

Instituto  
Tecnológico y de  
Estudios Superiores  
de Occidente –  
ITESO



**ITESO**  
Universidad Jesuita  
de Guadalajara

**Materia:** Sistemas de comunicaciones digitales

**Profesor:** Dr. Omar Longoria

**Fecha:** 17/11/2022

**Autor(es):** Alexis Luna Delgado

Neidys Marioly Vargas Montilva

## Fase 1:

1.- Conectar la salida de audio a la entrada de audio de otra computadora y transmitir una señal sinusoidal.

- Haga un programa que genere una señal sinodal de amplitud 1 (es decir, amplitud pico a pico igual a 2), frecuencia igual a 5 KHz y 10KHz (por separado) y duración de 10 segundos como mínimo. Utilice los comandos `soundsc(x,Fs)` para reproducir;

```
Fs = 96000; % Sampling Fs
Ts = 1/Fs;
F1 = 5000; % SIN_1 frequency
F2 = 10000; % SIN_2 frequency
T = 10; % Duration
t = 0:Ts:(T-Ts); % Vector time

sin5k = sin(2*pi*F1*t);
sin10k = sin(2*pi*F2*t)
```

- Reproduzca las señales de audio en la tarjeta de audio de una computadora.

```
soundsc(sin5k,Fs);
soundsc(sin10k,Fs);
```

- Observe, analice y mida las señales en el osciloscopio (Recuerde, la de 5KHz y luego la de 10KHz).

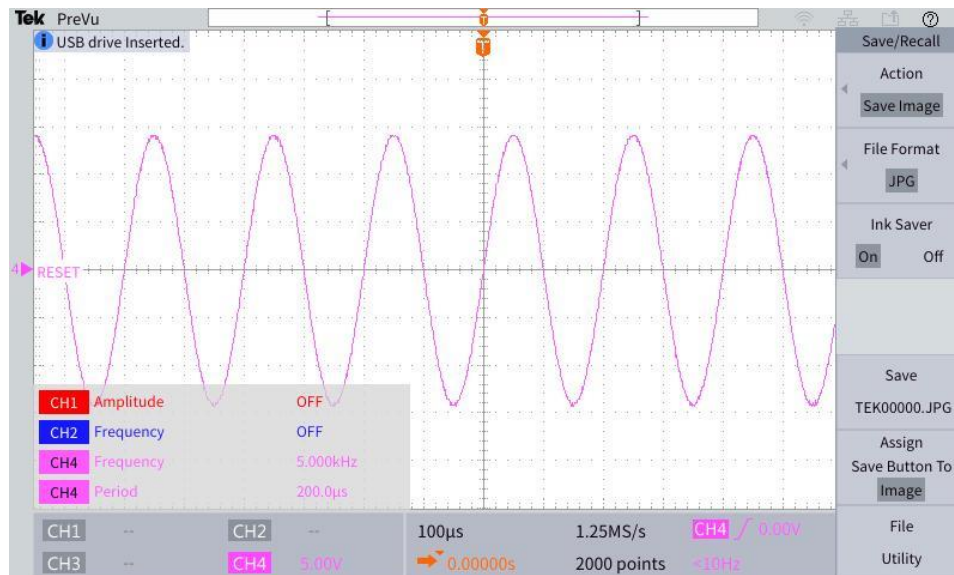
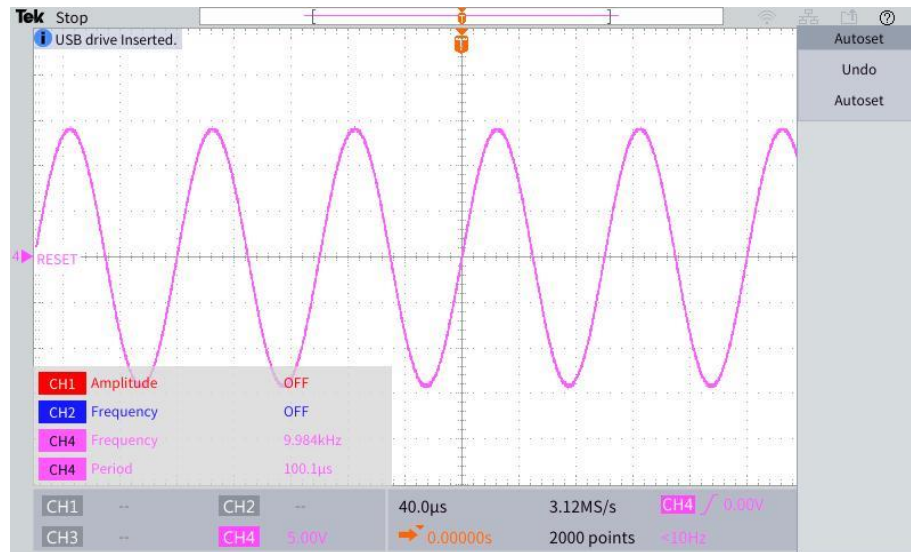


Ilustración 1 señal observada de 5Khz



*Ilustración 2 señal observada de 10Khz*

- Luego conecte el cable de audio al line-in de otra computadora. Transmita nuevamente las señales y grábelas en la computadora receptora utilizando el Software de Audacity. La idea es comenzar con niveles de volumen y sensibilidad muy bajos, e irlos subiendo paulatinamente. En caso de que la señal grabada no se vea bien, conecte la salida de la tarjeta de sonido a un osciloscopio para distinguir si el problema está en la generación o en la grabación de la señal.
- Una vez que estén preparados los niveles de salida del Tx y los niveles de sensibilidad del Rx, grabe la señal en Matlab utilizando `audiorecorder()` para la creación de un objeto y `record()` para grabar.

```

Fs=96000; Nb=16;Chs=1;
recObj = audiorecorder(Fs, Nb, Chs);
get(recObj);
disp('Start speaking.')
recordblocking(recObj, 5);
disp('End of Recording.');
```

*% Play back the recording.*

```

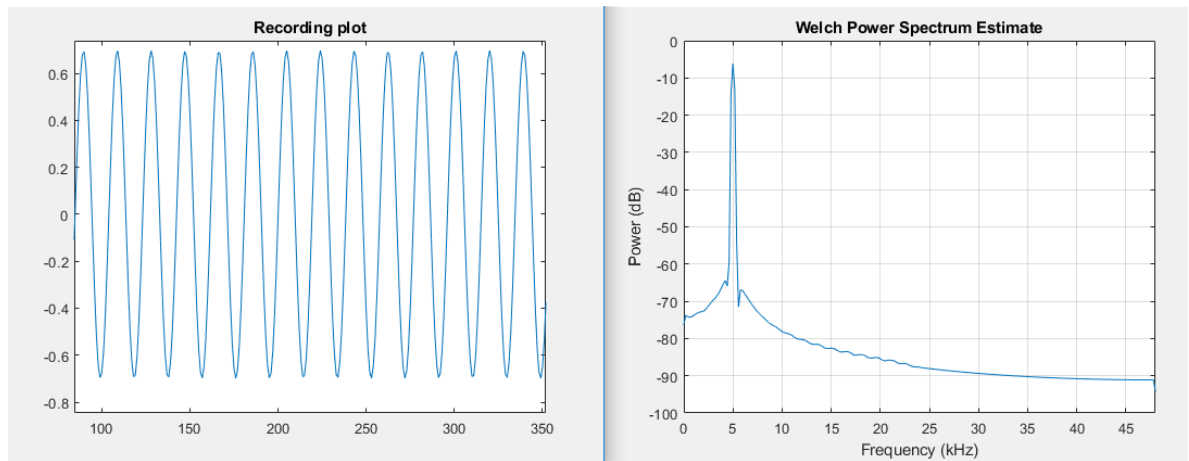
play(recObj);
% Store data in double-precision array.
myRecording = getaudiodata(recObj);
% Plot the waveform.
plot(myRecording);
title('Recording plot');
```

*% Power Spectrum Density:*

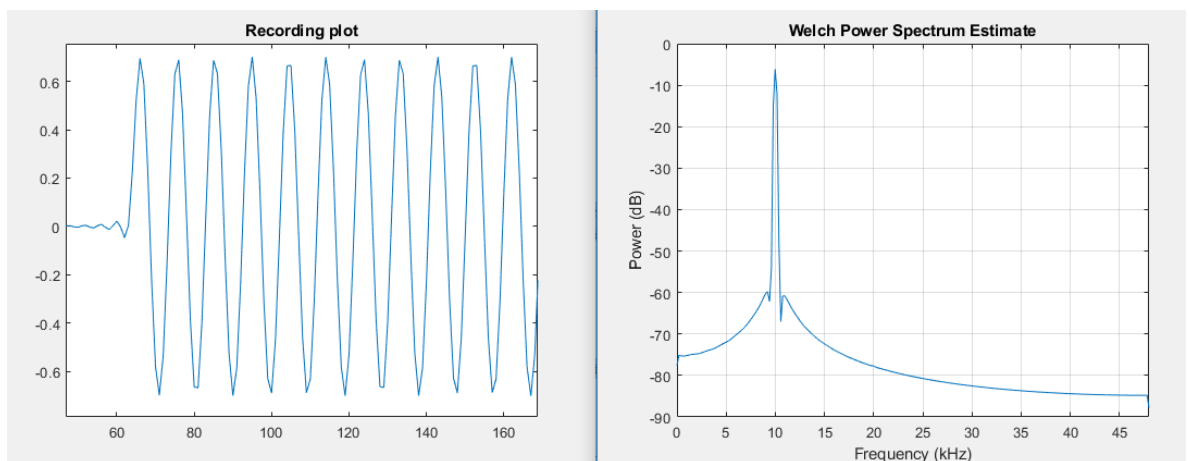
```

figure();
pwelch(myRecording,500,300,500,'one-
side','power',Fs)
```

- Analizar la señal en Matlab. Debe verse la misma sinodal, sin cambios de frecuencia, pero sobre todo sin saturación (debe observarse claramente la curva donde la señal cambia de creciente a decreciente).



*Ilustración 3: Señal seno de 5K Hz*



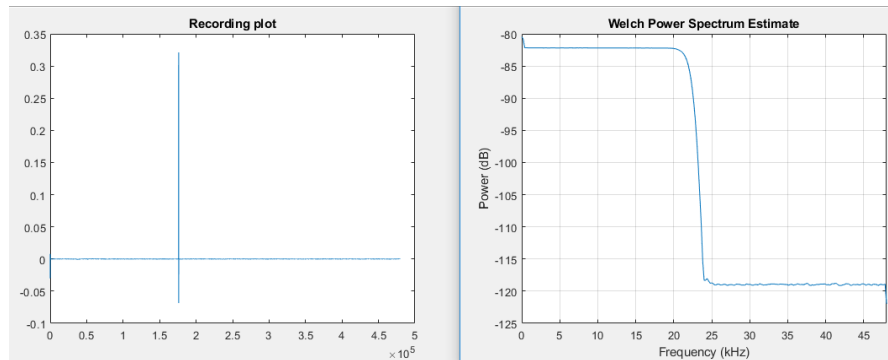
*Ilustración 4: Señal seno 10K Hz*

2.- Identificar el canal, es decir, obtener su respuesta en frecuencia. Vamos a emplear tres técnicas diferentes.

- Primero transmita un impulso, es decir, una señal formada de ceros (digamos, un segundo), después un uno (una sola muestra con voltaje en 1 volt) y después otro segundo de ceros. `soundsc( [zeros(1,Fs) 1 zeros(1,Fs)], Fs )` Según la teoría, la señal grabada es la respuesta al impulso del canal, y su transformada de Fourier es su respuesta en frecuencia (función de transferencia en el dominio de Z).

- Grafique la señal recibida en el tiempo y su PSD (Power Spectral Density)

```
soundsc( [zeros(1,Fs) 1 zeros(1,Fs)], Fs );
```

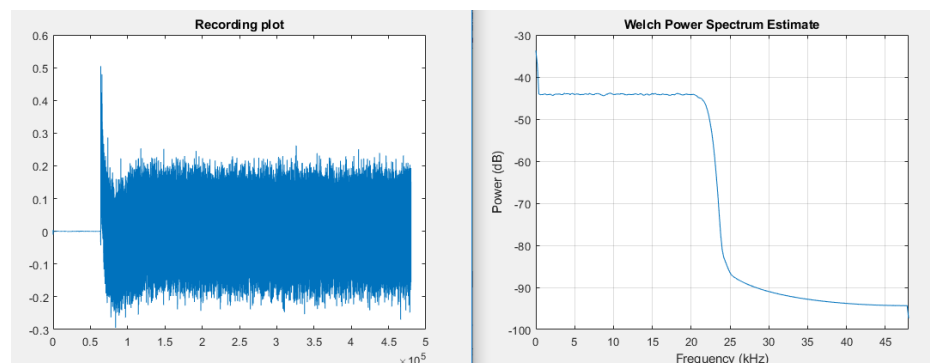


*Ilustración 5: Impulso transmitido*

- Como segunda técnica, utilizaremos una señal que tiene el mismo espectro que el impulso: el ruido gaussiano. Generar una señal de ruido de cinco segundos de duración (ver tutorial sobre el ruido). Utilizando el comando pwelch (u otro comando en Matlab relacionado a la estimación espectral como se vio en clase), estimar la densidad espectral de potencia de la señal. Debe ser muy cercana a plana entre 0 y 48,000 Hz, esto es,  $F_s/2$ .

```
xa = ones(1,Fs*5);
xa_noised = awgn(xa,10);
plot(xa_noised);

soundsc(xa_noised,Fs);
```



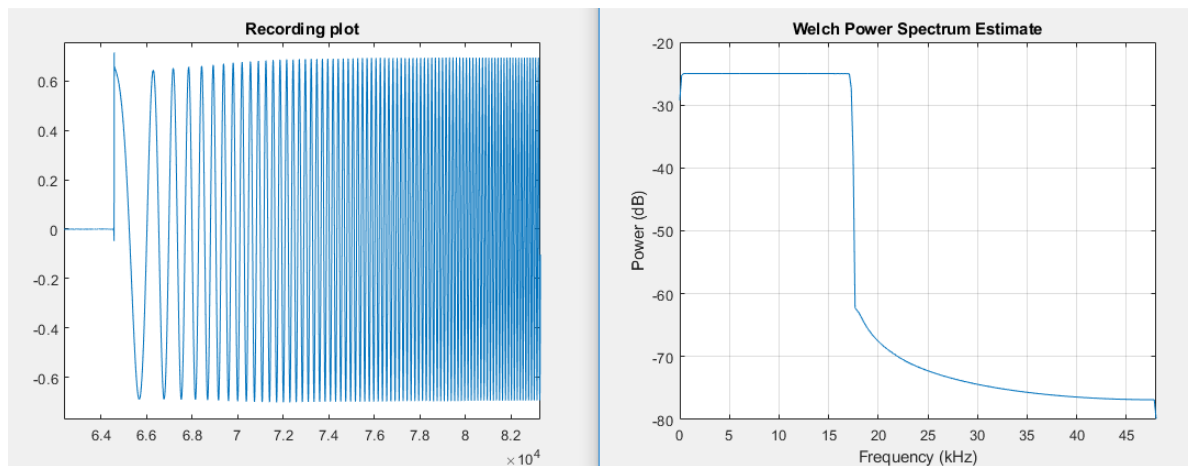
*Ilustración 6: Ruido Gaussiano*

- Transmita y grabe esta señal (puede usar el software Audacity), y grafique el espectro de la señal grabada. Esa es la función de transferencia del canal.

- Genere una señal “chirp” (-1 volt a 1 volt) y de frecuencias 20:20000 Hz, en un lapso de 5 segundos.

```
start=0;
end_c=5;
t = start:1/Fs:end_c;
fo = 20;
f1 = 20e3;
y = chirp(t,fo,end_c,f1,'linear');
figure(1)
pwelch(y,500,300,500,'one-side','power',Fs)
plot(y);

soundsc(y,Fs);
```



*Ilustración 7: Señal Chirp*

3.-Compare las respuestas en frecuencia obtenidas con cada método mostrando gráficas y señalando la información importante.

- ¿Qué conclusiones pueden obtenerse?  
Podemos conocer las limitaciones del canal y sus características, como el ancho de banda, o si invierte o no la señal, y así podemos escoger el código de línea y el tipo de pulso para no exceder las limitaciones del canal.
- ¿Cuál es la banda más plana y más ancha que es posible encontrar en este canal?  
La más plana está alrededor de los 17K Hz y la más ancha está a 22K Hz

## Fase 2:

- Preamble and Start Frame Delimiter (SFD): Construya un vector de 7 octetos
- Obtenga el vector de bits a partir de la imagen de Lena recortada o una señal de audio digital con similar cantidad de bits (pueden venir de un audio comprimido). A estos bits los llamaremos payload
- Construya un “header” que obtiene a partir del tamaño de la imagen (ancho y alto), o el “header” de un archivo wav (frecuencia de muestreo y bits por muestra). Lo llamaremos header.
- Concatene los bits a enviar en un vector de bits concatenados en el siguiente orden: bits2Tx = [preamble; SFD; DSA; header; payload].

```
preamble= [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0  
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0  
1 0]';  
SFD= [1 0 1 0 1 0 1 1]';  
DSA = de2bi(uint8('Practica II FASE II: NoExp1 y  
NoExp2'),8,'left-msb');  
DSA = reshape(DSA,numel(DSA),1);  
load lena512.mat; img = uint8(lena512);  
img = img(248:247+33,245:244+45,1); % Image size= 4(UDE1) x  
4(UDE2) pixels  
imshow(img); % Where UDE =Último Dígito Expediente  
size_img=de2bi(size(img),16,'left-msb')  
header= [size_img(1,:) size_img(2,:)]';  
payload = de2bi(img,8,'left-msb'); % Puede ser right-msb  
payload = payload';  
payload = payload(:);  
bits2Tx =[preamble; SFD; DSA; header; payload];
```

- Utilizando su Tarea 7-8 con los siguientes parámetros para el pulso base:  $F_s = 96000$  Hz, pulso tipo SRRC con  $\beta = 0.25$ ,  $D=10$  y ancho de banda  $B=4000$  Hz. Determine la tasa de bits  $R_b$ , el baud-rate  $R_s$ , y el número de muestras por pulso mp.

Fs	=	96e3;	% Samples per second
Ts	=	1/Fs;	% Sampling period
beta	=	0.25;	% Roll-off factor
B	=	4000;	% Bandwidth available
Rb	=	2*B/(1+beta);	% Bit rate = Baud rate
mp	=	ceil(Fs/Rb)	% samples per pulse
Rb	=	Fs/mp;	% Recompute bit rate
Tp	=	1/Rb;	% Symbol period
B	=	(Rb*(1+beta)/2)	% Bandwidth consumed
D	=	10;	% Time duration in terms of Tp
type	=	'srnc';	% Shape pulse: Square Root Rise
Cosine			
E	=	Tp;	% Energy
[pbase ~]	=	rcpulse(beta, D, Tp, Ts, type, E);	% Pulse
Generation			

- Utilice y mejore los algoritmos/códigos realizados para generar el tren de pulsos del código de línea de las tareas, utilizando los bits2Tx y el pulso base señalado. Lo llamaremos pulse\_train en este documento.

```
s1=int8(bits2Tx);
s1(s1==0)=-1;
s=zeros(1,numel(s1)*mp);
s(1:mp:end)=s1; %Impulse train

xPNRZ=conv(pbase,s);
```

- Realice las mediciones (tanto en la frecuencia y tiempo) correspondientes para el pulso y el tren de pulsos. Determine: ¿cuántos segundos duraría la transmisión?

```
pulse_time = numel(pbase)/Rb
pulse_train_time = numel(bits2Tx)/Rb
```

```
pulse_time =

    0.0236

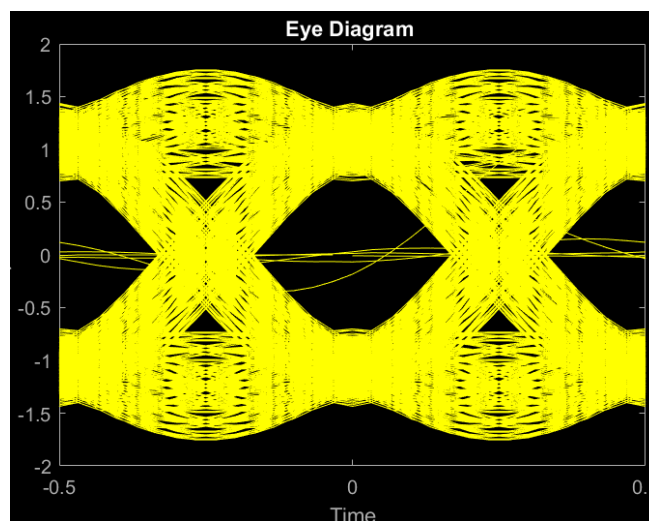
pulse_train_time =

    1.9163
```

*Ilustración 8 duración de la transmisión.*

- Grafique el diagrama de ojo del tren de pulsos que será transmitido.

```
eyediagram(xPNRZ,mp*2);
```



*Ilustración 9 eye diagram del tren de pulsos*



- Justo antes de transmitir la señal, agregue 0.5 segundos de silencio al inicio, de tal manera que al momento de la transmisión emplee la función soundsc( [zeros(1,Fs/2) pulse\_train], Fs ).

```
silence_time = zeros(1,(Fs/2));
soundsc( [silence_time xPNRZ], Fs );
```

- Transmita la señal con la tarjeta de audio y verifique la transmisión (tren de pulsos) en el osciloscopio (dominio del tiempo y frecuencia). Corrobore el tiempo de transmisión de un pulso,  $T_p$ , luego la tasa de transmisión de símbolos (Baud-rate)  $R_s$ , y la tasa de bits (Bit-rate),  $R_b$ .
- Capture la señal en la PC receptora, utilizando el mismo valor de  $F_s$  que el transmisor y considerando el tiempo necesario para capturar la transmisión (incluyendo el silencio). Puede utilizar Audacity o Matlab para la captura. Nota: También pudiera ser capturada por un teléfono celular que acepte una entrada de micrófono.
- 12. Elimine de la señal recibida, la parte que corresponde al silencio

```
Fs=96e3; sec=5; % Time duration of the whole communication
including the silence
recObj = audiorecorder(Fs,16,1);
disp('Start speaking');
recordblocking(recObj, sec);
disp('End of Recording');
Rx_signal = getaudiodata(recObj);

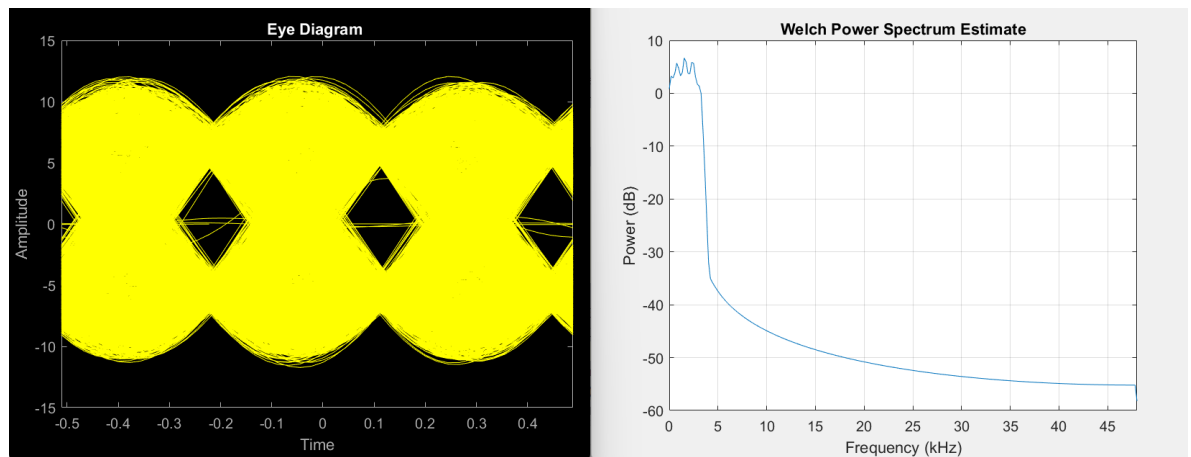
threshold = 0.1; % Detecting the
channel energization
start = find(abs(Rx_signal)> threshold,3,'first'); % Initial
stop = find(abs(Rx_signal)> threshold,1,'last'); % End
Rx_signal = Rx_signal (start:stop);
```

- Pase la señal recibida por el Match Filter correspondiente.

```
match = flip1r(pbase);%el pulso formador tambien lo conoce el
receptor
Rx_signal_match = conv(match, Rx_signal);
```

- Grafique el diagrama de ojo de la señal recibida. Observe y analice la distorsión.

- Grafique y analice la densidad espectral de potencia de la señal recibida



*Ilustración 10 eye diagram de la señal recibida y su espectro de potencia*

- Muestree la señal a la salida del Match Filter y trate de identificar los bits de preámbulo, inicio de trama, DSA y los bits del header.

```
>> preamble'

ans =

Columns 1 through 20
    1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0

Columns 21 through 40
    1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0

Columns 41 through 56
    1     0     1     0     1     0     1     0     1     0     1     0     1     0     1     0
```

*Ilustración 11 bits del Preámbulo de la señal*

```
>> sfd'

ans =

    1     0     1     0     1     0     1     1
```

*Ilustración 12 sfd de la señal*

dsa	
280x1 double	
	1
1	0
2	1
3	0
4	1
5	0
6	0
7	0
8	0

*Ilustración 13 bits de DSA de la señal*

- Repita los puntos (5 en adelante) pero utilice como pulso formador el SRRC con  $\beta = 0.25$  y  $B=12000\text{Hz}$

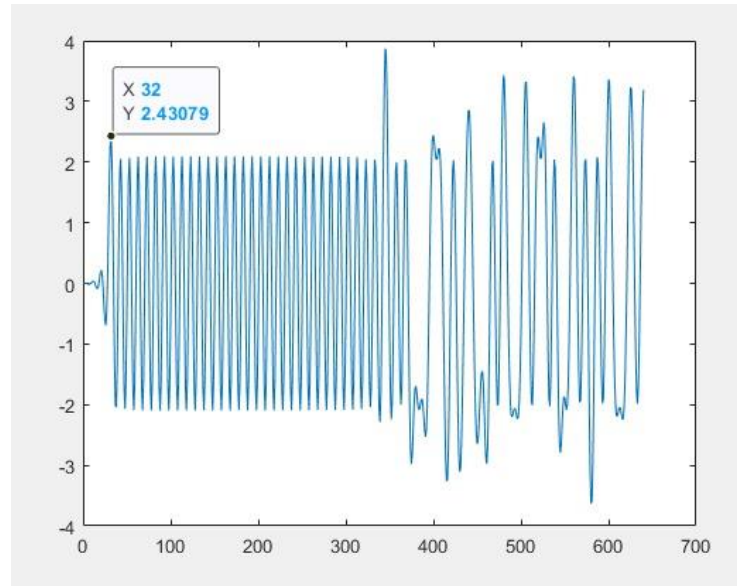


Ilustración 14 primeras 128 muestras de la señal

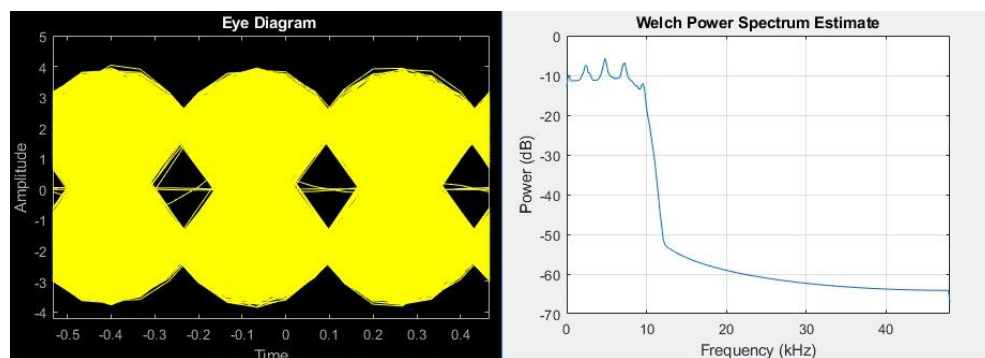


Ilustración 15 eye diagram y señal de potencia de la señal

Preámbulo:

```
ans =  
  
Columns 1 through 20  
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0  
  
Columns 21 through 40  
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0  
  
Columns 41 through 56  
1 0 1 0 1 0 1 0 1 0 1 0 1 0
```

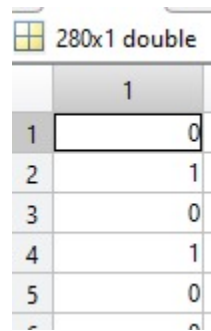
Ilustración 16 preámbulo de la señal

SFD:

```
ans =  
  
1 0 1 0 1 0 1 1
```

*Ilustración 17 SFD de la señal*

DSA:



280x1 double	
	1
1	0
2	1
3	0
4	1
5	0

*Ilustración 18 DSA de la señal*

- ¿Observa alguna diferencia al realizar el punto 8 y 14 con las dos transmisiones?  
Argumente sus conclusiones correspondientes a esta etapa de la práctica.

Al momento de comparar las dos transmisiones podemos notar ciertas diferencias en las gráficas de potencia, así como también en la resolución o claridad al momento de ver su diagrama de ojo. Donde podemos apreciar también el cambio del ancho de banda plana entre cada grafica.

### Fase 3:

```
end_preamble = 56*mp+(start_preamble-  
1)+8*mp+280*mp;  
  
Rx_signal_match =  
Rx_signal_match(1:numel(bits2Tx));  
Rx_signal_match = Rx_signal_match';  
  
bitsM1=  
Rx_signal_match(start_preamble:end_preamble);  
bitsM1 = reshape(bitsM1,16,2);  
bitsM1 = bitsM1';  
decval1 = bi2de(bitsM1,'left-msb');  
lenaRS1 = reshape(decval1, size(bits2Tx));  
imshow(uint8(lenaRS1))
```

## Resumen:

### *Experimental Characterization of RGB LED Transceiver in Low-Complexity LED-to-LED Link*

Hoy en día el internet de las cosas ha tomado gran parte de la industria, por lo que el artículo nos da una breve explicación de un sistema embebido aplicado a este ámbito. Dicho sistema se basa en la comunicación usando dos LEDs para la transmisión de información a una velocidad de 200kBits/seg. Teniendo en cuenta que el sistema se plantea basado en sus requerimientos y restricciones, una de estas sería el ruido, la cantidad de dispositivos conectados a la misma red. Etc.

Como solución se propuso usar células de menor tamaño y en rangos distintos de operación. Para esto se usa VLC (visible light communication), es un tipo de comunicación usando la luz visible, método de muy bajo consumo, con gran aporte de seguridad debido a su rango de comunicación, inmunidad a ser interferida por el espectro de comunicaciones de radiofrecuencia.

Para la resolución de este sistema también se busca rapidez a la hora de transmitir, por lo que el sistema implementado consta de un LED operando con un driver al que opera en frecuencias mucho más altas que las percibidas por el ser humano. La ventaja que se tiene de este tipo de sistemas o de comunicaciones es que es fácil de implementar, bajo costo, pero por otro lado se debe tener en cuenta que por sus distancias está a mayor grado de exposición al ruido blanco que otros sistemas, pero esto puede evitarse aplicando un filtro como bien lo han propuesto en el artículo.

Lo dicho anteriormente fue confirmado al momento de poner en práctica el sistema, por lo que efectivamente había ruido blanco en la transmisión, también se pudo apreciar el decaimiento de la señal entre el receptor y el transmisor. Por lo que cuando se aplicó el Match filter para mejorar la señal transferida, luego de esto fue posible calcular el BER utilizando un método de división a la diferencia del valor promedio de los voltajes transmitido, dando como resultado el saber que a menor distancia se podía tener tasas de transmisión más altas, pero si se separaba el receptor del transmisor se conservaba un rango superior al del BER, por lo que este rango constaba de por cada 1000 Bits transmitidos por 1 error. Por lo que se llegó a la conclusión que a mayores distancias a las regulares por este método, sin usarlo mejoraría hasta un 90% respecto a la eficiencia en presencia del ruido blanco.