ITESO

Universidad Jesuita
de Guadalajara

Sistemas de Comunicaciones Digitales

Práctica 3

Luis Fernando Rodriguez Gutierrez ie705694

Omar Humberto Longoria Gándara

11/05/2020

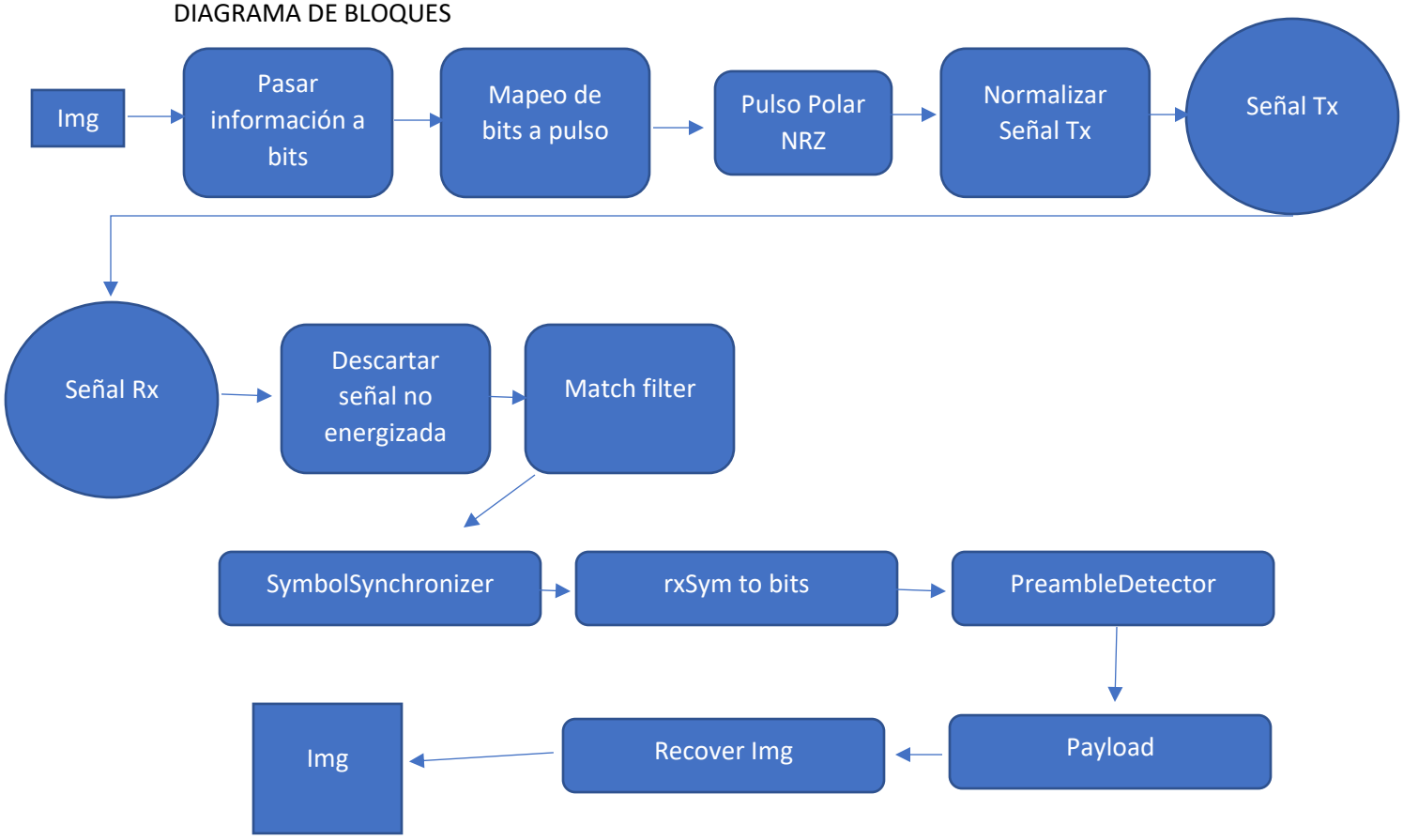
Contenido

INTRODUCCIÓN

El objetivo de esta práctica es el lograr enviar un paquete de información de una computadora a otra, de manera que se empleen los conceptos vistos en clase, así mismo como las técnicas.

Esta práctica hace énfasis a la importancia de la implementación de un transmisor y un receptor. Para que la información logre viajar por un medio y lograr recuperar la señal en la mejor calidad posible. Así mismo lograr una sincronización correcta para que la información

DIAGRAMA DE BLOQUES



El proceso principal que se debe de emplear es un experimento en el cual lo que buscamos es encontrar el ancho de banda máximo del sistema. Para de este punto llegar a una decisión y encontrar un valor óptimo para la transmisión de la señal. En mi caso opte por los siguientes valores.

```
Fs      =   96e3;               % Samples per second
Ts      =   1/Fs;               %
beta    =   0.22;                % Roll-off factor
B       =   16e3;               % 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       =   6;                  % Time duration in terms of Tp
type    =   'srrc';             % Shape pulse: Square Root Rise Cosine
E       =   Tp;                 % Energy
[pbase ~] = rcpulse(beta, D, Tp, Ts, type, E);    % Pulse Generation
```

Esto realizando un experimento por medio de una señal chirp como la siguiente.

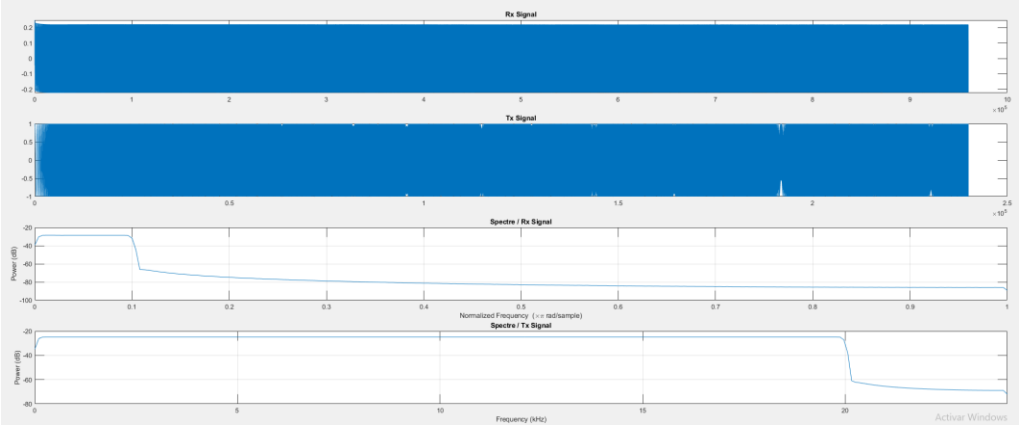
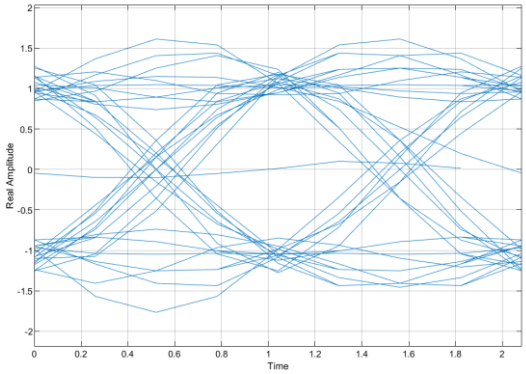


Ilustración 1. Señal chirp, empleada para encontrar el BW maximo.

SEÑAL TX

En la siguiente transmisión que se realizó, se tomó la decisión enviar información con formato tipo NRZ (POLAR) con una combinación de un pulso SRRC. Esto optando por un mejor método para el envió de la información.

B = 16e3



Por medio de la herramienta de Matlab wvtool podemos observar en su ancho de banda, su primer nulo se encuentra en aproximadamente los 16e3. Tomando como los valores normalizados que:

$$\frac{1}{0.328} = \frac{48e3}{15.75e3}$$

Ilustración 2. Diagrama de ojo de la señal con B = 16e3

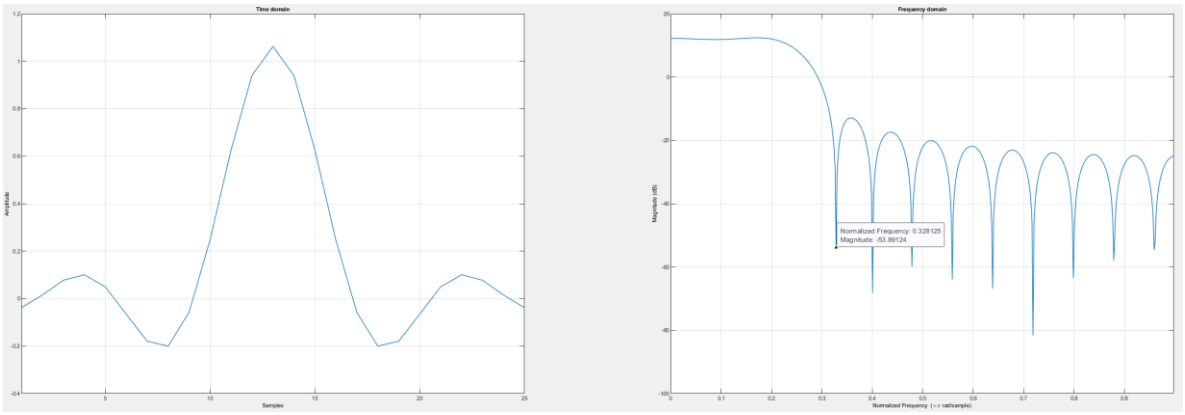


Ilustración 3. Analisis de la señal con WVtool

Con dichos valores obtengo un valor de mp = 4. Este valor es de suma importancia en cuanto a los valores que hay que decidir para lograr la transmisión, dado que, si se obtiene un valor de mp mejor de 3, se puede considerar que no se lograra la recuperación de la señal de manera adecuada. Ya que no se tendrán suficientes muestras por pulso.

De manera consiguiente obtenemos la forma siguiente del tren de pulsos generados con un SRRC. Por ultima etapa se corroboro que la forma del tren de pulsos fuera adecuada y de misma manera que sus valores fueran válidos. Uno de los procesos fue analizar su señal de potencia y corroborar que en realidad su ancho de banda corresponda al que le asignamos.

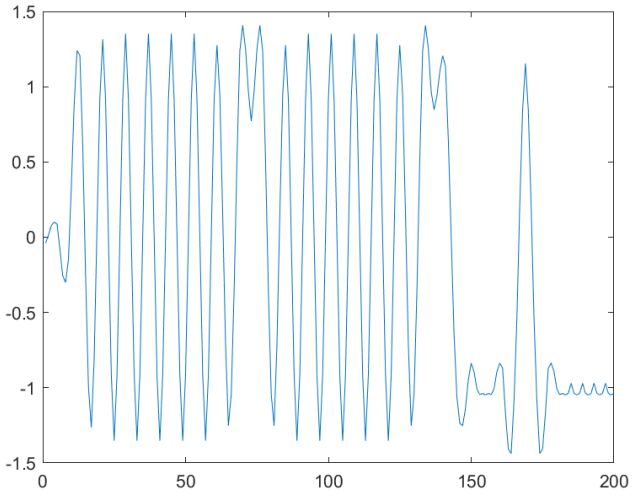


Ilustración 4. Tren de pulsos SRRC

Como se logra visualizar en la imagen 5, podemos ver que en definitiva el BW de la señal se aproxima a los 16,000 Hz que le asignamos. De manera que ahora podemos pasar a la siguiente fase, que es ya evitar la información por un medio hacia un receptor. En mi caso, se paso la señal hacia un dispositivo celular, y de ahí se envió hacia la computadora para ser recibida por Audacity.

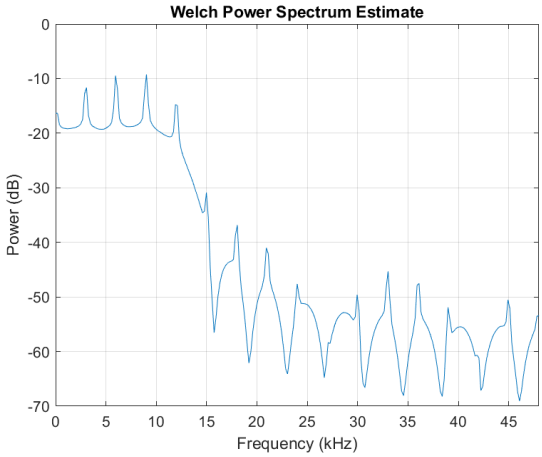


Ilustración 5. análisis de PSD del tren de pulsos.

SEÑAL RX

En esta fase, en la cual es el desarrollo de la practica 3. Lo que se realizo primero fue la optimización de los valores para lograr una recuperación correcta de la LENA512 completa. En donde dicha información lo que se hace en guardarla en un archivo .wav para ser enviada desde un teléfono.

Dicha grabación, tiene un tiempo aproximado de 87 segundos, esta para lograr la transmisión total de la imagen. Dicho tiempo de transmisión se calculó por medio de la siguiente formula;

```
%sec2Tx
sec2Tx = numel(bits2Tx)/Rb
```

Tomando los siguientes valores.

```
Fs      =   96e3;               % Samples per second
Ts      =   1/Fs;               %
beta    =   0.22;                % Roll-off factor
B       =   16e3;               % 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       =   6;                  % Time duration in terms of Tp
type    =   'srrc';             % Shape pulse: Square Root Rise Cosine
E       =   Tp;                 % Energy
[pbase ~] = rcpulse(beta, D, Tp, Ts, type, E);    % Pulse Generation
```

Una vez enviada la información que se quiere transmitir, en este caso una imagen de la LENA. Esta se recupera por medio de Audacity, para luego ser tomada en Matlab.

El siguiente proceso que se realizo fue el normalizar la señal con un rango de valores de 1 y -1. Esto buscando facilitar la automatización del sistema.

```
%% Rx side
clear all ; clc;
format longg

%Audio read from WAV file.
filename= 'lena_rx.wav';
INFO = audioinfo(filename)
nBits = INFO.BitsPerSample;
Fs_audio = INFO.SampleRate;
%samples = [1,sec*Fs_audio];
[Rx_signal,FsWav] = audioread(filename);
Rx_signal = normalize(Rx_signal, 'range', [-1 1]);
```

De manera consiguiente lo que se busco fue eliminar los silencios como se muestra en el código. Obtenemos los siguientes 500 valores.

```
%%
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);
```

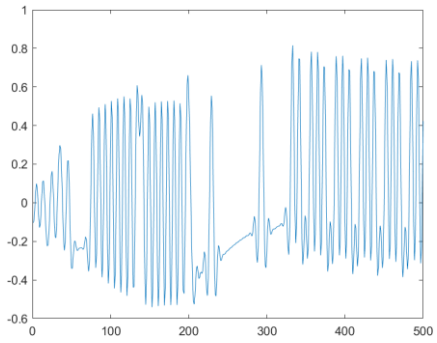


Ilustración 6 Señal recibida por Audacity

De manera consiguiente se realiza una convolución de la señal recibida con el pulso base, de manera que ahora que tenemos tanto una señal normalizada y “recuperada”. Obtenemos el diagrama de ojo para poder visualizar que tan abierto quedo este para la recuperación de información.

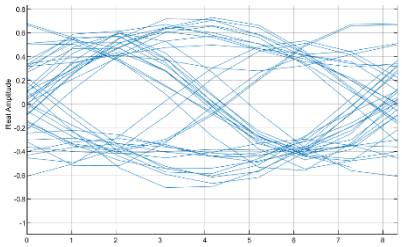


Ilustración 7. Diagrama de ojo de la señal Rx

En este punto del análisis de la señal Rx, me detuve y regresé un poco para verificar que el BW de la señal fuera el adecuado. De manera que a lo que se observa es que la información, o mejor dicho, la energía de la señal se mantiene debajo del BW original estipulado.

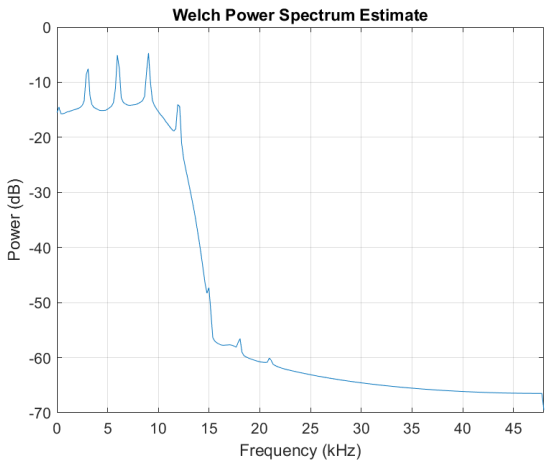


Ilustración 8. análisis de PSD de RX

SYNC

Este proceso lo que busca es evitar el buscar por medio de “a ojo de buen cubero” el punto adecuado para empezar a muestrear cada mp muestras. De misma forma que este proceso se busque automatizar este proceso y se logre recuperar información de paquetes mas grandes.

```
%% Sync
% Sampler:
symSync = comm.SymbolSynchronizer('TimingErrorDetector','Early-Late (non-data-aided)','SamplesPerSymbol', mp);
% El sincronizador utiliza la señal del Match Filter
% y también muestrea, dejando solamente los símbolos
rxSym = symSync(norm_signal);
release(symSync); % Liberar el objeto
```

Ahora los valores obtenidos, los decodificamos y convertimos a bits.

```
%% Convert rxSym to bits

bits_rxSym = rxSym(:);

bits_rxSym(bits_rxSym < 0) = 0;
bits_rxSym(bits_rxSym > 0) = 1;
figure; plot(bits_rxSym(1:60));
```

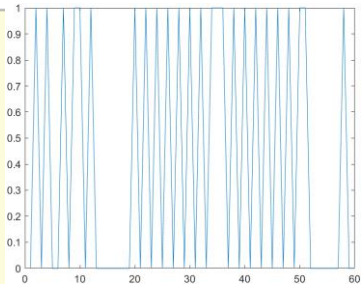


Ilustración 9. Valores recuperados a bits

Recuperamos el preámbulo por medio del siguiente código, en este lo que realiza es por medio del preámbulo ya conocido. La función de este pedazo de código es, en un máximo de muestras es que este mismo busque los bits. De manera consiguiente elimina los bits basura, para ya obtener los puros bits.

```
%% Preambulo
preamble_bits = [1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 1]';

preamble_detect = comm.PreambleDetector(preamble_bits,'Input','Bit');
% Deteccion de preámbulo. El índice indica dónde termina la trama
idx = preamble_detect(bits_rxSym(1:128)) % Ventana de 128 bits
% Una vez que encuentra el índice, se descartan los "bits basura"
% Una forma de hacerlo es la siguiente:
rxData= bits_rxSym(idx+1-numel(preamble_bits):end);
```

En el siguiente segmento de código lo que buscamos es reutilizar el código de la P02. Sin embargo, no se puede usar tal como se implementó, se buscó optimizar y que lográramos obtener los mismos resultados para recuperar lo que es el header.

```
%%
%preamble num elements
preamble_size = numel(preamble_bits);
header_size = 16;

header_bits = rxData(preamble_size+1:preamble_size+(2*header_size));

header_row = bi2de(header_bits(1:numel(header_bits)/2)','left-msb');
header_col = bi2de(header_bits(numel(header_bits)/2:end)','left-msb');
```

Por último, lo único que se hizo fue la conversión de información, y de misma manera recuperar lo que es el payload para recuperar la imagen de la LENA.

```
%% Payload
payload_size = header_row * header_col * 8;
bits_payload = rxData(preamble_size+(header_size*2)+1:end);

%Recover img
bi2mat = vec2mat(bits_payload,8);
lena_bin2dec = bi2de(bi2mat,'left-msb');
new_mat = vec2mat(lena_bin2dec,header_row);
new_mat = new_mat';
figure; imshow(uint8(new_mat));
```

Ilustración 10. Lena Recuperada, con bajo VER

Analizando la imagen podemos ver que el VER de la imagen seria bajo, dado que solamente tenemos uno que otro píxel mal recuperado de la señal transmitida. Sin embargo, esto es lo opuesto; esto ocasionado a lo que logre visualizar al ver el vector de la señal Rx con la Tx, en un punto tenemos un desfase de bits, esto ocasionando que el BER de la señal recuperada sea alto. De manera que este BER no se logra visualizar gracias al bloque sincronizador que mitiga y logra evitar este efecto.

```
b_error =

          53.2263724861912
```

Este efecto se logra visualizar desde la recuperación del header, en donde lo que debería corresponder a su contraparte de “header_row”, en la variable “header_col” recuperamos 768 en lugar de 512. Esto provocado por un error a la recuperación de ese bit en específico, como se mostrará en la siguiente imagen.

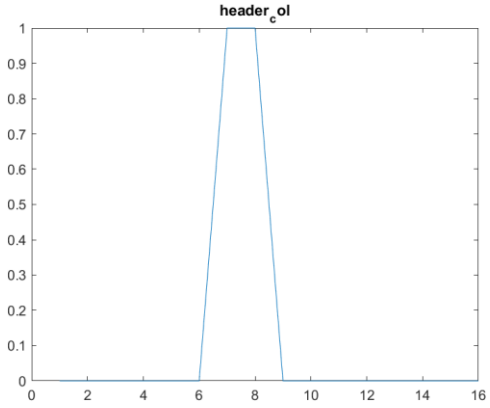


Ilustración 11. Recuperación del header_col de la imagen.

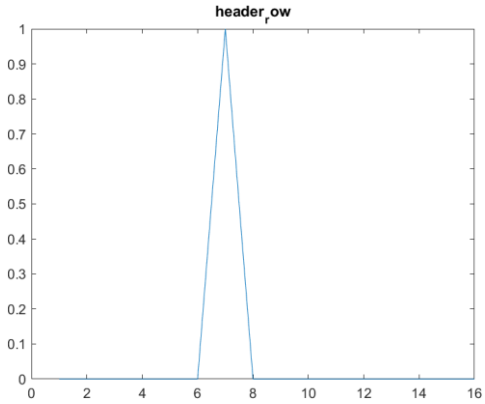


Ilustración 12. Recuperación del header_row de la imagen.

APRENDIDO EN EL CURSO

En general del transcurso del curso es la diferencia, pros y contras de los diferentes tipos de código de línea. Es decir, que ventajas nos puede dar PNRZ a diferencia del Uni polar, de misma manera, el código manchester. Que es de los mejores para poder llegar a recuperar información de una manera más fiable, ya que, en el transcurso de cada símbolo transmitido, se puede recuperar no solo la información, sino, también la frecuencia de esta a la que fue transmitida. Esto permitiendo una mejor sincronización. Así misma como la caracterización de un canal, por medio de cuanto bandwidth es lo que puede soportar, así mismo para poder seleccionar parámetros correctos al elegir que tipo de pulso base es el que se utilizara para la transmisión de información.

CONCLUSIONES

Principalmente mediante la experimentación, pude llegar a entender a más trasfondo el sistema de comunicación que se empleó para esta práctica. El cual solo es un fragmento de un sistema real que se aplica en el mundo real. Así mismo como que la cantidad de información que envíe fue sumamente pequeña a lo que se maneja. Sin embargo, se puede considerar también que fue pesada por la frecuencia y el bandwidth que se envió, de manera que tome la decisión de llegar a sacrificar tiempo de transmisión, por calidad de la información. Es decir, reducir la velocidad de envió de cada paquete (símbolo) para lograr una mayor calidad.

De misma manera la importancia de los códigos que llegan a facilitar la sincronización de información. Esto facilitando y optimizando trabajo. El cual llega a ser de gran ayuda al momento de implementar esto en algún sistema embebido.

ENLACE VIDEO

https://drive.google.com/file/d/12LJZ8YOLG3x9DhjAmKpVzHZBovJdf2Fl/view?usp=sharing