial de la señal recibida es la misma.

De la simulación se obtiene:

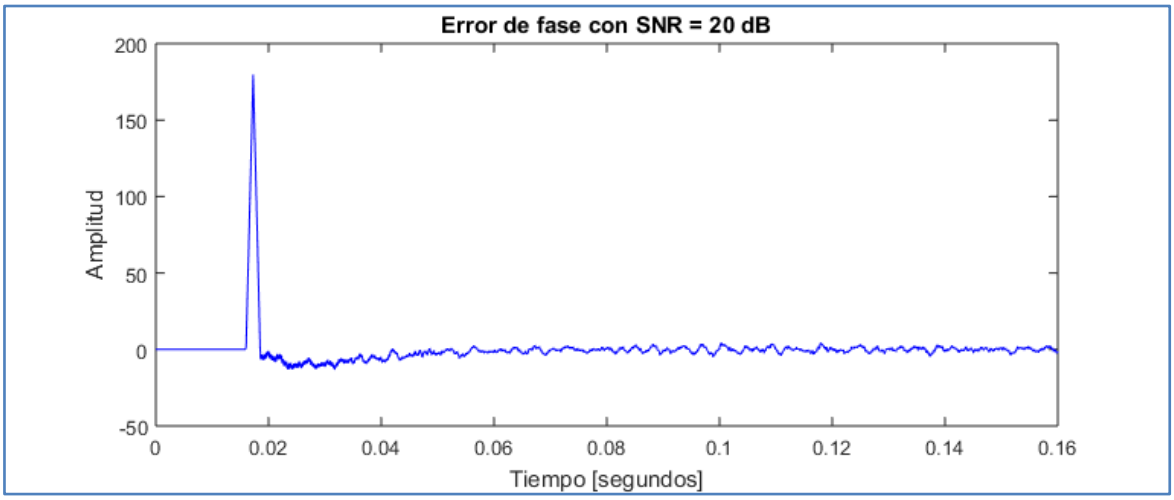

Imagen 6.12: Simulación del PLL integrando etapas de detección y estimación

En el gráfico se puede notar que existe un set de muestras el cual el PLL no procesa. Esas muestras son las que el receptor usa en la primera detección las cuales no pasan a los siguientes algoritmos de estimación y PLL debido a que es necesaria una segunda detección, como se explico en el capítulo de “Detección”.

Es de remarcar que todas las simulaciones fueron realizadas para la misma señal, un escalón de frecuencia de amplitud 10Hz, cambiando únicamente el nivel de ruido de las mismas.

El código implementado puede encontrarse en el [Anexo 5].

7.Filtro adaptado

Luego de la demodulación realizada por el PLL, lo que se tiene es una señal en banda base digital, es decir, una secuencia de símbolos que contienen el mensaje. A su vez esta secuencia se encuentra inmersa en ruido que se modela como aditivo, blanco y gaussiano (AWGN). Para poder decodificar este mensaje, el receptor en algún punto durante la duración de cada símbolo debe tomar la decisión de si lo que se recibió es un uno o un cero.

7.1. Desarrollo teórico

Como la secuencia se encuentra inmersa en ruido se plantea un test de hipótesis, donde las hipótesis bajo análisis son:

$$H_0: r[n] = S_0[n] + w[n] \quad \textbf{7.1}$$

$$H_1: r[n] = S_1[n] + w[n] \quad \textbf{7.2}$$

Siendo $S_0[n]$ y $S_1[n]$ las muestras correspondientes a los símbolos que representan 0 y 1 respectivamente.

Como el ruido tiene distribución normal con media nula y varianza σ^2, y a su vez las señales S_0 y S_1 son deterministas, la distribución para cada hipótesis queda:

$$fdp(r[n]; H_0) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[n]-S_0[n])^2}{2\sigma^2}} \quad \textbf{7.3}$$

$$fdp(r[n]; H_1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[n]-S_1[n])^2}{2\sigma^2}} \quad \textbf{7.4}$$

Pero esta distribución es para cada muestra en particular, y lo que se busca es decidir sobre el símbolo completo. Por lo cual se plantea la distribución conjunta de todas las muestras que forman el símbolo.

Dado que las muestras de ruido se modelan independientes entre sí, se obtiene:

$$fdp(r[0], r[1], r[2] \dots r[N-1]; H_0) = \prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[i]-S_0[i])^2}{2\sigma^2}} \quad \textbf{7.5}$$

$$fdp(r[0], r[1], r[2] \dots r[N-1]; H_1) = \prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[i]-S_1[i])^2}{2\sigma^2}} \quad \textbf{7.6}$$

El receptor debe tomar la decisión de que hipótesis es verdadera sobre la totalidad del símbolo, por lo cual se debe esperar a que lleguen todas las muestras que lo forman para tomar la decisión. Por este motivo se utilizara el criterio de máxima probabilidad a posteriori para analizar que símbolo se recibió.

$$P(R|H = H_0)P(H_0) \gtrless_{H_1}^{H_0} P(R|H = H_1)\, P(H_1) \quad \textbf{7.7}$$

Como se supone que la probabilidad de recibir S_0 o S_1 es la misma,

$$P(H_0) = P(H_1) = {}^1/_2 \quad \textbf{7.8}$$

Reemplazando esto y las distribuciones de probabilidad se obtiene:

$$\frac{1}{2}\prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[i]-S_0[i])^2}{2\sigma^2}} \quad \gtrless_{H_1}^{H_0} \quad \frac{1}{2}\prod_{i=0}^{N-1} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(r[i]-S_1[i])^2}{2\sigma^2}} \quad \textbf{7.9}$$

Simplificando:

$$\prod_{i=0}^{N-1} e^{-\frac{(r[i]-S_0[i])^2}{2\sigma^2}} \gtrless_{H_1}^{H_0} \prod_{i=0}^{N-1} e^{-\frac{(r[i]-S_1[i])^2}{2\sigma^2}} \quad \textbf{7.10}$$

Aplicando a ambos lados el logaritmo natural:

$$\sum_{i=0}^{N-1} -\frac{(r[i]-S_0[i])^2}{2\sigma^2} \gtrless_{H_1}^{H_0} \sum_{i=0}^{N-1} -\frac{(r[i]-S_1[i])^2}{2\sigma^2} \quad \textbf{7.11}$$

$$\sum_{i=0}^{N-1} (r[i]-S_1[i])^2 \gtrless_{H_1}^{H_0} \sum_{i=0}^{N-1} (r[i]-S_0[i])^2 \quad \textbf{7.12}$$

$$\sum_{i=0}^{N-1} (r[i]^2 - 2r[i]S_1[i] + S_1[i]^2) \gtrless_{H_1}^{H_0} \sum_{i=0}^{N-1} (r[i]^2 - 2r[i]S_0[i] + S_0[i]^2) \quad \textbf{7.13}$$

Si ahora se reagrupan los términos de manera más conveniente

$$\sum_{i=0}^{N-1} r[i](S_0[i]-S_1[i]) \gtrless_{H_1}^{H_0} \frac{\sum_{i=0}^{N-1} S_0[i]^2 - \sum_{i=0}^{N-1} S_1[i]^2}{2} \quad \textbf{7.14}$$

Si analizamos esta expresión puede verse que en el término de la izquierda se tiene la señal de entrada $r[i]$ sometida a un proceso y en el término de la derecha se tiene sumatorias que no dependen de los valores de la entrada por lo cual es equivalente a una constante. Es decir que se procesa la señal de entrada y se la compara con una constante, de valor K.

En este punto es necesario aclarar que existen varias formas de implementar este test de hipótesis utilizando un correlador, dos correladores, un filtro adaptado a la señal diferencia $(S_0 - S_1)$ u dos filtros

adaptados a cada una de las señales que representan los símbolos. Para este caso se utilizara un filtro adaptado a la señal diferencia $(S_0 - S_1)$.

La respuesta impulsional del filtro adaptado a la señal diferencia, es:

$$h_{FA}[n] = S_0[N-n] - S_1[N-n] \tag{7.15}$$

Es decir

Imagen 7.1: Respuesta del filtro adaptado

A su vez $h_{FA}[n]$ es una secuencia que puede expresarse:

$$h_{FA}[n] = 2A\sum_{k=0}^{\frac{N}{2}-1}\delta[n-k] - 2A\sum_{k=\frac{N}{2}}^{N-1}\delta[n-k] \tag{7.16}$$

Retomando la constante se puede calcular fácilmente su valor

$$K = \frac{\sum_{i=0}^{N-1} S_0[i]^2 - \sum_{i=0}^{N-1} S_1[i]^2}{2} \tag{7.17}$$

$$K = \frac{NA^2 - NA^2}{2} = 0 \tag{7.18}$$

Esta constante es el umbral de decisión óptimo con el cual se compara para tomar la decisión y como puede verse su valor es 0, esto quiere decir que las señales S_0 y S_1 son antipodales, es decir, que son opuestas. A este valor se lo llama umbral de decisión.

Con lo cual la etapa de decisión queda:

$$\sum_{i=0}^{N-1} r[i]h_{FA}[n-i] \gtrless_{H_1}^{H_0} K = 0 \qquad \textbf{7.19}$$

Siendo $n = kN$ para que la decisión sea óptima, donde k es un número natural que representa los símbolos transmitidos, es decir, k=1, 2, 3 … M, donde M es la cantidad de símbolos recibidos.

Como el tiempo de bit es de 2,5ms y la frecuencia de muestreo es de 260kHz, se tienen N=640 muestras para cada símbolo.

7.2. Implementación

La salida del filtro adaptado es la convolución entre la señal de entrada y la respuesta impulsional del filtro, es decir:

$$y[n] = \{r * h_{FA}\}[n] = \sum_{i=-\infty}^{\infty} r[i] h_{FA}[n-i] \qquad \textbf{7.20}$$

Reemplazando:

$$y[n] = \sum_{i=-\infty}^{\infty} r[i] \left(2A \sum_{k=0}^{\frac{N}{2}-1} \delta[n-k-i] - 2A \sum_{k=\frac{N}{2}}^{N-1} \delta[n-k-i] \right) \qquad \textbf{7.21}$$

$$y[n] = 2A \sum_{i=-\infty}^{\infty} r[i] \sum_{k=0}^{\frac{N}{2}-1} \delta[n-k-i] - 2A \sum_{i=-\infty}^{\infty} r[i] \sum_{k=\frac{N}{2}}^{N-1} \delta[n-k-i] \qquad \textbf{7.22}$$

Como $h_{FA}[n]$ va desde 0 a $(N-1)$ no es necesario que la sumatoria en "i" valla desde -∞ a ∞. Además se puede ver que cambiar el orden de las sumatorias no altera el resultado, por lo cual se puede escribir:

$$y[n] = 2A \sum_{k=0}^{\frac{N}{2}-1} \sum_{i=-0}^{N-1} r[i] \delta[n-k-i] - 2A \sum_{k=\frac{N}{2}}^{N-1} \sum_{i=-0}^{N-1} r[i] \delta[n-k-i] \qquad \textbf{7.23}$$

Con esta expresión se puede aplicar una propiedad de la delta de Kronecker, resultando:

$$y[n] = 2A \sum_{k=0}^{\frac{N}{2}-1} r[n-k] - 2A \sum_{k=\frac{N}{2}}^{N-1} r[n-k] \qquad \textbf{7.24}$$

Esta expresión utiliza $N-1$ términos de la entrada para obtener la salida $y[n]$ es decir que los coeficiente que multiplican a $r[n-k]$ son los coeficientes del filtro.

Conociendo esto se puede filtrar directamente desde MATLAB utilizando la función $filter$.

7.3. Simulación

Para la simulación del filtro se generó una señal de diez unos consecutivos codificados en Manchester y se observó la respuesta:

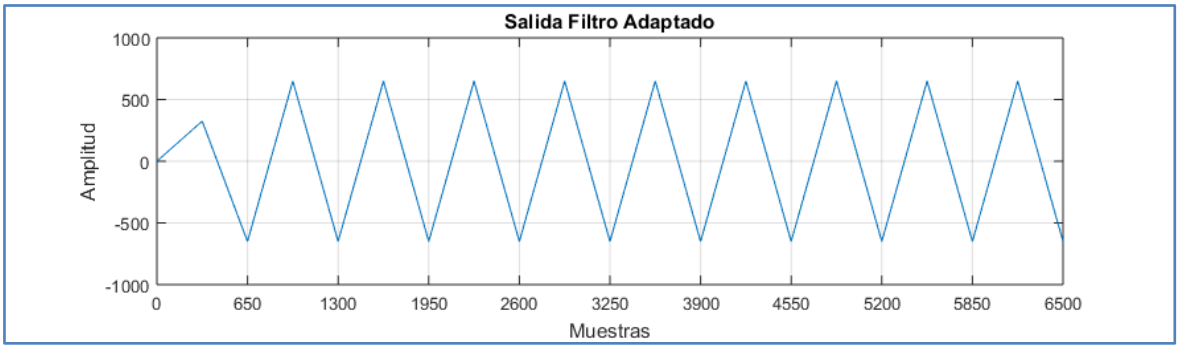

Imagen 7.2: Salida del filtro adaptado para diez unos consecutivos

Si ahora se analiza el gráfico se puede observar dos cosas, primero que cada 640 muestras se tiene la distancia máxima al umbral de decisión (umbral igual a cero) y segundo que para todos esos puntos la salida tiene un valor negativo. Como se dijo previamente la decisión se debe tomar en

los instantes $n = kN$ siendo N=640, con lo cual puede verse que el instante de toma de decisión es óptimo. Finalmente como el valor en todos los kN es negativo, se puede concluir que la hipótesis uno es verdadera en todos los instantes de toma de decisión, es decir, que lo que se recibió es una secuencia de unos.

Para poder realizar una prueba del comportamiento del filtro adaptado bajo ruido, es necesario realizar una simulación que integre tanto este filtro como todos los componentes previamente desarrollados del receptor. Esto se hace porque el ruido de la señal de entrada sufre varios procesamientos antes de llegar al filtro adaptado. Complementando esto, se debe simular la señal real que estaría recibiendo el receptor, es decir, 160ms de portadora únicamente al comienzo de la misma y luego los datos codificados en Manchester y modulados en fase.

La simulación fue realizada para distintas relaciones señal a ruido. El código utilizado para realizar esta simulación se encuentra en el [Anexo 7].

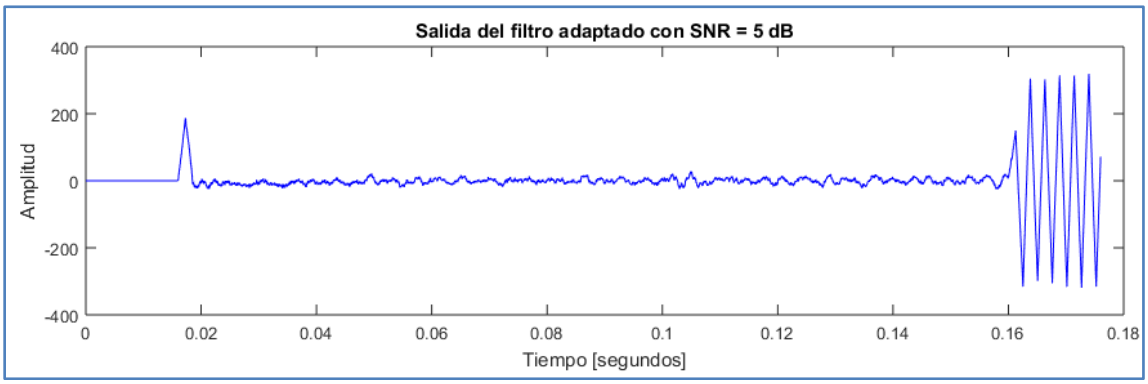

Imagen 7.3: Salida del filtro adaptado para SNR =5dB

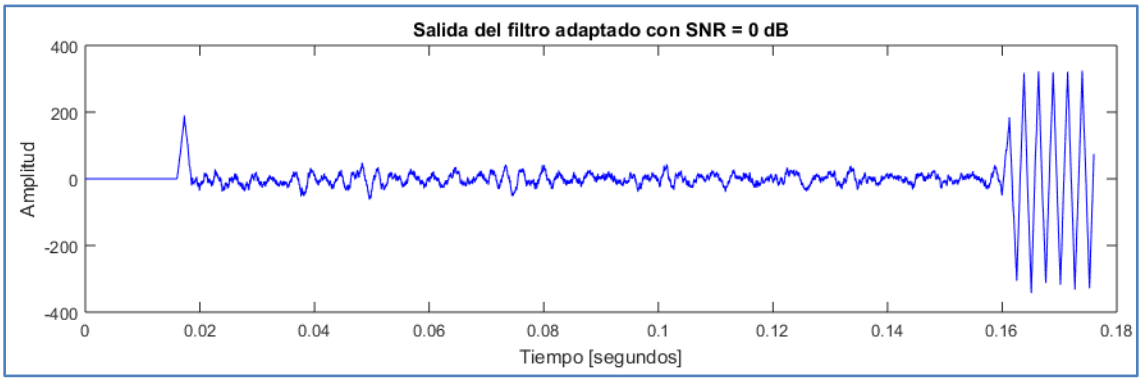

Imagen 7.4: Salida del filtro adaptado para SNR =0dB

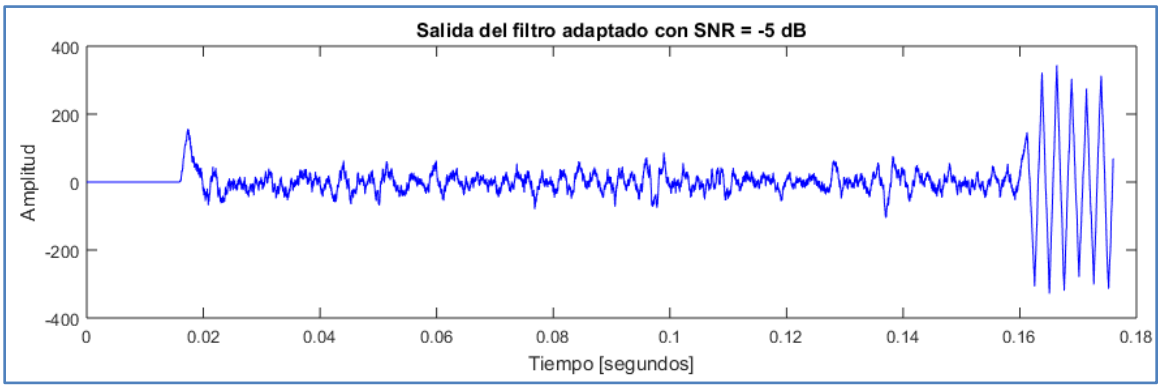

Imagen 7.5: Salida del filtro adaptado para SNR =-5dB

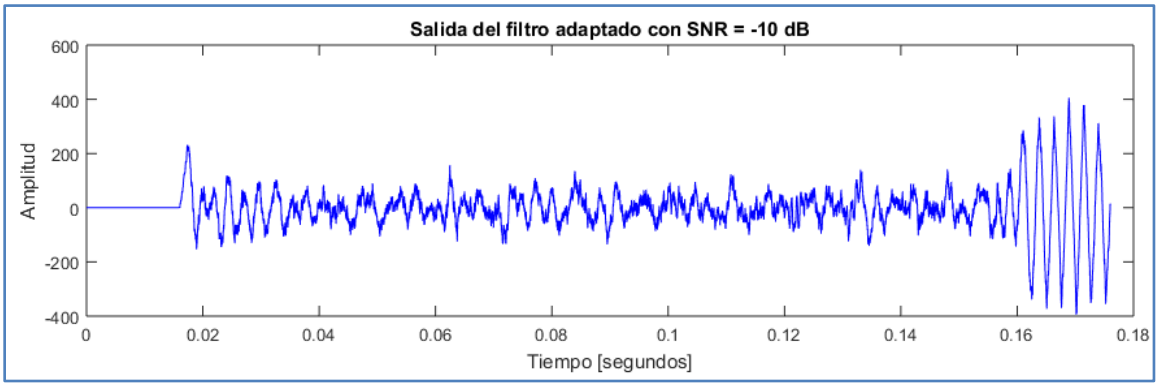

Imagen 7.6: Salida del filtro adaptado para SNR =-10dB

Puede observarse como el error de fase empeora a medida que la relación señal a ruido baja. Pero pese a esto, se ve que los datos a la salida del filtro todavía son detectables aún para la relación señal a ruido de -10dB.

A su vez, se puede ver a simple vista que el umbral con el cual se comparan las hipótesis (umbral igual a cero), no puede distinguir la señal del ruido, lo cual puede llevar a que se interpreten picos de ruido como datos transmitidos. Para solucionar esto se plantean umbrales diferenciados para cada hipótesis. En otras palabras se elije un umbral muy positivo para verificar la hipótesis cero y un umbral muy negativo para verificar la hipótesis uno. La función de dichos umbrales es la de detectar la presencia de un bit y no su valor (esto es realizado por el umbral de decisión). Estos umbrales solo se aplican al inicio de la detección hasta que la trama es detectada.

Gráficamente:

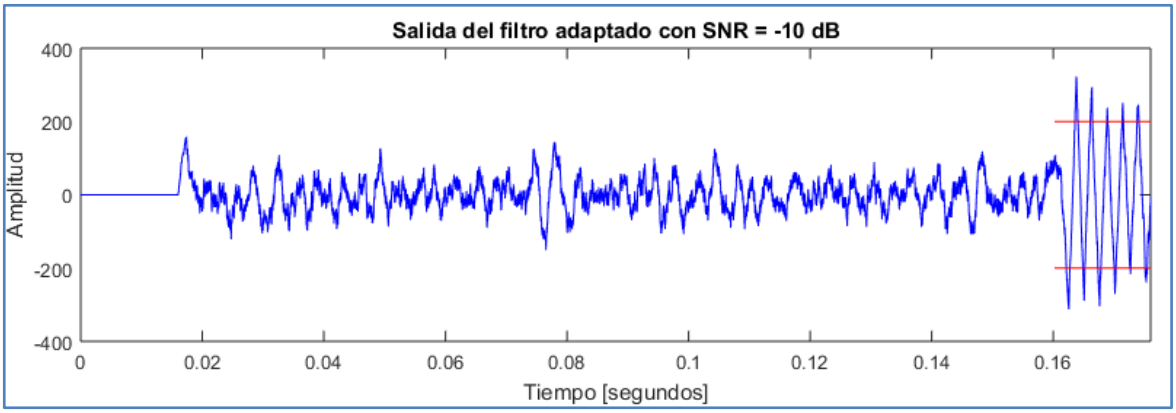

Imagen 7.7: Salida del filtro adaptado para SNR =-10dB con los umbrales de detección de bit

De esta forma puede apreciarse que lo niveles de ruido quedan en promedio desafectados aún para una relación señal a ruido de -10dB. Pueden existir casos en los cuales algún ruido supere dicho umbral, generando un error en la recepción.

8.Sincronismo de BIT

Dado que no se conoce en que instante de lo 160ms de portadora se detecta la señal no es posible, en principio, asegurar cuando se comenzaran a recibir los bits y menos aún cuando será el momento óptimo para tomar la decisión de si se recibió un uno o un cero.

En el capitulo anterior se vio que para tomar la decisión sobre el valor del bit recibido, era necesario comparar la salida del filtro adaptado con los umbrales de detección de bit. Pero el instante exacto en cual la señal cruza dichos umbrales es, difícilmente, el instante óptimo para tomar la decisión. Y agregado a esto no todas las veces que la señal cruza los umbrales se debe tomar una decisión. Para ilustrar esto en el siguiente grafico se muestran todos los cruces por los umbrales y los instantes óptimos donde tomar la decisión en una transmisión de bits aleatorios.

Imagen 8.1: Bits aleatorios a la salida del filtro adaptado

En este escenario, el sincronismo de bit, cumple la función de asegurar que siempre se esté tomando la decisión en instantes óptimos, es decir, aquel que presenta mayor distancia al umbral de decisión.

Es necesario remarcar, que el sincronismo de bit obtiene información de sincronismo de cada pico que cruce un umbral, sea este o no el correcto para la toma de decisión. Retomando la imagen anterior puede verse que el primer pico en superar el umbral es un instante de toma de decisión, pero

el segundo ubicado $^{N}/_{2}$ muestras más adelante no lo es ($N = 640$). El sincronismo de bit no va a poder diferenciarlos, pero si usar la información que poseen para poder asegurar lo mejor posible el instante óptimo de toma de decisión. Este problema de incertidumbre es resuelto por la siguiente etapa, el sincronismo de trama.

El algoritmo propuesto para implementar el sincronismo de bit, consta de dos etapas.

La etapa inicial, o primera sincronización, parte de la condición que al detectar el primer cruce por un umbral, en las siguientes $^{N}/_{2}$ muestras (mitad de la cantidad de muestras de cada bit) existe un máximo en el cual se da el punto óptimo para tomar la decisión.

Gráficamente:

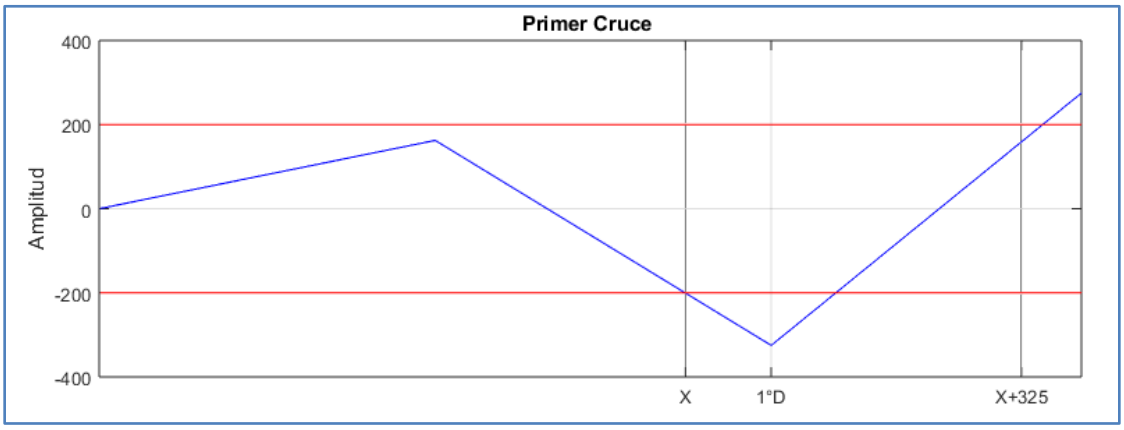

Imagen 8.2: intervalo y valor óptimo de la primera detección

Puede verse que dentro del rango de las $^{N}/_{2}$ muestras se encuentra contenido el primer punto de decisión óptimo.

La forma en la cual se implementa el algoritmo en MATLAB es, buscar el máximo del modulo de esas $^{N}/_{2}$ muestras y una vez obtenido, se establece dicho punto como referencia para la segunda etapa. A su vez en dicho punto se debe tomar una decisión, por lo cual se compara su valor con el umbral de decisión, es decir, se compara con cero.

La segunda etapa, o sincronización continúa, se realiza a continuación. Para esta etapa se sabe que una vez conocido el primer instante de toma de decisión óptimo, el siguiente instante debería encontrarse $^{N}/_{2}$ muestras (mitad del largo del bit) más adelante. Por lo cual se toma un intervalo de $^{N}/_{4}$ muestras centrado en dicho punto y se busca el máximo valor absoluto. Que el intervalo sea de $^{N}/_{4}$ muestras, es decir un cuarto del largo del bit, significa que el receptor completamente desincronizado necesita dos bits para sincronizarse. Si el máximo se encuentra en el centro del intervalo, no se realiza corrección alguna, pero si se encuentra desplazado, se debe realizar la corrección en el sentido opuesto a dicho desplazamiento. A su vez, cada máximo se compara con el umbral de decisión para saber si es un uno o un cero y luego se almacena dicho bit. La etapa de sincronización continua, como su nombre lo indica, debe realizarse en todos los instantes en los cuales se espera tener un punto de decisión óptimo, es decir cada $^{N}/_{2}$ muestras, pero puede suceder que en uno de estos instantes no exista un máximo que supere los umbrales de detección de bit. Si esto sucede, significa que ese intervalo no posee información de sincronismo, pero aún así se compara el máximo con el umbral de decisión y se almacena el valor del bit obtenido.

Simulando este proceso, puede mostrarse cuáles son los intervalos en los cuales se realiza la búsqueda de máximos tanto en la etapa de sincronización inicial, como en la etapa de sincronización continua. El siguiente gráfico (Imagen 8.3) muestra una secuencia de unos a la salida del filtro adaptado comparada con los umbrales de detección.

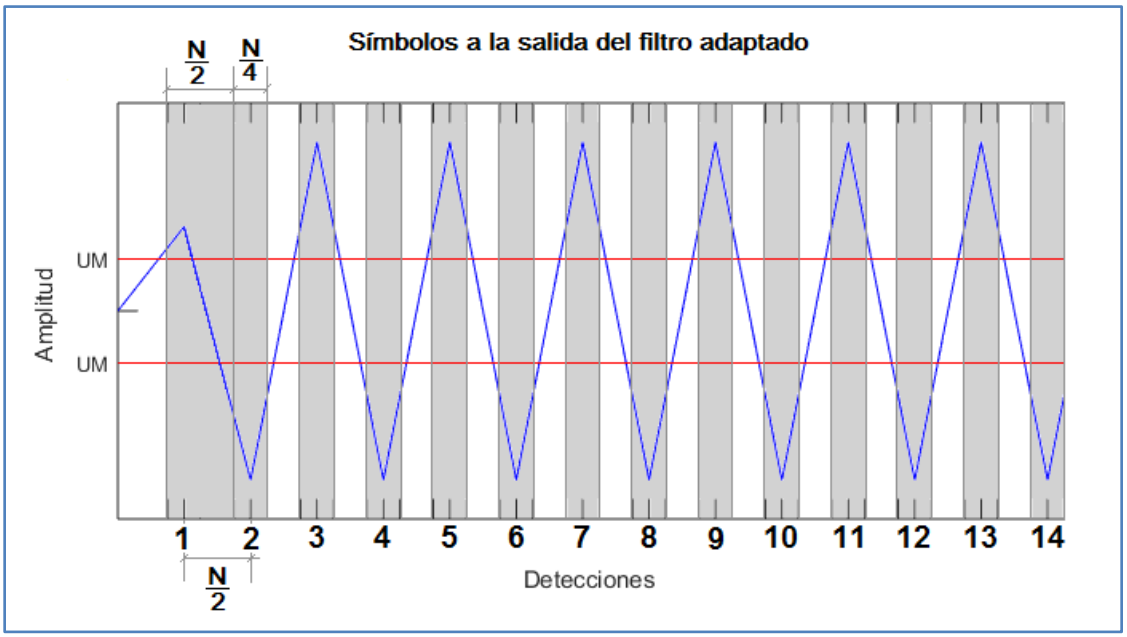

Imagen 8.3: Salida del filtro adaptado para secuencia de unos

Puede verse en el grafico que las zonas oscuras corresponden a los intervalos en los cuales se realiza la búsqueda del máximo. Es notable también como el primer intervalo de búsqueda de máximo es mayor tamaño a los demás, esto se debe a que el algoritmo de primera sincronización utiliza más muestras.

9.Sincronismo de Trama

El sincronismo de trama cumple la función de orientar al receptor en cuanto a en que parte del mensaje se encuentra y también resuelve una ambigüedad de π (radianes) en el sincronismo de portadora. Para esto se vale del campo de trama formador por 8 bits de valor conocido y el bit de inicio de mensaje (Tabla 1.2). A su vez como se mencionó previamente el sincronismo de bit, toma decisiones periódicamente cada $^{N}/_{2}$ muestras, sea o no ese, un instante de decisión optimo. Esto quiere decir que estoy almacenando el doble de bits y a su vez los bits correctos están intercalados con bit erróneos. El problema es que en principio no se puede saber cuáles son los correctos dentro de la secuencia.

La forma en la que se resuelve esto es, primero se separa la secuencia de bits en dos secuencias de 24 bits (cabecera y trama), de manera que una contenga los bits pares y la otra los impares. De estas dos secuencias se toman los primeros 9 bits y se los compara con la trama, luego se desplaza en un bit y se vuelven a comprar los 9 bits tomados con la trama, y así se sigue hasta el final de los 24 bits. Si en alguna de las dos secuencias la trama coincide en algún momento se toma esa secuencia como verdadera y se utiliza para sincronizar el reloj, lo que quiere decir que se cambia el sincronismo de bit. A partir del último bit de la secuencia verdadera se sabe que N muestras después debe haber otro bit verdadero. De ahí en más el sincronismo de bit busca máximos continuamente cada N muestras en vez de cada $^{N}/_{2}$ muestras como antes.

En caso de no encontrar coincidencia en ninguna de las dos secuencias se descarta el mensaje y se vuelva a la etapa de detección de portadora.

En la siguiente simulación se muestra como se aplica el sincronismo de bit en la trama. Pueden verse las zonas en las cuales la señal no supera los umbrales de detección de bit, es decir las que no poseen información de sincronismo.

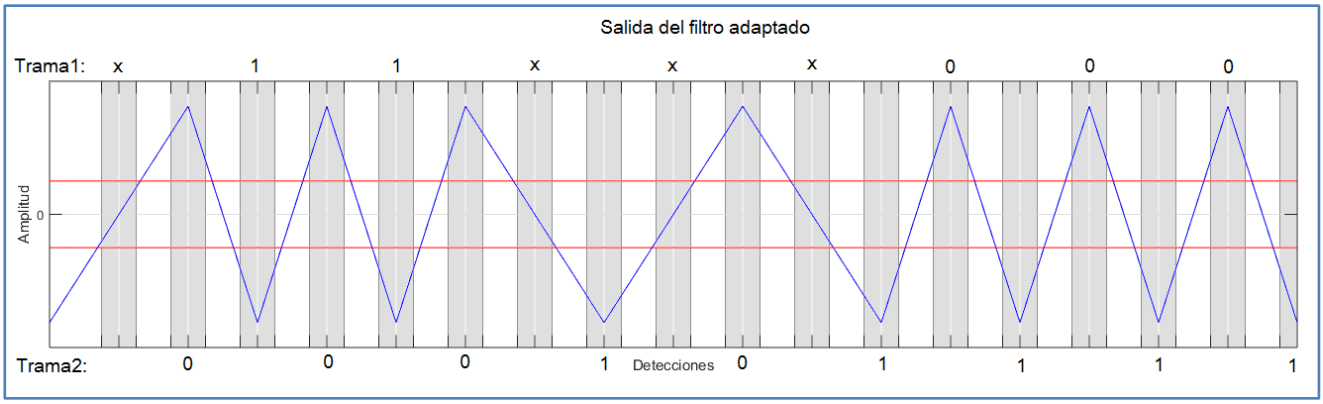

Imagen 9.1: Salida del filtro adaptado cuando se recibe la trama

Pueden verse las dos tramas obtenidas, las cuales deben ser comparadas con el valor predefinido.

10. Interfaz gráfica

En este capítulo se verán las distintas interfaces gráficas implementadas en el trabajo, las cuales tuvieron como fin la comprobación del correcto funcionamiento de determinadas instancias y la posibilidad de mostrar parámetros representativos del receptor de forma práctica.

Para el desarrollo de las interfaces gráficas se usó en su totalidad el entorno de desarrollo que brinda MATLAB, GUIDE (graphical user interface design environment).

10.1. GUIDE

El entorno GUIDE (graphical user interface design environment) es una herramienta de desarrollo gráfico de MATLAB, que permite crear una interfaz grafica la cual permite la fácil interacción con un usuario.

El programa consta de una ventana de diseño o workspace (imagen 10.1) donde se insertan los indicadores o controles del programa que dan forma a la interfaz gráfica, una barra de menús con opciones de manejo del archivo y de la propia interfaz del entorno de desarrollo, una barra de funciones que contiene los controles o indicadores que se pueden usar en el programa y una barra de herramientas con atajos a las funciones más frecuentemente usadas.

Paralelamente a esta ventana de diseño gráfico de la interfaz, se crea una ventana de funciones (imagen 10.2), las que controlan los distintos elementos que uno coloca en la ventana anterior, llamadas "Callbacks".

Esta ventana donde se crean las funciones es un Script el cual se abre en el editor de código de MATLAB.

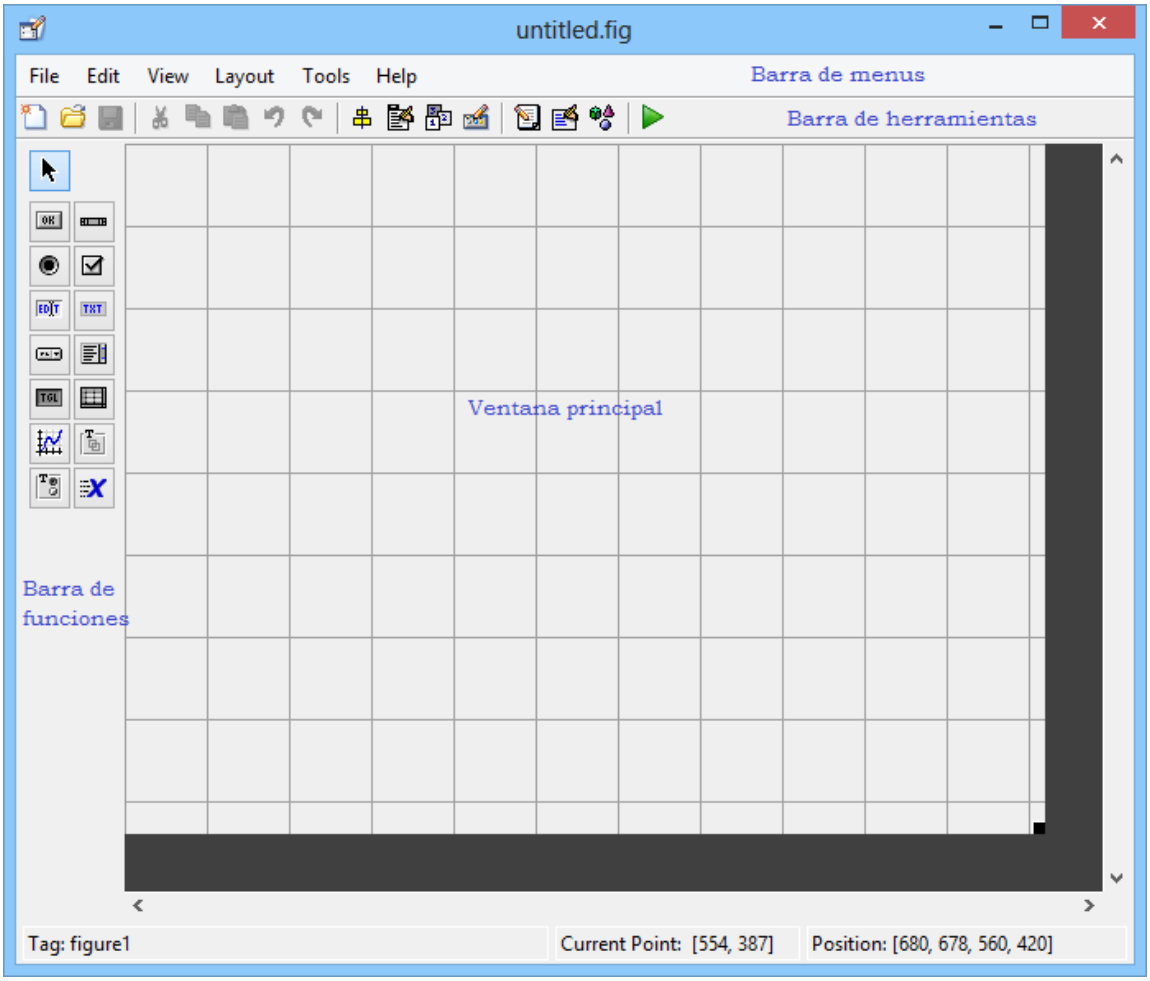

Imagen 10.1: Workspace

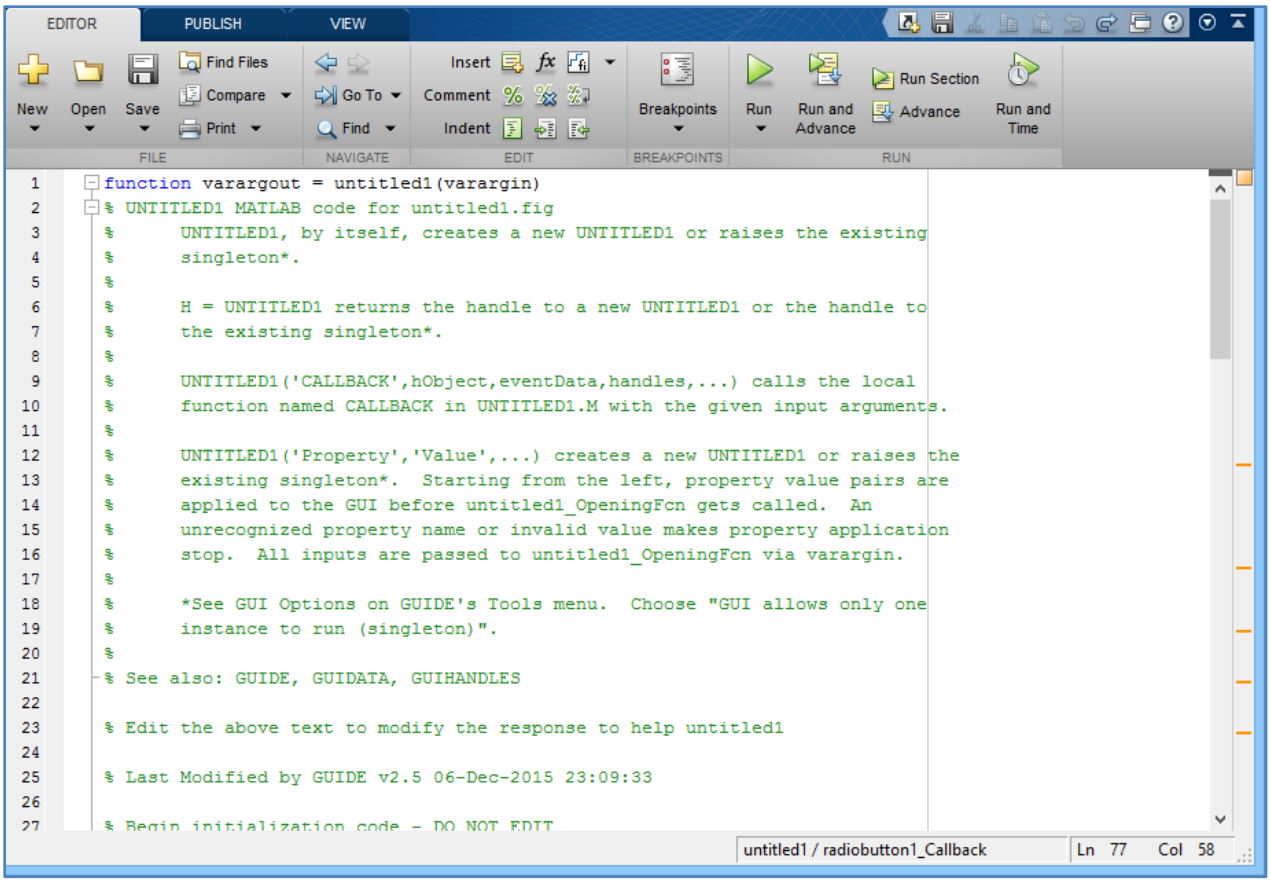

Imagen 10.2: Ventana de funciones

10.1.1. Indicadores y Controles

Los indicadores y controles son el conjunto de elementos gráficos que se pueden programar y formar parte de la interfaz grafica. A su vez dichos indicadores y controles poseen variedad de parámetros y características de apariencia y funcionalidad configurables, que ayudan a la personalización de la interfaz.

Los controles del programa, son los elementos que tiene a su disposición el usuario que maneja el funcionamiento del programa, es decir, pulsadores de inicio, parada e ingreso de datos. Mientras que los indicadores son todos aquellos elementos que sirven como salida de información, es decir, textos o gráficos.

A continuación se muestran todos ellos en su forma más estándar

- Pulsadores con y sin latch:

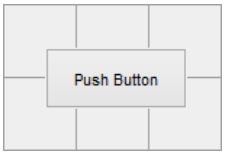

Imagen 10.3: Push Button

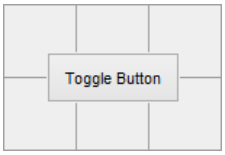

Imagen 10.4: Toggle Button

Estos pulsadores permiten que el programa ejecute funciones, corra scrips, abra ventanas nuevas o modifique parámetros de la ventana actual.

- Slider:

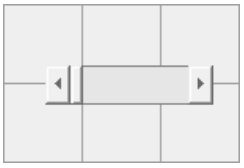

Imagen 10.5: Slider

Este control permite una variación gradual de una variable dentro del programa, la variación así como los valores posibles y los limites máximo y mínimo son fácilmente configurables. El valor de la variación en cada instante se guarda en el parámetro "Value" del Slider.

- Check Box:

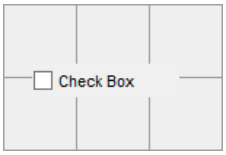

Imagen 10.6: Check Box

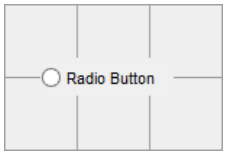

Imagen 10.7: Radio Button

Estos dos tipos de controles, permiten al usuario cambiar un valor binario en el programa, generalmente usado como flags. Si bien ambos tienen un comportamiento similar, el Radio Button posee un comportamiento particular cuando es usado con el Button Group.

- Cuadro de texto editable:

Imagen 10.8: Edit Text

Este cuadro permite al usuario ingresar valores al programa, los cuales son del tipo texto. Para introducir datos de tipo numérico es necesario realizar una conversión en el programa.

- Texto estático:

Imagen 10.9: Static Text

Este indicador, sirve para mostrar texto previamente definido en el programa, que el usuario no puede modificar. Sin embargo el mismo si puede ser modificado por el programa cuando éste está ejecutándose. Es generalmente usado para mostrar valores de salida, o simplemente texto que no cambie durante la ejecución, como títulos.

- Menú de opciones y listas de opciones:

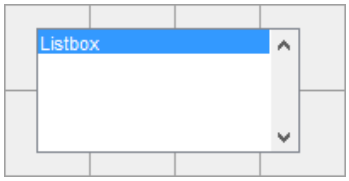

Imagen 10.10: Listbox

Imagen 10.11: Pop-up Menú

Estos dos elementos son similares en cuanto a su funcionalidad, no obstante, el "ListBox" presenta una función adicional como indicador. Utilizados como controles, estos elementos contienen opciones que al ser seleccionadas por el usuario realizan una acción o ejecutan alguna función o código. Pero el "ListBox" como puede ser modificado desde el programa en tiempo de ejecución, ya sea cambiando el texto de una línea o agregando nuevas líneas, puede ser usado para mostrar gran cantidad de valores de forma simultánea.

- Tabla:

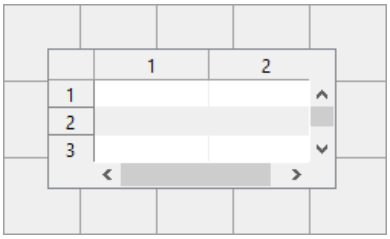

Imagen 10.12: Table

La tabla, como el "ListBox", se usa para mostrar gran cantidad de valores, pero a su vez permite la posibilidad de que el usuario pueda modificar el valor de las casillas, lo cual permite que el programa reciba datos de entrada de forma más ordenada.

- Gráfico:

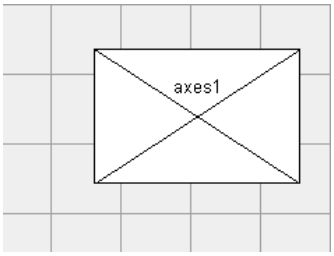

Imagen 10.13: Axes

Imagen 10.14: Axes con ejes graficados

Los gráficos o Axes, cumplen la función de mostrar datos de forma gráfica, pero además pueden mostrar cualquier tipo de información que se encuentre en formato de imagen.

- Panel y grupo de pulsadores:

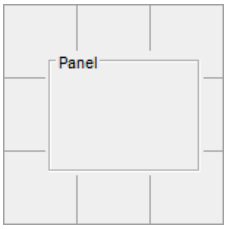

Imagen 10.15: Panel

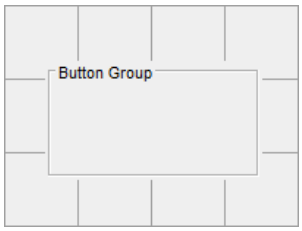

Imagen 10.16: Button Group

Estos dos elementos cumplen la función de organizar la interfaz gráfica en grupos, de los elementos antes enunciados. A su vez como se mencionó previamente el "Button Group" combinado con el "Radio Button" presenta la función adicional de combinar los "Radio Button" de manera que solo uno pueda estar seleccionado.

10.1.2. Direccionamientos

Los direccionamientos, son la forma en la cual se pueden obtener y modificar todas las propiedades tanto de indicadores como de controles.

La forma en la cual uno puede referirse a un objeto, ya sea a un control o indicador, para leer o modificar alguna propiedad del mismo es mediante Tags. Estos Tags pueden ser utilizados dentro del código que describe la interfaz gráfica o dentro de otros que en algún momento ejecute la interfaz.

Para referirse al elemento actualmente en uso se usa el Tag "`hObject`" generalmente utilizado cuando, por ejemplo, la función que ejecuta un pulsador cambia una de sus propiedades.

Para referirse a un elemento particular de la interfaz gráfica se utiliza el Tag compuesto por "`handles.`" y el Tag propio del elemento al que se quiere referir, por ejemplo, para el caso de un pulsador se tiene "`handles.button1`"

La función "`set`" usa estos Tags para modificar alguna propiedad de un elemento de la interfaz gráfica. Dicha función es usada para escribir en texto estático algún mensaje de salida o cambiar el valor de un indicador, como se ejemplifica seguidamente:

```
% Escribir un texto en un indicador de texto estático
set(handles.text1, 'String', 'Texto que se desea mostrar')
% Escribir un número en un indicador de texto estático
set(handles.text1, 'String',45)
% Escribir un texto valor numérico en un texto editable
set(handles.edit1, 'Value',45)
```

Por otro lado la función "`get`" se usa para leer algún parámetro de un indicador o control de la interfaz, y es necesaria para que cada control de ingreso de datos pueda transferir los mismos al programa.

Ejemplos:

```
% Obtiene el valor ingresado por el usuario en un texto editable
get(handles.edit1, 'String')
% Obtiene el valor actual de un pulsador con latch
get(handles.button1, 'UserData')
% Obtiene la opción actualmente seleccionada de un menú
get(handles.menu, 'Value')
```

Obsérvese que en las funciones antes descriptas, el parámetro que se desea obtener o editar se encuentra entre comillas simples (Ej: `'String'`). Existe gran variedad de parámetros, desde parámetros de funcionalidad hasta de aspecto, y también específicos de cada tipo de elemento.

Existe un grupo más de funciones que usa estos Tags, aquellas utilizadas para imprimir en gráficos de la interfaz o Axes. Por ejemplo:

```
% Grafico simple de dos dimensiones
plot(handles.axes1, X, Y)
% Grafico de barras
bar(handles.axes1, X)
% Grafico del histograma de una variable
hist(handles.axes1, X)
```

Como puede apreciarse el Tag correspondiente a un dado Axes se introduce al principio de la función.

Estas funciones son solo ejemplos de cómo se usan estos Tags, ya que existe gran variedad.

10.2. Interfaces Desarrolladas

En este capítulo se describirán las tres interfaces finales de este trabajo y sus respectivos propósitos y funcionamientos.

Como dichas interfaces comparten ciertos aspectos en su diseño, se optó por describirlos en primera instancia, estos son los controles de inicio, parada, ganancia y seteo de frecuencia, además de ciertos indicadores básicos de funcionamiento.

10.2.1. Controles comunes

El pulsador de inicio es del tipo “Push Button” lo cual indica que se ejecuta instantáneamente al pulsarlo y no tiene memoria de ningún tipo.

Este pulsador es el encargado de ejecutar el código principal, el cual se encuentra contenido en un Script y además cumple la función de setear la variable de estado del pulsador de parada en cero.

Para lograr esto, la función que se ejecuta al pulsarlo es:

```
set(handles.stop_button,'UserData',0) %Seteo de la variable de estado
DCS_RECEIVER  %Ejecución de la función principal
```

Además, el pulsador de inicio recoge los valores ingresados por el usuario que se necesitan para inicializar la función principal, con lo cual a la función anterior se le agrega:

```
global fc;
frecuencia=get(get(handles.button_group1,'SelectedObject'),'String');
switch frecuencia
    case '401.55 MHz (DCS)'
        fc = 401.5e6;
    case '401.62 MHz (SCD)'
        fc = 401.57e6;
    case '401.65 MHz (ARGOS)'
        fc = 401.6e6;
end
```

```
global gain;
ganancia=str2double(get(handles.edit1, 'String'));
if ganancia <= 45 % Fijado del límite máximo
    gain=ganancia;
else
    gain=45;
    set(handles.edit1, 'String',45)
end
```

En este código es necesario resaltar tres cosas:

Primero, que las frecuencias que se setean son 50kHz menores a las de los sistemas, esto se implementa de esta forma para lograr que la señal de interés quede centrada en el espectro de frecuencias ya que la sintonización del dongle establece la frecuencia que se le setea como comienzo del ancho de banda y no como frecuencia central.

Segundo, que se usan dos variables globales `gain` y `fc`, que son necesarias para que el programa principal pueda leer estos valores en cualquier momento que lo requiera.

Tercero, como el código mostrado anteriormente se ejecuta cuando se oprime el pulsador de inicio, puede deducirse que la frecuencia y la ganancia solo cambiaran al inicio del programa y no mientras este se encuentre ejecutándose. Ello es para evitar que una recepción en proceso se corte por un cambio en dichos parámetros.

Cabe destacar que el pulsador de inicio en la interfaz grafica funciona como una interrupción, es decir, que no importa si hay otro código ejecutándose, este dará la orden de ejecutar el código principal del receptor. Por lo cual, es necesario colocar un mecanismo que evite que el programa se ejecute nuevamente si ya se encuentra en ejecución. Debiéndose adicionar el siguiente código en la función del pulsador de inicio:

```
global iniciado;
if iniciado == 0
    DCS_RECEIVER  %Ejecución de la función principal
end
```

Y dentro de la función principal "DCS_RECEIVER" se añade:

```
global iniciado;
iniciado = 1;
```

De esta forma, si accidentalmente se vuelve a oprimir el pulsador de inicio, el programa no se ejecuta dos veces.

La frecuencia de funcionamiento se ingresa al programa mediante un conjunto de Button Group y Radio Buttons. Esto es así ya que la frecuencia debe ser seleccionada entre tres posibles valores de frecuencia según el sistema que se quiera recibir. Usar los Radio Buttons en conjunto con el Button Group trae la ventaja de permitir solo un Radio Button seleccionado a la vez, lo cual es necesario, ya que solo una frecuencia es la de funcionamiento del programa.

La ganancia de la etapa de radio frecuencia es ingresada mediante dos posibilidades, utilizando un Edit Text o un Slider, se implementó de esta forma para adecuarse a las distintas necesidades, ya que el Slider permite una variación lineal gradual, mientras que el Edit Text, permite variar rápidamente y a un valor especifico.

Se debe aclarar, que pese a que estos elementos son controles y los mismos cambian parámetros del funcionamiento, ninguno de los dos ejecuta algún código, esto se debe a que estos cambios de parámetros son realizados por el pulsador de inicio.

El pulsador de parada, cuenta con una variable de estado, que almacena información respecto a cuál de los pulsadores fue accionado anteriormente. Esto quiere decir, que si se oprimió el pulsador de inicio su valor será cero y si se oprimió el pulsador de parada será uno. Esta forma de implementación permite que en cada bucle del programa principal se verifique el valor de dicha variable para detectar lo más rápido posible la orden de parada.

Por lo que el pulsador de parada ejecuta en su función:

```
set(handles.stop_button,'UserData',1);
guidata(hObject, handles);
```

Y dentro del código principal, al final de cada bucle se agrega el siguiente código para verificar el valor de la variable de estado:

```
if get(handles.stop_button,'UserData') == 1
      msgbox('exit');
      return
end
```

En la imagen a continuación se muestra el grupo de controles resultante:

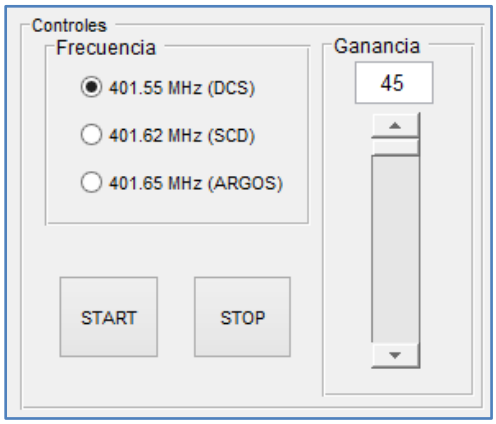

Imagen 10.17: Grupo de controles

Como puede verse en la imagen todos estos controles se encuentran contenidos en un panel, para diferenciarlos de los indicadores.

10.2.2. Indicadores básicos

En esta instancia, se abordará el conjunto de indicadores básicos, que muestran los parámetros del funcionamiento del receptor. Estos parámetros son: información que brinda la etapa de radio frecuencia, como cantidad de frames perdidos y el retardo; como así también información de procesamiento como la frecuencia estimada y la amplitud estimada (si es que existe alguna señal que sobrepase el umbral de detección).

La forma en la cual se implementaron estos indicadores es bastante simple, ya que solo deben mostrar en formato de texto un valor numérico. Por lo cual resulta:

```
%Ejecución de
set(handles.text1,'String',f_estimada)
%Ejecución de
set(handles.text2,'String',a_estimada)
%Ejecución de
set(handles.text3,'String',frames_lost)
%Ejecución de
set(handles.text4,'String',frames_retardo)
```

Finalmente este grupo de indicadores, presentan la siguiente forma gráfica:

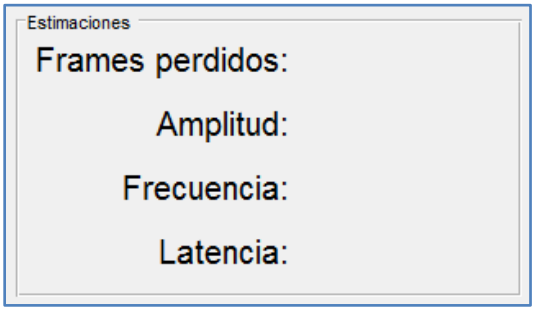

Imagen 10.18: Indicadores básicos

10.2.3. Interfaz 1: Comprobador de detección

Esta interfaz fue creada con la idea de mostrar en funcionamiento las etapas de toma de muestras y detección del receptor final. Para lograr este objetivo, la interfaz muestra la estimación del espectro en un rango de frecuencias centrado en la frecuencia que se desea recibir. A su vez, en el mismo gráfico, se visualiza simultáneamente el umbral de detección del receptor, permitiendo a simple vista, saber si existe alguna señal detectable o si existe alguna señal que pueda interferir en la recepción.

Para lograrlo es necesario graficar el módulo al cuadrado de la FFT de largo N. Lo cual significa que en cada ciclo de cálculo de la FFT se debe ejecutar:

```
plot(handles.axes1,f_eje,X,'b',f_eje,um,'r');
handles.axes1.Title.String = 'Espectro señal de entrada';
handles.axes1.YLabel.String = '|Y(f)|';
handles.axes1.XLabel.String = 'Frecuencia';
handles.axes1.XLim = [0 f_eje(length(f_eje))];
handles.axes1.YLim = [0 30];
pause(0.0001);
```

En este punto, cabe destacar el agregado de la función "`pause(0.0001)`", la cual genera una pausa en el programa necesaria para que MATLAB genere la información de forma gráfica. De no colocarse esta función, MATLAB no puede generar el gráfico a tiempo, por consiguiente la posterior orden de graficar llegaría antes de que se termine de generar la anterior y así sucesivamente, lo que llevaría a que no se grafique ninguna realización.

Las funciones que comienzan con "`handles.axes1`" se encargan del seteo del Axes, poniéndole titulo, colocando etiquetas en cada eje y fijando los límites máximos y mínimos para los mismos. Es necesario que estas funciones se ejecuten en cada ciclo, de lo contrario en el ciclo siguiente se borrarían dichos parámetros.

Finalmente la interfaz grafica que se obtiene es:

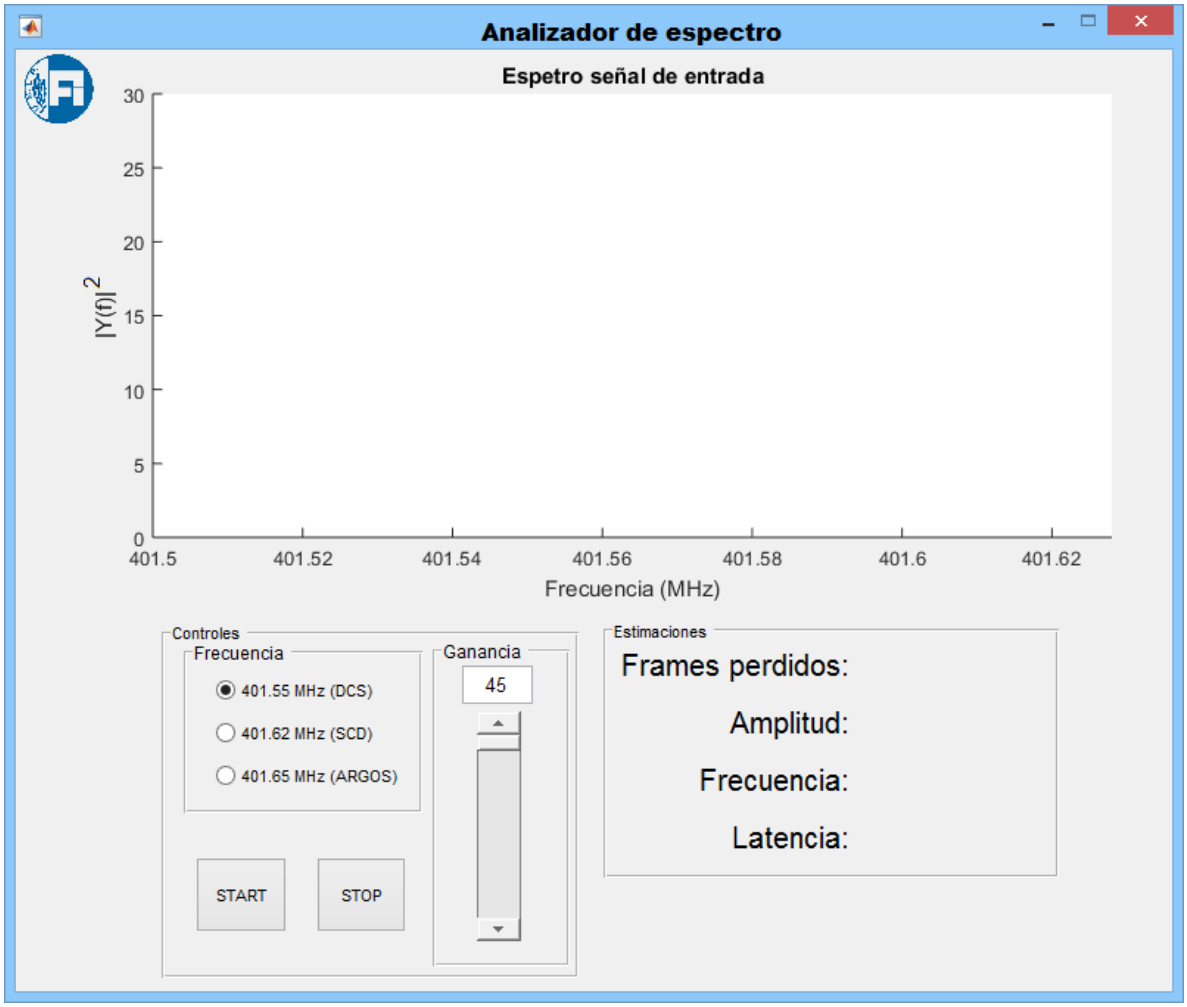

Imagen 10.19: Interfaz 1

Un ejemplo de esta interfaz en funcionamiento se muestra a continuación, donde se puede apreciar el umbral de comparación, y consecuentemente se puede saber cuándo una señal ha sido detectada.

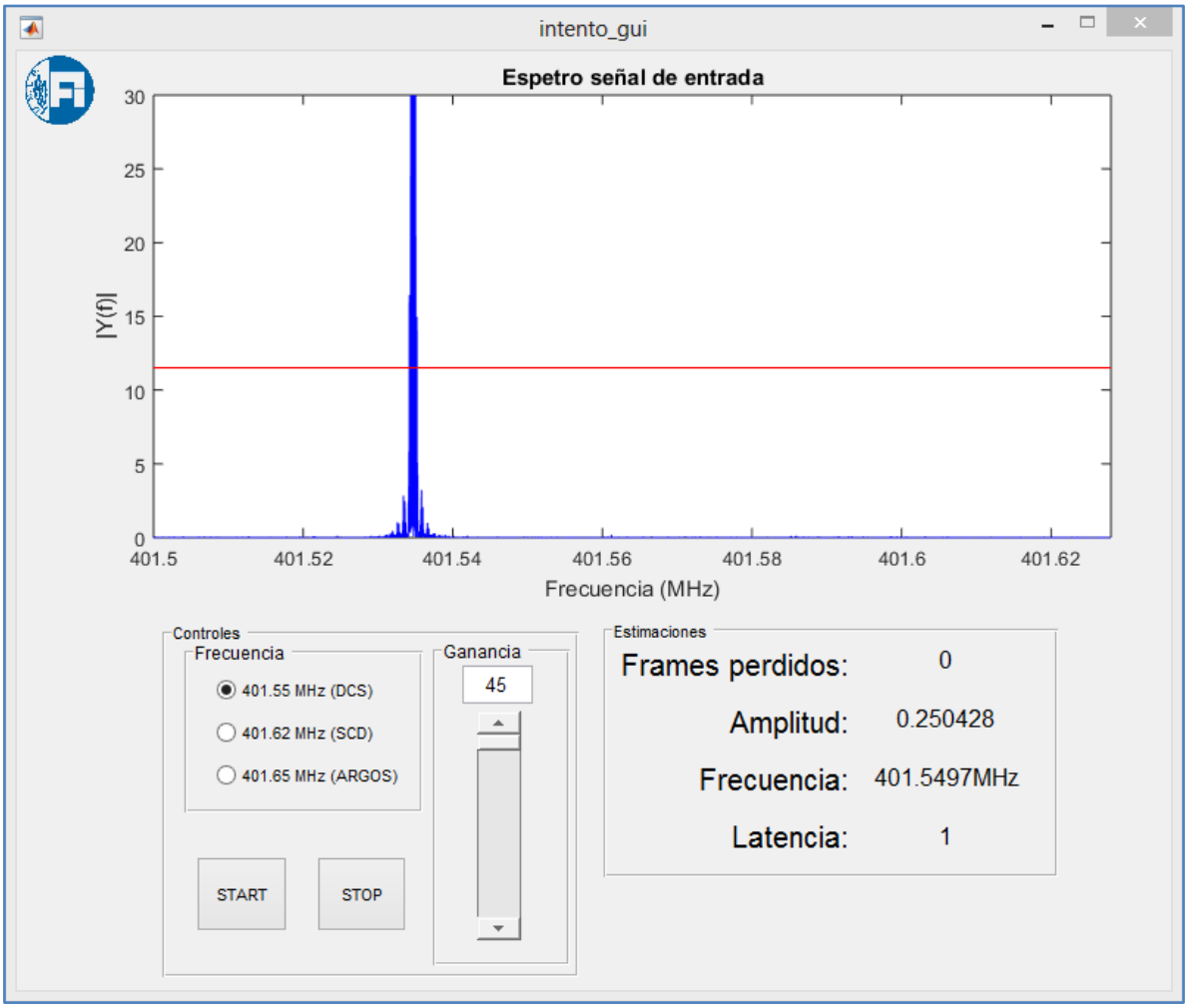

Imagen 10.20: Interfaz 1 funcionando

10.2.4. Interfaz 2: Diagrama de ojo

El objetivo de esta interfaz es mostrar en funcionamiento todas las etapas, a excepción de las etapas de decodificación del mensaje, es decir: detección, estimación, PLL, filtro adaptado, sincronismo de bit y sincronismo de trama.

A su vez, como su nombre lo indica, esta interfaz se vale del diagrama de ojo para mostrar en funcionamiento dichas etapas del receptor.

El diagrama de ojo, es utilizado para analizar el comportamiento de los enlaces de transmisión, permite analizar las formas de onda de los pulsos a la salida del filtro adaptado, para lograr observar sus, niveles de ruido, potencias de las señales y con ello apreciar la distorsión del canal (ISI), la severidad del ruido o interferencia y los errores de sincronismo en el receptor.

Al igual que en la interfaz anterior, los controles y los indicadores básicos son idénticos ya que, si bien la función de ambas interfaces es distinta, los parámetros a controlar son los mismos.

Para lograr graficar el diagrama de ojo, se debe conocer el momento de toma de decisión, siendo necesarias a tal fin, las etapas de sincronismo de bit y trama. Una vez conocido el instante de toma de decisión el diagrama consiste en graficar, en una ventana (con largo igual a la duración del bit) la salida del filtro adaptado, de tal forma que en el centro siempre quede el instante de toma de decisión.

A continuación se muestra la interfaz cuando la señal de entrada tiene buena relación señal a ruido.

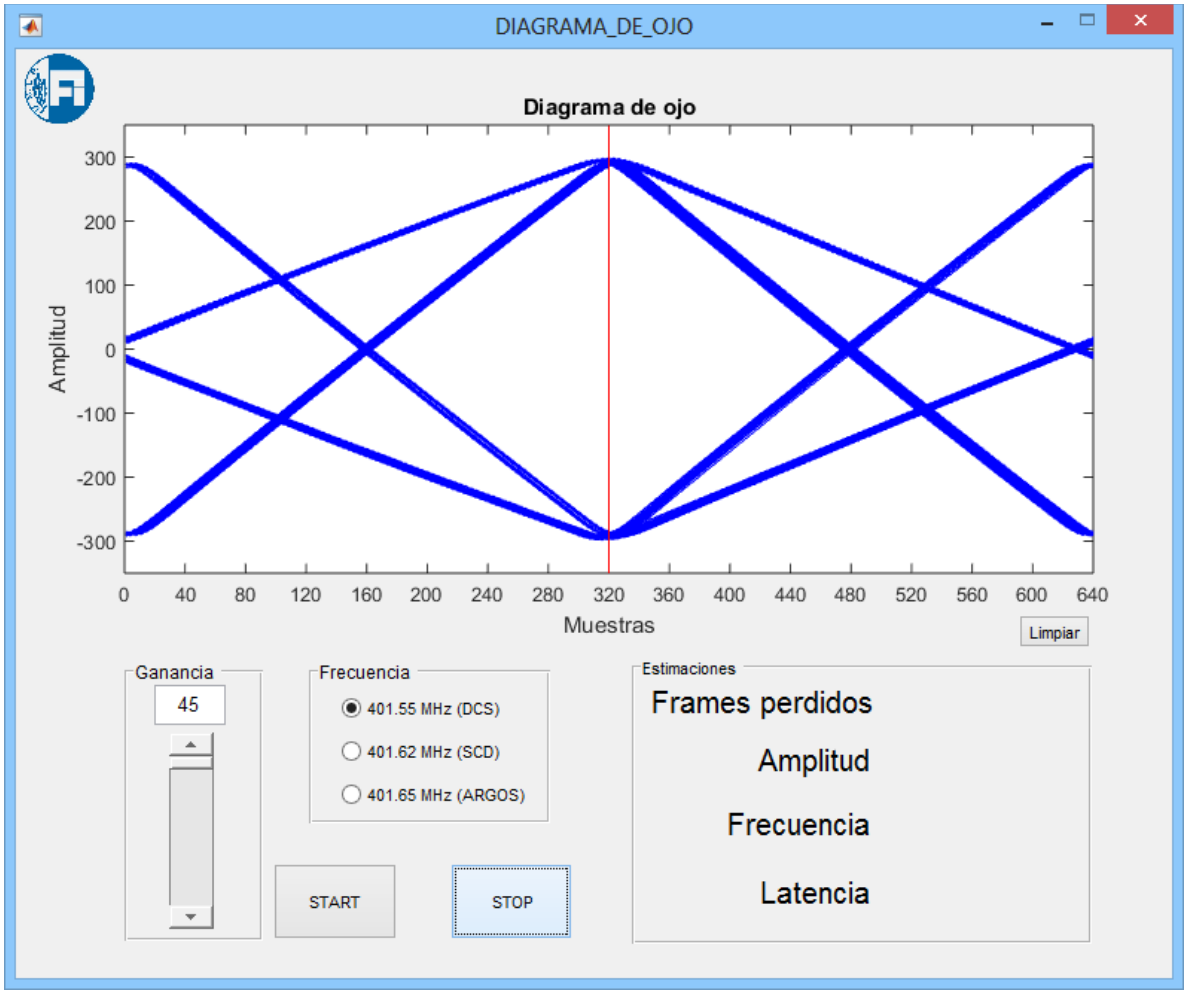

Imagen 10.21: Interfaz 2 con buena relación señal a ruido

En la siguiente imagen, se muestra la interfaz cuando la señal de entrada tiene baja relación señal a ruido.

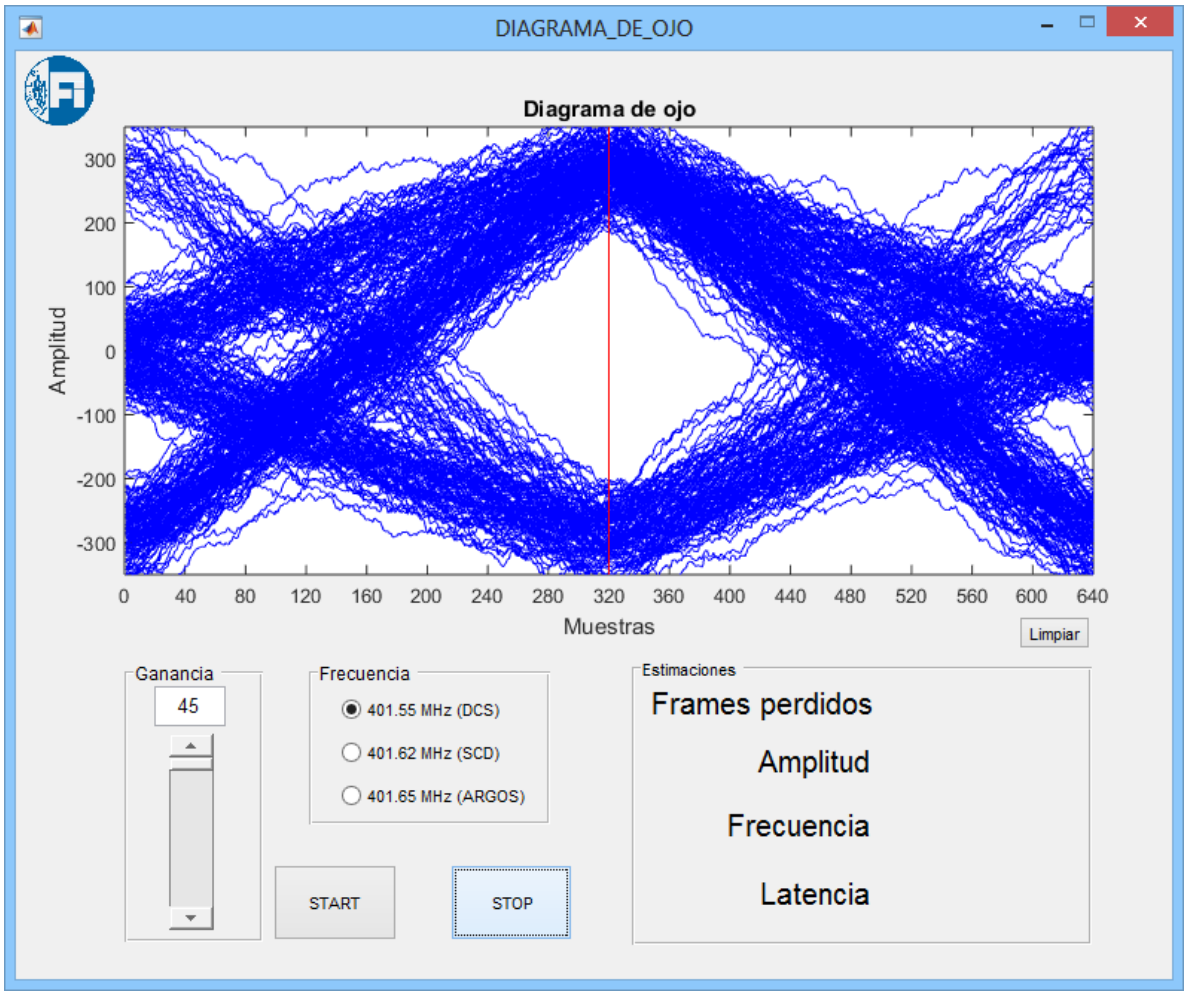

Imagen 10.22: Interfaz 2 con baja relación señal a ruido

10.2.5. Interfaz 3: Receptor

Finalmente la ultima interfaz desarrollada, es el receptor propiamente dicho, es decir, esta interfaz integra todas las etapas del desarrollo. A su vez, la misma tiene en consideración que el usuario final del sistema, no posea necesariamente conocimientos en el área de comunicaciones, es decir, está más orientada al mensaje que a los parámetros comunicacionales.

El grupo de controles de la interfaz se mantiene igual a las anteriores, pero los indicadores son diferentes. En esta interfaz, no se muestran ningún tipo de gráficos, pero sí se muestran, muchos más indicadores de texto con decodificaciones del mensaje.

El indicador que sí se mantiene de etapas anteriores es el de frames perdidos, habida cuenta que es de vital importancia para saber si la señal que se recibió es válida, ya que, si algún frame se pierde puede significar que se perdió una secuencia de varios bits de la señal.

Las decodificaciones que se muestran son, el largo del mensaje que se recibió y el número de plataforma siendo este ultimo mostrado en binario, hexadecimal y decimal.

A su vez, para saber si la recepción fue realizada con éxito, se tiene un indicador que verifica simultáneamente, que la señal haya sido detectada, que se haya enganchado el PLL y que se detectó la trama.

Para concluir, el último indicador muestra el mensaje recibido codificado en hexadecimal ya que queda a disposición del usuario el formato de los datos y almacena en pantalla las nuevas recepciones.

Gráficamente la interfaz resulta:

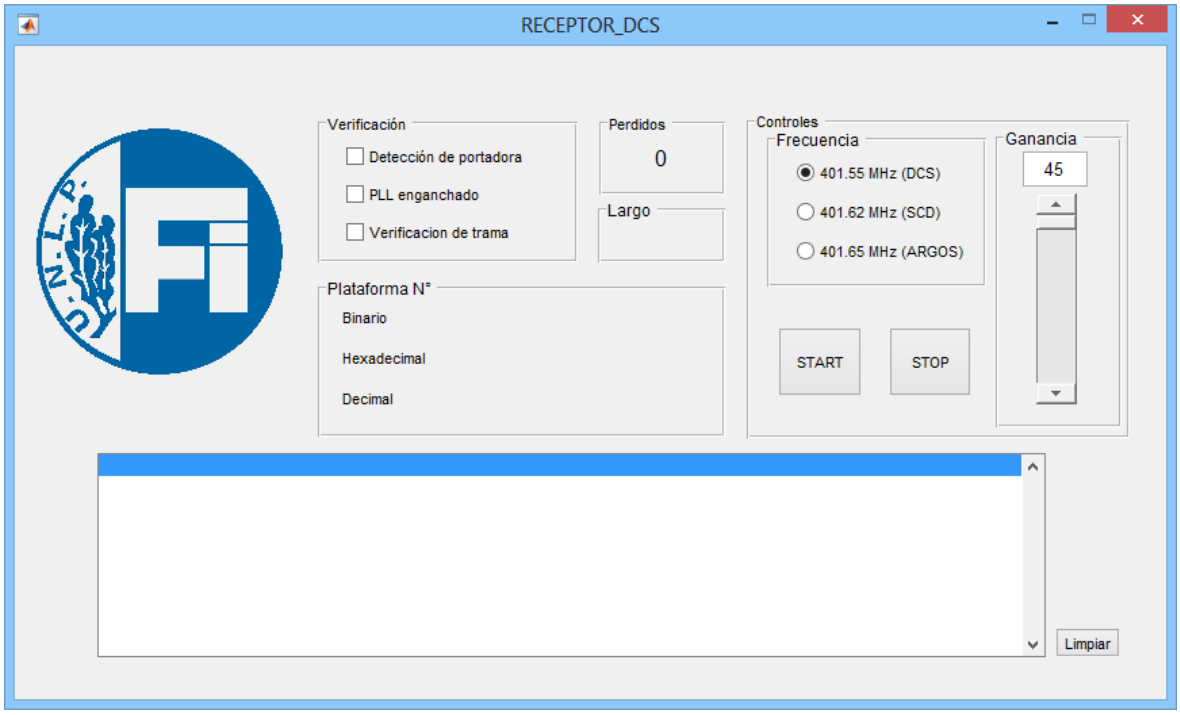

Imagen 10.23: Interfaz Receptor

11. Mediciones Finales

A continuación se muestran los resultados obtenidos para las mediciones finales de desempeño del receptor.

Las primeras pruebas que se le realizaron al receptor fueron con la idea de verificar su funcionamiento. Para realizar estas medidas se utilizo el sistema de la imagen 11.1, cuya función es la de transmitir un mensaje predefinido de forma única o con un ciclo de repetición de 2 segundos. Este mensaje fue recibido por las tres interfaces para su validación. A su vez la potencia con la cual se transmitió se fue variando para ver como esto se reflejaba en la recepción.

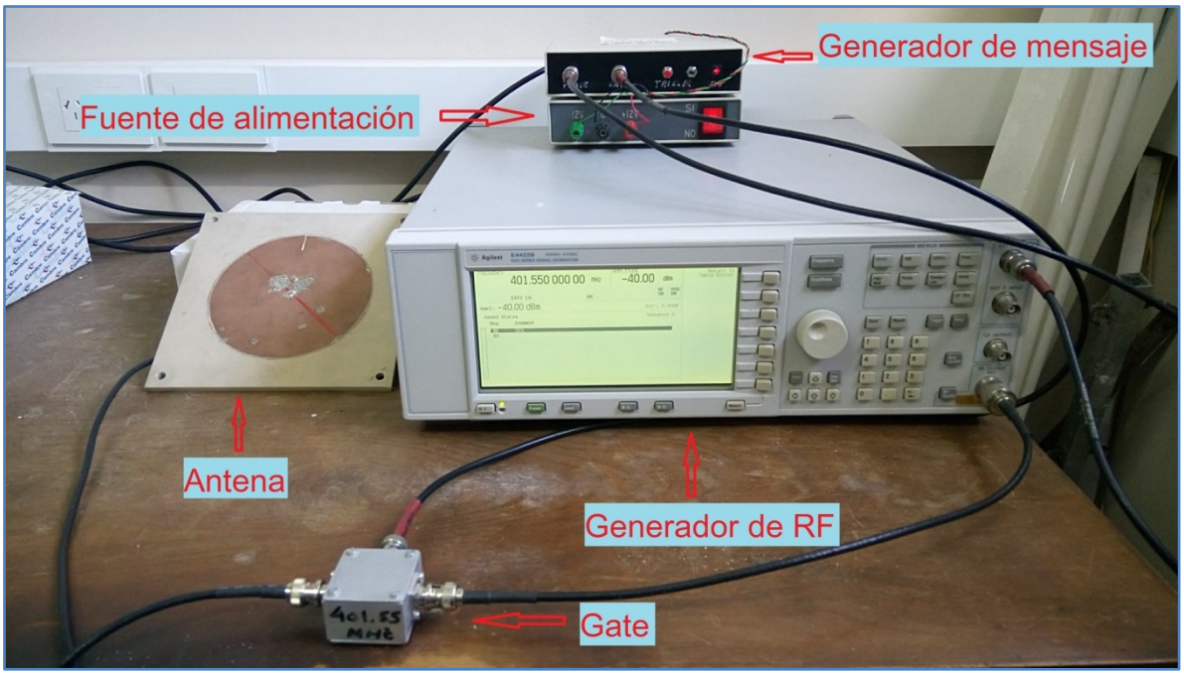

Imagen 11.1: Sistema utilizado para realizar la verificación

En la imagen 11.1 puede verse el instrumento generador de RF, la antena, el generador de mensaje con su fuente de alimentación y el gate que controla la señal hacia la antena. El funcionamiento de este setup comienza con el generador de mensaje, el cual posee un mensaje conocido.

Este mensaje entra al generador de RF donde es modulado con una variación de fase de ±1,1radianes para cada. El generador de RF se encuentra siempre encendido por lo cual a su salida, aun en ausencia del mensaje, siempre se tiene RF a la frecuencia de funcionamiento. Esto es controlado por el generador de mensaje se encarga de comandar el "gate" de manera que la señal de RF solo salga por la antena cuando comienza la transmisión de un mensaje.

A continuación de esto se planteó realizarle medidas de desempeño al receptor para de esta forma comparar el mismo con el receptor óptimo. Para realizar estas mediciones se planteó una simulación en la cual se dejó al receptor en un bucle infinito que permitiera recibir bits indefinidamente. Estos bits recibidos, son comparados con los bits transmitidos y si alguno es distinto, se considera dicho bit como erróneo. El objetivo de esta prueba es el de contar la cantidad de bits que son transmitidos hasta que ocurren 100 errores. Este método es una forma práctica de obtener aproximadamente la tasa de error de bit cometiendo un error máximo del 10%. Esta medida fue realizada para varias relaciones señal a ruido con la finalidad de conseguir una representación del receptor en el plano E_b/N_0 vs PEB, facilitando así la comparación del mismo con el receptor óptimo. Para lograr pasar de SNR a E_b/N_0 se tiene:

$$SNR = \frac{P_R}{N} \qquad \textbf{11.1}$$

Siendo P_R la potencia recibida y N la potencia de ruido recibida. Donde:

$$N = N_0 B_W \qquad \textbf{11.2}$$

Siendo B_W el ancho de banda del filtro antialiasing ($^{f_m}/_{2}$) y N_0 la densidad espectral potencia unilateral de ruido recibida.

Reemplazando 11.2 en 11.1 y multiplicando por $^{T_b}/_{T_b}$ se obtiene.

$$SNR = \frac{P_R T_b}{N_0 B_W T_b} = \frac{2P_R T_b}{N_0 f_m T_b} \quad \textbf{11.3}$$

A su vez $E_b = P_R T_b$, con lo cual se obtiene:

$$SNR = \frac{E_b}{N_0}\frac{2}{f_m T_b} = \frac{E_b}{N_0}\frac{2T_m}{T_b} \quad \textbf{11.4}$$

Expresado en dB

$$SNR_{[dB]} = \frac{E_b}{N_0}_{[dB]} + 10\log\frac{2T_m}{T_b} \quad \textbf{11.5}$$

Para este caso como $T_m = 3{,}9.10^{-6}\ [s]$ y $T_b = 0{,}0025\ [s]$ resulta:

$$SNR_{[dB]} = \frac{E_b}{N_0}_{[dB]} - 25{,}05_{[dB]} \quad \textbf{11.6}$$

Es importante notar, que para la realización de esta prueba, el mensaje que se debe simular es igual al original salvo por el campo de datos que es de longitud indeterminada. Lo cual significa que la señal atraviesa todas las etapas del receptor (detección, estimación, PLL, filtro adaptado, sincronismo de bit, sincronismo de trama) por lo menos una vez y luego se queda en la etapa de decodificación del mensaje hasta que se cometan los 100 errores.

Resultando:

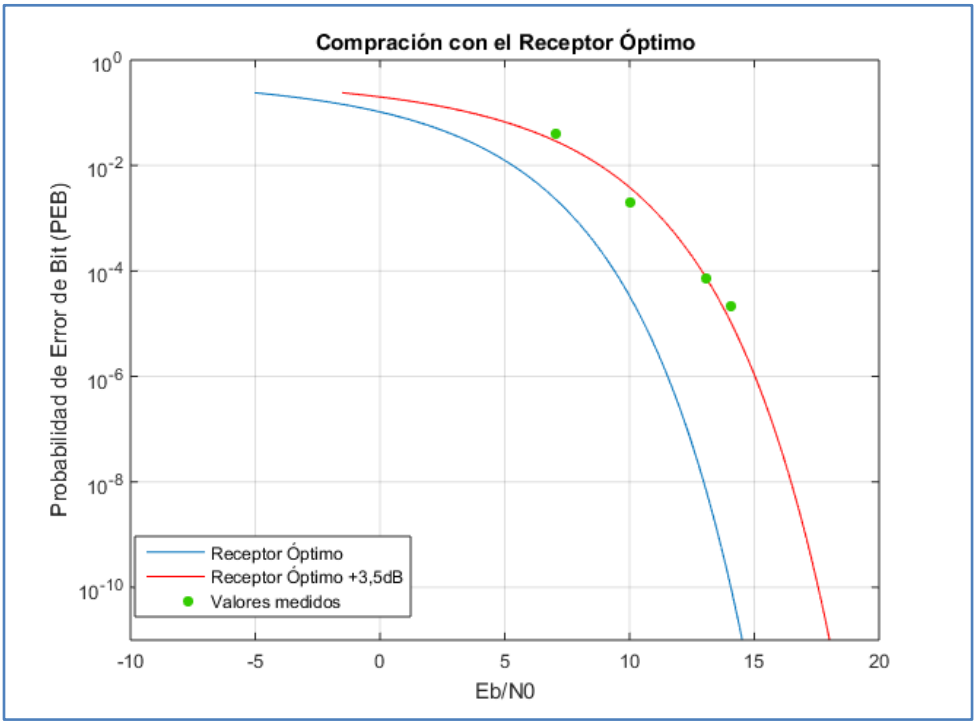

Imagen 11.2: Comparación con el receptor óptimo

En este gráfico puede observarse que en promedio el desempeño del receptor implementado es aproximadamente 3,5dB peor que receptor óptimo. La curva del receptor óptimo viene dada por [Ref 11.1]:

$$PEB_{BPSK} = Q\left(\sqrt[2]{2\frac{E_b}{N_0}}\right) \qquad \textbf{11.7}$$

Como en los sistemas no son exactamente BPSK (ya que la fase entre símbolo no es π). La degradación con respecto a este se puede calcular desde la constelación (imagen 1.3), siendo esta 1dB. Por lo cual el receptor optimo para el sistema es 1dB peor que BPSK.

Esto se debe a varios motivos, primero, como la señal que se tiene son muestras de la señal real, la salida del filtro adaptado también serán

muestras, por lo cual puede que dentro de las mismas no se encuentre el instante óptimo de toma de decisión. Gráficamente:

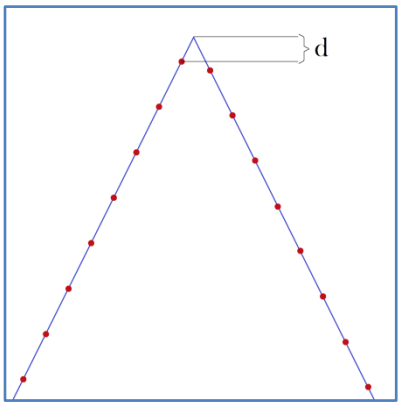

Imagen 11.3: Efecto de sincronismo de bit sub-óptimo por la discretización

En la imagen puede verse que la muestra que el receptor considerará como instante de toma de decisión difiere del óptimo, la cual no sólo genera que la amplitud sea menor, sino también que el reloj de bit no sea correcto. En resumen el sincronismo de bit implementado es sub-óptimo porque no es posible representar todos los valores de corrimiento posibles.

Segundo, el ancho de banda equivalente de ruido del lazo de portadora no es nulo, por lo tanto, existe una desviación (jitter) debido a que la portadora a la salida del lazo no es una señal sinusoidal pura, sino que tiene ruido agregado. Para el caso máximo de la desviación producida podría graficarse:

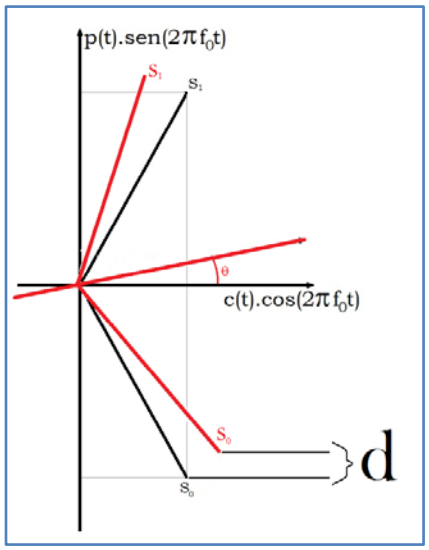

Imagen 11.4: Desviación máxima producida por el Jitter

Finalmente otro factor que empeora el desempeño del receptor es el mismo dispositivo Front-end. Ya que el mismo presenta una figura de ruido que ronda los 3,5dB.

12. Conclusiones

Primero que nada ante la falta de experiencia sobre el tema los objetivos del trabajo fueron simples, lograr la comunicación del dispositivo Front-end de RF con la computadora. Seguido de esto poder lograr comunicar MATLAB con el dispositivo o conseguir que las muestras lleguen al mismo.

Una vez resuelto lo anterior se siguió con el desarrollo teórico del receptor y en base a ello se consiguió la forma de implementación del receptor.

Finalmente la principal conclusión del trabajo, es la factibilidad de implementación del receptor DCS utilizando un dispositivo Front-end de RF como el mencionado, bajo el esquema de radio definida por software, es decir, implementando el procesamiento y la demodulación de la señal por software.

12.1. Resultados del trabajo

Como resultado del trabajo se obtuvo el receptor DCS propiamente dicho implementado en MATLAB así como las interfaces de “Comprobador de detección” y “Diagrama de Ojo” las cuales muestran parámetros de funcionamiento del receptor en tiempo real.

Estas interfaces quedan a disposición del laboratorio GrIDCom así como de la cátedra de comunicaciones.

12.2. Trabajos futuros

Así como el trabajo da por concluida la implementación de un receptor utilizando un dispositivo Front-end de RF en MATLAB, también deja visibles varias posibilidades de desarrollo siguiendo el precedente marcado. Entre algunas de las posibilidades de trabajos futuros a realizarse se encuentran:

- Mejoras en las interfaces visuales, permitiendo la integración de tanto la interfaz para el usuario de ciencia y las interfaces que muestran los parámetros de funcionamiento avanzados del receptor.
- Lograr que el programa sea ejecutable en una aplicación independiente de MATLAB.
- Otra posible mejora es la migración de plataforma de desarrollo, es decir, cambiar el sistema operativo e implementar el receptor en Android, lo cual permitiría una portabilidad mucho mayor del receptor, en un celular.

13. Referencias

Ref. 1.1: Cálculo del espectro de señales PAM. Apunte de la cátedra de Teoría de las Comunicaciones, FI-UNLP.

Ref. 2.1: Hoja de datos Elonics E4000
http://www.superkuh.com/gnuradio/Elonics-E4000-Low-Power-CMOS-Multi-Band-Tunner-Datasheet.pdf

Ref 2.2: Hoja de datos R820T
http://superkuh.com/gnuradio/R820T_datasheet-Non_R-20111130_unlocked.pdf

Ref. 4.1: Fundamentals of Statistical Signal Processing, Volume II: Detection Theory Pag[261-272]. ISBN-13: 007-6092032243

Ref. 5.2: Steven M. Kay. Fundamentals of Statistical Signal Processing: Estimation Theory. Pag[193-195]. ISBN-13: 978-0133457117

Ref 11.1: Principles Of Communications: Systems, Modulation, and Noise. Pag[350-364] ISBN 0-471-39253-7

14. Anexos

Anexo 1: Programa en MATLAB utilizado para simular la detección.

```
% parámetros
pfa=10^-5;
SNR=-2;
Amp=sqrt(2*10^(SNR/10));   % Amplitud del seno en función de la SNR
potruido=1;
largo=2048;   % Largo de la FFT
L=100*largo;  % Largo de la simulación
t=1:1:L;
fs=45.7e3;    % Frecuencia de la señal
fm=256000;    % Frecuencia de muestreo

% umbral
um=(sqrt(potruido))*(-log(pfa));
um(1:1:floor(largo/2))=um;

% señal simulada
tx=Amp*sin(2.*pi.*(fs/fm).*t);
nx=potruido*randn(1,L);
rx=tx+nx;

for i=1:floor(L/largo)

    %Modulo cuadrado de la FFT
    rxx=rx(((i-1)*largo)+1:1:(i*largo));
    X=((abs(fft(rxx,largo))).^2)./largo;
    %Grafico
    feje=(0:1:(floor(largo/2)-1)).*(fm/largo);
    plot(feje,abs(X(1:floor(largo/2))),'b',feje,um,'r'),um;
    xlabel('Frecuencia (Hz)'); ylabel('|Y(f)|'); title('FFT al cuadrado
    de la señal de entrada')
    axis([0 feje(largo/2) 0 80]);
    pause(0.0001);

    %Detección del máximo
    X=X(1:1:floor(length(X)/2));
    [A,n]=max(X);

end
```

Anexo 2: Programa en MATLAB utilizado para simular el error cometido por el estimador de amplitud para distintas cantidades de términos sumados a cada lado, sin presencia de ruido.

```
% Parámetros
Amp=100;

fm=1024;
fs=256.5;
largo=1024;

% Generación de la señal
t=1:1:largo;
gx=Amp.*sin(2.*pi.*(fs/fm).*t);

% Definición del largo de los vectores
Aest=zeros(40,1);
Err=zeros(40);

X=((abs(fft(gx,largo))).^2)/largo;

for prec=1:40

    [A,n]=max(X);

    % Algoritmo de precisión de estimación
    Xe=0;
    for j=1:(2*prec)+1
        Xe=Xe+X(n+j-prec-1);
    end

    % Algoritmo de estimación de amplitud
    Aest(prec)=mean((2/length(gx))*sqrt(Xe*length(gx)));
    Err(prec)=(abs(Amp-Aest(prec)))*100/Amp;

end
bar(Err,20); xlabel('Términos sumados'); ylabel('Error'); title('Error porcentual
Estimador')
```

Anexo 3: Programa en MATLAB utilizado para simular el error cometido cuando la señal recibida cae dentro de distintos valores dentro del bin.

```
% Parámetros
Amp=100;
fm=1024;
largo=1024;
prec =10;
t=1:1:largo;

% Definición del largo de los vectores
Aest=zeros(20,1);
Err=zeros(20);

for i=1:20
    %generación de la señal
    fs=256+((i-1)/20);
    gx=Amp.*sin(2.*pi.*(fs/fm).*t);

    X=((abs(fft(gx,largo))).^2)/largo;
    [A,n]=max(X);

    % Algoritmo de precisión de estimación
    Xe=0;
    for j=1:(2*prec)+1
        Xe=Xe+X(n+j-prec-1);
    end

    % Algoritmo de estimación de amplitud
    Aest(i)=mean((2/length(gx))*sqrt(Xe*length(gx)));
    Err(i)=(abs(Amp-Aest(i)))*100/Amp;

end
bar(Err,20); xlabel('Términos sumados'); ylabel('Error'); title('Error porcentual
Estimador')
axis([0 20 0 1]);
```

Anexo 4: Programa en MATLAB utilizado para simular el lazo de enganche de de fase.

```
% Constantes
kdf=0.5;
knco=31601;
k1=0.01125;
k2=-0.01125;
tm=1/256000;
primera_ejecucion=0;
fnco=64010;

% Relación señal a ruido
SNR=10;
Amp=sqrt(2*10^(SNR/10));

%Respuesta del Filtro adaptado
FA = 2*1.1*[ones(1,320) -ones(1,320)];

% Generación de la señal
t=1:1:40960;
rx=Amp.*sin(2.*pi.*(64000*tm).*t)+randn(1,40960);
rx=rx/Amp;

for q=1:10
    x=[0 rx(((q-1)*4096)+1:q*4096)];

    % Valores de inicialización
    if (primera_ejecucion==0)
        primera_ejecucion=1;
        Vnco=zeros(length(x),1);
        Vd=zeros(length(x),1);
        Vc=zeros(length(x),1);
        fi=zeros(length(x),1);
        zi=zeros(1,649);  %inicialización del filtro adaptado
    end

    % Algoritmo
    for n=2:length(x)

        fi(n)=fi(n-1)+(2*pi*fnco+knco*Vc(n-1))*tm;
        Vnco(n)=sin(fi(n));
        Vd(n)=x(n)*Vnco(n)*kdf;
        Vc(n)=Vc(n-1)+k1*Vd(n)+k2*Vd(n-1);

    end
    Vd(1)=Vd(length(x));
    Vc(1)=Vc(length(x));
    fi(1)=fi(length(x));
    [aux,zi]=filter(FA,1,Vd(2:4097),zi);
    errf(((q-1)*4096)+1:q*4096)=aux;
end

eje=t.*160e-3./length(t);
plot(eje,errf,'b')
title(['Error de fase con SNR = ',num2str(SNR),' dB'])
xlabel('Tiempo [segundos]')
ylabel('Amplitud')
```

Anexo 5: Programa en MATLAB utilizado para simular el PLL en conjunto con la detección y la estimación. En el Anexo 6 se muestra la función "PLL_FILTRO_ADAPTADO"

```
% Constantes
pfa=10^-5;
fm=256000;
tm=1/fm;
i=0; %contador de detecciones
prec=5; %Solo para la prueba de PLL
Vdi=0;
Vci=0;
fii=0;
zi=zeros(1,649);

% Relación señal a ruido
SNR=20;
Amp=sqrt(2*10^(SNR/10));

% Generación de la señal
t=1:1:40960;
tita=(2*pi)*rand();
rx=Amp.*sin(2.*pi.*(64000*tm).*t)+randn(1,40960);

%umbral
um=-log(pfa);

for q=1:10
    x=rx(((q-1)*4096)+1:q*4096);

    X=((abs(fft(x))).^2)/length(x);
    X(1)=0;
    X=X(1:1:floor(length(X)/2));
    [A,n]=max(X);

    % Protección contra única detección errónea
    if (A<=um) && (i==1)
        i=0;
    end

    if (A>um) && (i~=2)

        i=i+1; %contador de detecciones
        if (A>um) && (i==2)

            % Algoritmo de precisión de estimación
            Xe=0;
            for j=1:(2*prec)+1
                Xe=Xe+X(n+j-prec-1);
            end

            % Algoritmo de estimación de amplitud
            Aest=(2/length(x))*sqrt(Xe*length(x));

            % Estimación de frecuencia
            fmax=((n-1)/length(x))*fm;
        end
    end
```

```
    if (i==2)
        x=x./Aest; % normalización
        [Vdo,Vco,fio,zo,ysal]=PLL_FILTRO_ADAPTADO(fmax,x,Vdi,Vci,fii,zi);
        Vdi=Vdo;
        Vci=Vco;
        fii=fio;
        zi=zo;
        errf(((q-1)*4096)+1:q*4096)=ysal;
    end

end

eje=t.*160e-3./length(t);
plot(eje,errf,'b')
title(['Error de fase con SNR = ',num2str(SNR),' dB'])
xlabel('Tiempo [segundos]')
ylabel('Amplitud')
```

Anexo 6: Función "PLL_FILTRO_ADAPTADO"

```
function [ Vdo,Vco,fio,zo,y ] = PLL_FILTRO_ADAPTADO(fnco,rx,Vdi,Vci,fii,zi)
% Esta función implementa el algoritmo de seguimiento de fase mediante un
% PLL y el filtro adaptado a la salida del mismo
% ARGUMENTOS DE ENTRADA:
% fnco: valor estimado de la frecuencia
% primera_ejecucion: valor binario, indica si es la primera vez que se va a
%                    ejecutar el algoritmo
% Vdi: condición inicial de Vd
% Vci: condición inicial de Vc
% fii: condición inicial de fi
% zi: condición inicial del filtro adaptado
% rx: muestras de entrada previamente normalizadas con la estimación de la
% amplitud
%
% cuando se ejecuta por primera vez esta función o cuando las condiciones
% iníciales del filtro adaptado tienen que ser nulas se debe ejecutar
% zi=zeros(1,649); para hacerlas nulas.
%
% ARGUMENTOS DE SALIDA
% y: vector de muestras de salida
% Vdo: condición inicial de Vd a ingresar en el siguiente ciclo
% Vco: condición inicial de Vc a ingresar en el siguiente ciclo
% fio: condición inicial de fi a ingresar en el siguiente ciclo
% zo: condición inicial del filtro adaptado a ingresar en el siguiente ciclo
%
% Constantes
kdf=0.5;
knco=31601;
k1=0.01125;
k2=-0.01125;
tm=1/256000;
FA = 2*1.1*[ones(1,320) -ones(1,320)]; %Respuesta del Filtro adaptado

% especificación de tamaño para velocidad
x=[0 rx];
Vnco=zeros(length(x),1);
Vd=zeros(length(x),1);
Vc=zeros(length(x),1);
fi=zeros(length(x),1);

% Valores de inicialización
Vd(1)=Vdi;
Vc(1)=Vci;
fi(1)=fii;

% Algoritmo
for n=2:length(x)

    fi(n)=fi(n-1)+(2*pi*fnco+knco*Vc(n-1))*tm;
    Vnco(n)=sin(fi(n));
    Vd(n)=x(n)*Vnco(n)*kdf;
    Vc(n)=Vc(n-1)+k1*Vd(n)+k2*Vd(n-1);

end
Vdo=Vd(length(x));
Vco=Vc(length(x));
fio=fi(length(x));
[y,zo]=filter(FA,1,Vd(2:length(Vd)),zi);
end
```