Codificación de fuente (LZW)

Elizabeth Londoño Mora -Teodoro Calle Lara- Luis Eduardo Rodriguez Ascuntar
Departamento de Ingeniería Electrónica y de Telecomunicaciones
E-mail: elizabeth.londonom@udea.edu.co
teodore.calle@udea.edu.co
luis.rodriguez@udea.edu.co

I. RESUMEN

El objetivo de los sistemas de comunicaciones digitales es enviar, con la máxima fiabilidad y eficiencia, una secuencia de bits que proporcionan un flujo de información desde el emisor al receptor, la mayor parte de los contenidos de esta asignación se centran en el estudio de procedimientos que mejoren esta viabilidad y eficiencia en el sistema de comunicaciones.

En el presente trabajo se propone una comprensión de símbolos modulados en QPSK tipo LZW, los cuales se transmiten a través de un canal inalámbrico, de esta manera buscamos comprender los diferentes conceptos que se encuentran en esta codificación, se buscó situar la codificación del mensaje dentros de un sistema de comunicaciones genéricos .

palabras claves: codificador, decodificador, sistema , canal ,compresión

II. ABSTRACT

The objective of digital communication systems is to send, with maximum reliability and efficiency, a sequence of bits that provide a flow of information from the sender to the receiver. Most of the contents of this assignment focus on the study of procedures that improve this viability and efficiency in the communications system.

In the present work we propose an understanding of LZW-type QPSK modulated symbols, which are transmitted through a wireless channel, in this way we seek to understand the different concepts found in this encoding, we sought to locate the encoding of the message within a generic communications system.

Keywords:.encoder, decoder, system, channel

III. MARCO TEÓRICO

El algoritmo LZW proviene de una mejora propuesta por Terry Welch (1984) a los algoritmos propuestos por Abraham Lempel y Jacob Ziv. Se trata de un método de compresión sin pérdidas, ya que los datos cifrados pueden ser perfectamente reconstruidos en el receptor.[1]

La ventaja de LZW radica en su dinamismo, ya que a la vez realiza la codificación de los datos y la generación de nuevas entradas, creando un diccionario de tipo semi-adaptativo que no requiere ser conocido por el receptor. Los Patrones repetitivos forman una tabla para codificar, donde para cada entrada podrán agruparse varios símbolos concatenados. Este arreglo llamado diccionario o tabla de patrones es necesaria para asignar el código correspondiente.Con lo anterior se puede asignar código a una mayor cantidad de símbolos que como se designaría normalmente. Es decir, a dos símbolos se les pudiese otorgar el mismo código en longitud que se asignaría a uno solo normalmente, a tres o cuatro lo de dos, entre otras cosas que puedan presentarse. Con esta reducción se tiene la compresión correspondiente, la longitud final del código propicia que se reduzca la cantidad de sımbolos iniciales[2]

La clave del método LZW reside en que es posible crear sobre la marcha, de manera automática y en una única pasada, un diccionario de cadenas que se encuentren dentro del texto a comprimir mientras al mismo tiempo se procede a su codificación. Dicho diccionario no es transmitido con el texto comprimido, puesto que el descompresor puede reconstruirlo usando la misma lógica con que lo hace el compresor y, si está codificado correctamente, tendrá exactamente las mismas cadenas que el diccionario del compresor tenía. [3]

El diccionario comienza pre-cargado con 256 entradas, una para cada carácter (byte) posible más un código predefinido para indicar el fin de archivo. A esta tabla se le van agregando sucesivos códigos numéricos por cada nuevo par de caracteres consecutivos que se lean.

Al armar el diccionario sobre la marcha se evita hacer dos pasadas sobre el texto, una analizando y la otra codificando y dado que la regla de armado del diccionario es tan simple, el descompresor puede reconstruirlo a partir del texto comprimido mientras lo lee, evitando así incluir el diccionario dentro del texto comprimido. Se puede objetar que el diccionario estará plagado de códigos que no se utilizarán y por tanto será innecesariamente grande, pero en la práctica el diccionario no crece demasiado y aún si lo hiciera no importa mucho pues el objetivo es que el archivo comprimido sea pequeño aun cuando los procesos de compresión y descompresión pudieran ocupar mucha memoria con el diccionario.[3]

Las entradas del diccionario pueden representar secuencias de caracteres simples o secuencias de códigos de tal forma que un código puede representar dos caracteres o puede representar secuencias de otros códigos previamente cargados que a su vez representen, cada uno de ellos, otros códigos o caracteres simples, o sea que un código puede representar desde uno a un número indeterminado de caracteres. [3]

Cada vez que se lee un nuevo carácter se revisa el diccionario para ver si forma parte de alguna entrada previa. Todos los caracteres están inicialmente predefinidos en el diccionario así que siempre habrá al menos una coincidencia, sin embargo, lo que se busca es la cadena más larga posible. Si el carácter leído no forma parte de más de una cadena más larga, entonces se emite la más larga que se hubiera encontrado y se agrega al diccionario una entrada formada por cualquiera que hubiera sido el código previo y este nuevo código. Si el carácter leído sí forma parte de más de una cadena del diccionario, se lee un nuevo carácter para ver si la secuencia

formada por el carácter previo y el nuevo es alguna de las encontradas en el diccionario. En tanto los caracteres sucesivos que se vayan leyendo ofrezcan más de una entrada posible en el diccionario, se siguen leyendo caracteres. Cuando la cadena sólo tiene una entrada en el diccionario, entonces se emite el código correspondiente a esa entrada y se incorpora al diccionario una nueva entrada que representa el último código emitido y el nuevo.[3]

Para esto serían necesarios códigos de 9 bits, lo cual quiere decir que aún hay disponibles 255 códigos de 9 bits para representar cadenas de caracteres. Cuando se llenan estas 255 entradas del diccionario, se amplían los códigos con un nuevo bit, lo cual permite 512 nuevas entradas. Cuando se completan estas 512 entradas, se agrega un bit y se disponen de 1024 nuevas entradas y así sucesivamente. En la práctica, se verifica que las primeras entradas, correspondientes a códigos de 12 bits de longitud (4096 entradas) se llenan rápidamente por lo que es habitual comenzar el proceso no con códigos de 9 bits sino directamente con códigos de 12 bits.[3]

IV. DISEÑO DE SISTEMA DE COMUNICACIONES

Para este trabajo se procede a utilizar el sistema de comunicación PSK diseñado anteriormente, se modifica la forma de ingresar la información y cómo esta se extrae (ver figura 1), es decir para el caso de ingresar la información se reemplazó el bloque Bernoulli Binary Generator por el bloque from workspace con el fin de exportar los datos comprimidos con el algoritmo LZW desde matlab hasta simulink, por otro lado para extraer la información se adiciona el bloque to workspace con el fin de exportar los datos demodulados desde simulink hasta matlab para posteriormente decodificarlos.

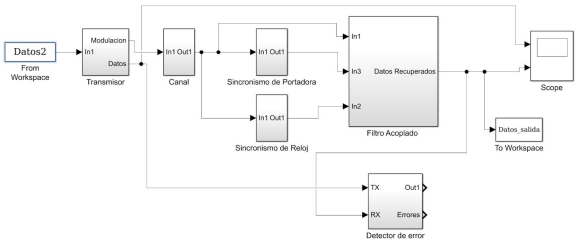

Figura 1. Sistema de comunicación modificado

Parámetros de diseño.

La información a transmitir en este caso es el siguiente texto :

"*Alimentando lluvias, caracolas y órganos mi dolor sin instrumento, a las desalentadas amapolas daré tu corazón por alimento.*"

y de acuerdo al algoritmo de codificación LZW y a los parámetros que exige el trabajo (longitud de $2^{(7+k)}$ donde $k = 7 \; mod \; 3 \; = 1$) se crea un diccionario inicial tanto para el receptor como el transmisor de 256 caracteres para lo cual se decidió utilizar el código ascii.

Para explicar de maner sencilla el codificador LZW se muestra el siguiente Pseudocódigo:

1. para i=0 hasta 255
   a. $pos(i) \leftarrow S_i$
2. mientras existan símbolos en la entrada
   b. X=siguiente símbolo de entrada
   c. Mientras S.X no se encuentre en el diccionario
      i.S ← S.X
      ii X=siguiente símbolo de entrada
   d.Escribir a la salida posición de S en el Diccionario
   e. .S ← X

Diseño de la fuente de información

Dado que se quiere transmitir una información específica (un texto) se reemplazó el bloque Bernoulli Binary Generator por el bloque from workspace con el fin de codificar el texto con el

algoritmo LZW a través de un script y generar un código binario para posteriormente ser exportado desde matlab hacia simulink, así se puede implementar y transmitir dicho código binario a través del módulo transmisor en el sistema de comunicación PSK ya anteriormente diseñado.

ya que en el sistema de comunicación los primeros bits que se reciben en el módulo receptor son erróneos es necesario implementar el sincronismo de trama, para ello en el script de codificación se agregaron bits al inicio de la cadena de bits de información con el objetivo de que el receptor se entere desde que momento o que bits debe comenzar o tomar para proceder a procesar la información correcta.

Por otra parte, como resultado propio de la simulación y además de los bits erróneos iniciales, el mensaje no llega completo al receptor por ende para solucionar dicho problema también se adicionan bits al final de la cadena de bits de información.

Decodificación de la información

Para poder decodificar la información que le llega al módulo receptor del sistema de comunicación PSK de acuerdo al algoritmo LZW es necesario exportar los datos binarios que llegan, desde simulink hacia matlab, para llevar a cabo tal tarea se agregó el bloque to workspace al sistema de comunicación, para posteriormente a través de un script implementar dicho algoritmo de decodificación y así obtener al final el mismo texto transmitido.

La decodificación en LZW se realiza de la siguiente forma: El diccionario se construye con las 256 primeras posiciones con los 256 símbolos. El decodificador lee el archivo comprimido, el cual consiste de apuntadores al diccionario. Cada apuntador indica la posición en el diccionario de donde se recuperan el o los símbolos no comprimidos para escribirlos al archivo de salida. El decodificador también construye el diccionario de la misma forma que el codificador conforme nuevos símbolos son recuperados del diccionario. Cuando se lee el primer apuntador del archivo comprimido, se recupera el símbolo X almacenado en esa posición en el diccionario y se asigna a S

1. P ← Primer apuntador en los datos de salida
2. S ← Pos(P).Escribe al archivo de salida S
3. P ← siguiente apuntador en los datos de salida
4. T ← concatenar P y S
5. S_1 ← primer símbolo de T
6. Agregar S_1 en el diccionario , si dicha cadena no se encuentra
7. S ← T regresa al paso 1

V. RESULTADOS

En la siguientes figuras 2 y 3 se presenta los datos binarios de entrada y salida del sistema de comunicación (datos codificados por LZW) y el mensaje recibido decodificado en matlab respectivamente :

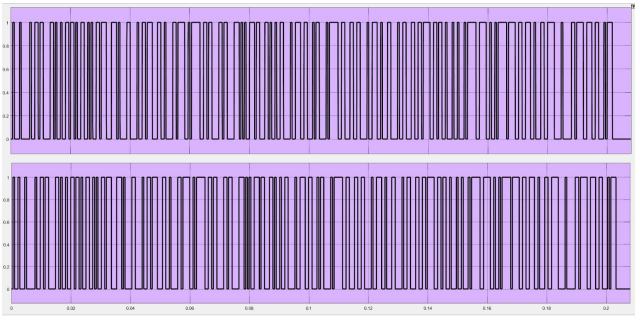

Figura 2. Mensaje codificado transmitido y recibido.

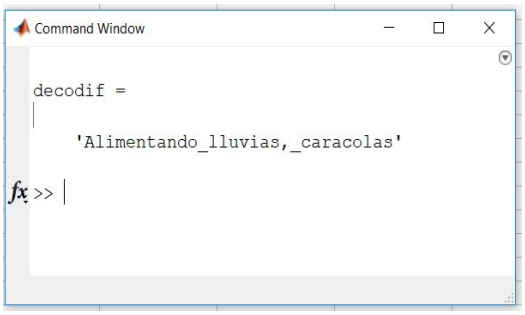

Figura 3. Mensaje recibido y decodificado

En la siguiente tabla se presenta los valores del carácter y su correspondiente índice generados en el diccionario tanto para el transmisor como para el receptor. Se omiten los valores del diccionario inicial.

Tabla 1. Diccionario creado en el transmisor y el receptor de manera adaptativa.

Índice	carácter
...	...
257	Al
258	li
259	im
260	me
261	en
262	nt
263	ta
264	an
265	nd
266	do
267	o_
268	_l
269	ll
270	lu
271	uv
272	vi
273	ia
274	as
275	s,
276	._
277	_c
278	ca
279	ar
280	ra
281	ac
282	co
283	ol
284	la
285	as.
286	.

Cabe mencionar que los resultados obtenidos corresponden a una parte de todo mensaje transmitir, esto debido a que al codificar el mensaje completo se generan muchos bits y la computadora encargada de la simulación no cuenta con el hardware suficiente para simular el sistema de transmisión con dicha cantidad, por lo tanto se optó por transmitir solo la siguiente parte del mensaje:

"*Alimentando lluvias, caracolas*"

VI. ANALISIS Y DISCUSION

Teniendo en cuenta que el diccionario inicial es de 256 caracteres y posterior a realizar la codificación del mensaje este queda con un total de 350 por lo que estos 94 nuevos caracteres se codifica en binario con 9 bits cada uno de ellos, con esta secuencia de unos y ceros es que se obtiene la fuente de la figura 2.

En esta imagen en la parte superior está el mensaje original donde podemos ver claramente los bits mencionados anteriormente tanto al principio como al final para que de esta manera llegue sea un mensaje confiable, y en la parte inferior se encuentra el mensaje recuperado, donde se evidencia claramente el retraso del sistema y estos bits basura que van llegando al

sistema mientras todo el sistema se sincroniza en esta para figura se puede entender de una mejor manera porque es necesario colocar secuencias conocidas de información tanto al principio como al final del mensaje.

En la figura 3 podemos observar el mensaje después de ser decodificado, en este no hay rastro de las diferentes cadenas de bits que se le colocaron (al principio y al final del mensaje), también es posible que a lo largo de la decodificación se encuentre algún error, pero estos se solucionarán con una codificación de canal adecuada

VI. CONCLUSIONES

- Este tipo de compresión posee varias ventajas sobres otros métodos debido a que no necesita información previa sobre el flujo de datos de entrada por lo que puede ser adaptativo, por ello construye un diccionario para la compresión a partir de la entrada de datos como tal.
- Un hecho interesante sobre LZW es que las cadenas en el diccionario se convierten continuamente en un solo carácter, por lo tanto se puede decir que LZW se adapta lentamente a sus datos de entrada.
- ya que en los sistemas de comunicación digital se presentan errores en los primeros bits debido al tiempo que demora el sistema receptor en engancharse con señal portadora de información, se recomienda utilizar y enviar una palabra clave adicional a los datos binarios que contienen la información esto con el fin de que el receptor se entere dónde comenzar a procesar la información.
- El algoritmo de compresión LZW para una cantidad pequeña de información (ej. un texto corto) no presenta muy buena reducción o compresión del espacio (bits) para representar dicha información, por el contrario para una cantidad bastante grande de información (ej. un libro) si presenta una reducción considerable de espacio lo cual es factible utilizar este algoritmo ya que la cantidad de bits para representar dicha información es mucho menor.

VII. BIBLIOGRAFÍA.

[1]codificacion de fuente: extraido de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/63345/5/Teor%C3%ADa%20de%20la%20codificaci%C3%B3n%20y%20modulaciones%20avanzadas_M%C3%B3dulo%201_Codificaci%C3%B3n%20de%20fuente.pdf

[2]Compresión LZW en un sistema de comunicaciones MIMO inalámbrico extraido de https://www.researchgate.net/publication/259575179_Compresion_LZW_en_un_sistema_de_comunicaciones_MIMO_inalambrico

[3] lzw extraido de https://es.wikipedia.org/wiki/LZW