“IMPLEMENTACIÓN DE UN SISTEMA FHSS”

INFORME 8

Laboratorio de Comunicaciónes Digitales-GR9

1st Melanny Dávila
Ingeniería en Telecomunicaciones
Facultad de Elétrica y Eléctronica
Quito, Ecuador
melanny.davila@epn.edu.ec

2nd Ronaldo Almachi
Ingeniería en Telecomunicaciones
Facultad de Elétrica y Eléctronica
Quito, Ecuador
ronaldo.almachi@epn.edu.ec

***Abstract*—En este documento se analizarán los resultados obtenidos en base a la práctica de laboratorio referente a Frequency Hopping Spread Spectrum. Todo esto será analizado en base a scripts en Matlab utilizando el toolbox de comunicaciones con el fin de analizar la tasa de bit errado (BER) tanto de la señal de datos como de la señal ensanchada.**

***Index Terms*—FHSS, BPSK, FFT.**

I. Introdución

Frequency Hopping Spread Spectrum es una tecnología de comunicación inalámbrica que es una variante de Spread Sprectrum [1]. En FHSS, la frecuencia de la portadora en el modulador varía dentro de un ancho de banda fijo, conocido como ancho de banda total de salto.

La medida en que las frecuencias cambian de una a otra es pseudoaleatoria, es decir, no exite orden ya que depende de una secuencia PN que únicamente es conocida por el emisor y el receptor. El término “salto” viene para representar la asignación de frecuencia con respecto al tiempo y la duración del tiempo entre saltos se denomina duración del salto o período de salto.

Esta tecnología es utilizada actualmente en el ejército de los EE. UU. en JTIDS/MIDS, HAVE QUICK y SINCGARS radios, así como en todos los dispositivos Bluetooth (TM). Debido a que sus ventajas principales son:

- Proporciona una ruta de transmisión muy robusta en presencia de interferencias como múltiples rutas, ruido y otras transmisiones inalámbricas, etc. debido al soporte de un ancho de banda amplio.
- Puede emplearse en aplicaciones de punto a multipunto.
- Proporciona seguridad contra cualquier tipo de intrusión ya que solo el transmisor y el receptor conocen los códigos PN [2].

II. Objetivos

- Familiarizar al estudiante con los conceptos de Spread Spectrum
- Identificar las ventajas y desventajas del uso de las señales Spread Spectrum en los sistemas de comunicación.
- Visualizar los saltos de frecuencia entre los canales empleados en FHSS [1].

III. Preguntas

A. Analizar y presentar los resultados obtenidos tanto en el trabajo preparatorio como en la práctica.

Resultados obtenidos en el trabajo preparatorio

En la figura 1, se presenta el resultado de la implementación de un sistema FHSS, en donde una señal de datos es tratada tomando como referencia el diagrama de bloques de un transmisor FHSS. La misma que tiene las siguientes características: longitud de 30 bits, técnica de modulación digital B-PSK y un sintetizador con 6 diferentes niveles de frecuencia; tanto los datos como el secuencia pseudoaleatoria se generan con una función de Matlab que permite obtener números aleatorios diferentes en cada ejecución del script, es por eso que en cada literal se analiza diferentes datos.

La señal FHSS resultante ques es obtenida como el producto de la señal modulada con BPSK y la secuencia PN. Muestra cambios en amplitud y periodo , que corresponden a los cambios de frecuencia del sintetizador.

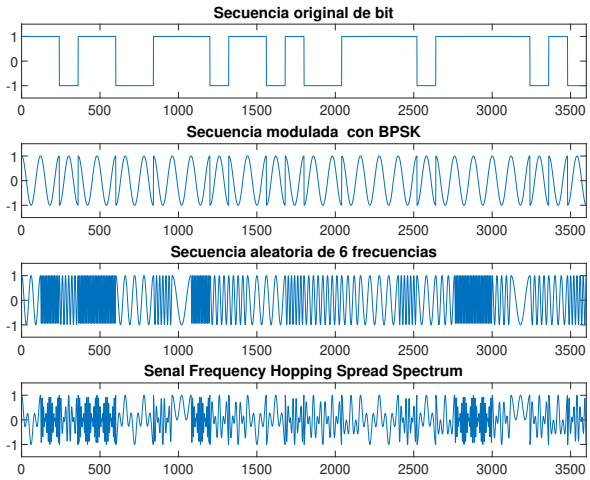

Fig. 1. Señales de datos obtenidas

Los resultados de la figura 1, se obtuvieron en base al Script 1, el mismo que es mostrado a continuación.

```
%% FHSS
clc
clear all
close all

N=30; %Numero de datos
datos=randi([0 1],1,N);    %Generacion del vector de datoss
senal=[]; %Vector senal vacio
portadora=[]; %Vector portadora vacio
t=[0:2*pi/119:2*pi];     % Creating 120 samples for one cosine
for k=1:N %lazo for que permite recorrer el vector de datos
    if datos(1,k)==0  %si el bit es 0
        se=-ones(1,120);    %se designan 120 muestras de valor 1 con signo negativo para cada bit
    else %si el bit es 1
        se=ones(1,120);     %se designan 120 muestras para cada bit
    end
    p=cos(t); %se designa la portadora cos(t)
    portadora=[portadora p]; %se registra la portadora de cada bit
    senal=[senal se]; %se registra la senal de datos extendida
end

%Grafico de la senal de datos
subplot(4,1,1);
plot(senal);
axis([0 length(senal) -1.5 1.5]);
title('Secuencia original de bit');

% Modulacion BPSK de la senal de datos
bpsk=senal.*portadora;   %modulacion BPSK
%Grafico de la senal modulada
subplot(4,1,2);
plot(bpsk)
axis([0 length(bpsk) -1.5 1.5])
title('Secuencia modulada  con BPSK');
% Creacion de las 6 frecuencias de portadora
t1=[0:2*pi/9:2*pi]; %vector de tiempo 1
t2=[0:2*pi/19:2*pi]; %vector de tiempo 2
t3=[0:2*pi/29:2*pi]; %vector de tiempo 3
t4=[0:2*pi/39:2*pi]; %vector de tiempo 4
t5=[0:2*pi/59:2*pi]; %vector de tiempo 5
t6=[0:2*pi/119:2*pi]; %vector de tiempo 6
c1=cos(t1); %portadora 1
c1=[c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1];
c2=cos(t2); %portadora 2
c2=[c2 c2 c2 c2 c2 c2];
c3=cos(t3); %portadora 3
c3=[c3 c3 c3 c3];
c4=cos(t4); %portadora 4
c4=[c4 c4 c4];
c5=cos(t5); %portadora 5
c5=[c5 c5];
c6=cos(t6); %portadora 6

% Frecuencias randomicas para FHSS
ss=[]; %vector de frecuencias vacio
for n=1:N %lazo for que designa las frecuencias potadoras a cada bit de dato
    p=randi([1 6],1,1); %obtencion de la secuencia aleatoria de frecuencias
    switch(p) %bucle switch que permite designar las frecuencias portadoras segun la secuencia aleatoria
        case(1)
            ss=[ss c1];
        case(2)
            ss=[ss c2];
        case(3)
            ss=[ss c3];
        case(4)
            ss=[ss c4];
        case(5)
            ss=[ss c5];
        case(6)
            ss=[ss c6];
    end
end

%Grafico de la senal aleatoria de frecuencias de salto
subplot(4,1,3)
plot([1:length(ss)],ss);
axis([0 length(ss) -1.5 1.5]);
title('Secuencia aleatoria de 6 frecuencias');

% Senal FHSS
fhss=bpsk.*ss; %producto entre la senal bpsk y la secuencia aleatoria
```

```
    de frecuencias
82
83 %Grafico de la senal FHSS
84 subplot(4,1,4)
85 plot([1:length(fhss)],fhss);
86 axis([0 length(fhss) -1.5 1.5]);
87 title('Senal Frequency Hopping Spread
    Spectrum');
```

Script 1. Frequency Hopping Spread Spectrum-BPSK

En la figura 2, se tiene el espectro en frecuencia de una señal FHSS, esta al ser tratada con 6 diferentes frecuencias, tiene 6 impulsos a diferentes frecuencias y amplitudes, con esto se puede corroborar que la señal original de datos ha sido ensanchada, debido a que tiene diferentes frecuencias portadoras.

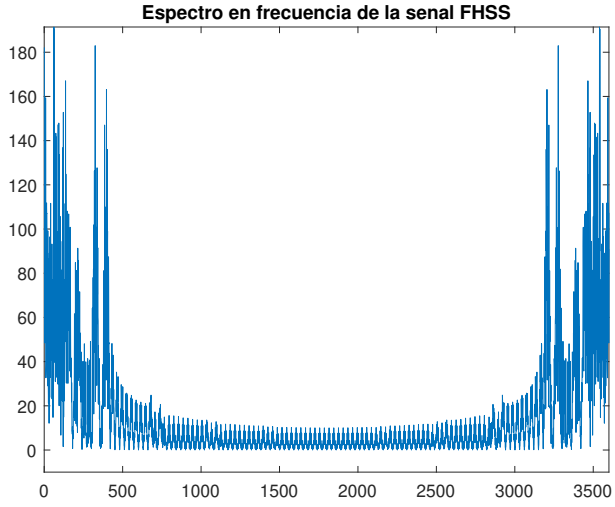

Fig. 2. Espectro en frecuencia de la señal ensanchada

Los resultados obtenidos en la figura 2, son en base al Script 2, el mismo que hace uso de la función fft disponible en Matlab.

```
1  %% FHSS
2  clc
3  clear all
4  close all
5
6  N=30; %Numero de datos
7  datos=randi([0 1],1,N);    %Generacion
       del vector de datoss
8  senal=[]; %Vector senal vacio
9  portadora=[]; %Vector portadora vacio
10 f=[0:2*pi/119:2*pi];     % Creating
    120 samples for one cosine
11 for k=1:N %lazo for que permite
    recorrer el vector de datos
12     if datos(1,k)==0  %si el bit es 0
13         se=-ones(1,120);    %se
    designan 120 muestras de valor 1
    con signo negativo para cada bit
14     else %si el bit es 1
15         se=ones(1,120);     %se
    designan 120 muestras para cada bit
16     end
17     p=cos(f); %se designa la portadora
       cos(t)
18     portadora=[portadora p]; %se
    registra la portadora de cada bit
19     senal=[senal se]; %se registra la
    senal de datos extendida
20 end
21
22 % Modulacion BPSK de la senal de datos
23 bpsk=senal.*portadora;   %modulacion
    BPSK
24
25 % Creacion de las 6 frecuencias de
    portadora
26 t1=[0:2*pi/9:2*pi]; %vector de tiempo
    1
27 t2=[0:2*pi/19:2*pi]; %vector de tiempo
     2
28 t3=[0:2*pi/29:2*pi]; %vector de tiempo
     3
29 t4=[0:2*pi/39:2*pi]; %vector de tiempo
     4
30 t5=[0:2*pi/59:2*pi]; %vector de tiempo
     5
31 t6=[0:2*pi/119:2*pi]; %vector de
    tiempo 6
32 c1=cos(t1); %portadora 1
33 c1=[c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1
    c1];
34 c2=cos(t2); %portadora 2
35 c2=[c2 c2 c2 c2 c2 c2];
36 c3=cos(t3); %portadora 3
37 c3=[c3 c3 c3 c3];
38 c4=cos(t4); %portadora 4
39 c4=[c4 c4 c4];
40 c5=cos(t5); %portadora 5
41 c5=[c5 c5];
42 c6=cos(t6); %portadora 6
43 fh=[ ];
44 % Frecuencias randomicas para FHSS
45 ss=[]; %vector de frecuencias vacio
46 for n=1:N %lazo for que designa las
    frecuencias potadoras a cada bit de
     dato
47     p=randi([1 6],1,1); %obtencion de
    la secuencia aleatoria de
    frecuencias
48     switch(p); %bucle switch que
```

```
    permite designar las frecuencias
    portadoras segun la secuencia
    aleatoria
49          case(1)
50              ss=[ss c1];
51          case(2)
52              ss=[ss c2];
53          case(3)
54              ss=[ss c3];
55          case(4)
56              ss=[ss c4];
57          case(5)
58              ss=[ss c5];
59          case(6)
60              ss=[ss c6];
61      end
62      fh=[fh p];
63  end
64  % Senal FHSS
65  fhss=bpsk.*ss; %producto entre la
    senal bpsk y la secuencia aleatoria
     de frecuencias
66
67  %Grafico del espectro de la senal fhss
68  plot([1:length(fhss)],abs(fft(fhss)));
     %Uso de la funcion fft y abs
69  axis([0 length(fhss) -10 max(abs(fft(
    fhss)))]);
70  title('Espectro en frecuencia de la
    senal FHSS');
```

Script 2. FHSS-Espectro en frecuencia

Resultados obtenidos en la práctica de laboratorio

En la sesión de laboratorio se presentó los resultados obtenidos en el trabajo preparatorio (figuras 1 y 2) los mismos que sirvieron como base para el desarrollo de la práctica de laboratorio.

En la figura 3, se tiene el resultado de la variación en frecuencia de una señal FHSS con respecto al tiempo de la misma, para dicha gráfica los cuadros de color rojo representan la frecuencia de subportadora usada en ese instante de tiempo. Estos cuadros a lo largo del tiempo van cambiando entre 6 niveles de frecuencia, estos saltos de frecuencia corresponden a una secuencia pseudoaleatoria generada con número establecido de bits correspondiente a lo requerido en la hoja guía.

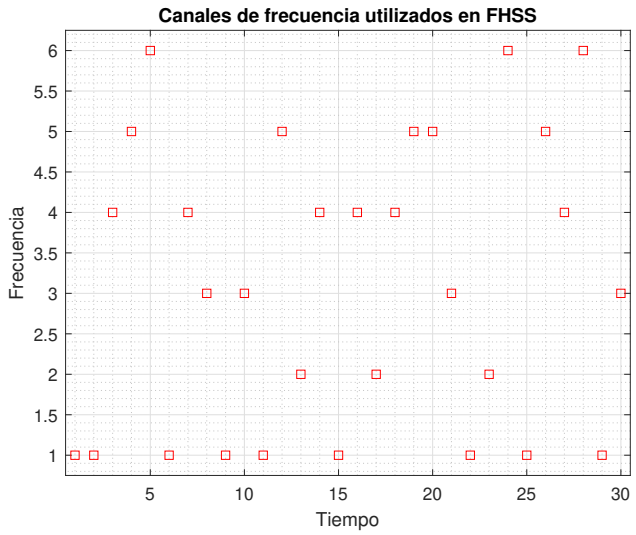

Fig. 3. Canales de frecuencia empleados en función del tiempo

El script 3,como modificación del script 1 permite obtener como resultado la gráfica de la figura 3.

```
1   %% FHSS
2   clc
3   clear all
4   close all
5
6   N=30; %Numero de datos
7   datos=randi([0 1],1,N);    %Generacion
     del vector de datoss
8   senal=[]; %Vector senal vacio
9   portadora=[]; %Vector portadora vacio
10  f=[0:2*pi/119:2*pi];     % Creating
    120 samples for one cosine
11  for k=1:N %lazo for que permite
    recorrer el vector de datos
12      if datos(1,k)==0  %si el bit es 0
13          se=-ones(1,120);    %se
    designan 120 muestras de valor 1
    con signo negativo para cada bit
14      else %si el bit es 1
15          se=ones(1,120);     %se
    designan 120 muestras para cada bit
16      end
17      p=cos(f); %se designa la portadora
     cos(t)
18      portadora=[portadora p]; %se
    registra la portadora de cada bit
19      senal=[senal se]; %se registra la
    senal de datos extendida
20  end
21
22  % Modulacion BPSK de la senal de datos
23  bpsk=senal.*portadora;   %modulacion
    BPSK
24
```

```
% Creacion de las 6 frecuencias de portadora
t1=[0:2*pi/9:2*pi]; %vector de tiempo 1
t2=[0:2*pi/19:2*pi]; %vector de tiempo 2
t3=[0:2*pi/29:2*pi]; %vector de tiempo 3
t4=[0:2*pi/39:2*pi]; %vector de tiempo 4
t5=[0:2*pi/59:2*pi]; %vector de tiempo 5
t6=[0:2*pi/119:2*pi]; %vector de tiempo 6
c1=cos(t1); %portadora 1
c1=[c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1];
c2=cos(t2); %portadora 2
c2=[c2 c2 c2 c2 c2 c2];
c3=cos(t3); %portadora 3
c3=[c3 c3 c3 c3];
c4=cos(t4); %portadora 4
c4=[c4 c4 c4];
c5=cos(t5); %portadora 5
c5=[c5 c5];
c6=cos(t6); %portadora 6
fh=[ ];
% Frecuencias randomicas para FHSS
ss=[]; %vector de frecuencias vacio
for n=1:N %lazo for que designa las frecuencias potadoras a cada bit de dato
    p=randi([1 6],1,1); %obtencion de la secuencia aleatoria de frecuencias
    switch(p); %bucle switch que permite designar las frecuencias portadoras segun la secuencia aleatoria
        case(1)
            ss=[ss c1];
        case(2)
            ss=[ss c2];
        case(3)
            ss=[ss c3];
        case(4)
            ss=[ss c4];
        case(5)
            ss=[ss c5];
        case(6)
            ss=[ss c6];
    end
    fh=[fh p];
end
% Senal FHSS
fhss=bpsk.*ss; %producto entre la senal bpsk y la secuencia aleatoria de frecuencias

f=1:1:30; %creacion de un vector de tiempo
%Grafico de los canales de frecuencia utilizados
figure
plot(f,fh,'s r');
axis([0.5 30.5 0.75 6.25])
title('Canales de frecuencia utilizados en FHSS')
grid on
grid minor
xlabel('Tiempo')
ylabel('Frecuencia')
```

Script 3. Obtención de los canales de frecuencia empleados

- Transmisor y receptor FHSS utilizando un canal afectado por ruido AWGN con un valor de $SNR = 5dB$:

  En la figura 4, se presenta la etapa de un transmisor FHSS, este proceso ya fue tratado en los puntos anteriores durante el análisis del desarrollo práctico.

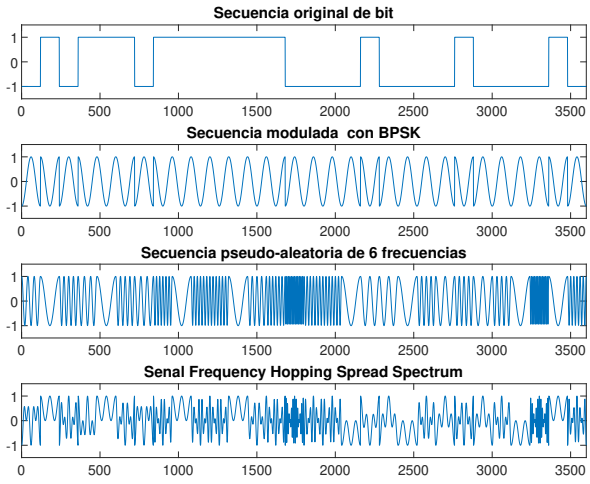

Fig. 4. Señales de datos en transmisión

En la figura 5, la etapa de un receptor FHSS ha pasado por un canal que ha añadido ruido AWGN con una $SNR = 5dB$. Al ser un valor de SNR bajo va a afectar en gran medida a la señal de datos, hasta el punto de distorsionarla inclusive después de pasarla por el sintetizador para poder demodularla.

En el proceso de demodulación a simple vista la señal es irreconocible, debido a que se presenta la señal ensanchada, la mismas que presenta mayor cantidad de bit errados. Sin embargo, sigue manteniendo la naturaleza

de la señal original que debe pasar por un dispositivo de decisión para interpretar de mejor manera la señal recibida.

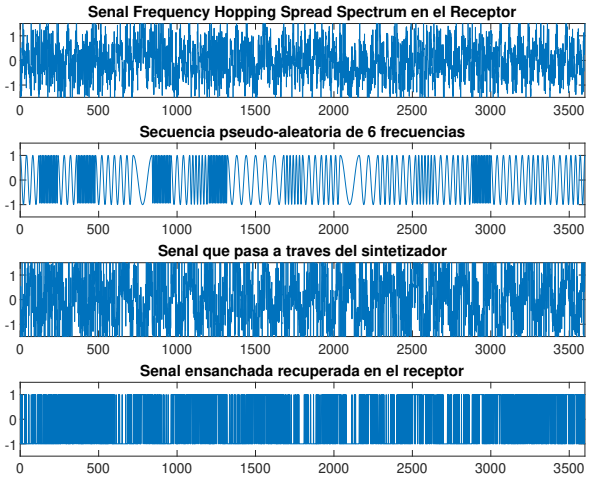

Fig. 5. Señales de datos en recepción

En la figura 6, se presenta la señal FHSS recibida y demodulada que paso a través de un dispositivo de decisión que permitió la correcta interpretación de la señal. Se puede apreciar que tiene un porcentaje de error menor al 30%, esto se debe a que se usó una modulación BPSK y a que la señal fue transmitida usando FHSS.

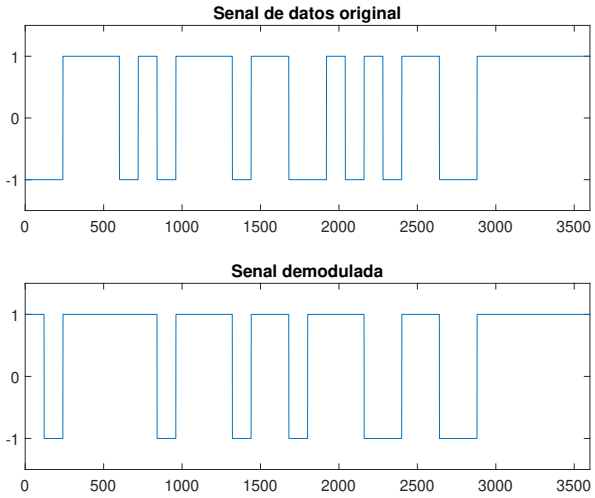

Fig. 6. Señales de datos original y demodulada

Estos resultados se pueden corroborar con el respectivo cálculo del BER como se puede ver en la figura 7, en donde el BER de la señal original es menor que de la señal ensanchada, estos resultados se deben a la cantidad de bits comparados, ya que una señal ensanchada tiene mayor cantidad de bit a comparar.

```
BER senal ensanchada es:0.2975
BER senal original es:0.16667
```

Fig. 7. Resultados obtenidos

- Transmisor y receptor FHSS afectado por un canal con ruido AWGN con un $SNR = 25dB$:

  En la figura 8, se presenta un transmisor FHSS con las mismas características que en el caso anterior (figura 4) sólo que la secuencia de datos y secuencia PN son diferentes.

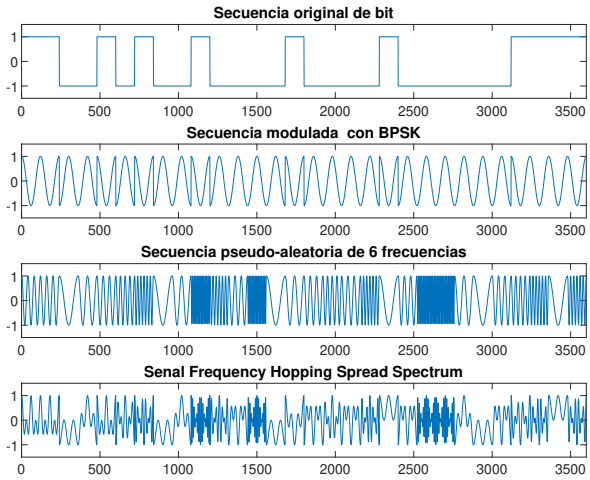

Fig. 8. Señales de datos en transmisión

En la figura 9, el receptor FHSS muestra la señal de datos afectada por ruido AWGN con un SNR=25dB, esto produce que la señal recibida sea mucho más fácil de interpretar en el bloque del sintetizador y demodulador, debido al hecho de tener menor cantidad de bits errados. Sin embargo, la señal aún se ve afectada por ruido, por lo que se debe en este caso también se debe utilizar un dispositivo de decisión al momento de demodular la señal.

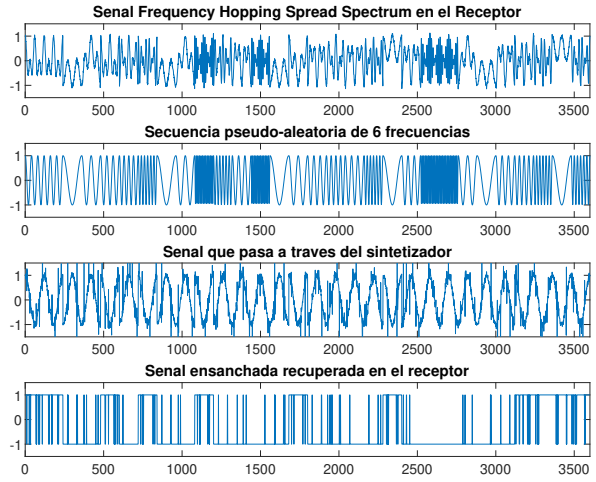

Fig. 9. Señales de datos en recepción

De la misma manera que en el caso anterior, los diferentes bloques utilizados en recepción ayudan a eliminar el ruido AWGN que ha sido añadido durante la transmisión de datos, con el fin de obtener como resultado la figura 10.

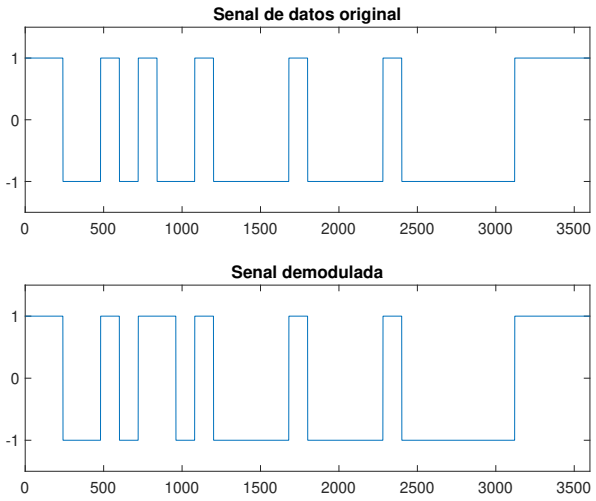

Fig. 10. Señales de datos original y demodulada

En la figura 11, se presenta que el BER de la señal recibida es mucho menor que el caso anterior, esto gracias al uso de un valor de SNR mucho mayor, lo que produce únicamente un bit errado, y un porcentaje de error que no supera el 5% en el caso de la señal original.

```
BER senal ensanchada es:0.044722
BER senal original es:0.033333
```

Fig. 11. Resultados obtenidos

Finalmente, en la figura 12, se presenta una comparación entre los resultados obtenidos al calcular el BER en la recepción de una señal FHSS con diferentes valores de SNR. Todo esto para poder llegar a una conclusión ya esperada, es decir, la cantidad de bit errados disminuye conforme el valor de SNR aumenta y el BER en el caso de la señal ensancha siempre será mayor al BER de la señal original por la cantidad de bits que forman parte de cada una de las señales.

SNR	BERSenalEnsanchada	BERSenalOriginal
5	0.2975	0.1666
25	0.0447	0.0333

Fig. 12. Resultados obtenidos

Todos los resultados mostrados anteriormente se obtuvieron en base al Script 4.

```
%% FHSS
clc
clear all
close all

%%Transmisor
N=30; %Numero de datos
datos=randi([0 1],1,N);    %Generacion del vector de datoss
senal=[]; %Vector senal vacio
SNR=5; %valor de la relacion senal a ruido
portadora=[]; %Vector portadora vacio
t=[0:2*pi/119:2*pi];     % Creating 120 samples for one cosine
for k=1:N %lazo for que permite recorrer el vector de datos
    if datos(1,k)==0  %si el bit es 0
        se=-ones(1,120);    %se designan 120 muestras de valor 1 con signo negativo para cada bit
    else %si el bit es 1
        se=ones(1,120);     %se designan 120 muestras para cada bit
    end
    p=cos(t); %se designa la portadora cos(t)
    portadora=[portadora p]; %se registra la portadora de cada bit
    senal=[senal se]; %se registra la senal de datos extendida
end

%Grafico de la senal de datos
subplot(4,1,1);
plot(senal);
```

```
axis([0 length(senal) -1.5 1.5]);
title('Secuencia original de bit');

% Modulacion BPSK de la senal de datos
bpsk=senal.*portadora;   %modulacion BPSK
%Grafico de la senal modulada
subplot(4,1,2);
plot(bpsk)
axis([0 length(bpsk) -1.5 1.5])
title('Secuencia modulada  con BPSK');
% Creacion de las 6 frecuencias de portadora
t1=[0:2*pi/9:2*pi]; %vector de tiempo 1
t2=[0:2*pi/19:2*pi]; %vector de tiempo 2
t3=[0:2*pi/29:2*pi]; %vector de tiempo 3
t4=[0:2*pi/39:2*pi]; %vector de tiempo 4
t5=[0:2*pi/59:2*pi]; %vector de tiempo 5
t6=[0:2*pi/119:2*pi]; %vector de tiempo 6
c1=cos(t1); %portadora 1
c1=[c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1];
c2=cos(t2); %portadora 2
c2=[c2 c2 c2 c2 c2 c2];
c3=cos(t3); %portadora 3
c3=[c3 c3 c3 c3];
c4=cos(t4); %portadora 4
c4=[c4 c4 c4];
c5=cos(t5); %portadora 5
c5=[c5 c5];
c6=cos(t6); %portadora 6

% Frecuencias randomicas para FHSS
ss=[]; %vector de frecuencias vacio
for n=1:N %lazo for que designa las frecuencias potadoras a cada bit de dato
    p=randi([1 6],1,1); %obtencion de la secuencia aleatoria de frecuencias
    switch(p) %bucle switch que permite designar las frecuencias portadoras segun la secuencia aleatoria
        case(1)
            ss=[ss c1];
        case(2)
            ss=[ss c2];
        case(3)
            ss=[ss c3];
        case(4)
            ss=[ss c4];
        case(5)
            ss=[ss c5];
        case(6)
            ss=[ss c6];
    end
end

%Grafico de la senal aleatoria de frecuencias de salto
subplot(4,1,3)
plot([1:length(ss)],ss);
axis([0 length(ss) -1.5 1.5]);
title('Secuencia pseudo-aleatoria de 6 frecuencias');

% Senal FHSS
fhss=bpsk.*ss; %producto entre la senal bpsk y la secuencia aleatoria de frecuencias

%Grafico de la senal FHSS
subplot(4,1,4)
plot([1:length(fhss)],fhss);
axis([0 length(fhss) -1.5 1.5]);
title('Senal Frequency Hopping Spread Spectrum');

SeTx=awgn(fhss,SNR); %paso a traves de un canal awgn

%%Receptor

defhss1=SeTx./ss; %obtencion de la senal bpsk
demodSe=defhss1./portadora; %demodulacion de la senal bpsk obtenida
v=[]; %vector vacio que permite guardar el valor de x
for i=1:119:length(demodSe)-119 %lazo for que recorre la senal demodulada
   x=mean(demodSe(i:i+119)); %se obtiene el valor medio de los primeros 120 valores
   v=[v x]; %registro del valor de x en el vector v
end

for i=1:length(v) %lazo for que recorre el vector v
    if v(i)<=0 %si la media obtenida es menor que 0, se designa el valor de 0
       v(i)=0;
```

```
    else
        v(i)=1; %caso contrario se designa el valor de 1
    end
end

mat_f=ones(1,120); %variable auxiliar que permite obtener 120 unos
modf=[]; %vector de datos vacios
for i=1:length(datos) %lazo for que recorre el vector v, el mismo que tiene la misma longitud de los datos originales
    mod=v(i)*mat_f; %se realiza el producto entre los 120 unos y el valor v designado anteriormente
    modf=[modf mod]; %se registra el producto obtenido en el vector modf
end

for i=1:1:length(senal) %lazo for que permite recorrer al vector senal
   if senal(i)==-1 %si el bit i es -1 se designa el valor de 0
       senal(i)=0;
   else
       senal(i)=1; %caso contrario se designa el valor de 1
   end
end

[b, ber2]= biterr(senal,modf); % se calcula el BER entre la senal de datos y la senal demodulada

for i=1:length(modf) %lazo for que recorre el vector modf
   if modf(i)==0 %si el dato i es 0, se designa el valor de -1
      modf(i)=-1;
   else %caso contrario se designa el valor de 1
      modf(i)=1;
   end
end
 for i=1:length(senal) %lazo for que permite recorrer el vector senal
   if senal(i)==0 %si el dato 1 es  0, se designa el valor de -1
     senal(i)=-1;
   else
      senal(i)=1; %de lo contrario se designa el valor de 1
   end
end
% Grafico de las  senales de datos original y de la senal demodulada
figure(2)
subplot(2,1,1)
plot(senal)
title('Senal de datos original');
axis([0 length(SeTx) -1.5 1.5]);
subplot(2,1,2)
plot(modf)
title('Senal demodulada');
axis([0 length(SeTx) -1.5 1.5]);

figure(3)

%Senal recibida
subplot(4,1,1)
plot([1:length(SeTx)],SeTx);
axis([0 length(SeTx) -1.5 1.5]);
title('Senal Frequency Hopping Spread Spectrum en el Receptor');

%grafico de la secuencia pseudoaleatoria de frecuencias
subplot(4,1,2)
plot([1:length(ss)],ss);
axis([0 length(ss) -1.5 1.5]);
title('Secuencia pseudo-aleatoria de 6 frecuencias');

%Senal que pasa a traves del sintentizador
subplot(4,1,3)
plot([1:length(defhss1)],defhss1);
axis([0 length(defhss1) -1.5 1.5]);
title('Senal que pasa a traves del sintetizador');

%Senal demodulada con BPSK
dembpsk1 = defhss1./portadora;
r=round(dembpsk1);


for i=1:1:length(r)
   if r(i)>0
       r(i)=1;
   else
       r(i)=-1;
   end
end
subplot(4,1,4)
plot([1:length(dembpsk1)],r);
axis([0 length(dembpsk1) -1.5 1.5]);
title('Senal ensanchada recuperada en el receptor');
for i=1:1:length(senal) %lazo for que permite recorrer al vector senal
   if senal(i)==-1 %si el bit i es -1
```

```
    se designa el valor de 0
191         senal(i)=0;
192     else %caso contrario se designa el
        valor de 1
193         senal(i)=1;
194     end
195 end
196
197 for i=1:1:length(r)%lazo for que
        permite recorrer el vector r
198     if r(i)==-1 %si el bit i es -1 se
        designa el valor de 0
199         r(i)=0;
200     else %caso contrario se designa el
        valor de 1
201         r(i)=1;
202     end
203 end
204
205 [a,ber1]=biterr(senal,r);%obtencion
        del ber entre la senal ensanchada
        en Rx y Rx
206
207 disp([' BER senal ensanchada es:',
        num2str(ber1)])
208 disp([' BER senal original es:',
        num2str(ber2)])
```

Script 4. Obtención del BER de B-PSK

Además, como parte final de la sesión de laboratorio se obtuvo una señal FHSS con una técnica de modulación diferente a BPSK.

En la figura 13, se presenta la etapa de transmisión FHSS, usando una técnica de modulación 8-PSK, donde la designación de las fases fue tomada en referencia a la tabla 1. Así al usar el Script 5 se obtendrá como resultado dichas gráficas.

Símbolo	Fase
000	$5\pi/8$
001	$7\pi/8$
010	$3\pi/8$
011	$\pi/8$
100	$-5\pi/8$
101	$-7\pi/8$
110	$-3\pi/8$
111	$-\pi/8$

Tabla I
FASES Y SÍMBOLOS DE 8-PSK

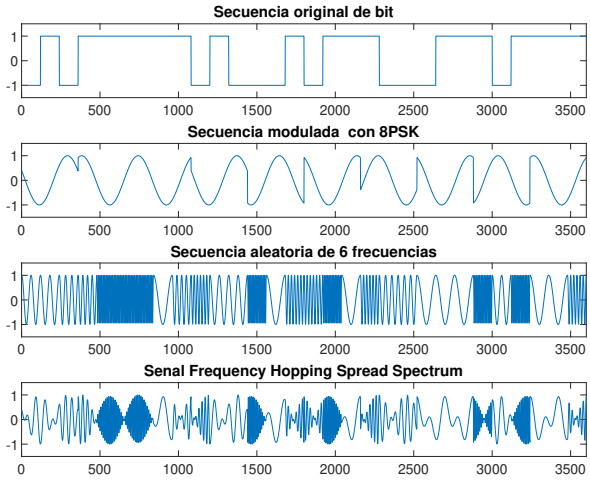

Fig. 13. Señales de datos obtenidas

```
1  %% FHSS
2  clc
3  clear all
4  close all
5
6  N=30; %Numero de datos
7  g=randi([0 1],1,N);    %Generacion del
       vector de datoss
8  senal=[]; %Vector senal vacio
9  portadora=[]; %Vector portadora vacio
10 t=[0:2*pi/119:2*pi];     % Creating
       120 samples for one cosine
11 f=1;
12
13 puntos=360;
14 for k=1:N %lazo for que permite
       recorrer el vector de datos
15     if g(1,k)==0  %si el bit es 0
16         se=-ones(1,120);     %se
       designan 120 muestras de valor 1
       con signo negativo para cada bit
17     else %si el bit es 1
18         se=ones(1,120);      %se
       designan 120 muestras para cada bit
19     end
20     p=cos(t); %se designa la portadora
        cos(t)
21     portadora=[portadora p]; %se
       registra la portadora de cada bit
22     senal=[senal se]; %se registra la
       senal de datos extendida
23 end
24
25 t1=0:2*pi/359:2*pi; %creacion de un
       vector de tiempo
26
```

```
cp=[]; %creacion de un vector de portadora coseno vacio
sp=[]; %creacion de un vector de portadora seno vacio

mod=[];%vector vacio que guarda la portadora I
mod1=[]; %vector vacio que guarda la portadora Q
bit=[]; %vector vacio que guarda los bits

for n=1:3:length(g) %lazo for que permite recorrer los datos con el fin de
    %comparar los simbolos y designar la respectiva senal potadora y fase
    if g(n)==0 && g(n+1)==1 && g(n+2)==1
        die=cos(pi/8)*ones(1,puntos);
        die1=sin(pi/8)*ones(1,puntos);
        se=[zeros(1,50) ones(1,50) ones(1,50)];
    elseif g(n)==0 && g(n+1)==1 && g(n+2)==0
        die=cos(3*pi/8)*ones(1,puntos);
        die1=sin(3*pi/8)*ones(1,puntos);
        se=[zeros(1,50) ones(1,50) zeros(1,50)];
    elseif g(n)==0 && g(n+1)==0  && g(n+2)==0
        die=cos(5*pi/8)*ones(1,puntos);
        die1=sin(5*pi/8)*ones(1,puntos);
        se=[zeros(1,50) zeros(1,50) zeros(1,50)];
    elseif g(n)==0 && g(n+1)==0  && g(n+2)==1
        die=cos(7*pi/8)*ones(1,puntos);
        die1=sin(7*pi/8)*ones(1,puntos);
        se=[zeros(1,50) zeros(1,50) ones(1,50)];
    elseif g(n)==1 && g(n+1)==0  && g(n+2)==1
        die=cos(-7*pi/8)*ones(1,puntos);
        die1=sin(-7*pi/8)*ones(1,puntos);
        se=[ones(1,50) zeros(1,50) ones(1,50)];
    elseif g(n)==1 && g(n+1)==0  && g(n+2)==0
        die=cos(-5*pi/8)*ones(1,puntos);
        die1=sin(-5*pi/8)*ones(1,puntos);
        se=[ones(1,50) zeros(1,50) zeros(1,50)];
    elseif g(n)==1 && g(n+1)==1  && g(n+2)==0
        die=cos(-3*pi/8)*ones(1,puntos);
        die1=sin(-3*pi/8)*ones(1,puntos);
        se=[ones(1,50) ones(1,50) zeros(1,50)];
    elseif g(n)==1 && g(n+1)==1  && g(n+2)==1
        die=cos(-pi/8)*ones(1,puntos);
        die1=sin(-pi/8)*ones(1,puntos);
        se=[ones(1,50) ones(1,50) ones(1,50)];
    end
    c=cos(f*t1); %portadora coseno
    s=sin(f*t1); %portadora seno
    cp=[cp die];    %Amplitud coseno
    sp=[sp -die1];  %Amplitud seno
    mod=[mod c];    %Portadora coseno (Q)
    mod1=[mod1 s];  %Portadora seno (I)
    bit=[bit se]; %bits de datos obtenidos
end

opsk=cp.*mod+sp.*mod1; %obtencion de la senal modulada con 8-psk
%Grafico de la senal de datos
subplot(4,1,1);
plot(senal);
axis([0 length(senal) -1.5 1.5]);
title('Secuencia original de bit');

% Modulacion BPSK de la senal de datos
bpsk=senal.*portadora;   %modulacion BPSK
%Grafico de la senal modulada
subplot(4,1,2);
plot(opsk)
axis([0 length(opsk) -1.5 1.5]);
title('Secuencia modulada  con 8PSK');
% Creacion de las 6 frecuencias de portadora
t1=[0:2*pi/9:2*pi]; %vector de tiempo 1
t2=[0:2*pi/19:2*pi]; %vector de tiempo
```

```
    2
t3=[0:2*pi/29:2*pi]; %vector de tiempo
    3
t4=[0:2*pi/39:2*pi]; %vector de tiempo
    4
t5=[0:2*pi/59:2*pi]; %vector de tiempo
    5
t6=[0:2*pi/119:2*pi]; %vector de
   tiempo 6
c1=cos(t1); %portadora 1
c1=[c1 c1 c1 c1 c1 c1 c1 c1 c1 c1 c1
   c1];
c2=cos(t2); %portadora 2
c2=[c2 c2 c2 c2 c2 c2];
c3=cos(t3); %portadora 3
c3=[c3 c3 c3 c3];
c4=cos(t4); %portadora 4
c4=[c4 c4 c4];
c5=cos(t5); %portadora 5
c5=[c5 c5];
c6=cos(t6); %portadora 6

% Frecuencias randomicas para FHSS
ss=[]; %vector de frecuencias vacio
for n=1:N %lazo for que designa las
   frecuencias potadoras a cada bit de
    dato
    p=randi([1 6],1,1); %obtencion de
   la secuencia aleatoria de
   frecuencias
    switch(p) %bucle switch que
   permite designar las frecuencias
   portadoras segun la secuencia
   aleatoria
        case(1)
            ss=[ss c1];
        case(2)
            ss=[ss c2];
        case(3)
            ss=[ss c3];
        case(4)
            ss=[ss c4];
        case(5)
            ss=[ss c5];
        case(6)
            ss=[ss c6];
    end
end

%Grafico de la senal aleatoria de
   frecuencias de salto
subplot(4,1,3)
plot([1:length(ss)],ss);
axis([0 length(ss) -1.5 1.5]);
title('Secuencia aleatoria de 6
   frecuencias');

% Senal FHSS
fhss=opsk.*ss; %producto entre la
   senal 8psk y la secuencia aleatoria
    de frecuencias

%Grafico de la senal FHSS
subplot(4,1,4)
plot([1:length(fhss)],fhss);
axis([0 length(fhss) -1.5 1.5]);
title('Senal Frequency Hopping Spread
   Spectrum');
```

B. Describir las ventajas y desventajas del uso de FHSS en los sistemas de comunicación.

Ventajas:

- FHSS es robusta ante el ruido, la señal se ve afectada mínimamente ante el ruido. [3]
- Es seguro ya que la secuencia pseudoaleatoria para interpretar la señal solo es conocida por el transmisor y el receptor. [3]
- La eficiencia espectral es un poco mejor comparado DSSS. [3]
- Presenta una probabilidad de bits errados baja al usarlo en conjunto con valores de SNR altos. [3]
- Se puede usar Fast-FHSS o Slow-FHSS, dependiendo de las necesidades del canal. [3]

Desventajas:

- Difícil de implementar debido al sintetizador de frecuencias. [3]
- Sensible a problemas de sicronización de frecuencias cuando es Fast-FHSS y se usa múltiples portadoras. [3]

C. Consultar y describir tres aplicaciones de sistemas comunicación que utilizan la técnica FHSS.

Las técnicas de espectro ensanchado por salto de frecuencia se han implementado con éxito en muchos sistemas espaciales y sistemas de aviónica para comunicaciones de acceso múltiple, protección contra perturbaciones e interferencias [4].

- Bluetooth: Los dispositivos Bluetooth se agrupan en tres clases según su potencia radiada. Las clases 1 y 2 son las más comunes y utilizan un nivel máximo de potencia de transmisión de 1 y 2,5 [mW], respectivamente. Debido a que Bluetooth funciona como PAN, los dispositivos de clase 1 y 2 utilizan un nivel de potencia de transmisión relativamente bajo y tienen un alcance de solo 35 pies. Los dispositivos Bluetooth de clase 3 ”industriales”, menos comunes, pueden funcionar a hasta 100 [mW].

  Se pueden emparejar o vincular hasta ocho dispositivos en un PAN, con un dispositivo que asume la función de maestro y los otros funcionan como esclavos. Los dispositivos funcionan en la banda ISM de 2,4 GHz, pero no son compatibles con el estándar 802.11. Bluetooth

utiliza una técnica de espectro ensanchado por salto de frecuencia (FHSS), con dispositivos que se mueven a través de una secuencia predefinida de 79 canales con un ancho de banda de 1 MHz cada uno.

Los transmisores Bluetooth podrían interferir potencialmente con la mayoría de la banda de 2,4 GHz porque sus canales se superponen con los tres canales 802.11 no superpuestos. Los dispositivos Bluetooth pueden interferir a corta distancia debido a su baja potencia de transmisión. Si hay muchos dispositivos Bluetooth en una celda 802.11, pueden crear un efecto de saturación que tiende a privar de tiempo aire a los dispositivos LAN inalámbricos. Tenga en cuenta que las personas suelen llevar teléfonos, auriculares y periféricos de computadora Bluetooth directamente a su WLAN. Es posible que tenga dificultades para encontrarlos y convencer a sus propietarios de que los dejen fuera de su red inalámbrica [5].

- HAVE QUICK: Es un sistema de salto de frecuencia/resistente a ECM que se utiliza para proteger el tráfico de radio móvil aeronáutico (OR) militar [6]. Especificamente, Have Quick I-II, están diseñados para proporcionar una comunicación aire-tierra y aire-aire efectiva y confiable sin verse afectados por amenazas como interferencias y engaños [7].
- eXtreme Radio Service (eXRS): Es una propiedad de la comunicación personal tecnología comercializada por TriSquare en los Estados Unidos [8]. Radios similares walkie-talkies, usar la banda de 915 MHz, y emplean la tecnología de transmisión de espectro ensanchado por salto de frecuencia para tratar de abordar algunas de las deficiencias percibidas de Servicio de Radio Familiar (FRS). Puesto que la frecuencia en uso se cambia rápidamente en un patrón definido por el número de canal, las transmisiones no se pueden monitorizar por los escáneres de radio disponibles [9] .

IV. Conclusiones

- Ronaldo Almachi:
  1) Cuando usamos FHSS en conjunto con una SNR baja, los resultados son eficientes, y al usar una SNR alta estos mejoran en gran medida tal y como se puede ver tanto en las gráficas resultantes y el cálculo del BER, sin embargo hay que recordar que los sistemas de comunicación que usan FHSS se basan en estándares ya establecidos, por lo que en un caso real no tendríamos la libertad de diseñar un sistema de comunicaciones con Spread Spectrum.
  2) Una de las etapas mas complicadas de implementar en un sistema de comunicación que use FHSS, es la recepción de la señal, ya que en este punto debemos implementar un dispositivo de decisión que nos ayude a recuperar la señal original, ya que la señal después de la demodulación si bien conserva una naturaleza similar a la original, esta es afectada por el ruido lo que provocara bits errados.
  3) La forma de espectro ensanchado es diferente a cuando usamos DSSS, en este caso el espectro ensanchado presenta múltiples impulsos a diferentes frecuencias con diferentes amplitudes, este efecto se puede observar en la señal de salida del transmisor FHSS.
  4) Si usamos técnicas de modulación con un numero de estados mas alto la señal FHSS, es mucho mas sencilla de interpretar en la etapa del receptor.
- Melanny Dávila:
  1) Es posible concluir que la técnica de transmisón FHSS presenta muy buenos resultados, ya que ofrece inmunidad frente a las perturbaciones externas, además de una seguridad elevada, ya que el espectro de la señal transmitida se camufla con el del ruido.
  2) Con la técnicas de transmisón FHSS, el mismo canal de frecuencia se puede compartir entre diversos usuarios con poca interferencia. Este es un uso eficiente del ancho de banda disponible.
  3) A través de FHSS, una señal de datos tiene inmunidad contra la difusión, el ruido eléctrico y la distorsión multitrayecto. Debido a que un recepctor no autorizado sólo puede obtener la información si tiene el mismo código de salto que utilizo el transmisor.
  4) Si la señal modulada se amplía considerablemente (secuencia PN muy grande), implica una reducción en el ancho de banda disponible del canal para otras transmisiones, por lo cual el número de usuarios que pueden acceder simultáneamente al sistema de comunicaciones se reduce considerablemente.

V. Recomendaciones

- Ronaldo Almachi:
  1) Para el sintetizador usar frecuencias bajas cerca de los kHz con la finalidad de que sea mas sencillo interpretar estas señales tanto en tiempo como en frecuencia.
  2) Al recuperar la señal en el receptor usar un segmento de código que funcione como un dispositivo de decisión para recuperar la señal de mejor manera posible.
  3) Cuando cambiamos la técnica de modulación de BPSK a 8PSK, tener en consideración que las longitudes de los vectores donde se guardan los valores cambian, y por eso se debe ajustar dicha longitud en base a los datos generados.
- Melanny Dávila:
  1) Al momento de calcular la tasa de bit errado, se debe ser cauteloso con las longitudes de los vectores a comparar; con el fin de obtener un valor adecuado.

2) Es importante presentar las gráficas de la señal de datos modulada como de la secuencia de saltos, todo esto para poder observar como se obtiene la señal FHSS y poder comparar si efectivamente la señal obtenida es la correcta.
3) Ser cautelosos al momento de obtener designar las fases y símbolos de cada una de las técnicas de modulación a utilizar.

REFERENCES

[1] E. Tatayo, "IMPLEMENTACIÓN DE UN SISTEMA FHSS". C.P. COMUNICACIÓN DIGITAL, Accedido: ago. 14, 2020. [En línea].

[2] D. J. Skinner, "An Introduction to Frequency-Hopping Spread- Spectrum (FHSS) Data Communication Techniques", p. 1.

[3] William Stallings, Wireless Communications and Networks, Second Edition, Pearson Prentice Hall, Upper Saddle River, NJ, 2005. ISBN 0-13-191835-4.

[4] P M C Lal, V S Palsule & K V Ravi (1986) "Applications of Frequency Hopping Spread Spectrum Techniques: An Overview", [En línea]. Disponible en: https://www.tandfonline.com/doi/abs/10.1080/02564602.1986.11437952 $\#text$ =Frequency%20hopping%20spread%20spectrum%20techniques %20have%20been%20successfully%20implemented%20in,%2C% 20Command%20and%20Control)%20applications.

[5] "Bluetooth - CCNA Wireless 640-722 Official Cert Guide [Book]". https://www.oreilly.com/library/view/ccna-wireless-640-722/9780133445725/ch19lev2sec2.html (accedido ago. 14, 2020).

[6] "HAVE QUICK", Wikipedia. dic. 05, 2019, Accedido: ago. 14, 2020. [En línea]. Disponible en: https://en.wikipedia.org/w/index.php?title=HAVE-QUICK&$oldid$ =929414570.

[7] "V/UHF Have Quick-II Waveform — ASELSAN". https://www.aselsan.com.tr/en/capabilities/military-communication-systems/waveforms/vuhf-have-quickii-waveform (accedido ago. 14, 2020).

[8] "Salto de frecuencia de espectro ensanchado - Frequency-hopping spread spectrum - qwe.wiki". https://es.qwe.wiki/wiki/Frequency-hopping-spread-spectrum (accedido ago. 14, 2020).

[9] "Radio Service eXtreme - eXtreme Radio Service - qwe.wiki". https://es.qwe.wiki/wiki/EXtreme-Radio-Service (accedido ago. 14, 2020).