ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELECTRÍCA Y ELECTRÓNICA

ESTUDIO DE LOS EFECTOS REALES DEL CANAL INALÁMBRICO POR MEDIO DE LA TECNOLOGÍA DE RADIO DEFINIDO POR SOFTWARE Y TÉCNICAS PARA SU COMPENSACIÓN.

IMPLEMENTACIÓN DE UN ALGORITMO BASADO EN SECUENCIAS CORTAS Y LARGAS DE ENTRENAMIENTO EMPLEADA EN EL ESTÁNDAR 802.11A PARA COMPENSACIÓN DE LOS EFECTOS DEL CANAL INALÁMBRICO

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERA EN TELECOMUNICACIONES

SILVIA ALEJANDRA VEINTIMILLA MENA

silvia.veintimilla@epn.edu.ec

DIRECTOR: DR. ROBIN ÁLVAREZ

robin.alvarez@epn.edu.ec

DMQ, febrero 2022

CERTIFICACIONES

Yo, Silvia Alejandra Veintimilla Mena declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

ALEJANDRA VEINTIMILLA

Certifico que el presente trabajo de integración curricular fue desarrollado por Silvia Alejandra Veintimilla Mena, bajo mi supervisión.

ROBIN ÁLVAREZ

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

ALEJANDRA VEINTIMILLA

ROBIN ÁLVAREZ

DEDICATORIA

Este trabajo lo dedico a mi familia y amigos que han sido un pilar fundamental en mi vida, y que sin su constante apoyo, esto no sería posible.

AGRADECIMIENTO

A mi madre, Silvia, cuyo amor y constantes lecciones, me ha guiado a lo largo de mi vida para llegar a ser la persona que soy. A su carácter y actitud perseverante, que me ha demostrado que todo es posible con dedicación y esfuerzo. A sus ocurrencias, que me han enseñado diferentes formas de ver la realidad y a disfrutar de los pequeños detalles.

A mi padre, Fabián, que, a pesar de la distancia, me ha demostrado su amor incondicional.

A mis hermanas, Verónica y Carol, que han sido mis mejores amigas, con las cuales compartimos una conexión especial, y que son mi adoración.

A mis amigos, Ricardo y Gabriel, que, pese a no compartir vínculos sanguíneos, son la familia que he elegido, y que puedo contar con su apoyo en cualquier parte del planeta.

A mis tíos y mi abuelita, que me han brindado el calor de un hogar y han estado pendientes de mis necesidades.

A mi tutor, que ha sabido motivarme, escucharme y guiarme con paciencia a lo largo de este camino.

ÍNDICE DE CONTENIDO

RESUMEN

El presente trabajo tiene como objetivo el estudio, comprensión y cálculo de los parámetros: desplazamiento de frecuencia portadora (CFO - Carrier Frequency Offset), sincronización de símbolos (STO - Symbol Timing Offset) y desplazamiento de fase de los sistemas de comunicación inalámbrica.

El primer capítulo presenta una descripción de los parámetros a estudiar además de una comparación con otros trabajos de investigación, con sistemas de comunicación inalámbricos similares, que utilizan diferentes equipos de transmisión y recepción, y, asimismo, manejan otras técnicas de estimación de dichos parámetros. También se detalla las características de los equipos de Radio Definida por Software (Software Defined Radio - SDR) a utilizar en este trabajo para el funcionamiento del prototipo de comunicaciones desarrollado en un trabajo anterior y que se lo empleará aquí.

En el segundo capítulo, se desarrollará métodos de estimación de valores reales de los parámetros mencionados dentro del prototipo de comunicación que emplea multiplexación por división de frecuencia ortogonal (OFDM), donde se expondrá un algoritmo para la estimación del CFO con base en la utilización del espectro de la señal enviada y de la señal recibida, también se realizará la sincronización de símbolos y una estimación del desplazamiento de fase por medio de un procedimiento basado en secuencias de entrenamiento cortas y largas

Por último, en el tercer capítulo se harán pruebas con diferentes distancias y diferentes tipos de modulaciones para observar la evolución en el tiempo de dichos parámetros y encontrar valores promedio de CFO desde 2.38 kHz con la función de distribución Nakagami como la que más se ajusta y valores promedio desde 87° para el desplazamiento de fase. Finalmente se expondrá conclusiones y recomendaciones a partir de los resultados obtenidos.

PALABRAS CLAVE: desplazamiento de frecuencia portadora, CFO, sincronización de símbolo, STO, desplazamiento de fase, OFDM, RDS.

ABSTRACT

The objective of this work is to study, understand and calculate the parameters: carrier frequency offset (CFO), symbol timing offset (STO) and phase shift of wireless communication systems.

The first chapter presents a description of the parameters to be studied as well as a comparison with other research works, with similar wireless communication systems, which use different transmission and reception equipment, and, in addition, handle other estimation techniques for these parameters. It also details the characteristics of the Software Defined Radio (SDR) equipment to be used in this work for the operation of the communications prototype developed in a previous work and which will be used here.

In the second chapter, methods for estimating real values of the parameters included within the communication prototype that uses orthogonal frequency division multiplexing (OFDM) will be developed, where an algorithm for estimating the CFO based on the use of the spectrum of the signal sent and the signal received, the acoustic signal of symbols and an estimation of the phase shift will also be performed by means of a procedure based on short and long training sequences.

Finally, in the third chapter, tests will be performed with different distances and different types of modulations to observe the time evolution of these parameters and find average CFO values from 2.38 kHz with the Nakagami distribution function as the best fit and average values from 87° for the phase shift. Finally, conclusions and recommendations will be presented based on the results obtained.

KEYWORDS: Carrier frequency offset, CFO, symbol timing offset, STO, phase shift, OFDM, SDR.

1. INTRODUCCIÓN

Los sistemas de comunicaciones basados en multiplexación por división de frecuencia ortogonal (OFDM) son sensibles a errores de frecuencia y sincronización, en particular, al desplazamiento de frecuencia portadora(Carrier Frequency Offset - CFO), sincronización de símbolos (Symbol Timming Offset - STO) y desplazamiento de fase, que son provocados por la falta de coincidencia del oscilador entre el transmisor y el receptor, el efecto Doppler, la propagación de trayecto múltiples, entre otros.

Por otro lado, los sistemas de comunicación actuales han evolucionado con la adopción de nuevas tecnologías que pueden mejorar su rendimiento de forma innovadora. Mientras tanto, la cuestión de la compensación de frecuencia de la portadora (CFO) sigue siendo importante en estos sistemas de comunicación actuales ya que las distorsiones de fase de los símbolos recibidos empeoran con el paso del tiempo, por lo que una pequeña cantidad de desplazamiento de frecuencia puede incrementar la tasa de errores y disminuir la confiabilidad de todo el proceso de recepción. [1][2]

El preámbulo 802.11a cuenta con una secuencia corta que posee 10 símbolos de entrenamiento cortos y con una secuencia larga que posee 2 símbolos de entrenamiento más un prefijo cíclico que sirve para estimar la detección de una trama OFDM, el CFO, el STO y el desplazamiento de fase. Para ello, el CFO puede ser estimado mediante un procedimiento basado en la correlación cruzada de las secuencias cortas y largas. Para la sincronización de símbolos, cuya finalidad es encontrar el punto de origen de una trama OFDM, se lo puede hacer utilizando secuencias de entrenamiento largas integradas en la señal enviada.

Finalmente, el desplazamiento de fase, que produce una rotación de los estados de las constelaciones, se puede estimar en base a los ángulos que forman cero grados entre la parte real e imaginaria de las secuencias de entrenamiento larga.

1.1. OBJETIVO GENERAL

Comprender el fundamento teórico del desplazamiento de frecuencia portadora (CFO), temporización de símbolo (STO) y corrección de fase de un sistema de comunicación OFDM a través de dispositivos SDR Adalm Pluto, permitiendo encontrar sus valores experimentales para obtener una mejor fiabilidad en la transmisión de información.

1.2. OBJETIVOS ESPECÍFICOS

1.2.1. Comprender el fundamento teórico de los fenómenos de desplazamiento de frecuencia portadora (CFO), temporización de símbolo (STO) y desplazamiento de fase, en un sistema de comunicación basado en OFDM.

1.2.2. Analizar y comparar los métodos de medición o estimación de los parámetros: desplazamiento de frecuencia portadora (CFO), sincronización de símbolo (STO) y desplazamiento de fase.

1.2.3. Realizar mediciones experimentales de los parámetros CFO y desplazamiento de fase con respecto a diferentes tipos de modulaciones y diferentes distancias.

1.3. ALCANCE

Este estudio técnico implementa un sistema de comunicación inalámbrico, utilizando un transmisor y receptor basados en tecnología SDR Adalm Pluto.

Para la configuración de los equipos SDR del sistema de comunicación se emplean las librerías de Matlab “Communications Toolbox”, “DSP System Toolbox”, “Signal Processing Toolbox” y “ADALM-PLUTO Radio Support from Communications”. Además, se establecen los parámetros de funcionamiento como la frecuencia de operación, frecuencia de muestreo, ganancias (transmisión y recepción), entre otros, en base a las especificaciones técnicas de los equipos. [3]

El sistema será implementado con un solo computador portátil al cual se conectan los dispositivos SDR Adalm Pluto en un puerto USB cada uno, a la vez que se conectará un cable de 9 metros tanto en la antena de transmisión como en la antena de recepción de los equipos para el envío y recepción de datos, por lo que la distancia máxima de separación entre transmisor y receptor será de 18 metros. Por otro lado, los datos a ser enviados constarán únicamente de información de texto que contiene 1454 caracteres alfanuméricos en formato ASCII, suficientes para apreciar los errores cometidos.

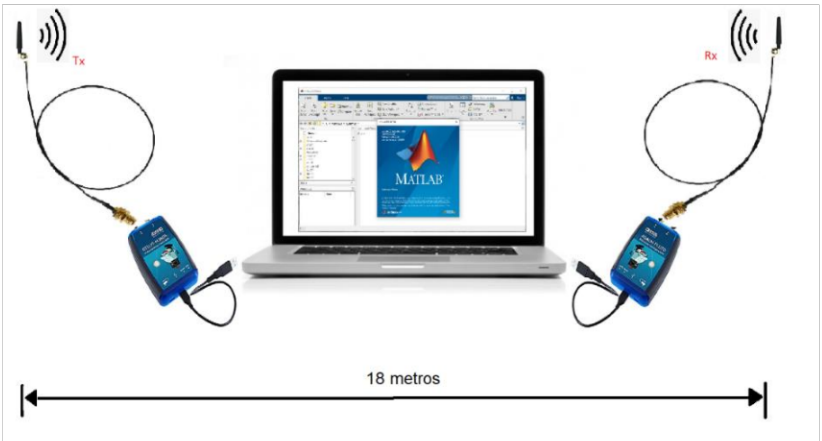


Figura 1. Esquema de hardware

1.4. MARCO TEÓRICO

1.4.1. ESTADO DEL ARTE

En el estudio “Physical Layer Authentication for Mobile Systems with Time-Varying Carrier Frequency Offsets” se desarrolló una plataforma de radio definida por software (SDR) basada en Universal Software Radio Peripheral (USRP1) para capturar los datos reales del CFO. El sistema implementado comprende dos transmisores y un receptor que operan a una frecuencia portadora de 2,47 GHz y se obtuvieron alrededor de 2 × 104 muestras de las estimaciones de CFO basadas en autocorrelación en el entorno inalámbrico real utilizando la señal de banda base de la plataforma SDR. La plataforma de hardware implementada muestra el CFO estimado en entornos inalámbricos interiores estáticos sin considerar los cambios Doppler. [4]

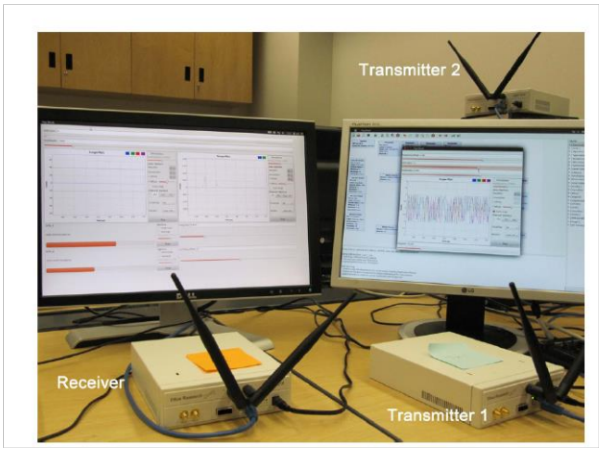


Figura 2. Plataforma de hardware implementada [4].

Con una frecuencia de muestreo de 20MHz y diferentes tipos de SNR, se obtienen los siguientes resultados de la estimación del CFO usando el algoritmo de Kalman para el sistema de comunicación inalámbrico.

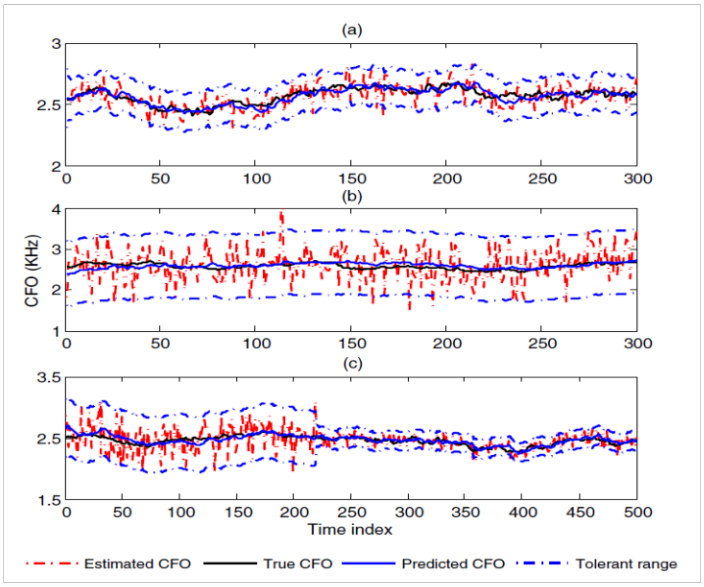


Figura 3. CFO predichos usando Kalman a)SNR= 30dB b) SNR= 15 dB c) SNR cambia desde SNR=20dB a SNR= 30dB en la trama número 220 [4].

En la Figura 3 se puede apreciar que mientras el CFO real y el CFO predicho para los tres casos permanece casi constante a 2.5 kHz, el CFO estimado varía en cada caso desde valores como 1.5 kHz hasta valores de 3 kHz aproximadamente. Para ello, se utilizó un sistema donde primero se estima el CFO con un modelo de transmisión basado en la secuencia de entrenamiento como el descrito a continuación:

$$y_m[n] = e^{j2\pi\varepsilon[m]n}u_m[n] + w_m[n], \qquad 0 \le n \le N_t - 1 \tag{1}$$

donde, $y_m[n]$ denota la señal de banda base recibida para el m-ésimo marco, $u_m[n]$ denota la secuencia de entrenamiento replicada de longitud N_t y $w_m[n]$ denota el ruido gaussiano blanco aditivo de media cero con varianza σ_n^2. Y luego se procede con el filtrado de Kalman para obtener los valores de CFO predicho, donde se rastrea la variación de CFO cuadro por cuadro para detectar cualquier variación atípica del mismo.

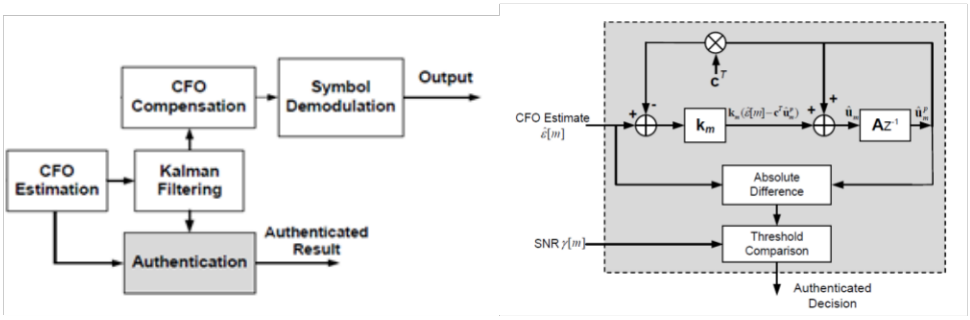


Figura 4. Diagrama de bloques del sistema propuesto, basado en la predicción de CFO del filtrado de Kalman

En el estudio "The Phase Noise and Clock Synchronous Carrier Frequency Offset based RF Fingerprinting for the Fake Base Station Detection" se desarrolla un entorno de laboratorio mediante el uso de diferentes plataformas SDR como USRP-B210, USRP-N210 y Adalm Pluto que imitan como una estación base falsa (FBS) de gamas alta, media y baja respectivamente y una fuente de reloj R&S externa para emular una red celular con estaciones de base regulares (RBS).[8]

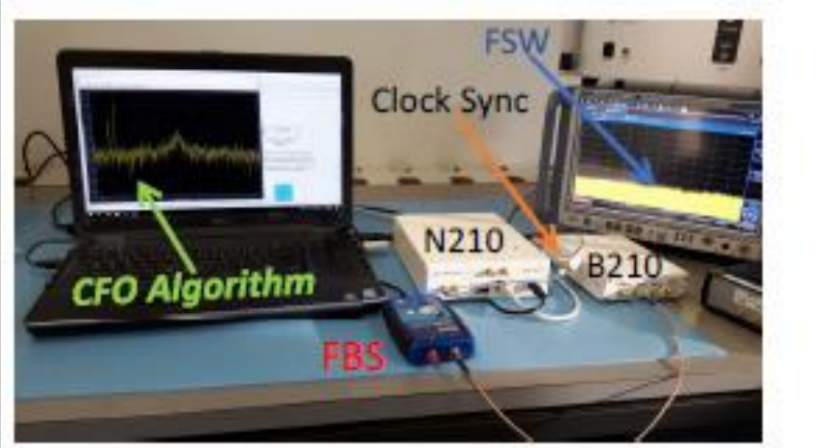


Figura 5. Plataforma de hardware desarrollada con 3 dispositivos SDR [8]

La frecuencia portadora utilizada es de 800 MHz y se ha obtenido los valores de CFO para los dispositivos SDR, como se muestra en la Figura 10.

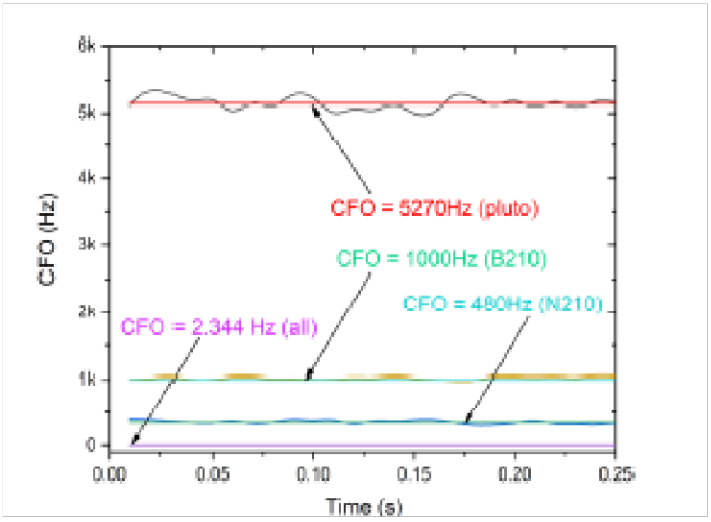


Figura 6. CFO cuando todos los SDR actúan como FBS en el dominio de reloj individual [8].

Obteniendo como resultado para el dispositivo SDR Adalm Pluto un valor promedio de CFO de 5270 Hz, mientras que para los dispositivos USRP-B210 y N210 valores promedio de CFO de 1000 Hz y 480 Hz respectivamente, dando un total entre los tres equipos de un CFO de 2344 Hz . Estos valores fueron determinados por el método de CFO aproximado

que utiliza la envolvente compleja $r(n)$ de la forma de onda de entrada, cuya expresión matemática es la siguiente:

$$r(n) = |r(n)|e^{j\theta_{r(n)}} \tag{2}$$

Luego la estimación aproximada del CFO a la frecuencia de muestreo f_s con desplazamiento de fase $\Delta\theta_r$ viene dado por:

$$\widehat{\Delta f} = \frac{\Delta\theta_r}{2\pi} f_s \tag{3}$$

1.4.2. PREÁMBULO 802.11A

El preámbulo del estándar IEEE 802.11a se compone de dos partes, denominadas parte corta LSTF (Legacy Short Training Field) y parte larga LLTF (Legacy Long Training Field) [3].

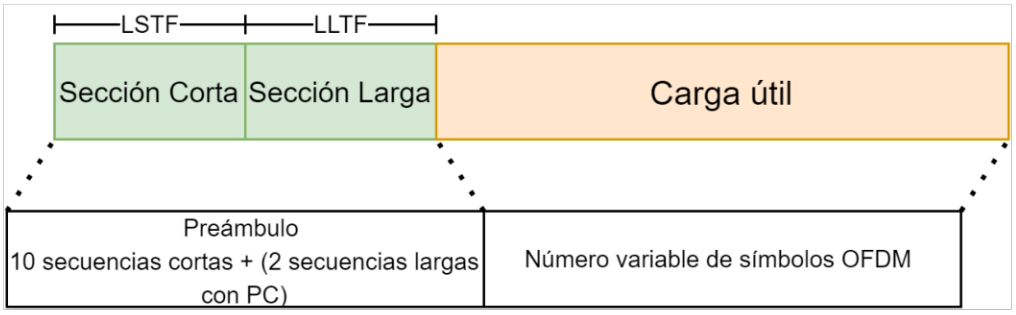


Figura 7. Preámbulo del estándar 802.11a [3]

La secuencia corta está formada por 16 muestras, las cuales se repiten 10 veces y la secuencia larga se compone de un prefijo cíclico (CP) de 32 muestras seguido de dos símbolos de entrenamiento largos idénticos (C1 y C2), cada uno de 64 muestras. El CP consiste en la mitad del segundo símbolo de entrenamiento largo [3][17].

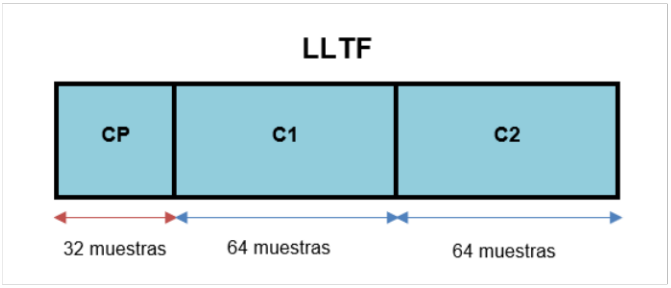


Figura 8. Composición del LLTF

1.4.3. DIAGRAMA DE BLOQUES

El esquema de transmisión utilizado con los dispositivos SDR Adalm Pluto se muestra en el siguiente diagrama de bloques de la Figura 9, donde se utilizará un script general que contiene cada etapa.

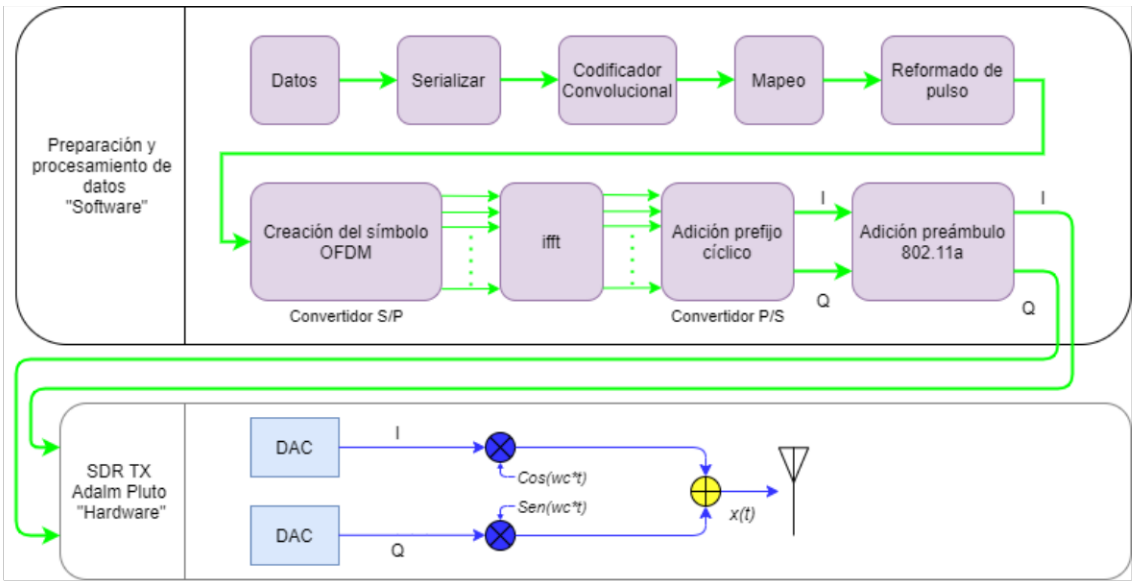


Figura 9. Etapa de transmisión [3]

En la etapa Datos se selecciona un archivo de texto denominado “ArchivoTextoRT”(VER ANEXO II) que contiene 1454 caracteres alfanuméricos, siguiendo con la etapa Serializar que convierte dichos caracteres en una secuencia de bits serializados para pasar por la etapa Codificador Convolucional donde se implementa un código que corrige errores a nivel de bit con el codificador de trellis. En la etapa de Mapeo, se toma los datos de la etapa anterior y se los implementa en diferentes tipos de modulaciones (BPSK, QPSK, 16-QAM, 64-QAM), para pasar a la siguiente etapa Reformado de Pulso que aplica un filtro a cada pulso de la señal con el fin de contrarrestar la interferencia entre símbolos (ISI -Inter-Symbol-Interference). La etapa Creación de símbolo OFDM permite dividir el flujo de datos de la entrada en múltiples subportadoras para evitar alteraciones del canal inalámbrico. La etapa IFFT se encarga de modular los símbolos OFM creados. La etapa Adición prefijo cíclico se encarga de copiar las últimas muestras de cada símbolo al inicio de estos. Finalmente, la etapa Adición preámbulo 802.11a permite al receptor detectar la trama OFDM transmitida, estimar el CFO, identificar el inicio exacto del paquete OFDM y corregir el paquete OFDM en fase.

El diagrama de bloques de recepción, Figura 10, las etapas en las que se centrará este trabajo son las siguientes: CFO y Corrección del CFO, donde se desarrollará un método

para estimar el valor real que el sistema de comunicaciones produce al enviar la señal por el canal inalámbrico; Sincronización de símbolo, donde se determinará el punto de inicio del símbolo OFDM; y Corrección de fase, donde se calculará el desfase promedio en grados de cada muestra.

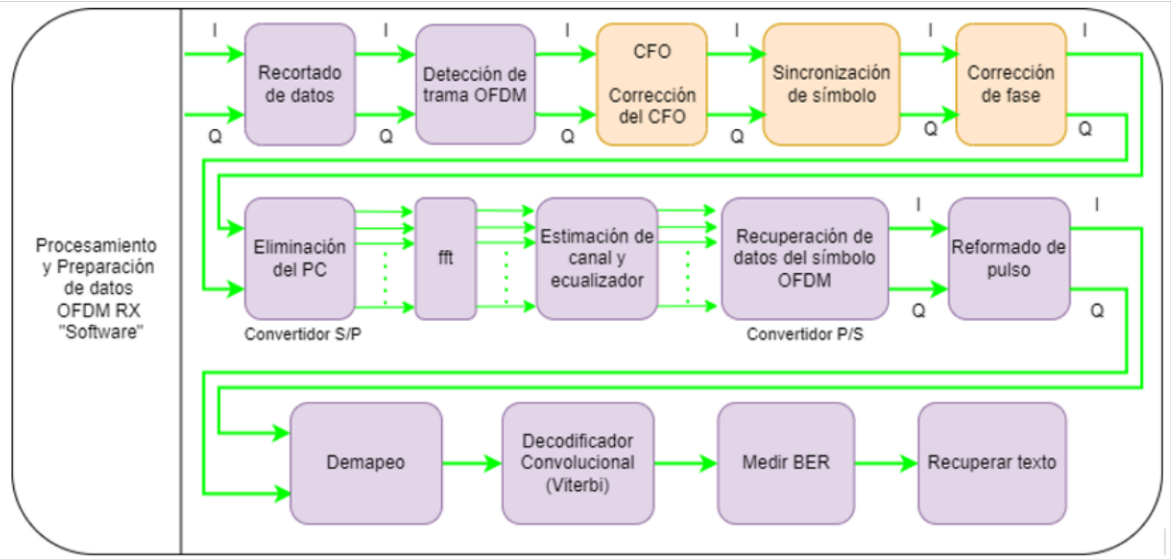


Figura 10. Etapa de recepción [3]

1.4.4. ESTIMACIÓN DEL CFO

1.4.4.1. FUNDAMENTO TEÓRICO

En los sistemas de comunicación inalámbricos prácticos, el CFO es un efecto combinado de la discordancia del oscilador de RF y el cambio de frecuencia Doppler. En particular, el CFO provocado por el desajuste del oscilador entre el transmisor y el receptor puede considerarse como una constante, esto se debe a que el desajuste del oscilador causado por la temperatura, el voltaje de suministro y los factores de envejecimiento, cambia muy lentamente generalmente en la escala de tiempo de horas o días. Por el contrario, el CFO causado por el efecto Doppler inducido por la movilidad puede cambiar más rápidamente durante la transmisión [4].

Específicamente, el CFO se puede dividir en dos partes: un componente constante causado por el desajuste del oscilador y un componente variable debido al desplazamiento Doppler inducido por la movilidad. Incluso si se pretende generar exactamente las mismas frecuencias portadoras en el transmisor y el receptor, puede haber una diferencia inevitable entre ellos debido a las características únicas de los osciladores. Siendo f_{tx} y f_{rx} las frecuencias portadoras del transmisor y el receptor, respectivamente, se puede considerar su “offset” como la diferencia entre las frecuencias, como se muestra en la Figura 11.

$$CFO = f_{tx} - f_{rx} \quad (4)$$

Mientras tanto, la frecuencia Doppler f_d está determinada por la frecuencia portadora f_{tx} y la velocidad v del receptor, como:

$$f_d = \frac{v \cdot f_{tx}}{c} \quad (5)$$

donde c es la velocidad de la luz.

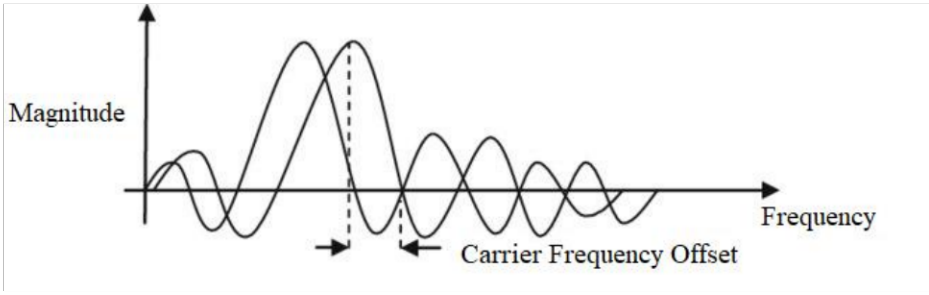


Figura 11. Carrier Frequency Offset (CFO) [6]

1.4.4.2. INTRODUCCIÓN MANUAL DE UN CFO

Para simular un desplazamiento de frecuencia dentro del prototipo de sistema de comunicación, se puede utilizar el objeto de comunicación de Matlab denominado **comm.PhaseFrequencyOffset** que permite añadir o compensar tanto frecuencia como fase a una señal de entrada.

La sintaxis del objeto de comunicación de Matlab es la siguiente:

```
H = comm.PhaseFrequencyOffset(Name1,Value1, Name2, Value2, …, NameN, ValueN)
```

donde se debe usar como primer argumento la propiedad “FrequencyOffset” para añadir un desplazamiento de frecuencia, y como segundo argumento la propiedad “SampleRate” que determina la frecuencia de muestreo a ser utilizada.

1.4.4.3. ESTIMACIÓN DE CFO MEDIANTE EL PREÁMBULO 802.11A

En la tesis “Desarrollo de un prototipo de comunicación inalámbrica de uso didáctico utilizando equipos SDR, incluyendo la técnica OFDM y el servicio de confidencialidad” (Hiidalgo & Gordon, 2020) se realiza la estimación de CFO por medio de las secuencias cortas del preámbulo 802.11a. Para ello, se usa el objeto de comunicación **comm.PhaseFrequencyOffset** que introduce un desplazamiento de frecuencia de 30 Hz de la siguiente manera:

```
desplazamientoFrecuencia = comm.PhaseFrequencyOffset('SampleRate',Fs,...
'FrequencyOffset',30);
```

y luego por medio del método de Schmidl and Cox que utiliza la autocorrelación de las secuencias cortas en base a una métrica M en función de su índice k, realiza la estimación para encontrar el desplazamiento que se acaba de añadir, como se muestra a continuación:

```
r = datosOFDM; % Señal recibibida para estimar el CFO
L=16; % Longitud de una secuencia corta
m=L-1; % Máximo valor de m
disp('****************************************')
% Se recorre diferentes valores de N, este valor es múltiplo de L y
% especifica el máximo valor que va a tomar k
for N=16:16:128
    fEst = zeros(N,1); % Iniciar vector que almacena el desplazamiento
    for k=1:N
        p = r(k:k+m)'*r(k+L:k+m+L);
        fEst(k) = (Fs/(2*pi*L))*(angle(p));
    end
    % Obtenemos el promedio del CFO estimado
    estimacionFrecuencia=-mean(fEst);
```

Donde en la variable estimacionFrecuencia se guarda el valor de CFO estimado, que es de 30. 25 Hz, similar al desplazamiento de frecuencia añadido previamente.

1.4.4.4. ESTIMACIÓN DEL CFO MEDIANTE CÁLCULO DEL ESPECTRO.

En este trabajo, se realiza la estimación del desplazamiento en frecuencia por medio del cálculo del espectro de la señal enviada y recibida. Para eso, se obtienen los espectros de una muestra de la señal transmitida y la señal recibida por medio de una función de Matlab. Luego, se recolectan todos los picos de las señales y se identifica el pico máximo de la señal recibida. Como el espectro de las dos señales es el mismo, solo desplazado en frecuencia cierta cantidad, al medir el pico máximo de la señal recibida y buscar el pico más cercano a dicho pico en la señal transmitida, se puede usar la Ecuación 4 para restar las frecuencias de ambos picos de la señal y encontrar el CFO estimado de la señal.

A continuación se muestra un código en el script cfo_estimation.m para la estimación de CFO de una muestra de una señal enviada y una señal recibida.

```
%Frecuencia de muestreo
frecuenciaMuestreo=2.0e6;

%Recolección de una muestra de la señal recibida
condicion=1;

disp('Empezando a sondear el canal inalámbrico')
disp(i)
while(condicion==1)
    datosRecibi2=rxAdalm();
    max(abs(datosRecibi2));
    if(max(abs(datosRecibi2))> umbralrx)

        break
```

```
    end

end

load("datosEnviados.mat",'datosEnviados') % Se carga los datos
transmitidos

%Procesamiento de datos
datosRecibi2=double(datosRecibi2); % Conversión a una variable compleja
% doble de los datos recolectados
datosEnviados1=double(datosEnviados);% Conversión a una variable
compleja
% doble de los datos transmitidos


%Cálculo de los espetros de las señales.
[ptx,ftx]=pspectrum(datosEnviados1,frecuenciaMuestreo); %Espectro de la
%señal enviada con sus respectivas frecuencias
[prx,frx]=pspectrum(datosRecibi2,frecuenciaMuestreo);%Espectro de la
%señal recibida con sus respectivas frecuencias

%Obtención de los picos y sus locaciones de las señales.
[picostx,locstx] = findpeaks(ptx);% Localización de los picos del
espectro
% de la señal transmitida y sus índices.
[picosrx,locsrx] = findpeaks(prx);% Localización de los picos del
espectro
% de la señal transmitida y sus índices.

%Pico máximo de la señal recibida y su índice
[picomaxrx,Irx]=max(picosrx);

%Localización del pico más cercano de la señal transmitida, con
%respecto al pico máximo de la señal recibida
l=find(locstx<Irx);
Itx=locstx(l(end));% Se almacena el índice del pico más cercano de
    % la señal transmitida, con respecto al pico máximo de la señal
    % recibida.

%Frecuencias de los picos de la señal transmitida y señal recibida.
Frecrx=frx(Irx);%Frecuencia señal recibida
Frectx=ftx(Itx);%Frecuencia señal transmitida

%Estimación del CFO
cfo=Frecrx-Frectx;

%Gráfica
figure()
stem(cfo)
title('Estimación del CFO')
xlabel('Muestras')
ylabel('Frecuencia [Hz]')
```

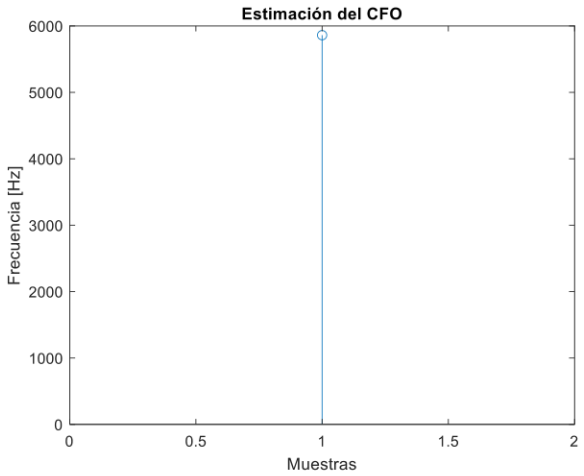


Figura 11. Estimación del CFO para una muestra de una señal enviada y una señal recibida.

1.4.5. ESTIMACIÓN DEL STO

1.4.5.1. FUNDAMENTO TEÓRICO

Una vez realizado la detección de paquetes, por medio de la autocorrelación de las secuencias cortas como se utiliza en el método de Schmild y Cox [3] y la corrección del desplazamiento frecuencia portadora (CFO), se puede realizar la sincronización de símbolos (STO) que proporciona una estimación de precisión a nivel de símbolo de las secuencias de preámbulo 802.11a en el dominio del tiempo, por lo que se debe detectar el punto de inicio de cada símbolo OFDM para facilitar la recuperación de las muestras exactas.

Según la ubicación del punto de inicio estimado del símbolo OFDM, el efecto de STO puede ser diferente. En el primer caso, cuando el punto de inicio estimado corresponde con el tiempo exacto del símbolo, se preserva la ortogonalidad entre componentes de frecuencia de la subportadora. Para el segundo caso, cuando el punto de inicio del símbolo OFDM es un poco antes del punto exacto, el n-ésimo símbolo no se sobrepone con el anterior, evitando incurrir en interferencia inter-símbolo (ISI), se preserva la ortogonalidad entre subportadoras, sin embargo, existe un desplazamiento de fase que es proporcional al STO normalizado y al índice de la subportadora. En el tercer caso, cuando el punto de inicio estimado del símbolo OFDM es después del punto exacto, ocurre interferencia entre portadoras (ICI) lo cual destruye la ortogonalidad e incurre a ISI del siguiente símbolo n-ésimo OFDM.[5]

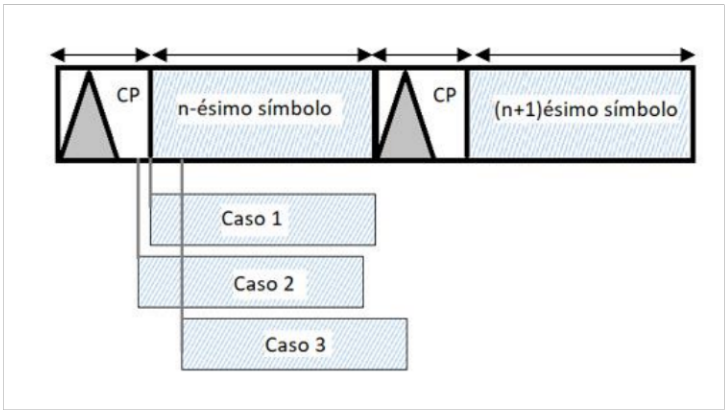


Figura 12. Diferentes casos de punto de inicio del símbolo OFDM sujeto al STO.

1.4.5.2. ESTIMACIÓN DEL STO USANDO EL PREÁMBULO 802.11A

Para estimar el STO se utiliza la correlación cruzada entre el símbolo de entrenamiento y la señal recibida, ya que determina el límite entre las secuencias de entrenamiento largas con precisión de nivel de muestra [5] [7].

La implementación de la correlación necesita solamente la sección larga LLTF y la señal recibida r . Para ello, se tomará las primeras 80 muestras de la sección larga, denominadas d_{LLTF}. La métrica usada c, corresponde a valor absoluto elevado al cuadrado de la correlación cruzada y se aplicará de la siguiente forma:

$$c(k) = \left| \sum_{m=0}^{L-1} d_{LLTF}^{*}(m) r(m+k) \right|^{2} \tag{5}$$

donde L es la longitud de d_{LLTF}, k es el índice de la métrica c, y d^{*} es la conjugación compleja.

1.4.5.3. INTRODUCCIÓN DE UN STO MANUAL

Para estimar el STO, se utilizará la señal enviada y se agregará un relleno de 45 ceros para detectar el punto de inicio de la trama OFDM. Es entonces donde se utiliza la correlación cruzada, usando la función de Matlab *xcorr(r,d),* donde r es la señal con los datos enviados más el relleno de ceros, y d es la secuencia larga LLTF. Como se mencionó en la sección 1.4.5.2, solo se tomará las 80 primeras muestras de la LLTF, y se sacará el valor absoluto, elevado al cuadrado de la correlación cruzada.

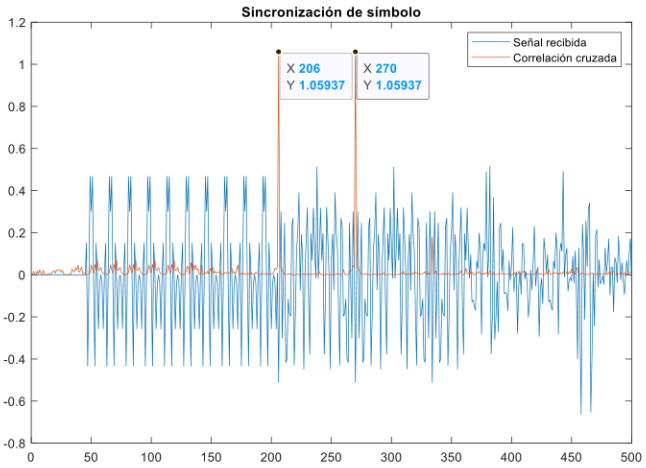


Figura 13. Estimación del STO usando correlación cruzada

Como se observa en la Figura 13, se obtienen dos picos de la correlación cruzada, en las posiciones 270 y 206, si restamos ambas, nos da como resultado 64 lo que determina la longitud de un símbolo de entrenamiento de la secuencia larga, obtenido de las 80 primeras muestras, las cuales corresponden a las 48 primeras muestras de C1 de LLTF y las 32 muestras del CP de LLTF como se muestra en la Figura 14.

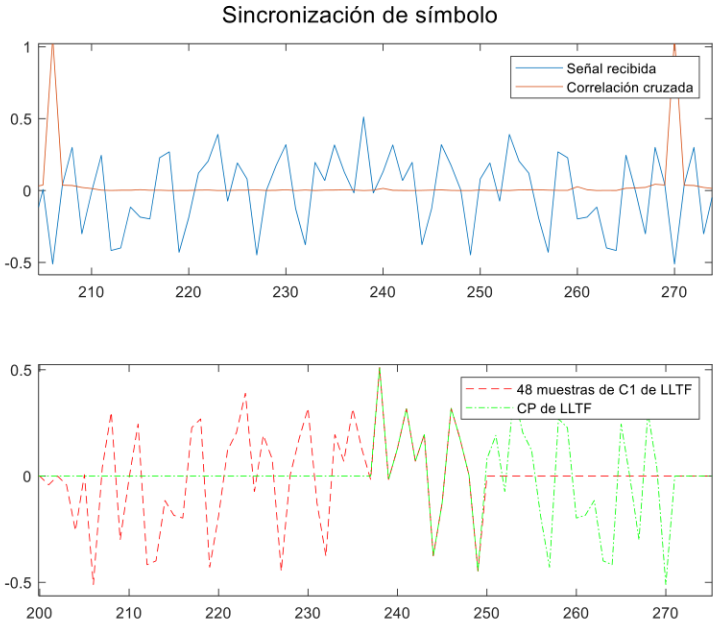


Figura 14. Resultado de la correlación cruzada.

En cambio, si se resta el pico de posición 206 menos la longitud de la secuencia corta (160) que contiene la señal recibida, da como resultado 46. Si se recuerda que se agregó 45 ceros a la señal recibida, esto nos indicaría que el primer símbolo de la sección corta se

encuentra en la posición 46, logrando encontrar el punto de inicio del símbolo OFDM incluido el preámbulo 802.11a.

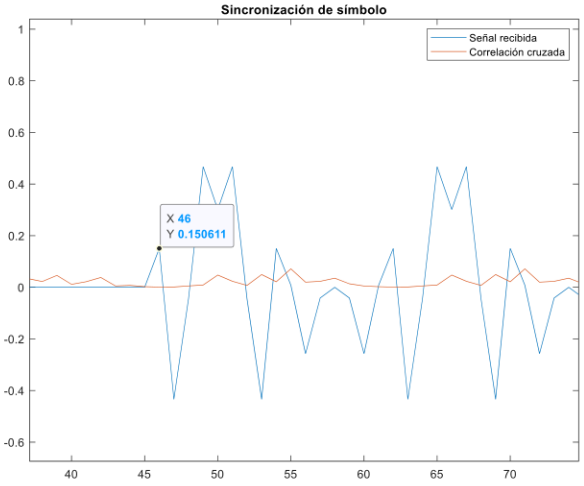


Figura 15. Punto de inicio del símbolo OFDM

Así como en Higaldo & Gordon(2020) se utiliza esta técnica para la estimación de STO, en este trabajo se emplea la misma técnica.

1.4.6. ESTIMACIÓN DEL DESPLAZAMIENTO EN FASE.

1.4.6.1. FUNDAMENTO TEÓRICO

Después de la sincronización de símbolos, puede existir un desplazamiento de fase provocado por el STO, así como el propio canal, para ello, al utilizar un ecualizador en el dominio de la frecuencia, diseñado para compensar el efecto del canal puede absorber esta consecuencia, siempre y cuando la STO sea pequeña, caso contrario, se utiliza la secuencia larga LLTF (Legacy Long Training Field) que posee dos símbolos OFDM completos que se utilizan para estimar el canal y corregir el desplazamiento de fase. Finalmente se implementa luego de la demodulación de datos un ecualizador que elimina cualquier compensación residual de fase o frecuencia que quede en la señal recibida [3][7].

1.4.6.2. ESTIMACIÓN DEL DESPLAZAMIENTO DE FASE USANDO EL PREÁMBULO 802.11A

La secuencia larga del preámbulo 802.11a tiene una componente real y una componente imaginaria, como un símbolo de entrenamiento largo está conformada por 64 muestras, para realizar la estimación de desplazamiento de fase, se toma las posiciones del símbolo de entrenamiento que correspondan a que el ángulo entre la parte real y la parte imaginaria sea igual a 0°. Una vez obtenidos las posiciones, que conforman 31 valores, se mide el

ángulo que existe en ellos al recibir una señal, y se calcula el promedio de los ángulos para obtener un ángulo total de desplazamiento de fase.

En Hidalgo & Gordon (2020), se usa el objeto de comunicación **comm.PhaseFrequencyOffset**, mencionado en la sección 1.4.4.2, para añadir un desplazamiento de fase y luego estimar por medio de las posiciones ese desplazamiento de fase añadido inicialmente. Sin embargo, en este trabajo se mide directamente los ángulos de las posiciones de la secuencia larga de una señal recibida y se saca un promedio para obtener el desplazamiento total de fase, y se usa el objeto de comunicación **comm.PhaseFrequencyOffset** para corregir aquel desplazamiento de fase hallado.

1.4.6.3. INTRODUCCIÓN MANUAL DE UN DESPLAZAMIENTO DE FASE

En el script desplazar_fase.m que se muestra a continuación, es una simulación de dos señales, la primera es la señal original que contiene datos aleatorios con modulación QPSK añadido ruido aditivo gaussiano para representar las alteraciones del canal inalámbrico y la segunda señal es una copia de la primera con la diferencia que está desplazada en fase 30° y 45° utilizando el objeto de comunicación **comm.PhaseFrequencyOffset** .

```
%% % Ejemplo de las alteraciones del canal inalámbrico
clear; close all;
% RUIDO ADITIVO GAUSSIANO
% Datos de entrada:
datos=randi([0 3],256,1); % Simbolos de entrada
% Mapeo QPSK:
M = 4; % Puntos en la constelación (símbolos)
datosM = pskmod(datos,M,pi/4,'gray'); % Datos modulados QPSK
% Factor de normalización = 1/sqrt(2)
datosMapeados=(1/sqrt(2))*datosM; % Vector con datos modulados y norm
% Añadir ruido:
datosMapeo= awgn(datosMapeados,20,'measured'); % Señal con ruido

%% DESPLAZAR EN FASE
Fs=2.0e6; % frecuencia de muestreo
DesfSenial1 = comm.PhaseFrequencyOffset('PhaseOffset',30,...
    'SampleRate',Fs);
DesfSenial2 = comm.PhaseFrequencyOffset('PhaseOffset',45,...
    'SampleRate',Fs);
datosDesfase1=DesfSenial1(datosMapeo); % Aplicar el desfase a datosMapeo
datosDesfase2=DesfSenial2(datosMapeo); % Aplicar el desfase a datosMapeo

% Graficar
figure()
subplot(1,2,1)
plot(datosMapeo,'r.','MarkerSize',10); hold on;
plot(datosDesfase1,'b.','MarkerSize',10); grid on;
title('Señal desplazada en fase 30°');
legend('Original', 'Desplazada');
```

```
xlabel('I (fase)');ylabel('Q (cuadratura)');axis([-1 1 -1 1]);

subplot(1,2,2)
plot(datosMapeo,'r.','MarkerSize',10); hold on;
plot(datosDesfase2,'b.','MarkerSize',10); grid on;
title('Señal desplazada en fase 45°');
legend('Original', 'Desplazada');
xlabel('I (fase)');ylabel('Q (cuadratura)');axis([-1 1 -1 1]);
```

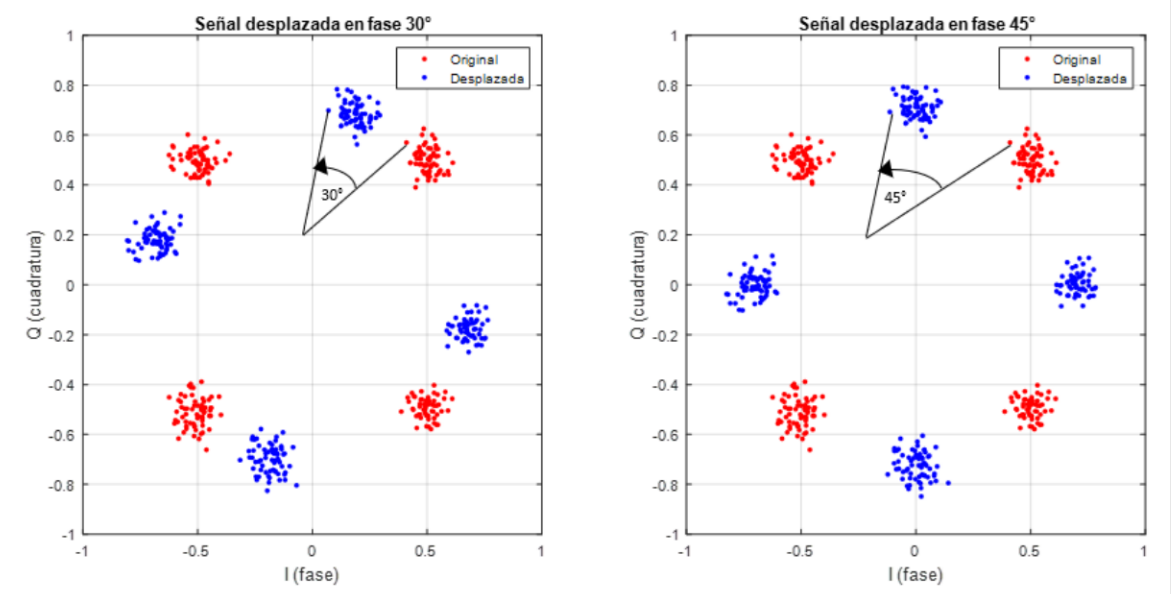


Figura 16. Señales desplazadas en fase. [3]

1.4.7. SDR ADALM PLUTO

El dispositivo SDR Adalm Pluto es un equipo que presenta canales de recepción y transmisión independientes que se pueden operar en half o full duplex, también puede generar o adquirir señales analógicas de radio frecuencia (RF) de 325 MHz a 3800 MHz con frecuencias de muestreo de hasta 61,44 megamuestras por segundo (MSPS). Cuenta con alimentación por USB 2.0 y está habilitado por las librerías libiio, C, C++ y API de Phyton, por lo que es compatible con los sistemas operativos OS X , Windows y Linux, además soporta programas como Matlab, Simulink o GNU Radio. [9]

El paquete llamado *"Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio"* permite utilizar el dispositivo SDR Adalm Pluto para diseñar y verificar sistemas prácticos, en condiciones reales, con Matlab y Simulink [10].

Para configurar los equipos SDR se utiliza un objeto del sistema para transmisión y otro para recepción []. El objeto de transmisión utilizado es sdrtx() que tiene una sintaxis *txAdalm = sdrtx('Pluto',Name,Value),* cuyas propiedades se muestran en la Tabla 1.

Tabla 1. Propiedades del objeto de transmisión [3]

PROPIEDAD	DETALLE	VALOR
DeviceName	Es el nombre o modelo del dispositivo SDR	'Pluto'
RadioID	Se refiere a un número de identificación de la radio, el cual está relacionado al puerto USB del computador en donde se ha conectado	'usb:0'
CenterFrecuency	Ajuste de la frecuencia central de RF. Se especifica como un número escalar que va desde 70.0e6 hasta 6.0e9 (de 70 MHz a 6 GHz). El valor por defecto de este parámetro es de 2.4e9 (2.4 GHz).	'860e6'
Gain	Es la ganancia en dB, la cual se especifica con un número escalar que va de -89.75 a 0 [dB]. Además, la resolución es de 0.25 y el valor predeterminado de este parámetro es -10.	'-1'
BasebandSampleRate	Es la frecuencia de muestreo en banda base en Hz, se especifica como con un escalar que va desde 65105 a 61.44e6 muestras por segundo (sps). El valor por defecto es 1.0e6.	'2e6'

Para el objeto de recepción se utiliza sdrtx() que tiene una sintaxis *rxAdalm = sdrrx('Pluto',Name,Value)* cuyas propiedades configuradas se muestran en la Tabla 2

Tabla 2. Propiedades del objeto de recepción [3]

PROPIEDAD	DETALLE	VALOR
DeviceName	Es el nombre o modelo del dispositivo SDR	'Pluto'
RadioID	Se refiere a un número de identificación de la radio, el cual está relacionado al puerto USB del computador en donde se ha conectado	'usb:0'
CenterFrecuency	Ajuste de la frecuencia central de RF. Se especifica como un número escalar que va desde 70.0e6 hasta 6.0e9 (de 70 MHz a 6 GHz). El valor por defecto de este parámetro es de 2.4e9 (2.4 GHz).	'860e6'

GainSource	Es la fuente de ganancia y se puede especificar como: -'AGC Slow Attack' que se utiliza para señales que los niveles de potencia cambian lentamente, además es el valor por defecto. -'AGC Fast Attack' que se utiliza para señales que los niveles de potencia cambian rápidamente. -'Manual' que se utiliza para configurar de manera manual la ganancia con la propiedad Gain.	'Manual'
Gain	Es la ganancia en dB del receptor de radio, se especifica como un número escalar de -4 a 71 y el valor predeterminado es 10. Esta propiedad depende de que en GainSource se haya seleccionado 'Manual'. Además, cabe mencionar que los valores máximos y mínimos dependen también de la frecuencia central, si se genera una combinación incompatible de estos parámetros, se devuelve un error.	'70'
BasebandSampleRate	Es la frecuencia de muestreo en banda base en Hz, se especifica como con un escalar que va desde 65105 a 61.44e6 muestras por segundo (sps). El valor por defecto es 1.0e6.	'2e6'
OutputDataType	Es el tipo de datos de la señal de salida y pueden ser: -'int16' (predeterminado): entero de 16 bits con signo, rango de valores de -32768 a 32767 y emplea 16 bits por cada valor. -'single': punto flotante de doble precisión, con un rango de valores de -1 a 1 y emplea 32 bits por cada valor. -'double': punto flotante de precisión simple, con un rango de valores de -1 a 1 y emplea 64 bits por cada valor.	'single'
SamplesPerFrame	Es el número de muestras por trama, se especifica como un entero positivo par que va desde 2 a 16,777,216 y el valor por defecto es de 20000. Los valores inferiores a 3660 puede producir un rendimiento deficiente.	'800000'

2. METODOLOGÍA

El desarrollo de este trabajo consta de una computadora portátil a la cual se conectan en un puerto USB diferente cada dispositivo SDR Adalm Pluto como se muestra en la Figura .Además requiere que se ejecute dos sesiones de Matlab, la primera sesión tiene los scripts config_SDR.m y tx_real_ofdm.m (VER ANEXO II) que sirven para que el dispositivo SDR funcione como transmisor, mientras que la segunda sesión de Matlab posee los scripts config_SDR.m y rx_real_ofdm.m (VER ANEXO II) que configuran al equipo SDR como receptor.

Figura 17. Funcionamiento del sistema de comunicación.

Para la recolección y estudio de los datos, se procede a tomar una muestra de la señal transmitida y recibida para observar tanto el desplazamiento de frecuencia como el desplazamiento de fase y buscar mecanismos para su corrección. En tal sentido, se empezará con la configuración de los dispositivos Adalm Pluto, uno como transmisor y otro como receptor con la ayuda de script config_SDR.m (VER ANEXO II) que establece los parámetros básicos para su correcto funcionamiento.

En la sesion de Matlab donde se configuró al dispositivo SDR como transmisor, se corre el script *tx_real_ofdm.m*(VER ANEXO II) que es una recopilación de los scripts: *abrir_archivo.m, serializar_datos.m, cod_conv.m, mapeo_datos.m, reformado_tx.m, ofdm_tx.m* y *preámbulo_tx.m* [3] que sirven para seguir paso a paso cada etapa en el transmisor. Posteriormente, se añade una etapa de transmisión donde se acondiciona la señal para que cumpla con los parámetros configurados en el dispositivo SDR Adalm Pluto, en este caso, una longitud de 800000 muestras por trama, además los datos son transmitidos en tiempo real cada segundo en un bucle infinito, hasta detenerlo manualmente. La variable que contiene los datos a ser enviados se llama *datosEnviados* y

es de tipo complejo, por lo que, para graficarla, se va a dividir en su componente real e imaginaria y se obtiene la Figura 18:

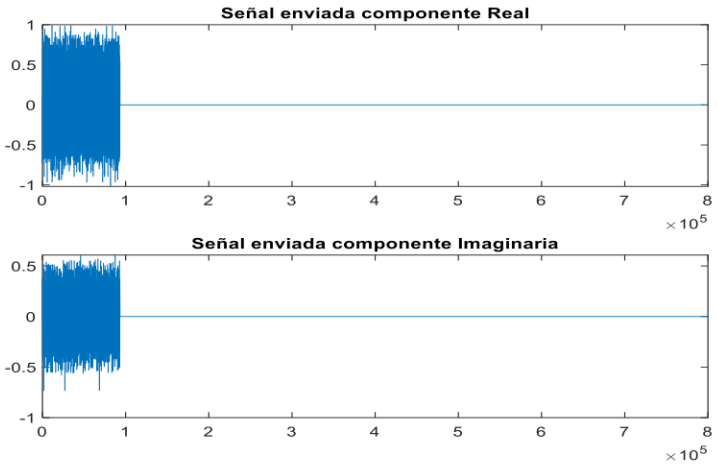


Figura 18. Gráfica de la señal enviada con su componente real e imaginario.

En la sesión de Matlab que funciona como recepción, después de ejecutar el script config_SDR,m (ANEXO II) para configurar el objeto de recepción, se necesita leer los datos recibidos mientras que el transmisor envía al mismo tiempo la información, es necesario solucionar el problema de sincronismo entre ambos dispositivos con una variable de decisión la cual tomará el nombre de umbral, y permitirá determinar si los datos corresponden a una señal o son producto del ruido en el canal [3].

Para estimar el umbral se realizaron 10 lecturas de datos mediante el objeto de recepción configurado previamente. Se inicializan dos vectores: *lectura*, que permitirá almacenar los valores recibidos proporcionados por el receptor y *maxi*, que permitirá almacenar el valor máximo del valor absoluto de los datos del vector lectura. Luego se procederá a crear un lazo *for* con n número de muestras a considerar, en este caso 10 lecturas, que nos permitirá estimar el umbral. Finalmente, en la variable *umbralrx* se establecerá el umbral en base al promedio.

Una vez determinado el valor de umbral, se requiere sondear el canal inalámbrico, con el objetivo de establecer lo que se está recibiendo, una señal o solo ruido. Para ello se ha desarrollado un script llamado *pruebaRecepcion.m* (VER ANEXO III). A continuación, una vez ejecutado el script, se obtiene la gráfica de la parte real e imaginaria de la señal recibida como se muestra en la Figura 19.

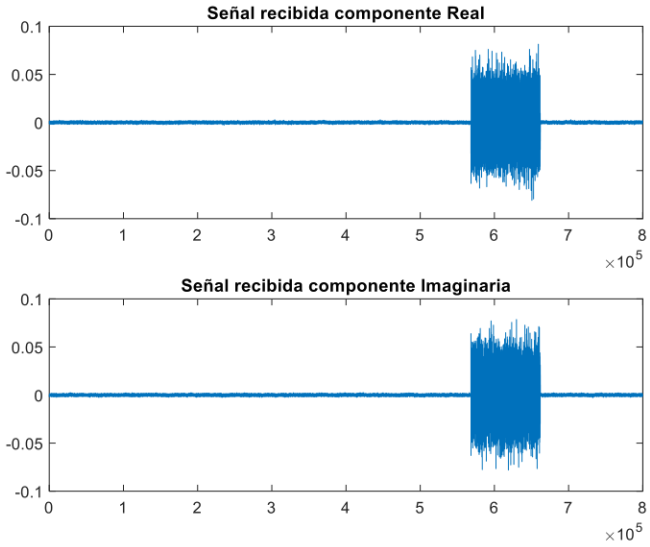


Figura 19. Gráfica de la señal recibida con su componente real e imaginario

Obtenidas ambas gráficas, tanto de la señal enviada como de la señal recibida, se las muestra en un solo gráfico, como se evidencia en la Figura 20, para poder observar su desplazamiento.

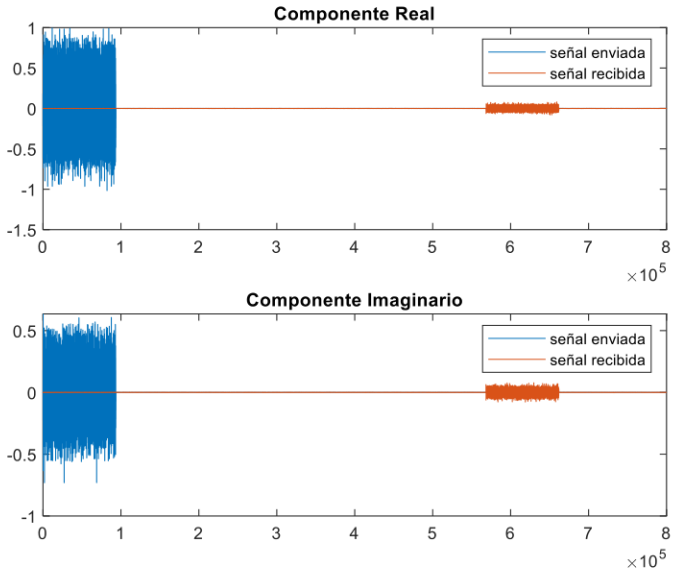


Figura 20. Gráfica de la señal enviada y recibida con sus componentes real e imaginario.

2.1. MEDICIÓN DEL DESPLAZAMIENTO DE FRECUENCIA.

Ahora bien, determinado que se puede enviar y recibir la señal al mismo tiempo, se procede a realizar un recorte de los datos, ya que de todos los datos recibidos solo el 11,68% contiene información que ha pasado por varias etapas antes de ser transmitida y también se debe detectar que de los datos que se han recibido y recortado, todos pertenezcan a

una trama OFDM, así pues, se ha utilizado el preámbulo IEEE 802.11 compuesto por secuencias cortas y largas.

Finalmente, se procede a graficar la señal enviada antes de ser acondicionada (datosConPre) y la señal recibida luego de detectar la trama OFDM (datosOFDM) usando la herramienta proporcionada por Matlab, signalAnalyzer (Analizador de señales)[11] cuya sintaxis es la siguiente:

```
signalAnalyzer(sig1,…,sign, 'SampleRate', fs)
```

Donde se agregan las señales a ser graficadas en conjunto con su frecuencia de muestreo, además con la herramienta signalAnalyzer se despliega una ventana nueva y nos muestra las gráficas de las señales en tiempo y frecuencia, usando para la última un analizador de espectro, donde se han convertido a la señal enviada (*datosConPre*) y la señal recibida (*datosOFDM*) en variables complejas de tipo *double*, debido a que es requisito del analizador de espectros para poder graficar.

En la Figura 21, se ha ejecutado el script signal_analyzer.m (VER ANEXO III) que nos permite observar las gráficas en tiempo y frecuencia de las señales mencionadas, por lo que ahora, realizando una ampliación en la gráfica, se intentará determinar el CFO de las señales, usando la herramienta cursor, que proporciona Matlab para sus gráficas como se muestra en la Figura 22.

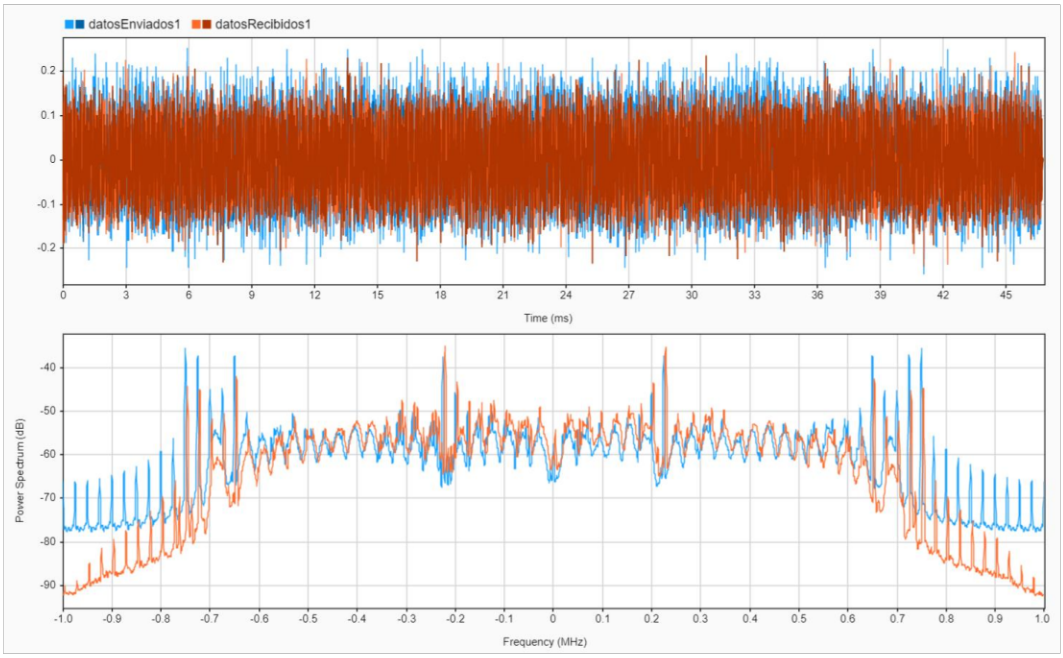


Figura 21. Gráfica de los espectros superpuestos usando el Analizador de señales

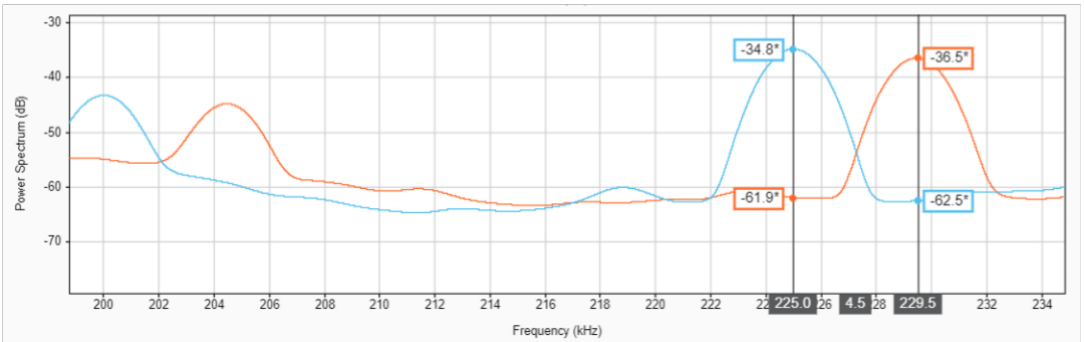


Figura 22. Cálculo del CFO de las señales, usando los cursores del analizador de señales.

El CFO para la muestra de estas señales es de 4.5 kHz aproximadamente.

2.2. AUTOMATIZACIÓN DE LA MEDICIÓN DEL CFO

Para calcular el CFO de manera automática, primero se procede a recolectarlas y almacenarlas dentro de un vector, para luego procesarlas. Una vez obtenidas las muestras, se realiza el cálculo del CFO utilizando un analizador de espectro con la función de Matlab ***pspectrum*** [12], cuya sintaxis es la siguiente:

```
[p,f] = pspectrum(x)
```

que devuelve las frecuencias correspondientes(f) a las estimaciones espectrales contenidas en p. Es así que se procede a sacar el espectro de frecuencia de la señal enviada y de cada una de las muestras de la señal recibida. Después, se buscan los picos de potencia de las señales (enviada y recibidas) con la función de Matlab ***findpeaks***[13] con la siguiente sintaxis:

```
[pks,locs] = findpeaks(data)
```

que además, devuelve los índices en los que se producen los picos. Y se selecciona solo el valor máximo de cada señal recibida y su posición dentro del vector, con la función de Matlab ***max***[14] con la siguiente sintaxis:

```
[M, I] = max(A)
```

Luego, se realiza una comparación en la señal enviada, buscando el valor del pico anterior a la posición del valor del pico máximo de la señal recibida con la función de Matlab ***find*** [15] que permite filtrar los elementos de una matriz aplicando condiciones a la matriz.

Obtenidos ambos valores, se procede a determinar la frecuencia, ya que, para cada valor de potencia, se corresponde un valor de frecuencia en el espectro, por lo que, al restar estos dos valores, se obtiene el valor real del CFO para cada muestra.
Finalmente, se muestra una gráfica con el valor de CFO de cada muestra para proceder a realizar un histograma y buscar una función de probabilidad que se ajuste.

A continuación, se detalla el código utilizado en el script *cfo_calculo.m* :

```
%Script que realiza el cálculo del CFO de n muestras recolectadas
%Para su funcionamiento ejecutrar primero los scripts config_SDR.m
% umbral_rx.m, y recoleccion_datosrx.m

frecuenciaMuestreo=2.0e6;
n=300; %Número de muestras que se calculará el CFO.

%Procesamiento de los datos recibidos y transmitidos
load('D164QAMs.mat','datosRecibi2')
datosRecibi2=double(datosRecibi2); % Conversión a una variable compleja
% doble de los datos recolectados
load("datosEnviados.mat",'datosEnviados') % Se carga los datos transmitidos
datosEnviados1=double(datosEnviados);% Conversión a una variable compleja
% doble de los datos transmitidos

%Espectro de la señal transmitida
[ptx,ftx]=pspectrum(datosEnviados1,frecuenciaMuestreo);
[picostx,locstx] = findpeaks(ptx);% Localización de los picos del espectro
% de la señal transmitida y sus índices.

for i=1:n
    %Espectro de la señal recibida
    [prx(:,i),frx(:,i)]=pspectrum(datosRecibi2(:,i),frecuenciaMuestreo);

    %Localización de los picos del espectro de la señal recibida y sus
índices
    picosrx=zeros(600,i); %Se inicializa la variable con 600 ceros
    % donde se almacenarán todos los picos de cada muestra.

    %Longitud del relleno de los picos de cada muestra
    if length(findpeaks(prx(:,i)))< length(picosrx)

        aux1(:,i)=length(picosrx)-length(findpeaks(prx(:,i)));
    end
end

for i=1:n
    c1 =findpeaks(prx(:,i));
    picosrx(:,i)=[c1 ;zeros(aux1(i),1)]; % Se almacena todos los picos de
    % cada muestra

    %Selección del pico máximo y su índice de cada muestra de la señal
    % recibida
    [c2 c3] =max(picosrx(:,i));
    picomaxrx(i)=c2; %se almacena el valor del pico máximo de cada muestra
    Irx(i)=c3; %Se almacena el índice del pico máximo de cada muestra
```

```
    %Localización del pico más cercano de la señal transmitida, con
    %respecto al pico máximo de la señal recibida
    l=find(locstx<Irx(i));
    Itx(i)=locstx(l(end));% Se almacena el índice del pico más cercano de
    % la señal transmitida, con respecto al pico máximo de la señal
    % recibida.

    %Frecuencias de los picos de la señal transmitida y señal recibida.
    Frecrx(i)=frx(Irx(i));%Frecuencia señal recibida
    Frectx(i)=ftx(Itx(i));%Frecuencia señal transmitida

    %Cálculo del CFO
    cfo(i)=Frecrx(i)-Frectx(i);

end

%Gráfica del valor del CFO para cada muestra recibida.
figure()
stem(cfo)
title('CFO con modulación BPSK y distancia 18m')
xlabel('Muestras')
ylabel('Frecuencia [Hz]')


%Corrección del CFO
correctioncfo= comm.PhaseFrequencyOffset('FrequencyOffset',cfo,...
    'SampleRate',frecuenciaMuestreo);
datosCorrCFO=correctioncfo(datosOFDM);
```

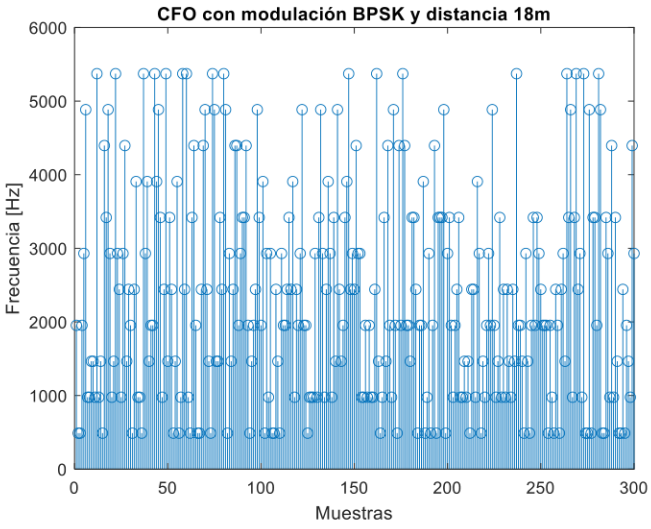


Figura 23. Resultado del script cfo_calculo.m

Determinado el CFO tanto gráficamente como matemáticamente con el script cfo_calculo.m(VER ANEXO III), se procede a corregirlo usando el objeto ***comm.PhaseFrequencyOffset*** como lo mencionado en la sección 1.4.4.2

2.3. SINCRONIZACIÓN DE SÍMBOLO

Por consiguiente, para la estimación del STO, se toman los datos ya corregidos el CFO y se procede a ejecutar el script sto_rx.m (VER ANEXO III) luego de haber sido ejecutado el script cfo_calculo.m (VER ANEXO III) donde se utilizará la autocorrelación de la secuencia larga para determinar el punto de inicio de la trama OFDM

```
%Script para determinar la sincronización de símbolo STO

datosSTO=datosCorrCFO(1:5000); % Se toma dolos primeros 5000 datos
% para reducir el tiempo de procesamiento


% Secuencia larga (parte de LLTF) en el dominio de la frecuencia
parteLarga=[...
    0;1;-1;-1;1;1;-1;1;-1;1;-1;-1;-1;-1;-1;1
    1;-1;-1;1;-1;1;-1;1;1;1;1;0;0;0;0;0
    0;0;0;0;0;0;1;1;-1;-1;1;1;-1;1;-1;1
    1;1;1;1;1;-1;-1;1;1;-1;1;-1;1;1;1;1];
% Conversión del dominio de la frecuencia al tiempo, y adición de PC
% de la sección larga:
parteLargaTime=ifft(parteLarga); % conversión al dominio del tiempo
LLTF=[parteLargaTime(33:64);parteLargaTime;parteLargaTime]; % LLTF
dLLTF80 = LLTF(1:80); % Primeras 80 muestras de la LLTF
%Correlación cruzada
datosCorrRell = (abs(xcorr(datosSTO,dLLTF80))).^2;
% Se corrige en tiempo los datos correlacionados, para que
% coincidan con las posiciones de los datos de entrada:
datosCorr = datosCorrRell(length(datosSTO):end);
% Almacenar la posición de los dos picos:
[~,posPico1]=max(datosCorr); % Posición del primer pico
datosCorr(posPico1)=0; % Eliminar el primer pico para poder
% encontrar el segundo
[~,posPico2]=max(datosCorr); % Posición del segundo pico
% Se comprueba que la distancia entre los dos picos sea 64
if abs(posPico2-posPico1)==64
    % Se encuentra el inicio real del LTTF y de LSTF
    inicioLLTF=min([posPico1 posPico2]); % Se toma la posición
    % menor y esa es la ubicación del primer dato del LLTF
    inicioLSTF=inicioLLTF-160; % Determinar el inicio del preambulo
    % Comprobar que la posición inicial sea mayor a 0
    if (inicioLSTF>0)
        % Recortar los datos
        datosSincronizados=datosCorrCFO(inicioLSTF:end); % Datos
        % de salida
        %Graficar:
        figure()
        plot(real(datosSincronizados(1:500)))
        title('Parte real de los datos sincronizados')
    else
        disp('Falla en la sincronizacion de simbolo (2)')
    end
else
    disp('Falla de sincronización de símbolo')

end
```

2.4. DEZPLAZAMIENTO DE FASE

Realizado la sincronización de símbolo, se procede a realizar el cálculo de desplazamiento de fase, para ello se toma como referencia la sección LLTF del preámbulo 802.11a en el dominio de la frecuencia, debido a que cuando la parte real de la misma es igual a 1, el ángulo entre la componente real e imaginaria es igual a 0. En ese sentido, ejecutando el script fase.m se obtendrán las posiciones de la secuencia larga, para verificar el desplazamiento de fase real de la señal recibida. [3]

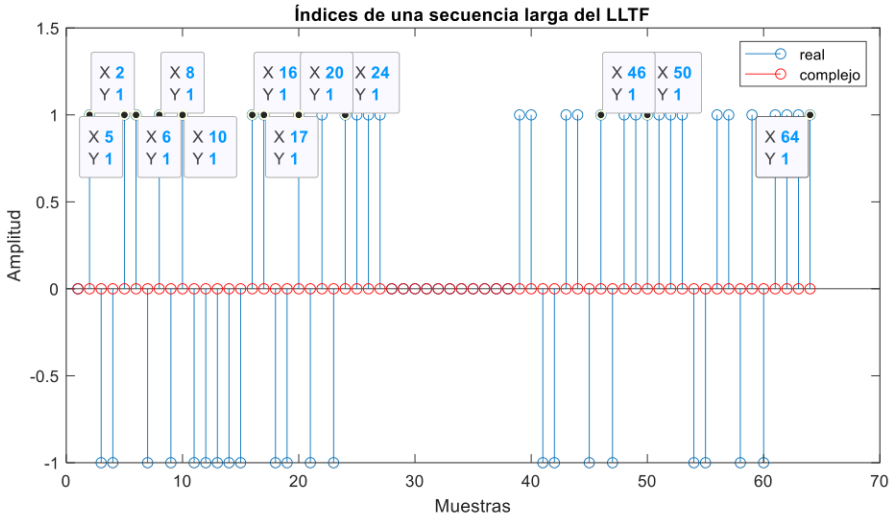


Figura 24. Índices de una secuencia larga LLTF

Ejecutando el script **fase.m** (VER ANEXO III) se puede obtener las posiciones donde la parte real de la secuencia larga es igual a 1, como se muestra en la Figura 24.

Luego se utiliza esas posiciones para calcular los ángulos de la señal recibida, usando el script **fase_rx.m** (VER ANEXO III) que utiliza la función angle de Matlab, que devuelve un valor en radianes, y se lo convierte en grados dando como resultado la Figura 25.

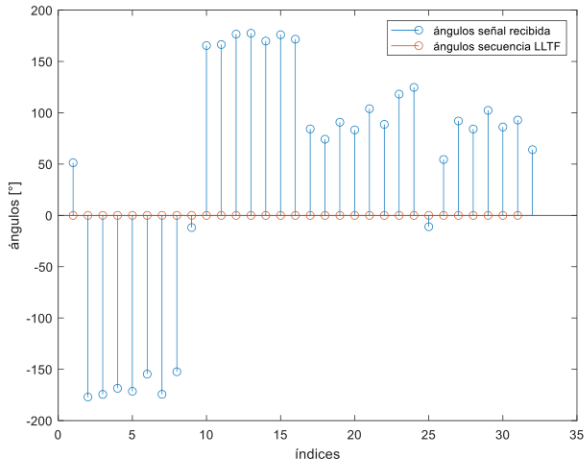


Figura 25. Ángulos de la señal recibida y de la secuencia LLTF

Finalmente se saca el promedio de los ángulos de cada índice, para poder ser corregido con el objeto ***comm.PhaseFrequencyOffset*** cuya sintaxis se mencionó en la sección 1.4.4.2 y continuar con el resto del proceso de recepción.

Para observar cómo cambia el desplazamiento de fase con cada muestra, se recolectan 15 muestras que han sido corregidas el CFO, se han sincronizado los símbolos y se ha sacado el ángulo promedio de todos los índices, obteniéndose como resultado la Figura 26.

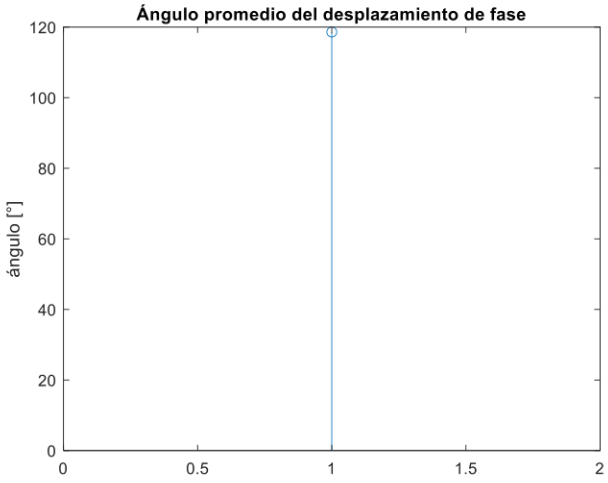


Figura 26. Desplazamiento de fase

3. RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1. PRUEBAS DE RECEPCIÓN PARA EL PARÁMETRO CFO

3.1.1. CON LÍNEA DE VISTA (LoS)

3.1.1.1. DISTANCIA MÁXIMA CON DIFERENTES MODULACIONES

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 18m, con línea de vista, con los diferentes tipos de modulaciones usados en el esquema de transmisión OFDM para un número de 300 muestras.

- **Mapeo 1: BPSK**

Las muestras se encuentran almacenadas en el archivo D1BPSK.mat y se procede junto al script **histograma.m** (VER ANEXO III) a obtener la siguiente gráfica:

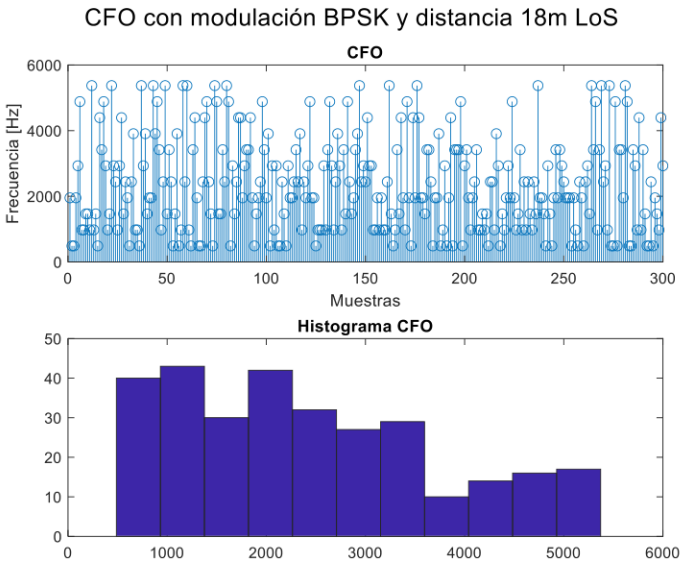


Figura 27. Cálculo del CFO a 18m de distancia con modulación BPSK.

- **Mapeo 2: QPSK**

Las muestras se encuentran almacenadas en el archivo D1QPSK.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

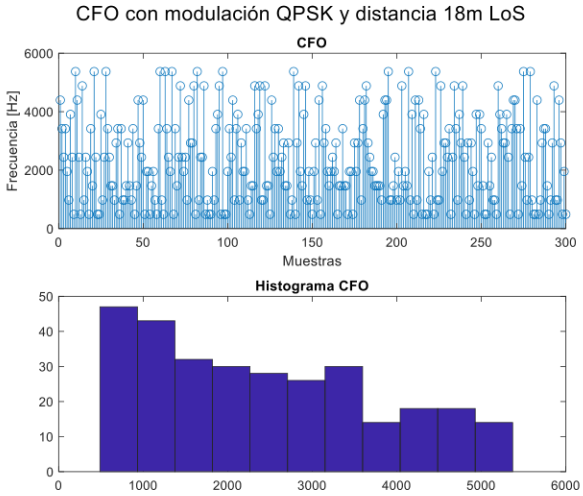


Figura 28. Cálculo del CFO a 18m de distancia con modulación QPSK.

- **Mapeo 3: 16-QAM**

Las muestras se encuentran almacenadas en el archivo D116QAM.mat y se procede junto al **script histograma.m**(VER ANEXO III) a obtener la siguiente gráfica:

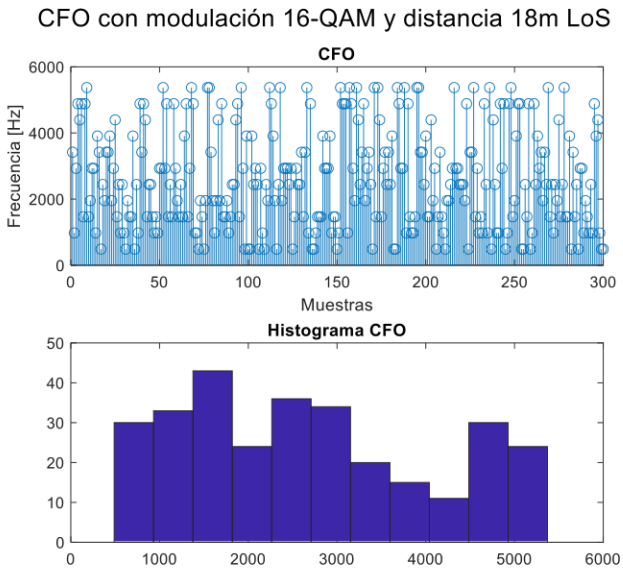


Figura 29. Cálculo del CFO a 18m de distancia con modulación 16-QAM.

- **Mapeo 4: 64-QAM**

Las muestras se encuentran almacenadas en el archivo D164QAM.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

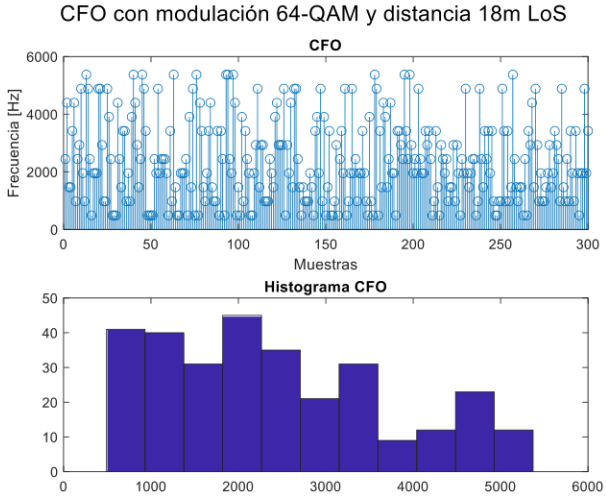


Figura 30. Cálculo del CFO a 18m de distancia con modulación 64-QAM.

3.1.1.1.1. Resultados con diferentes modulaciones.

La modulación BPSK tiene un CFO promedio de 2.38 kHz y una distribución gamma como se muestra en la Figura 31.

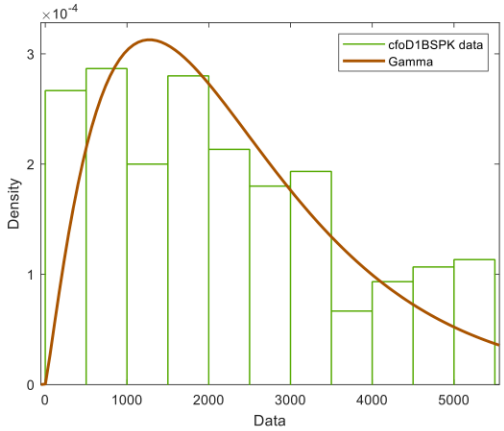


Figura 31. Distribución Gamma para el CFO con modulación BPSK a 18m

La modulación QPSK tiene un CFO promedio de 2.38 kHz y una distribución lo más parecida a una exponencial, como se muestra en la Figura 32.

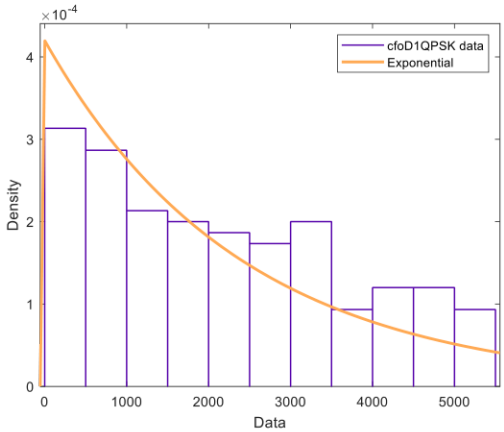


Figura 32. Distribución exponencial para el CFO con modulación QPSK a 18m

La modulación 16-QAM tiene un CFO promedio de 2.65 kHz y una distribución que se puede ajustar a Weibull mejor que una distribución gamma, como se muestra en la Figura 33.

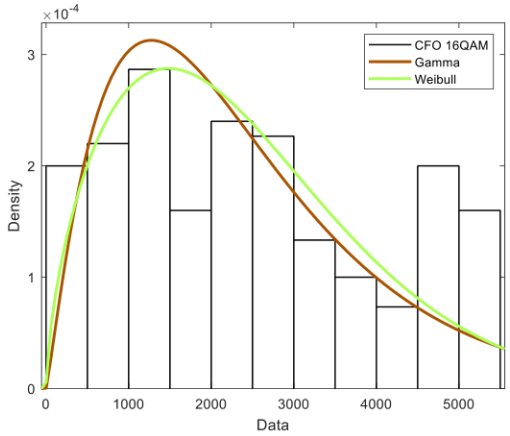


Figura 33. Distribución gamma y Weibull para el CFO con modulación 16-QAM a 18m

La modulación 64-QAM tiene un CFO promedio de 2.38kHz y se puede ajustar a una distribución gamma o una distribución de valor extremo generalizado, como se muestra en la Figura 34.

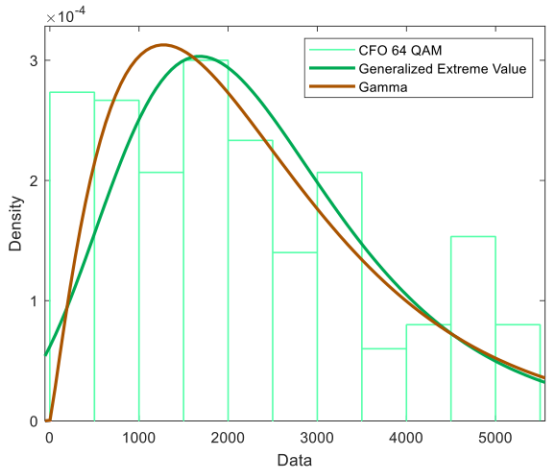


Figura 34. Distribución de valor extremo generalizado y gamma para el CFO con modulación 64-QAM a 18m

3.1.1.2. DISTANCIA 2 AL 70% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 12.6 m, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D2BPSK.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

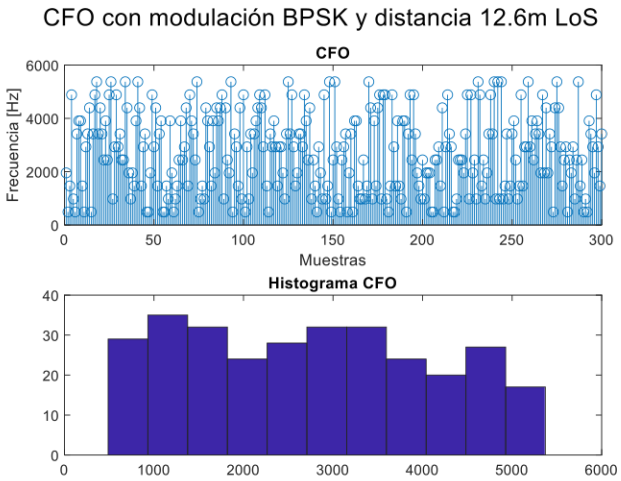


Figura 35 . Cálculo del CFO a 12.6 m de distancia con modulación BPSK

3.1.1.3. DISTANCIA 3 AL 50% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 9 m, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D3BPSK.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

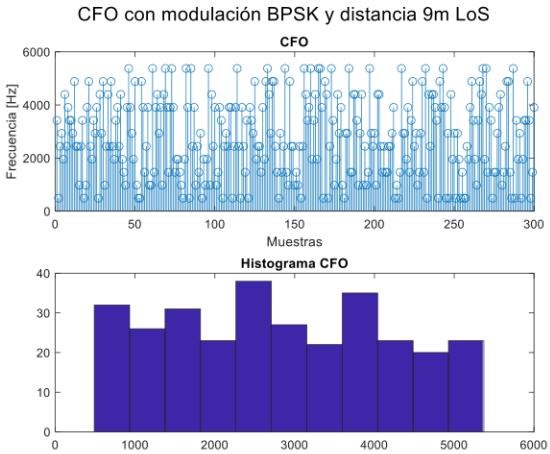


Figura 36. Cálculo del CFO a 9 m de distancia con modulación BPSK

3.1.1.4. DISTANCIA 4 AL 30% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 5.4 m, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D4BPSK.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

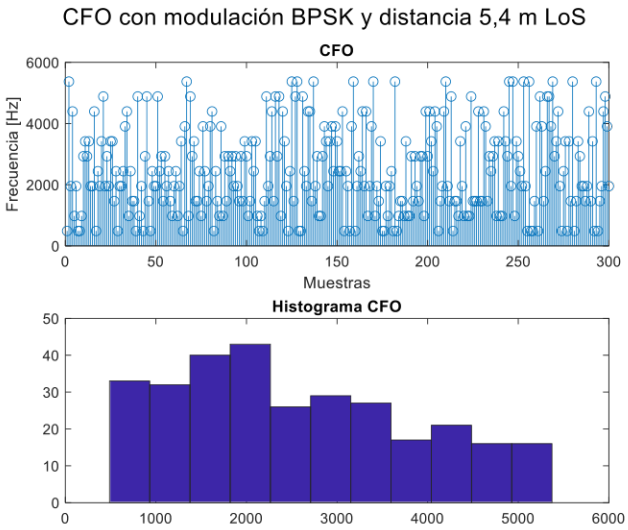


Figura 37. Cálculo del CFO a 5.4 m de distancia con modulación BPSK

3.1.1.5. RESULTADOS CON DIFERENTES DISTANCIAS

El CFO promedio para una distancia de 12,6m es de 2,78 kHz y se puede ajustar tanto a una distribución Nakagami o una distribución Weibull como se muestra en la Figura 38.

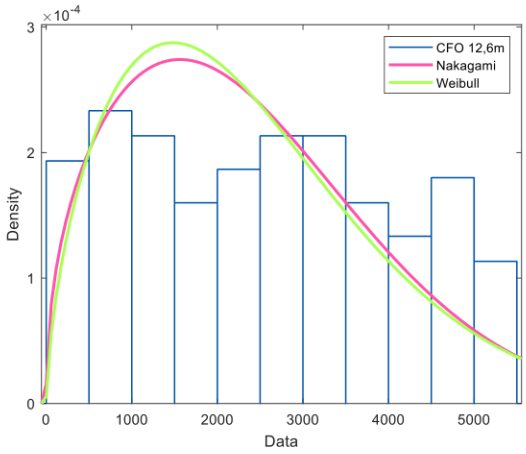


Figura 38. Distribución Nakagami y Weibull para el CFO con modulación BPSK a 12,6m

El CFO promedio para una distancia de 9m es de 2,79 kHz y se podría ajustar a una distribución normal como se muestra en la Figura 39.

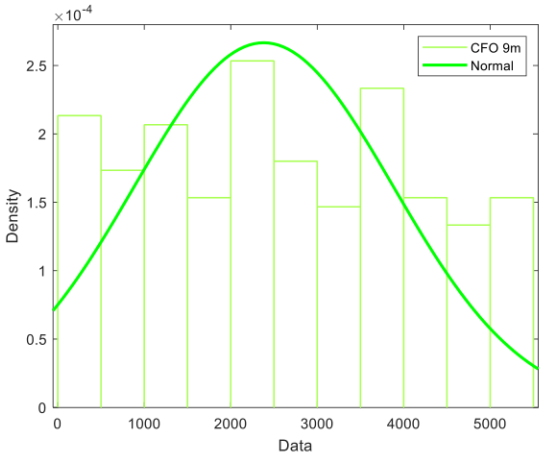


Figura 39. Distribución Normal para el CFO con modulación BPSK a 9m

El CFO promedio para una distancia de 5,4m es de 2,51 kHz y se ajusta a una distribución Weibull como se muestra en la Figura 40.

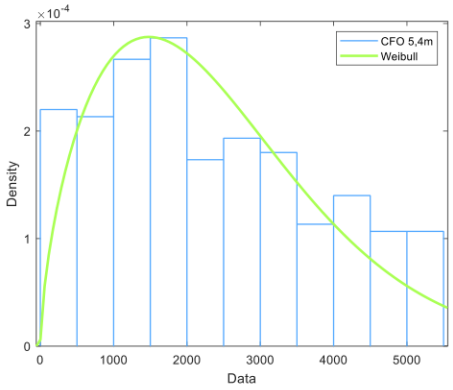


Figura 40. Distribución Normal para el CFO con modulación BPSK a 5,4m

3.1.2. SÍN LÍNEA DE VISTA

3.1.2.1. DISTANCIA MÁXIMA CON DIFERENTES MODULACIONES

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 18m, sin línea de vista, con los diferentes tipos de modulaciones usados en el esquema de transmisión OFDM para un número de 300 muestras.

- **Mapeo 1: BPSK**

Las muestras se encuentran almacenadas en el archivo D1BPSKs.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

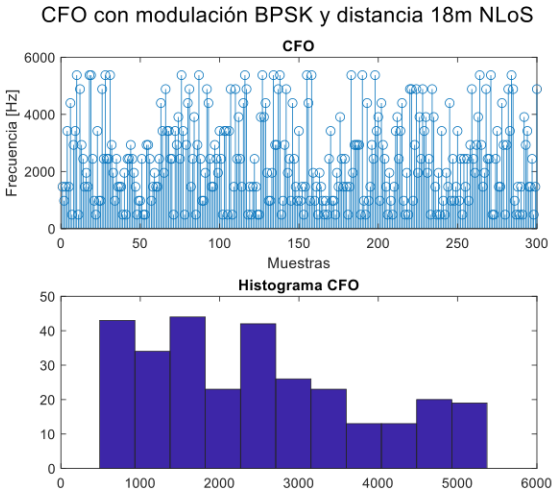


Figura 41. Cálculo del CFO a 18 m de distancia con modulación BPSK

- **Mapeo 2: QPSK**

Las muestras se encuentran almacenadas en el archivo D1QPSKs.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

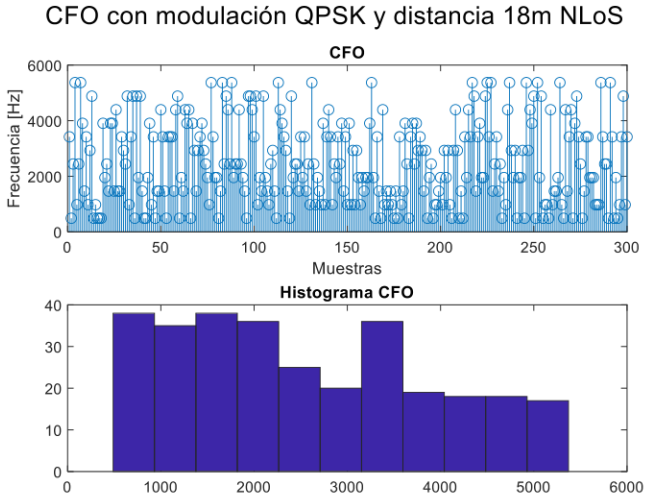


Figura 42. Cálculo del CFO a 18 m de distancia con modulación QPSK

- **Mapeo 3: 16-QAM**

Las muestras se encuentran almacenadas en el archivo D116QAMs.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

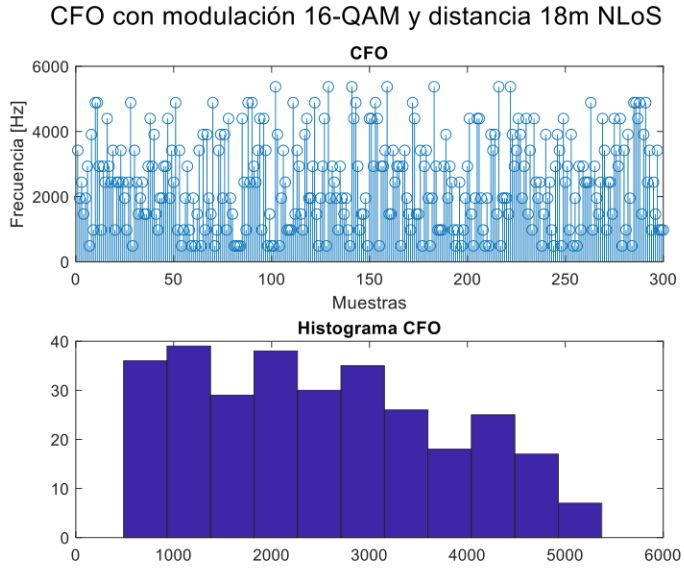


Figura 43. Cálculo del CFO a 18 m de distancia con modulación 16-QAM

- **Mapeo 4: 64-QAM**

Las muestras se encuentran almacenadas en el archivo D164QAMs.mat y se procede junto al script histograma.m(VER ANEXO III) a obtener la siguiente gráfica:

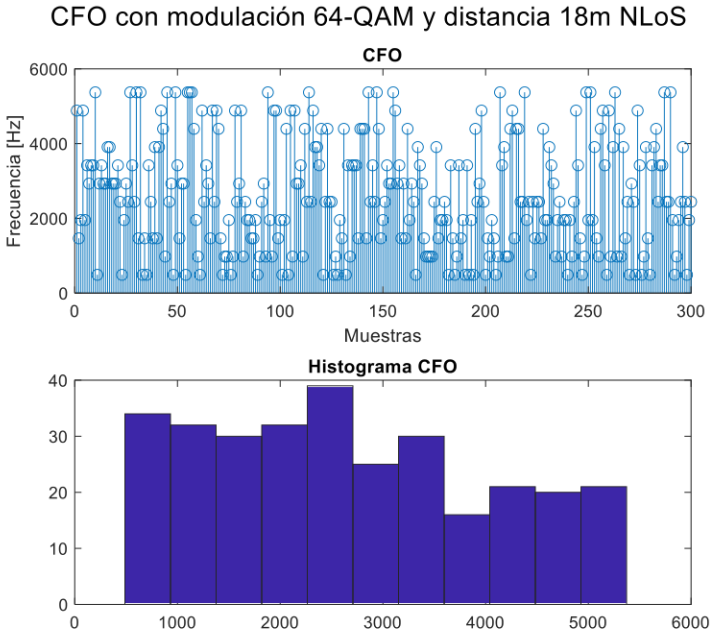


Figura 44. Cálculo del CFO a 18 m de distancia con modulación 64-QAM

3.1.2.1.1. Resultados con diferentes modulaciones

Para la modulación BPSK sin línea de vista se obtiene un CFO promedio de 2.43kHz y se podría ajustar a una distribución gamma, como se muestra en la Figura 45.

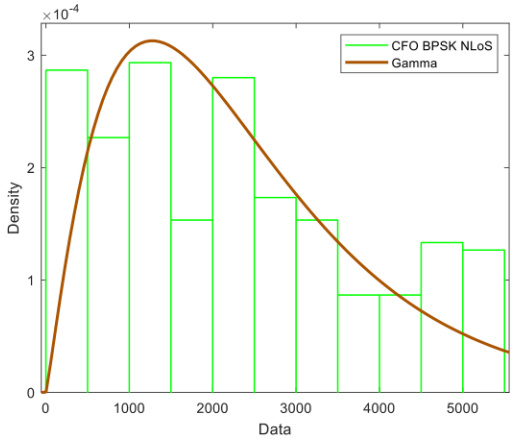


Figura 45. Distribución gamma para el CFO con modulación BPSK a 18m NLoS.

Para la modulación QPSK sin línea de vista se obtiene un CFO promedio de 2.51kHz y se podría ajustar a una distribución nakagami, como se muestra en la Figura 46.

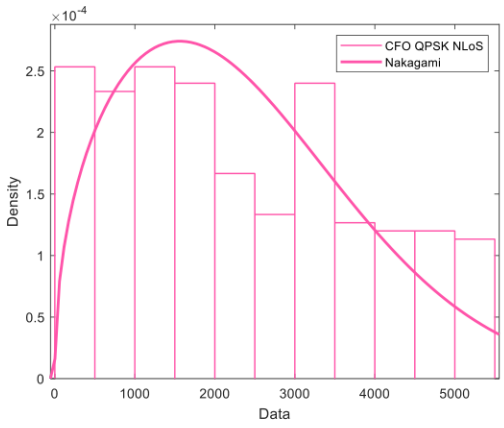


Figura 46. Distribución gamma para el CFO con modulación QPSK a 18m NLoS

Para la modulación 16-QAM sin línea de vista se obtiene un CFO promedio de 2.46kHz y se podría ajustar a una distribución Nakagami, como se muestra en la Figura 47.

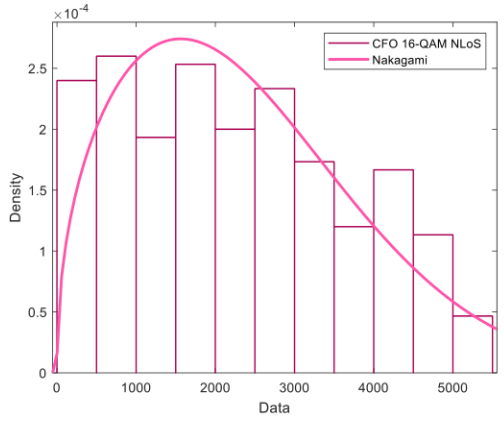


Figura 47. Distribución Nakagami para el CFO con modulación 16-QAM a 18m NLoS

Para la modulación 64-QAM sin línea de vista se obtiene un CFO promedio de 2.64kHz y se puede ajustar a una distribución Nakagami como se muestra en la Figura 48.

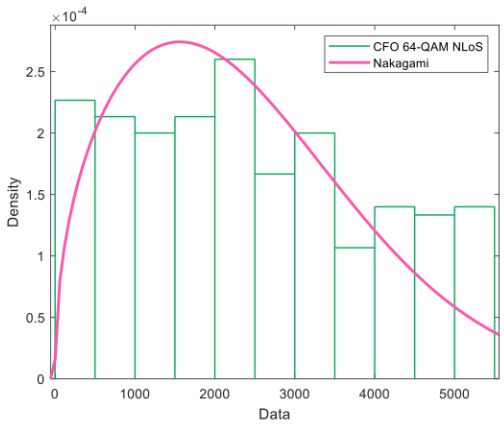


Figura 48. Distribución Nakagami para el CFO con modulación 64-QAM a 18m NloS

3.1.2.2. DISTANCIA 2 AL 70% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 12.6 m, sin línea de vista, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D2BPSKs.mat y se procede junto al script histograma.m a obtener la siguiente gráfica:

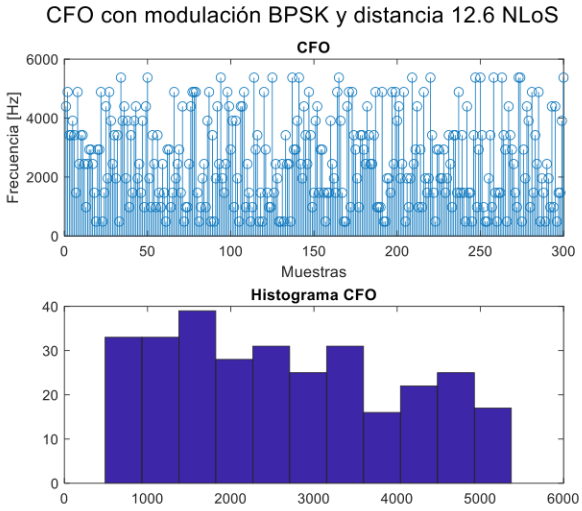


Figura 49. Cálculo del CFO a 12.6 m de distancia con modulación BPSK

3.1.2.3. DISTANCIA 3 AL 50% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 9 m, sin línea de vista, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D3BPSKs.mat y se procede junto al script histograma.m a obtener la siguiente gráfica:

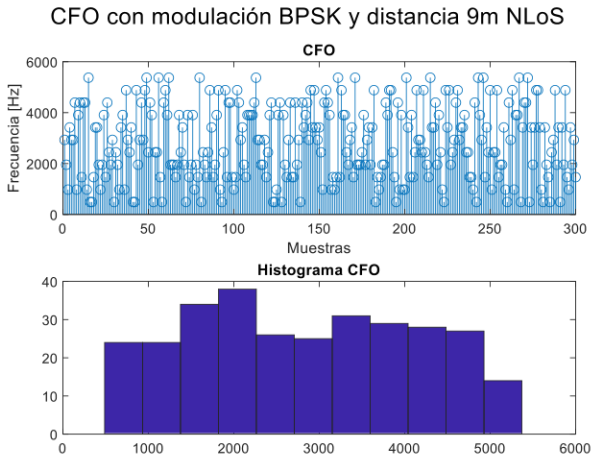


Figura 50. Cálculo del CFO a 9 m de distancia con modulación BPSK

3.1.2.4. DISTANCIA 4 AL 30% DE LA DISTANCIA MÁXIMA.

Se procede a realizar las pruebas para el cálculo del CFO con una distancia de separación entre transmisor y receptor de 5.4 m, sin línea de vista, con modulación BPSK en el esquema de transmisión OFDM para un número de 300 muestras.

Las muestras se encuentran almacenadas en el archivo D4BPSKs.mat y se procede junto al script histograma.m a obtener la siguiente gráfica:

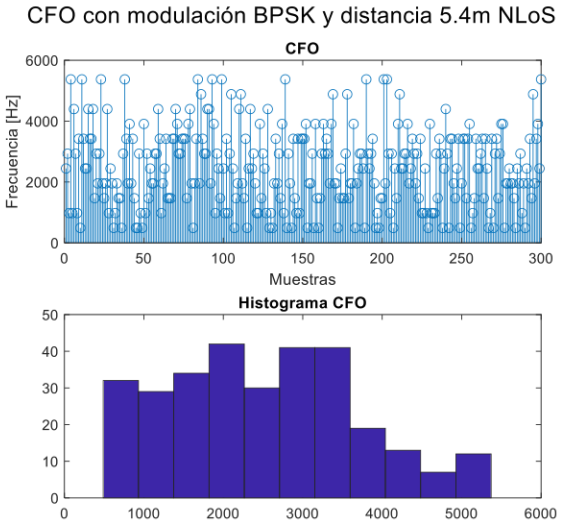


Figura 51. Cálculo del CFO a 5.4m m de distancia con modulación BPSK

3.1.2.5. RESULTADOS CON DIFERENTES DISTANCIAS.

El CFO promedio para una distancia de 12,6 m es de 2,63kHz y se ajusta a una distribución Nakagami como se muestra en la Figura 52.

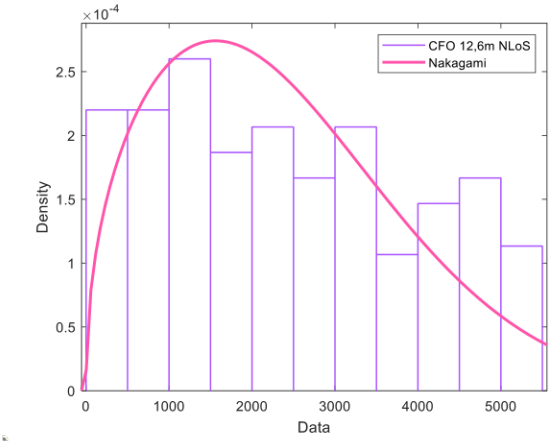


Figura 52. Distribución Nakagami para el CFO con modulación BPSK a 12,6m NLoS

El CFO promedio para una distancia de 9 m es de 2,82kHz y se ajusta a una distribución Nakagami como se muestra en la Figura 53.

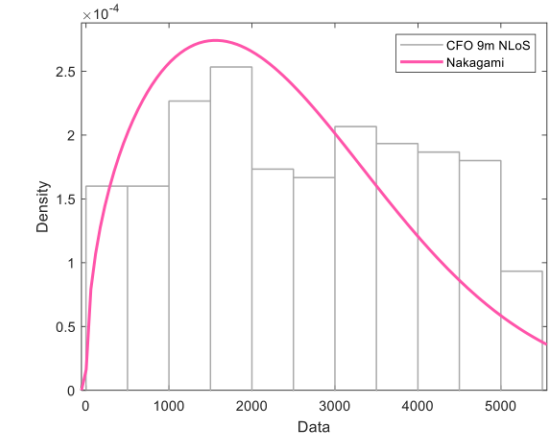


Figura 53. Distribución Nakagami para el CFO con modulación BPSK a 9m NLoS

El CFO promedio para una distancia de 5.4 m es de 2,46kHz y se ajusta a una distribución Rayleigh como se muestra en la Figura 54.

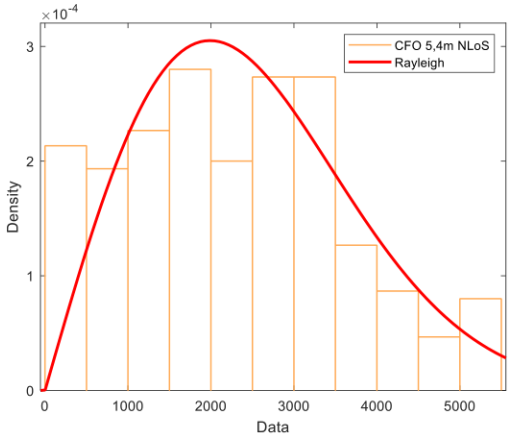


Figura 54. Distribución Rayleigh para el CFO con modulación BPSK a 5,4m NloS

3.2. PRUEBAS DE RECEPCIÓN PARA EL DESPLAZAMIENTO DE FASE.

3.2.1. DISTANCIA

Se procede a realizar las pruebas para el cálculo del desplazamiento de fase con distancias de separación entre transmisor y receptor de 1 metro y 2 metros y modulación BPSK en el esquema de transmisión OFDM para un número de 15 muestras.

Los datos se encuentran almacenados en Fase1m.mat y Fase2m.mat y junto la script fase_aut.m(VER ANEXO III) se obtienen las siguientes gráficas:

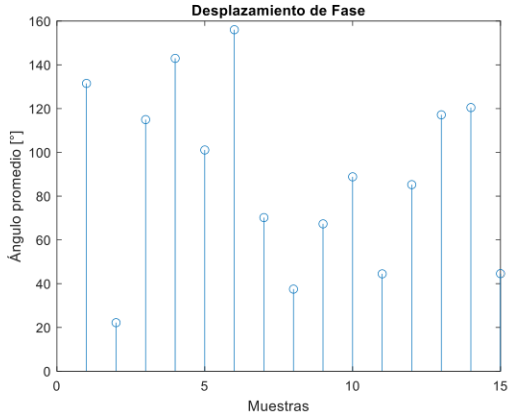


Figura 55. Desplazamiento de fase a una distancia de 1m.

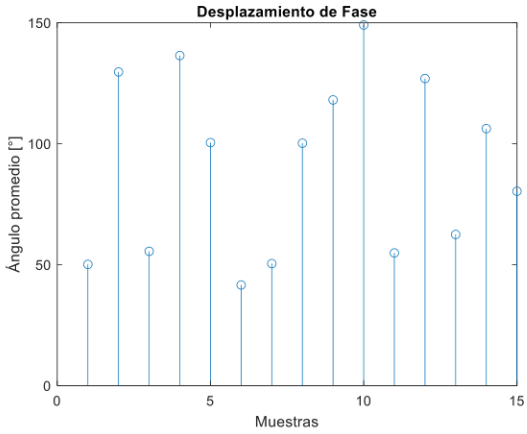


Figura 56. Desplazamiento de fase a una distancia de 2m.

3.2.1.1. RESULTADOS CON DIFERENTES DISTANCIAS

A la distancia de un metro, el promedio del desplazamiento de fase para las 15 muestras de 89.64°, mientras que, a una distancia de 2 metros , el promedio es de 90.8°, sin embargo, en la Figura 57, los datos se encuentran más dispersos que en la Figura 58, donde se observa valores casi constantes entre los 50 grados.

3.2.2. MODULACIONES

Se procede a realizar las pruebas para el cálculo del desplazamiento de fase con una distancia de separación entre transmisor y receptor de 1 m y diferentes modulaciones en el esquema de transmisión OFDM para un número de 15 muestras.

- **BPSK**

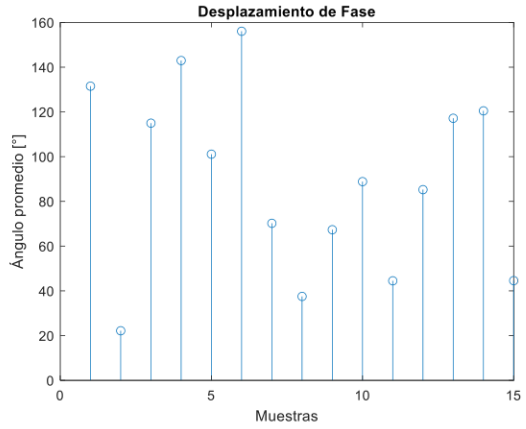


Figura 57. Desplazamiento de fase para una modulación BPSK.

- **QPSK**

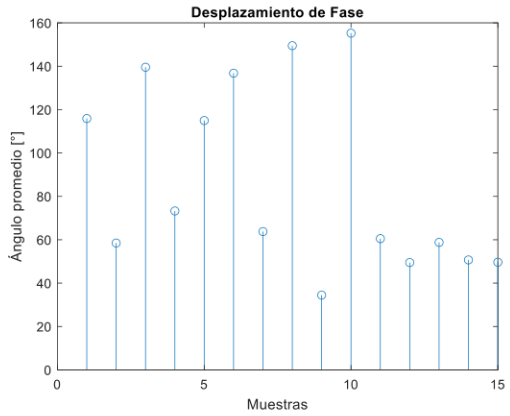


Figura 58. Desplazamiento de fase para una modulación QPSK.

- **16-QAM**

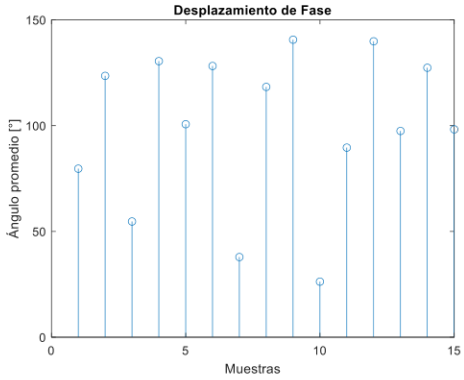


Figura 59. Desplazamiento de fase para una modulación 16-QAM.

- **64-QAM**

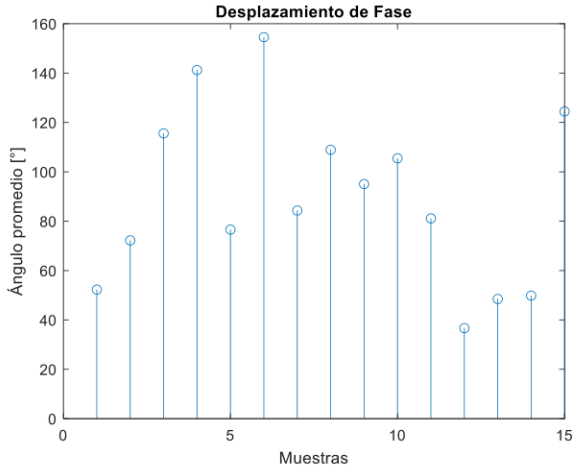


Figura 60. Desplazamiento de fase para una modulación 64-QAM.

3.2.2.1. RESULTADOS CON DIFERENTES MODULACIONES.

El promedio del desplazamiento de fase para la modulación BPSK es de 89.64° mientras que para QPSK es 87.39°. Para 16-QAM es 99.53° y para 64-QAM es de 89.79°.

Tanto en BPSK, QPSK como 16-QAM, los datos se encuentran dispersos, mientras que en la modulación 64-QAM los datos podrían encajar con una distribución de probabilidad normal.

3.3. RESULTADOS

Efectuado los cálculos y correcciones planteados en este trabajo, se dispone a poner en funcionamiento el prototipo de sistema de comunicación, para comparar mediante la medida Bit Error Rate (BER), cuan efectivo resulta las técnicas planteadas para compensar los errores.

A continuación, se presenta un cuadro, donde se muestran los resultados del BER, utilizando diferentes tipos de mapeo y variando la corrección de CFO y de fase.

Tabla 3. Resultado de prueba realizada a 1 metro.

MODULACIÓN	CORRECCIÓN DE (CFO)	CORRECCIÓN DE FASE	BER (Bit Errados/ Bit Totales)
BPSK	ON	ON	3006/11632
	ON	OFF	3506/11632
	OFF	ON	3515/11632
	OFF	OFF	3520/11632
QPSK	ON	ON	1016/11632
	ON	OFF	3440/11632
	OFF	ON	1437/11632
	OFF	OFF	2623/11632
16-QAM	ON	ON	4245/11632
	ON	OFF	5690/11632
	OFF	ON	5495/11632
	OFF	OFF	5754/11632
64-QAM	ON	ON	5262/11632
	ON	OFF	5607/11632
	OFF	ON	5615/11632
	OFF	OFF	5618/11632

3.4. CONCLUSIONES

En base a las pruebas realizadas se obtuvo que, a una distancia de 18 metros con línea de vista, el CFO permanece constante con 2,38kHz para las modulaciones BPSK, QPSK, 64-QAM a excepción de la modulación 16-QAM donde varía a 2,65kHz. Por otro lado, a la misma distancia pero sin línea de vista, el CFO varía para cada modulación, obteniéndose el menor valor 2,43kHz para la modulación BPSK y el mayor valor 2,64kHz para la modulación 64-QAM.

También se pudo notar que con línea de vista las funciones de densidad que más se ajustan presentan asimetría hacia la izquierda y para modulaciones BPSK y QPSK las muestras tienen mayor ocurrencia hasta valores de 3000Hz y luego empiezan a decaer. Por otra parte, para modulaciones 16-QAM y 64-QAM, pese a decaer pasado los 3000 Hz, aumenta de manera significativa a partir de valores de 4500 Hz. En cambio, sin línea de vista, pese a que se mantiene la asimetría a la izquierda, la ocurrencia luego de valores de 3000 Hz no es tan notoria para modulaciones 16-QAM y 64-QAM, mientras que para las modulaciones BPSK y QPSK se mantiene casi constante una vez pasado los 3000Hz.

Por lo anterior se podría concluir que el CFO para diferentes tipos de modulaciones no presenta mayor variación en cuanto a su valor numérico, haciendo que no sea relevante considerar el tipo de mapeo para la estimación de CFO. Por otro lado, la distribución de los datos que si presenta variación con el cambio de modulaciones, no se lo consideraría como un factor que influye en el cálculo y compensación del desplazamiento de frecuencia portadora.

En cuanto a la distancia, tanto con línea de vista como sin línea de vista, el CFO tiene densidades más homogéneas con distancias de 9m y 12.6m, lo que no sucede a la distancia de 5.4 m, por lo que, mientras más cerca se encuentre el receptor del transmisor, los valores de CFO se mantienen por debajo de los 3000 Hz. A distancias mayores, los valores de CFO de 5000 Hz son tan probables como los valores de 3000 Hz.

Para el desplazamiento de fase, mientras más cerca se encuentre el receptor del transmisor, en el caso de este trabajo una distancia de 1metro, los datos no siguen ningún patrón y varían desde 20° hasta 160°. Por el contrario, a mayor distancia, en este caso el doble de la distancia anterior, el 40% de los datos se mantienen con un ángulo de 50° aproximado. Sin embargo, el promedio de todas las muestras, tanto para una distancia de

1 metro(89.64°) como para 2 metros(90.79°), apenas varían en 1°, en consecuencia, la distancia, no es un factor a considerar para la compensación del desplazamiento de fase.

Respecto a las modulaciones con el desplazamiento de fase, el valor promedio de los ángulos varía hasta 2° para BPSK(89.64°), QPSK(87.39°) y 64-QAM(89.79°) siendo la excepción la modulación 16-QAM con 99.53°. Por lo que, se debe tomar en cuenta, la modulación utilizada para compensar el desplazamiento de fase.

Respecto al funcionamiento del sistema de comunicaciones implementado con las diferentes técnicas de compensación tanto para el CFO, con el algoritmo para medir la diferencia de picos, y para el desplazamiento de fase, con la utilización de las secuencias de entrenamiento, se obtuvo que es eficiente los métodos utilizados para una modulación QPSK, obteniéndose el mejor resultado con un BER de 1016/11632 , equivalente a que el 8,73% de la información llega incorrecta. Para otro tipo de modulaciones, las técnicas utilizadas sí brindan una leve mejoría; sin embargo, se debería buscar otras alternativas para la compensación de errores.

3.5. RECOMENDACIONES

El sistema de comunicaciones está creado para trabajar con bloques de longitud igual a 800000 muestras, lo cual necesita una gran cantidad de procesamiento, por lo que se recomienda no mantener más de media hora ejecutándolo en computadores con bajo procesador. Una alternativa sería disminuir en el código la longitud de cada muestra a 400000 , ya que todos los datos se encuentran almacenados en los primeros 200000.

Los equipos SDR Adalm Pluto soportan comunicaciones half y full dúplex, por lo que cada equipo viene integrado con dos antenas, una para transmisión y otra para recepción. En el sistema de comunicaciones utilizado, se realiza una comunicación simplex, por lo que no es necesario conectar las dos antenas en cada equipo y se recomienda solo conectar una de ellas en cada equipo para obtener mejores resultados y evitar interferencias con las antenas que no se están utilizando.

El sistema de comunicaciones está desarrollado completamente en script, sin usar alguna interfaz gráfica, por lo que, para utilizarlo con los diferentes tipos de mapeo, es necesario, realizar el cambio manualmente, descomentando la sección pertinente. Cada sección se encuentra debidamente rotulada.

4. REFERENCIAS BIBLIOGRÁFICAS

[1] K. Guo, W. Xu, G. Zhou, "Differential Carrier Frequency Offset and Sampling Frequency Offset Estimation for 3GPP LTE," 2011 IEEE 73rd Vehicular Technology Conference (VTC Spring), 2011, pp. 1-5, doi: 10.1109/VETECS.2011.5956576.

[2] M. Jeong, S. Choi, J. Kim, Juyeop. "Designing a Robust Frequency Offset Estimation Scheme For Target Decoding Performance in an OFDM System" 2021. [En línea] Disponible: https://www.researchgate.net/publication/353208688_Designing_a_Robust_Frequency_Offset_Estimation_Scheme_Satisfying_Target_Decoder_Performance_in_an_OFDM_System

[3] Madelin Gabriela Gordón Enríquez, Francisco Javier Hidalgo Pazmiño, "Desarrollo de un prototipo de comunicación inalámbrica de uso didáctico utilizando equipos SDR, incluyendo la técnica OFDM y el servicio de confidencialidad", EPN, Octubre 2020, Quito-Ecuador.

[4] W. Hou, X. Wang, J. Chouinard and A. Refaey, "Physical Layer Authentication for Mobile Systems with Time-Varying Carrier Frequency Offsets," in IEEE Transactions on Communications, vol. 62, no. 5, pp. 1658-1667, May 2014, doi: 10.1109/TCOMM.2014.032914.120921.

[5] Yong Soo Cho; Jaekwon Kim; Won Young Yang; Chung G. Kang, "Synchronization for OFDM," in MIMO-OFDM Wireless Communications with MATLAB®, IEEE, 2010, pp.153-185, doi: 10.1002/9780470825631.ch5.

[6] Vijay, Chanamala & Rao, Gottapu & Minchula, Vinodh. (2019). Carrier Frequency Offset Impact on LTE-OFDM Systems: Proceedings of the Fourth ICMEET 2018. 10.1007/978-981-13-1906-8_42.

[7] T. F. Collins, R. Getz, D. Pu, A. M. Wyglinski, "Software-Defined Radio for Engineers", 2018. 1era. Ed., pp 275-284. [En línea]. Disponible: https://www.analog.com/media/en/training-seminars/design-handbooks/Software-Defined-Radio-for-Engineers-2018/SDR4Engineers.pdf

[8] A. Ali and G. Fischer, "The Phase Noise and Clock Synchronous Carrier Frequency Offset based RF Fingerprinting for the Fake Base Station Detection," 2019 IEEE 20th Wireless and Microwave Technology Conference (WAMICON), 2019, pp. 1-6, doi: 10.1109/WAMICON.2019.8765471.

[9] Analog Devices, "Adalm Pluto" 2022. Accedido: 07 de enero de 2022. [En línea]. Disponible en: https://www.analog.com/en/design-center/evaluation-hardware-and-software/evaluation-boards-kits/adalm-pluto.html

[10] MathWorks, “Communications Toolbox Support Package for Analog Devices ADALM-Pluto Radio”. Accedido: 20 de noviembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/supportpkg/plutoradio/index.html?s_cid=doc_ftr

[11] Mathworks, “Signal Analyzer”. 2021. Accedido: 13 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/signal/ref/signalanalyzer-app.html

[12] Mathworks, “pspectrum”. 2021. Accedido: 18 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/signal/ref/pspectrum.html

[13] Mathworks, “findpeaks”. 2021. Accedido: 18 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/signal/ref/findpeaks.html

[14] Mathworks, “max” 2021. Accedido: 20 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/matlab/ref/max.html

[15] Mathworks, “Find Arrays Elements That Meet a Condition” 2021. Accedido: 20 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/matlab/matlab_prog/find-array-elements-that-meet-a-condition.html

[16] Mathworks “Frequency Correction for Adalm-Pluto Radio”Accedido: 20 de diciembre de 2021. [En línea]. Disponible en: https://www.mathworks.com/help/supportpkg/plutoradio/ug/frequency-correction-for-adalm-pluto-radio.html

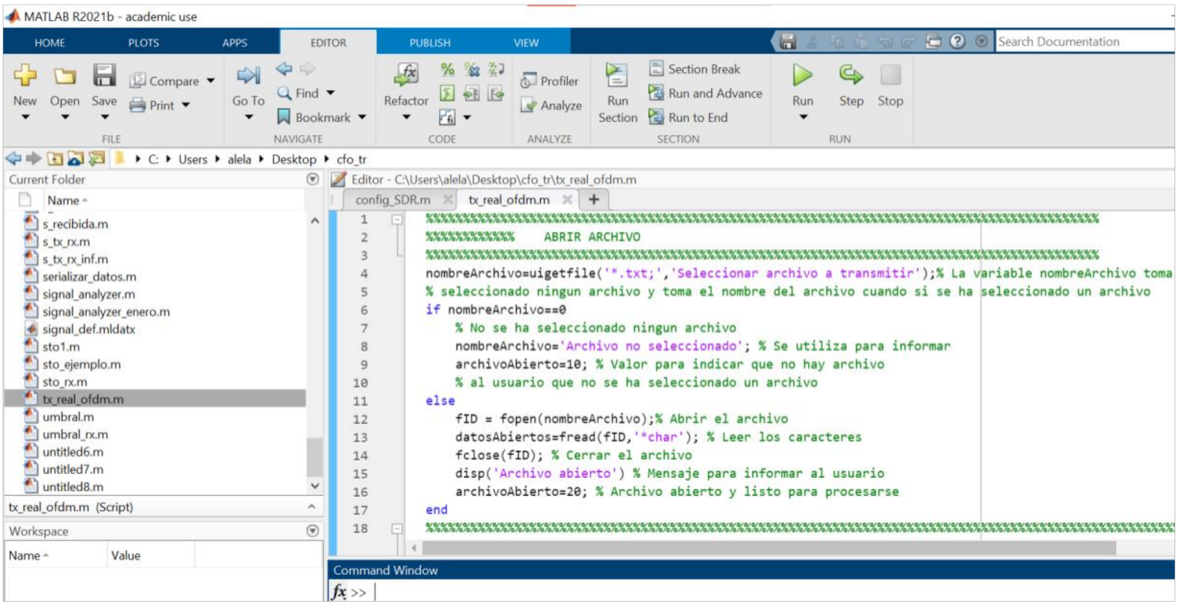
[17] Mathworks, “WLAN PPDU Structure” 2022. Accedido: 15 de enero de 2022. [En línea]. Disponible en: https://www.mathworks.com/help/wlan/gs/wlan-ppdu-structure.html

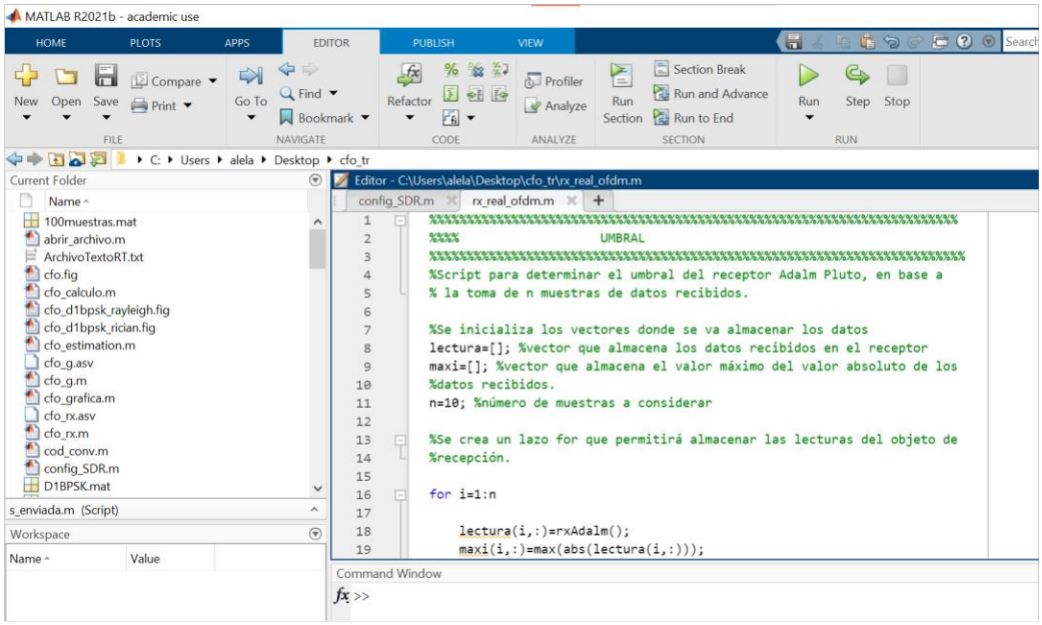
5. ANEXOS

ANEXO I. MANUAL DE USUARIO

1. Poner en funcionamiento el sistema de comunicaciones, requiere conectar en dos puertos USB diferentes del computador, cada equipo SDR Adalm Pluto.

2. Se debe abrir dos sesiones de Matlab. En la primera se debe abrir los archivos config_SDR.m y tx_real_ofdm.m. En la segunda sesión, se debe abrir los archivos config_SDR.m y rx_real_ofdm.m, como se muestra a continuación



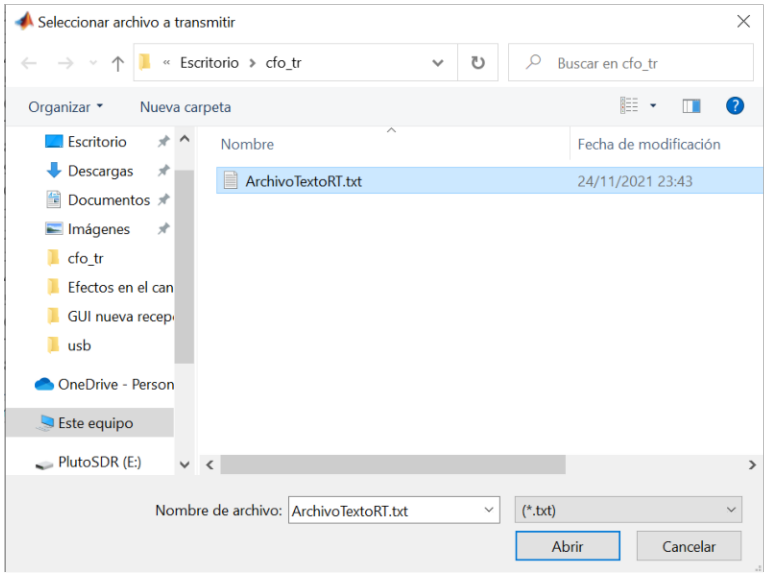


3. Se debe ejecutar en cada sesión el script **config_SDR.m** para activar los equipos como transmisor y receptor, donde se obtendrá lo siguiente:

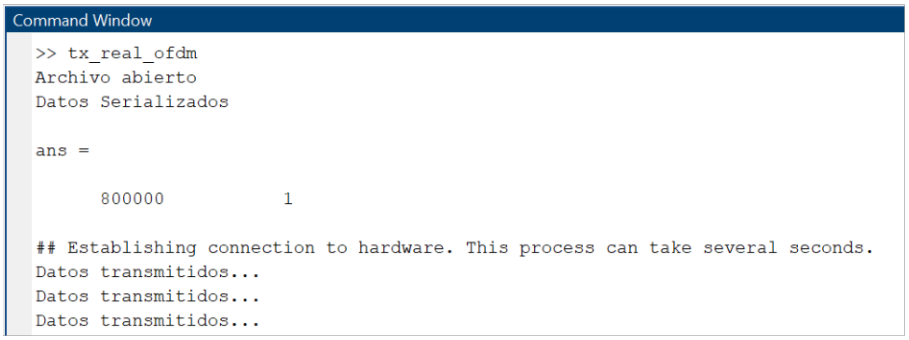
```
Información de los objetos:
Objeto para transmisión
## Establishing connection to hardware. This process can take several seconds.

ans =

  struct with fields:

                    Status: 'Full information'
           CenterFrequency: 859999998
         BasebandSampleRate: 2000000
                  SerialNum: '104400b839910013fbff260095c7642567'
                       Gain: -1
      RadioFirmwareVersion: "0.31"
    ExpectedFirmwareVersion: "0.31"
           HardwareVersion: "B1"
```

```
Objeto para recepción
## Establishing connection to hardware. This process can take several seconds.

ans =

  struct with fields:

                    Status: 'Full information'
           CenterFrequency: 859999998
         BasebandSampleRate: 2000000
                  SerialNum: '104400b839910013fbff260095c7642567'
                       Gain: 70
      RadioFirmwareVersion: "0.31"
    ExpectedFirmwareVersion: "0.31"
           HardwareVersion: "B1"
   EnableQuadratureTracking: 1
         EnableRFDCTracking: 1
   EnableBasebandDCTracking: 1
        FrequencyCorrection: 0
```

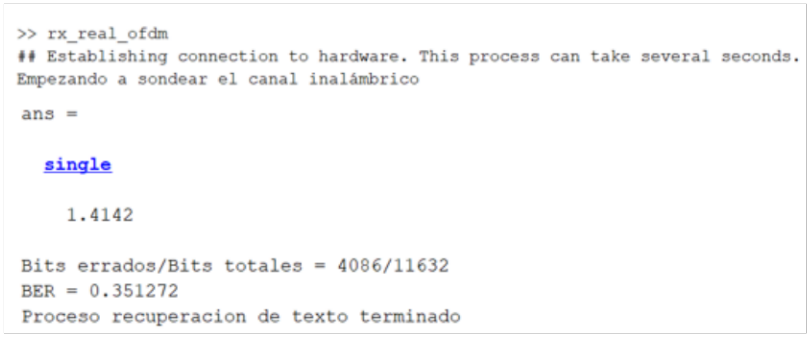
4. Una vez ejecutado el script **config_SDR** en ambas sesiones, se procede a ejecutar en primera instancia el script **tx_real_ofdm.m** y seleccionar el archivo “ArchivoTextoRT.txt”



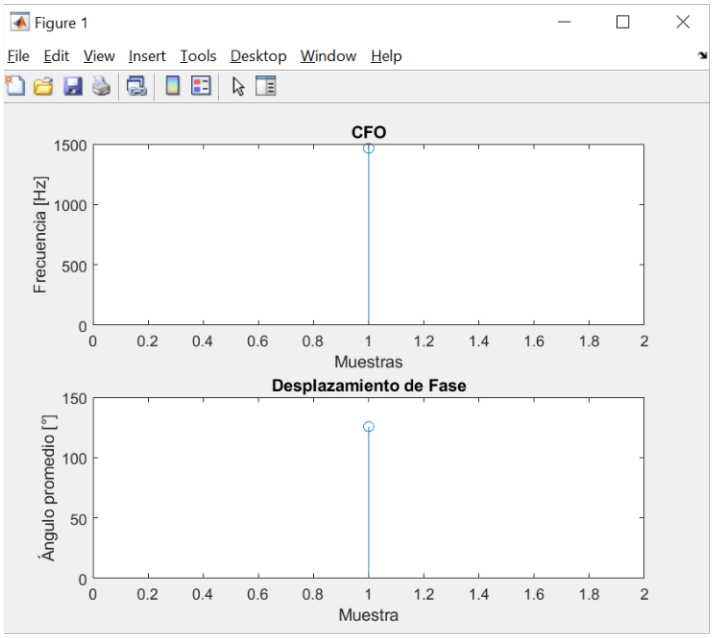
5. Seleccionado el archivo, comenzará la transmisión y se observará en el Command Window, lo siguiente:



6. En la otra sesión de Matlab, se dispondrá a ejecutar el script **rx_real_ofdm.m**, y se empezará a sondear en canal en busca de una muestra que supere el umbral.



7. Finalmente, cuando una muestra se considere válida, se mostrará una figura que muestra el CFO y el ángulo desplazado de esa muestra. Y se irá actualizando a medida que se encuentren más muestras válidas.



8. Para obtener las gráficas de distribución, se ejecuta primero el script cfo_cálculo.,m y luego el script histograma.m que se encontrarán en los anexos siguientes

ANEXO II. SCRIPTS PARA EL FUNCIONAMIENTO DEL SISTEMA DE COMUNICACIONES

ArchivoTextoRT.txt

Este texto se va a transmitir y recebir utilizando dispositivos SDR Adalm Pluto con Matlab 2021a.
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
a e i o u á é í ó ú ñññññ(x5) [] {} () ** ¿? ¡!
Fragmento de texto de la novela Crónica de una muerte anunciada de Gabrial García Márquez.

"Prometió ocuparse de eso al instante, pero entró en el Club Social a confirmar una cita de dominó para esa noche,
y cuando volvió a salir ya estaba consumado el crimen.Cristo Bedoya cometió entonces su único error mortal: pensó
que Santiago Nasar había resuelto a última hora desayunar en nuestra casa antes de cambiarse de ropa, y allá se
fue a buscarlo. Se apresuró por la orilla del río, preguntándole a todo el que encontraba si lo habían visto pasar,
pero nadie le dio razón. No se alarmó, porque había otros caminos para nuestra casa. Próspera Arango, la cachaca,
le suplicó que hiciera algo por su padre que estaba agonizando en el sardinel de su casa, inmune a la bendición
fugaz del obispo. «Yo lo había visto al pasar -me dijo mi hermana Margot-, y ya tenía cara de muerto.» Cristo Bedoya
demoró cuatro minutos en establecer el estado del enfermo, y prometió volver más tarde para un recurso de urgencia,
pero perdió tres minutos más ayudando a Próspera Arango a llevarlo hasta el dormitorio. Cuando volvió a salir sintió
gritos remotos y le pareció que estaban reventando cohetes por el rumbo de la plaza..."

config_SDR.m

```
% Script para mostrar un ejemplo de configuración del dispositivo SDR
% adalm Pluto
encontrarSDR=findPlutoRadio; %Busca la identificación de la radio
```

```
% y el número de serie
numeroEncontrado=length(encontrarSDR); % toma el valor de 0 si no
% se encuentra ningun dispositivo conectado
if(numeroEncontrado==0) %No está conectado ningún dispositivo
    disp('No se encontró ningun dispositivo SDR conectado.')
else
    % Se ha detectado almenos un dispositivo SDR conectado
    frecuenciaCentral=860e6; % Frecuencia central
    frecuenciaMuestreo=2.0e6; % Frecuencia de muestreo en banda base
    % Configuración del objeto para la tx:
    txAdalm=sdrtx('Pluto',...
        'RadioID','usb:0',...
        'CenterFrequency',frecuenciaCentral,...
        'Gain',-1,...
        'BasebandSampleRate',frecuenciaMuestreo);
    disp('Objeto para la transmisión configurado:')
    txAdalm % Para comprobar las propiedades configuradas
    % Configuración del objeto para la rx:
    rxAdalm = sdrrx('Pluto',...
        'RadioID','usb:0',...
        'CenterFrequency',frecuenciaCentral,...
        'GainSource','Manual',...
        'Gain',70,...
        'BasebandSampleRate',frecuenciaMuestreo,...
        'OutputDataType','single',...
        'SamplesPerFrame',800000,...
        'ShowAdvancedProperties',1,...
        'FrequencyCorrection',0);
    disp('Objeto para la recepción configurado:')
    rxAdalm % Para comprobar las propiedades configuradas
    disp('Información de los objetos:')
    disp('Objeto para transmisión')
    info(txAdalm) %Muestra algunas propiedades de txAdalm
    disp('Objeto para recepción')
    info(rxAdalm) %Muestra algunas propiedades de rxAdalm
end
```

tx_real_ofdm.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%%%%%%    ABRIR ARCHIVO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
nombreArchivo=uigetfile('*.txt;','Seleccionar archivo a transmitir');% La
variable nombreArchivo toma el valor de 0 cuando no se ha
% seleccionado ningun archivo y toma el nombre del archivo cuando si se ha
seleccionado un archivo
if nombreArchivo==0
    % No se ha seleccionado ningun archivo
    nombreArchivo='Archivo no seleccionado'; % Se utiliza para informar
    archivoAbierto=10; % Valor para indicar que no hay archivo
    % al usuario que no se ha seleccionado un archivo
else
    fID = fopen(nombreArchivo);% Abrir el archivo
    datosAbiertos=fread(fID,'*char'); % Leer los caracteres
```

```
    fclose(fID); % Cerrar el archivo
    disp('Archivo abierto') % Mensaje para informar al usuario
    archivoAbierto=20; % Archivo abierto y listo para procesarse
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%  SERIALIZAR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%
textoBinario = dec2bin(datosAbiertos,8); % representacion de cada caracter
con 8 bits (codificación)
% Para serializar los datos se utiliza dos lazos for(). El primer
% lazo for() recorre cada fila, length(textoBinario) representa el
% número de filas. El segundo lazo for() en cada fila recorre todos
% los elementos (8 columnas). Y con una variable auxiliar (aux) se
% almacena los datos en la posición que dicte la variable aux.
aux=1; % Inicializar variable auxiliar
for i = 1:length(textoBinario)
    for j=1:8
        textoBitsSerial(aux)=textoBinario(i,j); % tipo char
        aux=aux+1; % Sumar 1 en la variable auxiliar
    end
end

% Para convertir los datos del tipo char al tipo str se utiliza un
% lazo for para recorrer todos los elementos del vector serializado
for aux=1:length(textoBitsSerial)
    % Se utiliza una variable auxiliar para almacenar los datos
    datosS_aux(aux)=str2double(textoBitsSerial(aux));
end

% Se copia los datos en una variable
datosSerializados=datosS_aux; %datos serializados
disp('Datos Serializados') % Mostrar un msj en el Command Window
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%
%%%%%%% CODIFICADOR CONVOLUCIONAL ( CORRECCION DE ERRORES)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%
datosBina=datosSerializados; % copiar el vector de entrada

% Primero se define los valores a considerar de las etapas posteriores
NPdatos=40; % Número de portadoras de datos por símbolo OFDM
Nbps_M=2; % Bits por símbolo en la modulación QPSK
Nsps=4; % Muestras por símbolo en la etapa reformado

% Parámetros del codificador convolucional:
nc=2; % n es la longitud por cada ingreso de k bits
kc=1; % k es los bits de entrada
mc=3; % m es el tamaño de registro de la memoria
g1=5; % Polinomio generador
g2=7; % Polinomio generador
Rcc=kc/nc; % tasa del codificador convolucional

% Cálculo del número de bits serializados que se tiene por cada símbolo OFDM:
Nbits=Nbps_M*NPdatos*Rcc*(1/Nsps);
% Cálculo para rellenar el vector de entrada, y así obtener un vector
% que sea múltiplo de Nbits:
```

```
NumSimb=ceil(length(datosBina)/Nbits); % Número de simbolos OFDM
longitudPadd=Nbits*NumSimb-length(datosBina); % Longitud de relleno
vector_padd=zeros(1,longitudPadd); % vector de relleno de ceros
datosConPadd=[datosBina vector_padd]; % Datos serializados con relleno

% Creación de la estructura de Trellis:
P_Trellis = poly2trellis(mc, [g1 g2]);

% Codificador:
datosCodConv=[]; % se inicializa la variable que va a almacenar los
% datos codificados y serializados
% Se recorre los todos los símbolos OFDM con un lazo for
for i=1:NumSimb
    auxDatosCc = datosConPadd(1,(Nbits*(i-1)+1):Nbits*i); % variable
    % auxiliar para almacenar los datos a codificar
    % Codificación convolucional:
    CodConv = convenc(auxDatosCc, P_Trellis);
    datosCodConv=[datosCodConv CodConv]; % Datos de salida
end
datosBinario=datosCodConv; % Copiar los datos de salida en una variable

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
%%%%%%%                 MAPEO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%5
% Script para mostrar un ejemplo para la etapa de mapeo, se utiliza
% la modulación digital QPSK, implementada con función directa
% Como argumento de entrada se tiene una vector fila tipo double,
% este vector se obtiene en el scritp cod_conv.m
% datosBinario es el vector de entrada

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%
% Mapeo BPSK
    datosColumna=datosBinario.'; % Se transpone la matriz que contiene
    % los datos de entrada para obtener un vector columna
    % Se recorre los cada bit de entrada y se asigna un símbolo
    for i=1:length(datosColumna)
        % Con if se discrimina si es igual a 1 y se asigna el
        % valor 0.707+0.707i, caso contrario es 0 y se asigna el
        % valor de -0.707-0.707i
        if(datosColumna(i)==1)
            datosMapeados(i,1)=complex(0.707,0.707);
        else
            datosMapeados(i,1)=complex(-0.707,-0.707);
        end
    end

%%
% MAPEO QPSK
% M = 4; % Puntos en la constelación (símbolos), para QPSK igual a 4
% Nb=log2(M); % Bits por símbolo
% % Hay que asegurarse que la longitud de los datos de entrada sea
% % múltiplo de Nb, por lo que se rellena de ceros cuando no es múltiplo
% % de Nb.
% % Se calcula la longitud de relleno y se almacena en long_padd0
```

```
% long_padd0=ceil(length(datosBinario)/Nb)*Nb-length(datosBinario);
% paddQ=zeros(1,long_padd0); % Vector de relleno con ceros
% datosbin=[datosBinario paddQ]; % Vector con relleno
% % Pasar de binario a símbolos decimales:
% dataMatriz=reshape(datosbin,Nb,[]).'; %Nb columnas, cada fila es
% % un símbolo
% dataSimb=bi2de(dataMatriz,'left-msb'); % Símbolos a modular
% % Modulador:
% % Factor de normalización = 1/sqrt(2)
% datosMapeados = (1/sqrt(2))*pskmod(dataSimb,M,pi/4,'gray'); % Datos
% % mapeados con QPSK

%%
%MAPEO CON 16-QAM

%  M = 16; % Puntos en la constelación (símbolos)
%  Nb=log2(M); % Bits por símbolo, se usan 4 bits por símbolo
% % Hay que asegurarse que la longitud de los datos de entrada sea
% % múltiplo de Nb, por lo que se rellena de ceros cuando no es
% % múltiplo de Nb.
% % Se calcula la longitud de relleno y se almacena en long_padd0
% long_padd0=ceil(length(datosBinario)/Nb)*Nb-length(datosBinario);
% paddQ=zeros(1,long_padd0); % Vector de relleno con ceros
% datosbin=[datosBinario paddQ]; % Vector con relleno
% %Pasar de binario a símbolos decimales:
% dataMatriz=reshape(datosbin,Nb,[]).'; % Matriz de Nb columnas y
% % cada fila corresponde a un símbolo
% dataSimb=bi2de(dataMatriz,'left-msb'); % Símbolos a modular
% % Modulador:
% % Factor de normalización = 1/sqrt(10)
% datosMapeados = (1/sqrt(10))*qammod(dataSimb,M); % Modulación 16-QAM

%%
%MAPEO 64-QAM
% M = 64; % Puntos en la constelación (símbolos)
% Nb=log2(M); % Bits por símbolo, se usan 6 bits por símbolo
% % Hay que asegurarse que la longitud de los datos de entrada sea
% % múltiplo de Nb, por lo que se rellena de ceros cuando no es
% % múltiplo de Nb.
% % Se calcula la longitud de relleno y se almacena en long_padd0
% long_padd0=ceil(length(datosBinario)/Nb)*Nb-length(datosBinario);
% paddQ=zeros(1,long_padd0); % Vector de relleno con ceros
% datosbin=[datosBinario paddQ]; % Vector con relleno
% %Pasar de binario a símbolos decimales:
% dataMatriz=reshape(datosbin,Nb,[]).'; % Matriz de Nb columnas y
% % cada fila corresponde a un símbolo
% dataSimb=bi2de(dataMatriz,'left-msb'); % Símbolos a modular
% % Modulador:
% % Factor de normalización = 1/sqrt(42)
% datosMapeados = (1/sqrt(42))*(qammod(dataSimb,M)); % Datos
% % mapeados con 64-QAM
%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%
%%%%%%% REFORMADO DE PULSO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%
```

```
% Parámetros del filtro de transmisión:
rolloff = 0.5;  % Factor de caída del filtro
span = 6;       % Define el tamaño de la ventana en símbolos
sps = 4;        % # de muestras por cada símbolo
% En el filtro de transmisión se define el parámetro 'Shape' como
% 'Square root' para obtener un filtro SRRC
% Creación del filtro SRRC:
txfilter = comm.RaisedCosineTransmitFilter(...
    'Shape','Square root',...
    'RolloffFactor',rolloff, ...
    'FilterSpanInSymbols',span, ...
    'OutputSamplesPerSymbol',sps);
%fvtool(txfilter,'Analysis','impulse') % Análisis de la despuesta al
% impulso del filtro SRRC creado.
% A los datos de entrada datosMapeados de aplica el filtro txfilter
% y se almacena en la variable datosReformado
datosReformado=txfilter(datosMapeados); % datos filtrados
% % % Graficar el digramas de constelación de los datos filtrados y de
% % % los datos modulados normalizados (datos de entrada):
% figure()
% % % Graficar los primeros datos de los datos modulados normalizados,
% % % ya que la constelación QPSK solo tiene 4 símbolos
% plot(datosMapeados(1:100,1),'r.','MarkerSize',20);
% hold on
% % % Graficar los puntos de los datos filtrados, con azul
% plot(datosReformado,'b.');
% title('Señal antes y después del filtrado SRRC')
% legend('Antes','Después')
% xlabel('I (fase)');
% ylabel('Q (cuadratura)');
% axis([-0.6 0.9 -0.6 0.6]) % Set axis ranges
% hold off


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%
%%%%%%%                 CREACIÓN DEL SÍMBOLO OFDM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%
% El vector de salida de la etapa es simbolosOFDMCreados
% La estructura del símbolo OFDM consta de una subportadora DC,
% 4 pilotos, 19 de guarda y 40 de datos.
nPdatos=40; % Número de portadoras de datos
% El vector de entrada debe ser múltiplo de nPdatos, cuando no se
% se cumple dicha condición se realiza un relleno con los datos del
% inicio al final de vector. Esto con el objetivo de hacer que el
% vector sea múltiplo de nPdatos:
longDatos=length(datosReformado); % Longitud del vector de entrada
nSimbOFDM=ceil(longDatos/nPdatos); % Calcular el número de símbolos
longPadd=nSimbOFDM*nPdatos-longDatos; % Calcular la longitud del
% relleno
datosEntradaPadd=datosReformado; % Copio los datos de entrada en una
% nueva variable auxiliar
% Si la longitud de relleno es cero no se realiza ninguna operación
% adicional, si es mayor que cero se realiza el relleno
if(longPadd>0)
    % Se hace el relleno con los datos del inicio del vector de entrada
    datosEntradaPadd(end+1:end+longPadd)=datosReformado(1:longPadd);
```

```
end
% Se procede a crear los simbolos OFDM recorriendo en múltiplos de
% nPdatos, mediante un lazo for:
numeroSimbolos=1; % Inicializo la variable que sirve para almacenar los
% símbolos OFDM creados, en una matriz.
longDatosPadd=length(datosEntradaPadd); % Longitud de los datos de
% entrada con relleno
for i=1:nPdatos:longDatosPadd
    datosAux=datosEntradaPadd(i:i+nPdatos-1); %Copio los datos nPdatos,
    % es decir los 40 datos a introducirse dentro del símbolo OFDM
    DC=0;% Valor para la subportadora DC
    % Las variables p1, p2, p3, p4 son los valores que toman las
    % subportadoras piloto, usadas para poder hacer una estimación del
    % canal en recepción
    p1=1;
    p2=1;
    p3=1;
    p4=1;
    simbOFDM=zeros(64,1); % Inicializo símbolo OFDM, con todos los
    % valores en cero
    simbOFDM(1)=DC; % Portadora DC
    % Asignación del valor a cada subportadora de datos, y a las
    % subportadoras piloto:
    simbOFDM(2)=datosAux(21);
    simbOFDM(3)=datosAux(22);
    simbOFDM(4)=datosAux(23);
    simbOFDM(5)=datosAux(24);
    simbOFDM(6)=datosAux(25);
    simbOFDM(7)=datosAux(26);
    simbOFDM(8)=p1; % Piloto
    simbOFDM(9)=datosAux(27);
    simbOFDM(10)=datosAux(28);
    simbOFDM(11)=datosAux(29);
    simbOFDM(12)=datosAux(30);
    simbOFDM(13)=datosAux(31);
    simbOFDM(14)=datosAux(32);
    simbOFDM(15)=datosAux(33);
    simbOFDM(16)=datosAux(34);
    simbOFDM(17)=datosAux(35);
    simbOFDM(18)=datosAux(36);
    simbOFDM(19)=datosAux(37);
    simbOFDM(20)=datosAux(38);
    simbOFDM(21)=datosAux(39);
    simbOFDM(22)=p2;% Piloto
    simbOFDM(23)=datosAux(40);
    % Indices no usados, dado el problema de la banda de guarda:
    simbOFDM(24)=1;
    simbOFDM(25)=1;
    % Valores nulos iguales a 0, desde simbOFDM(26) a simbOFDM(40),
    % usados como banda de guarda.
    % Indices no usados, dado el problema de la banda de guarda:
    simbOFDM(41)=1;
    simbOFDM(42)=1;
    % Asignación del valor a cada subportadora de datos, y a las
    % subportadoras piloto:
    simbOFDM(43)=datosAux(1);
    simbOFDM(44)=p3; % Piloto
    simbOFDM(45)=datosAux(2);
```

```
    simbOFDM(46)=datosAux(3);
    simbOFDM(47)=datosAux(4);
    simbOFDM(48)=datosAux(5);
    simbOFDM(49)=datosAux(6);
    simbOFDM(50)=datosAux(7);
    simbOFDM(51)=datosAux(8);
    simbOFDM(52)=datosAux(9);
    simbOFDM(53)=datosAux(10);
    simbOFDM(54)=datosAux(11);
    simbOFDM(55)=datosAux(12);
    simbOFDM(56)=datosAux(13);
    simbOFDM(57)=datosAux(14);
    simbOFDM(58)=p4; % Piloto
    simbOFDM(59)=datosAux(15);
    simbOFDM(60)=datosAux(16);
    simbOFDM(61)=datosAux(17);
    simbOFDM(62)=datosAux(18);
    simbOFDM(63)=datosAux(19);
    simbOFDM(64)=datosAux(20);
    simbolosOFDMCreados(1:64,numeroSimbolos)=simbOFDM; % Matriz de
    % salida que almacena el símbolo OFDM completo, según la columna
    % que dicte la variable numeroSimbolo
    numeroSimbolos=numeroSimbolos+1; % Aumento en uno la variable
end
numeroSimbolos=numeroSimbolos-1; % Disminuyo en uno el valor de la
% variable para que concuerde con la cantidad total de símbolos OFDM
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% IFFT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% En esta sección se realiza la modulación de los símbolos OFDM
% mediante la ifft. La matriz de entrada es simbolosOFDMCreados y la de
% salida es simbolosOFDM.
% La función ifft admite como entrada un vector columna, pero tambien
% una matriz. En la matriz aplica la ifft a cada columna. Como la
% matriz de entrada contiene un spimbolo en cada columna se puede
% aplicar directamente la función ifft
simbolosOFDM=ifft(simbolosOFDMCreados); % matriz de salida
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% AÑADIR PC %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% El prefijo cíclico (PC) se añade copiando la parte final de cada
% símbolo, al inicio del mismo.
% La matriz de entrada es simbolosOFDM y a la salida se tiene una
% matriz llamada simbolosOFDMpc con los símbolos en cada columna y
% un vector contodos los símbolos serializados llamado datosEnviar
% Con un lazo for se recorre cada símbolo modulado, se añade el PC
% y se serializa los datos
datosEnviar=[]; % Variable que almacenara los datos serializados
for auxPC=1:numeroSimbolos
    % Añadir el prefijo cíclico, de tamaño igual a la cuarta parte del
    % símbolo:

simbolosOFDMpc(1:80,auxPC)=[simbolosOFDM(49:64,auxPC);simbolosOFDM(:,auxPC)];
    datosEnviar=[datosEnviar;simbolosOFDMpc(:,auxPC)]; % Datos
    % serializados
end
% %% Graficar el primer símbolo con PC en el dominio del tiempo
% t=1:80; % Variable que representa el tiempo
% figure()
% subplot(2,1,1)
% % Graficar el PC y el símbolo sin PC
% plot(t,real(simbolosOFDMpc(:,1)),'r')
```

```
% hold on
% plot(t(16:80),real(simbolosOFDMpc(16:80,1)),'b')
% title('Símbolo OFDM con Prefijo Cíclico')
% xlabel('Muestras de tiempo')
% legend('Prefijo cíclico','Símbolo OFDM')
% xlim([0 105])
% hold off
% subplot(2,1,2)
% % Graficar el PC y las muestras finales que se usan como PC
% plot(t,real(simbolosOFDMpc(:,1)),'r')
% hold on
% plot(t(65:80),real(simbolosOFDMpc(65:80,1)),'g')
% plot(t(16:65),real(simbolosOFDMpc(16:65,1)),'b')
% title('Símbolo OFDM con Prefijo Cíclico')
% xlabel('Muestras de tiempo')
% legend('Prefijo cíclico','Muestra finales')
% xlim([0 105])
% hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%          PREÁMBULO 802.11a
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script para añadir el preámbulo 802.11a a los símbolos OFDM creados
% El script corresponde a la etapa Adición preámbulo 802.11a
% Previamente a ejecutar el script actual es necesario ejecutar en
% orden los scripts: abrir_archivo.m, serializar_datos.m, cod_conv.m,
% mapeo_datos.m, reformado_tx.m y ofdm_tx.m
% Como argumento de entrada del script actual se tiene un vector columna
% que contiene símbolos OFDM serializados, el vector tiene el nombre
% de datosEnviar y se obtiene del script ofdm_tx.m
% Se inicia por definir parte de la sección corta y larga en el dominio
% de la frecuencia para luego obtener el preambulo en el dominio del
% tiempo:
% 4 secuencias cortas (parte de LSTF) en el dominio de la frecuencia
parteCorta64=[0;0;0;0;-1.4720-1.4720i;0;0;0
    -1.4720-1.4720i;0;0;0;1.4720+1.4720i;0;0;0
    1.4720+1.4720i;0;0;0;1.4720+1.4720i;0;0;0
    1.4720+1.4720i;0;0;0;0;0;0;0;0;0;0;0;0;0;0
    1.4720+1.4720i;0;0;0;-1.4720-1.4720i;0;0;0
    1.4720+1.4720i;0;0;0;-1.4720-1.4720i;0;0;0
    -1.4720-1.4720i;0;0;0;1.4720+1.4720i;0;0;0];
% Secuencia larga (parte de LLTF) en el dominio de la frecuencia
parteLarga=[...
    0;1;-1;-1;1;1;-1;1;-1;1;-1;-1;-1;-1;-1;1
    1;-1;-1;1;-1;1;-1;1;1;1;1;0;0;0;0;0
    0;0;0;0;0;0;1;1;-1;-1;1;1;-1;1;-1;1
    1;1;1;1;1;-1;-1;1;1;-1;1;-1;1;1;1;1];
% Conversión del dominio de la frecuencia al tiempo, y creación de la
% sección corta completa (LSTF):
parteCorta64Time=ifft(parteCorta64); % pasar al dominio del tiempo
parteCorta80Time=parteCorta64Time(49:64);% copiar las ultimas
% 16 muestras al inicio de la variable
parteCorta80Time(17:80)=parteCorta64Time; % 5 secuencias cortas
preambulo=parteCorta80Time; % copiar las 5 secuencias cortas
preambulo(81:160)=parteCorta80Time; % añadir las otras 5 secuencias
% cortas para obtener la sección corta (LSTF)
% Conversión del dominio de la frecuencia al tiempo, y adición de PC
% de la sección larga:
parteLargaTime=ifft(parteLarga); % conversión al dominio del tiempo la
```

```
% secuencia larga
preambulo(161:192)=parteLargaTime(33:64); % PC en el preámbulo
preambulo(193:256)=parteLargaTime; % primera secuencia larga
preambulo(257:320)=parteLargaTime; % segunda secuecia larga y se obtiene
% el preámbulo completo.
% Unir el preambulo a los símbolos OFDM serializados:
datosConPre=[preambulo;datosEnviar]; % Preambulo con símbolos OFDM
% creados previamente
% Inicializar con ceros una variable auxiliar, que tenga el tamaño del
% bloque a enviar:
tamanioBloque=800000; % tamaño del bloque a transmitir
auxDat=zeros(tamanioBloque,1); % inicializar variable
[longDat,~]=size(datosConPre); % longitud de los datos a enviar
auxDat(1:longDat,1)=datosConPre; % copiar el preambulo con los símbolos
% OFDM y los demás datos del bloque se mantienen en cero
datosEnviarBloque=auxDat;% Datos de salida a enviar


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%            TRANSMISIÓN
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Procesar los datos a transmitir:
valorMaximoAux1=max(real(datosEnviarBloque));
valorMaximoAux2=max(imag(datosEnviarBloque));

if(valorMaximoAux1<valorMaximoAux2)
    valorMaximo=valorMaximoAux2*1.001;
else
    valorMaximo=valorMaximoAux1*1.001;
end

% Almaceno los datos a transmitir en una variable auxiliar
auxDat1=datosEnviarBloque/valorMaximo;
% Obervar el tamaño de los datos, como indicador de pantalla
size(auxDat1)

% Convertimos los datos a un tipo single que emplea 32bits con respecto a
double 64 bits para la representacion de cada valor
datosEnviados=single(auxDat1);

%datosEnviados=single(datosEnviar);

% se ingresa al lazo para transmitir cada intervalo intTime
txRealTime=20;% Variable para entrar al lazo de transmisión repetitiva
intTime=1;%Intervalo de tiempo en segundos para espera entre trasnmisiones
while (txRealTime==20)
    % se procede a transmitir el bloque de datos OFDM
    txAdalm(datosEnviados(:,1));
    % mostramos un mensaje de transmision terminada
    disp('Datos transmitidos...');
    pause(intTime)

end
```

rx_real_ofdm.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                      UMBRAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Script para determinar el umbral del receptor Adalm Pluto, en base a
% la toma de n muestras de datos recibidos.

%Se inicializa los vectores donde se va almacenar los datos
lectura=[]; %vector que almacena los datos recibidos en el receptor
maxi=[]; %vector que almacena el valor máximo del valor absoluto de los
%datos recibidos.
n=10; %número de muestras a considerar

%Se crea un lazo for que permitirá almacenar las lecturas del objeto de
%recepción.

for i=1:n

    lectura(i,:)=rxAdalm();
    maxi(i,:)=max(abs(lectura(i,:)));

end

%Se saca el promedio del vector maxi, que será el umbral que se ha
%determinado del receptor.

umbralrx=mean(maxi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%           RECEPCIÓN DE DATOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
condicion=1;

disp('Empezando a sondear el canal inalámbrico')
while(condicion==1)
    tic
    datosRecibidos=rxAdalm();
    max(abs(datosRecibidos))
    if(max(abs(datosRecibidos))> umbralrx)


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                   RECORTAR DATOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script que permite recortar los datos de información del bloque
% de datos recibido en el script recibir.m
% Antes de correr este script es necesario ejecutar el script
% config_SDR.m y recibir.m o bien se necesita cargar los datos
% guardados con load('datosRecibidos150.mat')
% En vector de entrada de esta etapa es datosRecibidos
% RECORTAR DATOS
%umbralrx=0.0040;
% Se recorre el vector de entrada y  se obtiene la posición del
% primer valor por arriba del umbral:
pos_inic=0; % Inicializar variable
for i=1:length(datosRecibidos)
    if(max(abs(datosRecibidos(i)))>umbralrx)
```

```
        pos_inic=i; % Primera posición arriba del umbral
        break;
    end
end
% Guardar desde 400 datos antes de superar el umbral, hasta el final:
if(pos_inic>400)
    datosRec=datosRecibidos(pos_inic-400:end); % Datos recortados
else
    % El umbral se superó antes de las primeras 400 posiciones, solo
    % se copia los datos
    datosRec=datosRecibidos;
end
datosInv=flip(datosRec); % Invierte el vector, tal que los datos
% finales queden al inicio, con esto se puede repetir el proceso
% anterior.
% Se recorre el vector de invertido y  se obtiene la posición del
% primer valor por arriba del umbral:
pos_fin=0; % Inicializar variable
for i=1:length(datosRec)
    if (max(abs(datosInv(i)))>umbralrx)
        pos_fin=i; % Primera posición arriba del umbral
        break
    end
end
% Guardar desde 200 datos antes de superar el umbral, hasta el final:
if(pos_fin>200)
    datosInvRec=datosInv(pos_fin-200:end); % Recortar datos
else
    % El umbral se superó antes de las primeras 200 posiciones, solo
    % se copia los datos
    datosInvRec=datosInv;
end
datosRecortados=flip(datosInvRec); % Invertir nuevamente el vector
% para obtener los datos recortados
% Graficar datos de entrada de la etapa y los datos recortados
% figure()
% subplot(2,2,1)
% plot(real(datosRecibidos))
% title('Datos recibidos (parte real)')
% subplot(2,2,3)
% plot(imag(datosRecibidos))
% title('Datos recibidos (parte imaginaria)')
% subplot(2,2,2)
% plot(real(datosRecortados),'g')
% title('Datos recortados (parte real)')
% subplot(2,2,4)
% plot(imag(datosRecortados),'g')
% title('Datos recortados (parte imaginaria)')

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%    DETECCIÓN DE TRAMA OFDM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script que permite encontrar la trama OFDM y realizar un ajuste en
% tiempo
% En vector de entrada de esta etapa es datosRecortados
% DETECCIÓN TRAMA OFDM:
datosRX=datosRecortados; % Copiar los datos en una nueva variable
SamplesPerFrame=800000; % tamaño del bloque recibido
```

```
% Los datos enviados nunca pueden ser mayor a SamplesPerFrame+600,
% el 600 se
c1=1; % Variable de control
if length(datosRX)>(SamplesPerFrame+600)
    disp('***** Muestras recibidas inválidas *****');
    c1=0;
end
% El preambulo se encuentra al inicio de cada paquete y por tanto
% se trabaja con las primeras muestras, con esto se evita
% procesamiento innecesario
Mmues=1400; % Muestras para calcular la métrica
if(length(datosRX)>Mmues)
    r=datosRX(1:Mmues);
else
    % Si es menor a Mmues se trabaja con todas las muestras
    r=datosRX;
end
L = 48; % Múltiplo dela longitud de una secuencia corta (16*x)
inMax = length(r)-2*L+1; % k+m puede tomar valores de hasta 2L-1
% el indice maximo a evaluar es la longitud de r menos 2L.
m=L-1; % Máximo valor de m
M = zeros(Mmues,1); % Inicializar la variable para mayor velocidad
% Determinar la métrica M(k) para comprobar si es un paquete OFDM:
for k=1:inMax
    Numerador = (r(k:k+m)'*r(k+L:k+m+L)); % Sumatorio del numerador
    Denom = sum((abs(r(k+L:k+m+L))).^2); % Sumatorio del denominador
    M(k)=Numerador/Denom; % Métrica
end
M2=(abs(M)).^2; % Cuadrado de la magnitud de cada valor
umbral=0.60; % Umbral para estimar si existio un paquete OFDM, debe
% ser mayor a 0.40
tramaEncontrada=0; % 0 trama no encontrada
auxET=0; % Inicializar variable que almacena la posición en donde
% se superera el umbral de la métrica
for i=1:length(M2)
    %Se obtiene la posición del primer valor por arriba del umbral:
    if(M2(i)> umbral)
        auxET=i;
        tramaEncontrada=1; % Trama encontrada
        break;
    end
end
datosOFDM=[]; % Iniciar variable que contiene la trama OFDM
if(tramaEncontrada==1)
    % Para el caso de que se haya encontrado una paquete OFDM
    datosOFDM(:,1)=datosRX(auxET:end);
else
    % Si no existe un paquete OFDM se empieza de nuevo el bucle while
    disp('***** Trama OFDM NO encontrada *****');
    continue
end
% datosOFDM son los datos de salida
% % Graficar datos de entrada de la etapa, la métrica y la trama
% % encontrada
% if (tramaEncontrada==1 && c1==1)
%     disp('Trama OFDM detectada');
%     figure()
%     subplot(3,1,1); plot(M2); title('Métrica M(k)'); hold on
```

```
%     stem(auxET,M2(auxET),'filled'); hold off
%     subplot(3,1,2)
%     plot(real(datosRecortados(1:Mmues))); hold on;
%     stem(auxET,real(datosRecortados(auxET)),'filled'); hold off
%     title('Primeras muestras (parte real)')
%     subplot(3,1,3)
%     plot(real(datosOFDM));
%     title('Trama detectada y recortada (parte real)')
% end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%            CÁLCULO DEL CFO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Script que permite calcular el CFO de la señal recibida

load("datosEnviados.mat", 'datosConPre')% Se carga los datos transmitidos
datosEnviados=double(datosConPre);% Conversión a una variable compleja
datosRecibi2=datosOFDM;

frecuenciaMuestreo=2.0e6;

%Espectro de la señal  enviada y recibida
[ptx,ftx]=pspectrum(datosEnviados,frecuenciaMuestreo);
[prx,frx]=pspectrum(datosRecibi2,frecuenciaMuestreo);

%Localización de picos
[picostx,locstx] = findpeaks(ptx);
picosrx=findpeaks(prx);

%Localización del pico máximo de la señal recibida
[picomaxrx Irx]=max(picosrx);

%Localización del pico más cercano de la señal transmitida, con
%respecto al pico máximo de la señal recibida
l=find(locstx<Irx);
Itx=locstx(l(end));

%Cálculo del CFO
Frecrx=frx(Irx);
Frectx=ftx(Itx);

CFO=Frecrx-Frectx;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%              CORRECCIÓN CFO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

correctioncfo= comm.PhaseFrequencyOffset('FrequencyOffset',CFO,...
    'SampleRate',frecuenciaMuestreo);
datosCorrCFO=correctioncfo(datosOFDM);
 %datosCorrCFO=datosOFDM;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%                SINCRONIZACIÓN DE SÍMBOLO:
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

datosSTO=datosCorrCFO(1:5000); % Se toma dolos primeros 5000 datos
% para reducir el tiempo de procesamiento
```

```
% Secuencia larga (parte de LLTF) en el dominio de la frecuencia
parteLarga=[...
    0;1;-1;-1;1;1;-1;1;-1;1;-1;-1;-1;-1;-1;1
    1;-1;-1;1;-1;1;-1;1;1;1;1;0;0;0;0;0
    0;0;0;0;0;0;1;1;-1;-1;1;1;-1;1;-1;1
    1;1;1;1;1;-1;-1;1;1;-1;1;-1;1;1;1;1];
% Conversión del dominio de la frecuencia al tiempo, y adición de PC
% de la sección larga:
parteLargaTime=ifft(parteLarga); % conversión al dominio del tiempo
LLTF=[parteLargaTime(33:64);parteLargaTime;parteLargaTime]; % LLTF
dLLTF80 = LLTF(1:80); % Primeras 80 muestras de la LLTF
%Correlación cruzada
datosCorrRell = (abs(xcorr(datosSTO,dLLTF80))).^2;
% Se corrige en tiempo los datos correlacionados, para que
% coincidan con las posiciones de los datos de entrada:
datosCorr = datosCorrRell(length(datosSTO):end);
% Almacenar la posición de los dos picos:
[~,posPico1]=max(datosCorr); % Posición del primer pico
datosCorr(posPico1)=0; % Eliminar el primer pico para poder
% encontrar el segundo
[~,posPico2]=max(datosCorr); % Posición del segundo pico
% Se comprueba que la distancia entre los dos picos sea 64
if abs(posPico2-posPico1)==64
    % Se encuentra el inicio real del LTTF y de LSTF
    inicioLLTF=min([posPico1 posPico2]); % Se toma la posición
    % menor y esa es la ubicación del primer dato del LLTF
    inicioLSTF=inicioLLTF-160; % Determinar el inicio del preambulo
    % Comprobar que la posición inicial sea mayor a 0
    if (inicioLSTF>0)
        % Recortar los datos
        datosSincronizados=datosCorrCFO(inicioLSTF:end); % Datos
        % de salida
        %Graficar:
%         figure()
%         plot(real(datosSincronizados(1:500)))
%         title('Parte real de los datos sincronizados')
    else
        disp('Falla en la sincronizacion de simbolo (2)')
    end
else
    disp('Falla de sincronización de símbolo')
    continue
end
% datosSincronizados es los datos de salida

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%              CORRECIÓN DE FASE
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Pasar las dos secuencias largas del LLTF del dominio del tiempo al
% dominio de la frecuencia:
a=fft(datosSincronizados(257:320)); % segunda sección larga de LLTF
b=fft(datosSincronizados(193:256)); % Primera sección larga de LLTF
%  Valores de las secuencias largas a calcular el ángulo:
angulosAux=[a(2) a(5) a(6) a(8) a(10) a(16) a(17) a(20) ...
    a(53) a(56) a(57) a(59) a(61) a(62) a(63) a(64) ...
```

```
    b(2) b(5) b(6) b(8) b(10) b(16) b(17) b(20) ...
    b(53) b(56) b(57) b(59) b(61) b(62) b(63) b(64)];
angulos=(angle(angulosAux)*180)/pi; % Calcular el ángulo y pasar de
% radianes a grados
%Promedio de los ángulos
anguloProm=mean(abs(angulos));
figure(1)
subplot(2,1,1)
stem(CFO)
drawnow
title('CFO')
xlabel('Muestras')
ylabel('Frecuencia [Hz]')
subplot(2,1,2)
stem(anguloProm)
drawnow
xlabel('Muestra')
ylabel('Ángulo promedio [°]')
title('Desplazamiento de Fase')

%Corrección de fase
correctionPhase=comm.PhaseFrequencyOffset('PhaseOffset',anguloProm,...
    'SampleRate',frecuenciaMuestreo);

datFase=correctionPhase(datosSincronizados);

    %datFase=datosSincronizados;

% Se verifica que los datos corregidos sean multiplo de 80,
% caso contrario se realiza un padding de las ultimas muestras
longDatF=length(datFase);
padd80=ceil(longDatF/80)*80-longDatF;
if(padd80>=1)
    datosCorrFase=[datFase;datFase(end-padd80+1:end)];
else
    datosCorrFase=datFase;
end
% El vector de salida es datosCorrFase

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%              ELIMINACIÓN DEL PREFIJO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nSimOFDM=0; % Iniciar contador de símbolos OFDM
tramaDatos=datosCorrFase(321:end); % Quitar el preambulo
% Recorrer cada símbolo y eliminar el PC de cada uno
for i=1:80:length(tramaDatos)
    nSimOFDM=nSimOFDM+1; % Sumar 1 en la cantidad de símbolos OFDM
    simbolosSinPC(:,nSimOFDM)=tramaDatos(i+16:i+79); % Guardar el
    % símbolo sin PC
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%              FFT
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% La función fft aplica la DFT a cada columna de la matriz
% simbolosSinPC
```

```
simbolosfft=fft(simbolosSinPC); % Demodular los símbolos sin PC

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%          ESTIMACIÓN DE CANAL Y ECUALIZADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
simbolosEcualizados=[]; % Iniciar vector que almacena los simb ecualiz
% Con un lazo for se recorre todos los símbolos OFDM en frecuencia
for auxnSimb=1:nSimOFDM
    simbF=simbolosfft(:,auxnSimb); % Símbolo en frecuencia
    % Se realiza la interpolacion lineal utilizando las portadoras
    % piloto, que estan en las posiciones 8 22 44 y 58:
    x1=[8 22]; % Puntos de la muestra
    v1=[simbF(8) simbF(22)]; % Valores de pilotos recibidos
    xq1=(2:23); % Puntos a estimar
    x2=[44 58]; % Puntos de la muestra
    v2=[simbF(44) simbF(58)]; % Valores de pilotos recibidos (2)
    xq2=(43:64); % Puntos a estimar
    val_interp1 = (interp1(x1,v1,xq1,'linear','extrap')).';
    val_interp2 = (interp1(x2,v2,xq2,'linear','extrap')).';
    % Se aplica la interpolación a las portadoras de datos:
    DatosEc=zeros(64,1); % Iniciar variable q almacena datos ecualiz
    DatosEc(2:23)=simbF(2:23)./val_interp1; % Ecualizar datos
    DatosEc(43:64)=simbF(43:64)./val_interp2; % Ecualizar datos
    simbolosEcualizados(:,auxnSimb)=DatosEc; % Datos de salida
end
nSimOFDM=auxnSimb; % Copiar el número de símbolos OFDM

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%     RECUPERACIÓN DE DATOS DEL SÍMBOLO OFDM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

nPdatos=40; % Número de portadoras de datos
datosReformado=zeros(nPdatos*nSimOFDM,1); % Iniciar variable que
% contiene los datos seriaizados
% Con un lazo for se recorre todos los símbolos OFDM:
for auxnSimb=1:nSimOFDM
    % obtener los datos OFDM
    simAux=simbolosEcualizados(:,auxnSimb); % Símbolo ecualizado
    % Desamblaje del símbolo OFDM:
    datosAux=zeros(40,1); % Inicializar en 0 un vector auxiliar
    %               simAux(1) es Portadora DC
    datosAux(21:26)=simAux(2:7);
    %               simAux(8) es la primera piloto
    datosAux(27:39)=simAux(9:21);
    %               simAux(22) es la primera piloto
    datosAux(40)=simAux(23);
    %            simAux(24:42) son las portadoras de guarda
    datosAux(1)=simAux(43);
    %           simAux(44) es la primera piloto
    datosAux(2:14)=simAux(45:57);
    %              simAux(58) es la primera piloto
    datosAux(15:20)=simAux(59:64);
    % Serializar los datos recuperados:
    datosReformado((auxnSimb-1)*nPdatos+1:auxnSimb*nPdatos)=datosAux;
end
% Los datos de salida son datosReformado

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%%%%%            REFORMADO DE PULSO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% El vector de entrada de esta etapa es datosReformado
% Parámetros del filtro de recepción:
rolloff = 0.5;  % Factor de caída del filtro
span = 6;       % Define el tamaño de la ventana en símbolos
sps = 4;        % # de muestras por cada símbolo
% En el filtro de recepción se define el parámetro 'Shape' como
% 'Square root' para obtener un filtro SRRC
% Creación del filtro SRRC:
rxfilter = comm.RaisedCosineReceiveFilter('Shape','Square root',...
    'RolloffFactor',rolloff,'FilterSpanInSymbols',span, ...
    'InputSamplesPerSymbol',sps,'DecimationFactor',sps);
% Agregar relleno para asegurar que los datos a filtrar sean multipo
% de sps:
longDatR=length(datosReformado); % Longitud de datos
mltSPS=ceil(longDatR/sps); % Calcular un multiplo de sps
nceros=mltSPS*sps-longDatR; % Calcular el relleno de ceros
datosRellAux=[datosReformado; zeros(nceros,1)]; % Rellenar el vector
% A los datos de entrada datosRellAux se aplica el filtro rxfilter
% y se almacena en la variable datosMapAux
datosMapAux=rxfilter(datosRellAux);
datosMapeo=datosMapAux(span+1:end,1);  % Datos de salida
% Graficar el diagrama de constelación de los datos de entrada y
% de salida:
% figure()
% plot(datosReformado,'r.'); hold on
% plot(datosMapeo,'b.');
% title('Señal QPSK recuperada')
% legend('Señal recibida','Datos QPSK')
% xlabel('I (fase)'); ylabel('Q (cuadratura)'); hold off

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                DEMAPEO
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Demapeo BPSK
    M = 2; % Puntos en la constelación (símbolos)
    Nb=log2(M); % Bits por símbolo
    % Los datos filtrados no se normalizan
    longitud=length(datosMapeo); % Longitud datos de entrada
    datosDemod=zeros(longitud,1); % Inicializar vector
    % Implementar el Demapeo BPSK con el lazo for
    for i=1:longitud
        if(real(datosMapeo(i))>=0 && imag(datosMapeo(i))>=0)
            datosDemod(i)=1;
        elseif(real(datosMapeo(i))<0 && imag(datosMapeo(i))<0)
            datosDemod(i)=0;
        end
    end

 %%
 % Demapeo QPSK
%     M = 4; % Puntos en la constelación (símbolos)
%     Nb=log2(M); % Bits por símbolo
%     % Relleno de ceros:
%     long_pM=ceil(length(datosMapeo)/Nb)*Nb-length(datosMapeo);
%     paddM=zeros(long_pM,1); % relleno con ceros
%     % Los datos filtrados solo se multiplican por sqrt(2)
```

```
%     ModDat=(sqrt(2))*[datosMapeo;paddM]; %Datos con relleno
%     % Demodulación QPSK:
%     datosDemodS=pskdemod(ModDat,M,pi/4,'gray');
%     % Pasar de símbolos a bits:
%     demodMatriz=de2bi(datosDemodS,Nb,'left-msb');
%     datosDemod=reshape(demodMatriz.',1,[]).'; % Datos
%     % recuperados (bits Serializados)
 %%
 % Demapeo 16-QAM
%     M = 16; % Puntos en la constelación (símbolos)
%     Nb = log2(M); % Bits por símbolo
%     % Relleno de ceros:
%     long_pM=ceil(length(datosMapeo)/Nb)*Nb-length(datosMapeo);
%     paddM=zeros(long_pM,1); % relleno con ceros
%     % Los datos filtrados solo se multiplican por sqrt(10)
%     ModDat=(sqrt(10))*[datosMapeo;paddM]; % Datos con relleno
%     % Demodulación 16-QAM:
%     datosDemodS=qamdemod(ModDat,M);
%     % Pasar de símbolos a bits:
%     demodMatriz=de2bi(datosDemodS,Nb,'left-msb');
%     datosDemod=reshape(demodMatriz.',1,[]).'; %Datos
%     %recuperados (bits Serializados)
  %%
  % Demapeo 64-QAM
%     M = 64; % Puntos en la constelación (símbolos)
%     Nb = log2(M); % Bits por símbolo
%     % Relleno de ceros:
%     long_pM=ceil(length(datosMapeo)/Nb)*Nb-length(datosMapeo);
%     paddM=zeros(long_pM,1); % relleno con ceros
%     % Los datos filtrados solo se multiplican por sqrt(42)
%     ModDat=(sqrt(42))*[datosMapeo;paddM]; % Datos con relleno
%     % Demodulación 64-QAM:
%     datosDemodS=qamdemod(ModDat,M);
%     % Pasar de símbolos a bits:
%     demodMatriz=de2bi(datosDemodS,Nb,'left-msb');
%     datosDemod=reshape(demodMatriz.',1,[]).'; %Datos
%     % recuperados (bits Serializados
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%           DECODIFICADOR CONVOLUCIONAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Calcular el número de bits codificados por cada símbolo OFDM:
NPdatos=40; % número de portadoras de datos por símbolo OFDM
%%
% BPSK
 %M=2;
%%
% QPSK
 %M = 4; % Puntos en la constelación (símbolos); 4 para QPSK
%%
% 16-QAM
 %M= 16;
%%
% 64-QAM
%M= 64;
%%
Nb=log2(M); % Número de bits utilizado por cada símbolo
Nsps=1/4; % Este valor tiene que ser igual 1/sps, sps se define en
```

```
% la etapa reformado de pulso y es igual a 4 cuando está activado
nBc=Nb*NPdatos*Nsps; % Número de bits codificados por simb OFDM
% Relleno de bits para que el vector de entrada sea multiplo de nBc:
nGrupos=ceil(length(datosDemod)/nBc); % Calcular el número de grupos
% de bits a decodificar
longiPadd=nBc*nGrupos-length(datosDemod); % Longitud de relleno
v_relleno=zeros(longiPadd,1); % relleno de ceros
datosDemodRell=[datosDemod;v_relleno]; % Datos serializados con
% relleno de ceros
% Parámetros del codificador convolucional:
nc=2; % n es la longitud por cada ingreso de k bits
kc=1; % k es los bits de entrada
mc=3; % m es el tamaño de registro de la memoria
g1=5; % Polinomio generador
g2=7; % Polinomio generador
Rcc=kc/nc; % tasa del codificador convolucional
% Estructura de Trellis:
P_Trellis = poly2trellis(mc, [g1 g2]); % Igual que en tx
% Decodificación convolucional:
tb=nBc*Rcc; % La profundidad de rastreo debe ser igual al número de
% bits que se tuvo a la entrada del codificador
datosDecodConv=[]; % Inicilizar vector que almacena los datos
% decodificados
% Con un lazo for se recorre los grupos de bits a ser decodificados
for i=1:nGrupos
    auxDecod = datosDemodRell((nBc*(i-1)+1):nBc*i,1); % Se toma el
    % grupo correspondiente
    DecodVit= vitdec(auxDecod,P_Trellis,tb,'trunc','hard'); % Se
    % decodifica los bits
    datosDecodConv=[datosDecodConv;DecodVit]; % Almacenar los bits
    % decodificados
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%             MEDICIÓN DE BER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% %%%%%%%%%%%%%%%%%%%%%% ABRIR ARCHIVO: %%%%%%%%%%%%%%%%%%%%%%

% nombreArchivo=uigetfile({'*.txt'},'Seleccionar archivo'); %Seleccionar
% % un archivo mediante un cuadra de diálogo
fid =fopen('ArchivoTextoRT.txt'); % Abrir el archivo seleccionado
x=fread(fid,'*char'); % Leer y guardar el archivo
fclose(fid); % Cerrar el archivo
textoBinario = dec2bin(x,8); % Representar cada caracter con 8 bits
% Serializar datos del vector x:
aux=1; % Iniciar variable auxiliar
for i = 1:length(textoBinario)
    for j=1:8
        textoBitsSerial(aux)=textoBinario(i,j);
        aux=aux+1;
    end
end
% Pasar del tipo char al tipo double;
for aux=1:length(textoBitsSerial)
    datosMedirBER(aux,1)=str2double(textoBitsSerial(aux)); % Datos
    % originales serializados
end
```

```
longDatosBER=length(datosMedirBER); % Longitud de bits originales
% %%%%%%%%%%%%%%%%%%%%% MEDIR BER: %%%%%%%%%%%%%%%%%%%%%%%%%
LongDatDecod=length(datosDecodConv); % longitud de bits recuperados
if (longDatosBER<=LongDatDecod)
    % La longitud del archivo recuperado es mayor o igual que la
    % del original
    datosSerializadosRx=datosDecodConv(1:longDatosBER); % Recortar
    bitsErrado=0; % Iniciar contador de bits errados
    % Recorrer todo el vector de bits recuperados
    for aux=1:longDatosBER
        % Comparar bit recuperado con el bit original
        if(datosSerializadosRx(aux)~=datosMedirBER(aux))
            % Se no son iguales se aumenta en uno los bits errados
            bitsErrado=bitsErrado+1;
        end
    end
    % Mostrar la cantidad de bits errados sobre la longitud total:
    v_BER=bitsErrado/longDatosBER; % Valor de BER
    fprintf('Bits errados/Bits totales = %g/%g \n',...
        bitsErrado,longDatosBER);
    fprintf('BER = %g \n',v_BER);
else
    % La longitud del archivo recuperado es menor que la del original
    disp('Archivo no recuperado completamente, BER no medido')
    disp('Longitud del archivo recuperdado menor al original');
    datosSerializadosRx=datosDecodConv; % Copiar los datos
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%              DECODIFICADOR
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 % Relleno de ceros:
    bPc=8; % Bits por cada caracter
    numC=ceil(length(datosSerializadosRx)/8); %Número caracteres
    LongRellnumC=bPc*numC-length(datosSerializadosRx); % Longitud
    % de relleno para hacer múltiplo de 8
    rellenoC=zeros(LongRellnumC,1); % Vector de relleno ceros
    datosSerializados=[datosSerializadosRx ; rellenoC]; %Rellenar
    % en caso de ser necesario
    % Pasar de los bits serializados a una matriz, en donde las
    % filas contiene a los caracteres en binario:
    datosMtz=reshape(datosSerializados,[8 numC]).'; % numC filas
    % x 8 columnas
    % Pasar de binario a decimal, los datos de cada fila:
    datosTexto=[]; % Inicializar vector
    for i=1:numC
        % Se pasa cada fila a decimal y se almacena en datosTexto
        datosTexto(1,i)=double(bi2de(datosMtz(i,:),'left-msb'));
    end
    % Pasar de decimal a caracteres:
    archivoRecuperado=char(datosTexto); % Texto recuperado
    disp('Proceso recuperacion de texto terminado');
    % Imprimir el texto recuperado en un static text de la GUI
    handles.txtARecuperado.String=archivoRecuperado;
    pause(4)
    toc
```

```
    else
    end
end
```

ANEXO III. SCRIPTS PARA EL FUNCIONAMIENTO DE LAS ESTIMACIONES DE LOS PARÁMETROS CFO, STO Y DESPLAZAMIENTO DE FASE

pruebaRecepcion.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%          UMBRAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Script para determinar el umbral del receptor Adalm Pluto, en base a
% la toma de n muestras de datos recibidos.

%Se inicializa los vectores donde se va almacenar los datos
lectura=[]; %vector que almacena los datos recibidos en el receptor
maxi=[]; %vector que almacena el valor máximo del valor absoluto de los
%datos recibidos.
n=10; %número de muestras a considerar

%Se crea un lazo for que permitirá almacenar las lecturas del objeto de
%recepción.

for i=1:n

    lectura(i,:)=rxAdalm();
    maxi(i,:)=max(abs(lectura(i,:)));

end

%Se saca el promedio del vector maxi, que será el umbral que se ha
%determinado del receptor.

umbralrx=mean(maxi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%        SONDEO DEL CANAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generación de un bucle para estar sondeando el canal, además se
% establece un umbral de amplitud que permitirá distinguir la señal
%de lo que es el ruido del canal inalámbrico.

condicion=1;

disp('Empezando a sondear el canal inalámbrico')
while(condicion==1)
    datosRecibidos=rxAdalm();
    max(abs(datosRecibidos))
    if(max(abs(datosRecibidos))> umbralrx)
       break;
    else
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%                        GRÁFICAS
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

figure();
subplot(2,1,1)
plot(real(datosRecibidos));
title('Señal recibida componente Real');

subplot(2,1,2)
plot(imag(datosRecibidos));
title('Señal recibida componente Imaginaria');


load('datosEnviados.mat','datosEnviados')

%Señal enviada con sus componentes real e imaginario
tx1=real(datosEnviados);
tx2=imag(datosEnviados);

%Señal Recibida con sus componentes real e imaginario
rx1=real(datosRecibidos);
rx2=imag(datosRecibidos);

%Gráfica de las dos señales superpuestas
figure()

subplot(2,1,1)
plot(tx1)
hold on
plot(rx1)
legend('señal enviada','señal recibida')
title('Componente Real');

subplot(2,1,2)
plot(tx2)
hold on
plot(rx2)
legend('señal enviada','señal recibida')
title('Componente Imaginario');

end
```

signal_analyzer.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%           UMBRAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Script para determinar el umbral del receptor Adalm Pluto, en base a
% la toma de n muestras de datos recibidos.

%Se inicializa los vectores donde se va almacenar los datos
lectura=[]; %vector que almacena los datos recibidos en el receptor
maxi=[]; %vector que almacena el valor máximo del valor absoluto de los
%datos recibidos.
n=10; %número de muestras a considerar

%Se crea un lazo for que permitirá almacenar las lecturas del objeto de
%recepción.
```

```
for i=1:n

    lectura(i,:)=rxAdalm();
    maxi(i,:)=max(abs(lectura(i,:)));

end

%Se saca el promedio del vector maxi, que será el umbral que se ha
%determinado del receptor.

umbralrx=mean(maxi);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%         SONDEO DEL CANAL
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Generación de un bucle para estar sondeando el canal, además se
% establece un umbral de amplitud que permitirá distinguir la señal
%de lo que es el ruido del canal inalámbrico.

condicion=1;

disp('Empezando a sondear el canal inalámbrico')
while(condicion==1)
    datosRecibidos=rxAdalm();
    max(abs(datosRecibidos))
    if(max(abs(datosRecibidos))> umbralrx)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%               RECORTAR DATOS
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script que permite recortar los datos de información del bloque
% de datos recibido en el script recibir.m
% Antes de correr este script es necesario ejecutar el script
% config_SDR.m y recibir.m o bien se necesita cargar los datos
% guardados con load('datosRecibidos150.mat')
% En vector de entrada de esta etapa es datosRecibidos
% RECORTAR DATOS
%umbralrx=0.0040;
% Se recorre el vector de entrada y  se obtiene la posición del
% primer valor por arriba del umbral:
pos_inic=0; % Inicializar variable
for i=1:length(datosRecibidos)
    if(max(abs(datosRecibidos(i)))>umbralrx)
        pos_inic=i; % Primera posición arriba del umbral
        break;
    end
end
% Guardar desde 400 datos antes de superar el umbral, hasta el final:
if(pos_inic>400)
    datosRec=datosRecibidos(pos_inic-400:end); % Datos recortados
else
    % El umbral se superó antes de las primeras 400 posiciones, solo
    % se copia los datos
    datosRec=datosRecibidos;
end
datosInv=flip(datosRec); % Invierte el vector, tal que los datos
```

```
% finales queden al inicio, con esto se puede repetir el proceso
% anterior.
% Se recorre el vector de invertido y  se obtiene la posición del
% primer valor por arriba del umbral:
pos_fin=0; % Inicializar variable
for i=1:length(datosRec)
    if (max(abs(datosInv(i)))>umbralrx)
        pos_fin=i; % Primera posición arriba del umbral
        break
    end
end
% Guardar desde 200 datos antes de superar el umbral, hasta el final:
if(pos_fin>200)
    datosInvRec=datosInv(pos_fin-200:end); % Recortar datos
else
    % El umbral se superó antes de las primeras 200 posiciones, solo
    % se copia los datos
    datosInvRec=datosInv;
end
datosRecortados=flip(datosInvRec); % Invertir nuevamente el vector
% para obtener los datos recortados

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%       ENCONTRAR TRAMA OFDM
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Script que permite encontrar la trama OFDM y realizar un ajuste en
% tiempo
% Antes de correr este script es necesario ejecutar el script
% config_SDR.m, recibir.m, recortar_rx.m o bien cargar los datos
% guardados en datosRecibidos150.mat y ejecutar recortar_rx.m
% En vector de entrada de esta etapa es datosRecortados
% DETECCIÓN TRAMA OFDM:
datosRX=datosRecortados; % Copiar los datos en una nueva variable
SamplesPerFrame=800000; % tamaño del bloque recibido
% Los datos enviados nunca pueden ser mayor a SamplesPerFrame+600,
% el 600 se
c1=1; % Variable de control
if length(datosRX)>(SamplesPerFrame+600)
    disp('***** Muestras recibidas inválidas *****');
    c1=0;
end
% El preambulo se encuentra al inicio de cada paquete y por tanto
% se trabaja con las primeras muestras, con esto se evita
% procesamiento innecesario
Mmues=1400; % Muestras para calcular la métrica
if(length(datosRX)>Mmues)
    r=datosRX(1:Mmues);
else
    % Si es menor a Mmues se trabaja con todas las muestras
    r=datosRX;
end
L = 48; % Múltiplo dela longitud de una secuencia corta (16*x)
inMax = length(r)-2*L+1; % k+m puede tomar valores de hasta 2L-1
% el indice maximo a evaluar es la longitud de r menos 2L.
m=L-1; % Máximo valor de m
M = zeros(Mmues,1); % Inicializar la variable para mayor velocidad
% Determinar la métrica M(k) para comprobar si es un paquete OFDM:
```

```
for k=1:inMax
    Numerador = (r(k:k+m)'*r(k+L:k+m+L)); % Sumatorio del numerador
    Denom = sum((abs(r(k+L:k+m+L))).^2); % Sumatorio del denominador
    M(k)=Numerador/Denom; % Métrica
end
M2=(abs(M)).^2; % Cuadrado de la magnitud de cada valor
umbral=0.60; % Umbral para estimar si existio un paquete OFDM, debe
% ser mayor a 0.40
tramaEncontrada=0; % 0 trama no encontrada
auxET=0; % Inicializar variable que almacena la posición en donde
% se superera el umbral de la métrica
for i=1:length(M2)
    %Se obtiene la posición del primer valor por arriba del umbral:
    if(M2(i)> umbral)
        auxET=i;
        tramaEncontrada=1; % Trama encontrada
        break;
    end
end
datosOFDM=[]; % Iniciar variable que contiene la trama OFDM
if(tramaEncontrada==1)
    % Para el caso de que se haya encontrado una paquete OFDM
    datosOFDM(:,1)=datosRX(auxET:end);
else
    % Si no existe un paquete OFDM se empieza de nuevo el bucle while
    disp('***** Trama OFDM NO encontrada *****');

end
% datosOFDM son los datos de salida


    break;
    else
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%        SIGNAL ANALYZER
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Carga de datos de la señal enviada
load('datosEnviados.mat', 'datosConPre')


%Gráfica de la señal enviada y recibida superpuesta, usando un analizador
%de señales

a1= max(abs(datosConPre));
a2= max(abs(datosOFDM));

a=a1/a2;


%Señales a ser graficadas
datosRecibidos1=double(a.*datosOFDM); %nueva señal recibida multiplicada
                                      %por el promedio de la relación.
datosEnviados1=double(datosConPre);%Señal enviada convertida en double.
```

```
%Analizador de señales

signalAnalyzer(datosEnviados1,datosRecibidos1, 'SampleRate', 2e6)
```

cfo_calculo.m

```
%Script que realiza el cálculo del CFO de n muestras recolectadas
%Para su funcionamiento ejecutrar primero los scripts config_SDR.m
% umbral_rx.m, y recoleccion_datosrx.m

frecuenciaMuestreo=2.0e6;
n=300; %Número de muestras que se calculará el CFO.

%Procesamiento de los datos recibidos y transmitidos
load('D164QAMs.mat','datosRecibi2')
datosRecibi2=double(datosRecibi2); % Conversión a una variable compleja
% doble de los datos recolectados
load("datosEnviados.mat",'datosEnviados') % Se carga los datos transmitidos
datosEnviados1=double(datosEnviados);% Conversión a una variable compleja
% doble de los datos transmitidos

%Espectro de la señal transmitida
[ptx,ftx]=pspectrum(datosEnviados1,frecuenciaMuestreo);
[picostx,locstx] = findpeaks(ptx);% Localización de los picos del espectro
% de la señal transmitida y sus índices.

for i=1:n

    %Espectro de la señal recibida
    [prx(:,i),frx(:,i)]=pspectrum(datosRecibi2(:,i),frecuenciaMuestreo);

    %Localización de los picos del espectro de la señal recibida y sus
índices
    picosrx=zeros(600,i); %Se inicializa la variable con 600 ceros
    % donde se almacenarán todos los picos de cada muestra.

    %Longitud del relleno de los picos de cada muestra
    if length(findpeaks(prx(:,i)))< length(picosrx)

        aux1(:,i)=length(picosrx)-length(findpeaks(prx(:,i)));

    end
end

for i=1:n
    c1 =findpeaks(prx(:,i));
    picosrx(:,i)=[c1 ;zeros(aux1(i),1)]; % Se almacena todos los picos de
    % cada muestra

    %Selección del pico máximo y su índice de cada muestra de la señal
    % recibida
    [c2 c3] =max(picosrx(:,i));
    picomaxrx(i)=c2; %se almacena el valor del pico máximo de cada muestra
    Irx(i)=c3; %Se almacena el índice del pico máximo de cada muestra

    %Localización del pico más cercano de la señal transmitida, con
    %respecto al pico máximo de la señal recibida
```

```
    l=find(locstx<Irx(i));
    Itx(i)=locstx(l(end));% Se almacena el índice del pico más cercano de
    % la señal transmitida, con respecto al pico máximo de la señal
    % recibida.

    %Frecuencias de los picos de la señal transmitida y señal recibida.
    Frecrx(i)=frx(Irx(i));%Frecuencia señal recibida
    Frectx(i)=ftx(Itx(i));%Frecuencia señal transmitida

    %Cálculo del CFO
    cfo(i)=Frecrx(i)-Frectx(i);


end

%Gráfica del valor del CFO para cada muestra recibida.
figure()
stem(cfoD1BSPK)
title('CFO con modulación BPSK y distancia 18m')
xlabel('Muestras')
ylabel('Frecuencia [Hz]')


save('D164QAMscfo.mat','cfo')


%Corrección del CFO
correctioncfo= comm.PhaseFrequencyOffset('FrequencyOffset',cfoD1BPSK,...
    'SampleRate',frecuenciaMuestreo);
datosCorrCFO=correctioncfo(datosOFDM);
```

histograma.m

```
% Modificar el archivo cargado para obtener las gráficas para cada distancia
% y tipo de modulación
load('D4BPSKscfo.mat')

figure ()
subplot(2,1,1)
stem(cfo)
title('CFO')
ylabel('Frecuencia [Hz]')
xlabel('Muestras')

subplot(2,1,2)
hist(cfo,11)
title('Histograma CFO')

sgtitle('CFO con modulación BPSK y distancia 5.4m NLoS')
```

fase.m

```
% Graficar una secuencia larga en el dominio de la frecuencia y
% almacenar las posiciones cuando la secuencia larga es 1.
% Crear la parte larga del preámbulo 802.11a:
```

```
% Secuencia larga (parte de LLTF) en el dominio de la frecuencia
parteLarga=[...
    0;1;-1;-1;1;1;-1;1;-1;1;-1;-1;-1;-1;-1;1
    1;-1;-1;1;-1;1;-1;1;1;1;1;0;0;0;0;0
    0;0;0;0;0;0;1;1;-1;-1;1;1;-1;1;-1;1
    1;1;1;1;1;-1;-1;1;1;-1;1;-1;1;1;1;1];
% Conversión del dominio de la frecuencia al tiempo, y adición de PC
% de la sección larga:
parteLargaTime=ifft(parteLarga); % conversión al dominio del tiempo
secLargaF=fft(parteLargaTime); % Conversión del tiempo a frecuencia
j=1; % Inicializar variable
posiciones=[]; % Inicializar vector que almacena las posiciones de los
% valores que son igual a 1 en secLargaF
% Recorrer cada posición de secLargaF y almacenar las que sean mayor
% a 0.9:
for i=1:length(secLargaF)
    if  (real(secLargaF(i)) > 0.9)
        posiciones(j,1)=i;
        j=j+1;
    end
end
% Graficar la parte real e imaginaria de la secuencia larga
figure()
stem(real(secLargaF))
hold on
stem(imag(secLargaF))
xlabel('Muestras'); ylabel('Amplitud')
title('Índices de una secuencia larga del LLTF cuando es igual a 1')
legend('real','complejo')
```

fase_rx.m

```
% Pasar las dos secuencias largas del LLTF del dominio del tiempo al
% dominio de la frecuencia:
a=fft(datSF1(257:320)); % segunda sección larga de LLTF
b=fft(datSF1(193:256)); % Primera sección larga de LLTF
%  Valores de las secuencias largas a calcular el ángulo:
angulosAux=[a(2) a(5) a(6) a(8) a(10) a(16) a(17) a(20) ...
    a(53) a(56) a(57) a(59) a(61) a(62) a(63) a(64) ...
    b(2) b(5) b(6) b(8) b(10) b(16) b(17) b(20) ...
    b(53) b(56) b(57) b(59) b(61) b(62) b(63) b(64)];


% Secuencia larga (parte de LLTF) en el dominio de la frecuencia
parteLarga=[...
    0;1;-1;-1;1;1;-1;1;-1;1;-1;-1;-1;-1;-1;1
    1;-1;-1;1;-1;1;-1;1;1;1;1;0;0;0;0;0
    0;0;0;0;0;0;1;1;-1;-1;1;1;-1;1;-1;1
    1;1;1;1;1;-1;-1;1;1;-1;1;-1;1;1;1;1];
% Conversión del dominio de la frecuencia al tiempo, y adición de PC
% de la sección larga:
parteLargaTime=ifft(parteLarga); % conversión al dominio del tiempo
secLargaF=fft(parteLargaTime); % Conversión del tiempo a frecuencia
s=real(secLargaF);
angulosAux1=[s(2) s(5) s(6) s(8) s(10) s(16) s(17) s(20) ...
    s(22) s(24) s(25) s(26) s(27) s(39) s(40) s(43) ...
    s(44) s(46) s(48) s(49) s(50) s(51) s(52) ...
    s(53) s(56) s(57) s(59) s(61) s(62) s(63) s(64)];
```

```
%Cálculo de los ángulos de cada índice y de la secuencia LLTF, en grados
angulos=(angle(angulosAux)*180)/pi; % Calcular el ángulo y pasar de
% radianes a grados
angulos1=(angle(angulosAux1)*180)/pi;

%Promedio de los ángulos
anguloProm=mean(abs(angulos));

%Gráfica de los ángulos de cada índice
figure()
stem(angulos);
hold on
stem(angulos1)
xlabel('índices')
ylabel('ángulos [°]')
legend('ángulos señal recibida','ángulos secuencia LLTF')
```

fase_aut.m

```
load('Fase2m.mat')

for i=1:15

    datSF=datFase(:,i);
    % Pasar las dos secuencias largas del LLTF del dominio del tiempo al
    % dominio de la frecuencia:
    a(:,i)=fft(datSF(257:320)); % segunda sección larga de LLTF
    b(:,i)=fft(datSF(193:256)); % Primera sección larga de LLTF

    %  Valores de las secuencias largas a calcular el ángulo:
    angulosAux(:,i)=[a(2,i) a(5,i) a(6,i) a(8,i) a(10,i) a(16,i) a(17,i)
a(20,i) ...
        a(53,i) a(56,i) a(57,i) a(59,i) a(61,i) a(62,i) a(63,i) a(64,i) ...
        b(2,i) b(5,i) b(6,i) b(8,i) b(10,i) b(16,i) b(17,i) b(20,i) ...
        b(53,i) b(56,i) b(57,i) b(59,i) b(61,i) b(62,i) b(63,i) b(64,i)];
    angulos(:,i)=(angle(angulosAux(:,i)).*180)/pi; % Calcular el ángulo y
pasar de
    % radianes a grados

    %Promedio de los ángulos
    anguloProm(:,i)=mean(abs(angulos(:,i)));


end

figure()
stem(anguloProm)
xlabel('Muestras')
ylabel('Ángulo promedio [°]')
title('Desplazamiento de Fase')
```