

Procesamiento Digital de Señales

*Prof.: Dr. Juan Carlos Rojas*

Proyecto 1

Estenografía con Eco

*Integrantes*:

Diego Herrera

Fernando Paris

Esteban Rivera

Kevin Víquez

200324558

200510153

201831949

200944341

*Octubre 2018*

Tabla de Contenidos

[1 Introducción 3](#_Toc528481571)

[2 Metodología 4](#_Toc528481572)

[2.1 Codificador 4](#_Toc528481573)

[2.2 Decodificador 5](#_Toc528481574)

[3 Análisis de Resultados 6](#_Toc528481575)

[3.1 Codificación con Eco 6](#_Toc528481576)

[3.2 Decodificación 9](#_Toc528481577)

[3.3 Codificación en LSB 21](#_Toc528481578)

[4 Conclusiones 22](#_Toc528481579)

[5 Referencias 23](#_Toc528481580)

Tabla de Ilustraciones

[Ilustración 1. Diagrama de bloques del sistema 4](#_Toc528481513)

[Ilustración 2. Diagrama de bloques de la etapa de codificación 4](#_Toc528481514)

[Ilustración 3. Diagrama de bloques de la etapa de decodificación 5](#_Toc528481515)

[Ilustración 4. Componentes de la muestra de audio con un 0 codificado en el eco 6](#_Toc528481516)

[Ilustración 5. Muestra de audio con un 0 codificado en el eco 7](#_Toc528481517)

[Ilustración 6. Datos a codificar por medio de enmascaramiento con eco. 7](#_Toc528481518)

[Ilustración 7. Ventana para codificar un bit 0. 8](#_Toc528481519)

[Ilustración 8. Ventana para codificar un bit 1. 9](#_Toc528481520)

[Ilustración 9.Picos de la función Cepstro para una muestra con un 0 codificado en el eco 10](#_Toc528481521)

[Ilustración 10. Auto correlación de la primera ventana. 11](#_Toc528481522)

[Ilustración 11. Auto correlación de una ventana que decodifica un bit 0. 12](#_Toc528481523)

[Ilustración 12. Auto correlación de una ventana que decodifica un bit 1. 13](#_Toc528481524)

[Ilustración 13. Auto correlación de una ventana que decodifica un bit 0. 14](#_Toc528481525)

[Ilustración 14. Auto correlación de una ventana que decodifica un bit 1. 15](#_Toc528481526)

[Ilustración 15. Auto correlación de una ventana que decodifica un bit 0. 16](#_Toc528481527)

[Ilustración 16. Auto correlación de una ventana que decodifica un bit 0. 17](#_Toc528481528)

[Ilustración 17. Auto correlación de una ventana que decodifica un bit 1. 18](#_Toc528481529)

[Ilustración 18. Auto correlación de una ventana que decodifica un bit 0. 19](#_Toc528481530)

[Ilustración 19. Letra “R” en decimal, binario y ASCII. 20](#_Toc528481531)

[Ilustración 20. Resultado de la decodificación de la canción “Rosa de vientos”. 21](#_Toc528481532)

# Introducción

La estenografía es una técnica mediante la cual se codifica un mensaje dentro de una señal potadora de manera que este pase inadvertido en la misma. Típicamente se aprovechan las limitaciones perceptuales del humano para esconder dicho mensaje y así lograr que la señal portadora original sufra mínimas perturbaciones. Con el auge de la tecnología digital es común observar aplicaciones estenográficas en el campo de las imágenes, video, audio, seguridad y telecomunicaciones. Los sistemas LTI facilitan la implementación y análisis de muchos sistemas estenográficos.

Audio es una disciplina con gran participación en la estenografía. Entre las aplicaciones principales destacan la seguridad contra piratería, restricciones geográficas de contenido y adición de metadatos. Así mismo existen gran variedad de técnicas que logran codificar un mensaje en una señal, entre las cuales se puede mencionar Codificación de Bit, Espectro Disperso, Codificación de Fase y Enmascaramiento con Eco[1, 2].

# Metodología

Para la implementación del algoritmo deseado se utilizó un modelo de sistema de dos etapas, un codificador, donde se introducen diferentes versiones de eco para encriptar los metadatos, y un decodificador que se encarga de leer los metadatos.

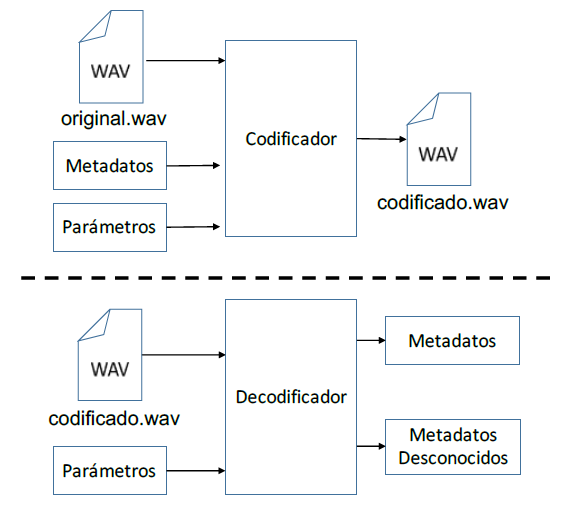


Ilustración 1. Diagrama de bloques del sistema

En la Ilustración 1 se muestra el modelo del sistema, con las etapas de codificación y decodificación, y sus respectivas entradas y salidas.

El sistema fue diseñado por medio del lenguaje MATLAB, dado que las utilidades del lenguaje en cuanto a manejo de señales permitieron una implementación más sencilla y mejores herramientas para el análisis de resultados.

## Codificador

Para la etapa de codificación se utilizó el siguiente modelo.

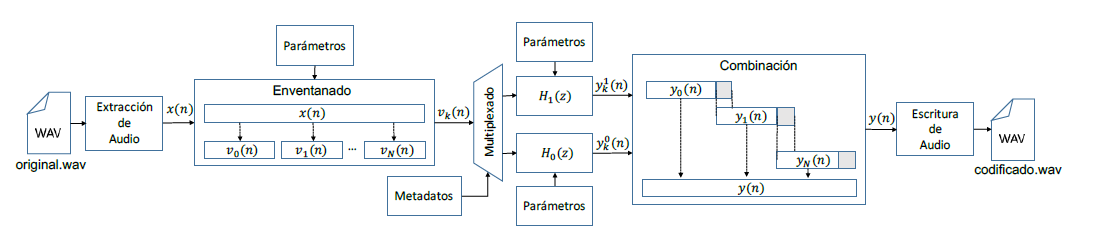


Ilustración 2. Diagrama de bloques de la etapa de codificación

En la Ilustración 2 se observan los componentes utilizados para la codificación estenográfica utilizada.

El sistema cuenta con una etapa de lectura que se realiza fácilmente con Matlab por medio de la función *audioread*, que obtiene un vector con todas las muestras y un escalar que representa la frecuencia de muestreo. Este vector se utilizó para generar ventanas utilizando la función *mat2cell*.

*v = mat2cell(y(:,1),diff([0:samplesSegment:totalSamples-1,totalSamples]));*

Donde *totalSamples* contiene el tamaño del vector y *samplesSegment* trae el número de muestras por ventana, obtenido de dividir *totalSamples* entre el número de ventanas deseadas, definidas como 10000 ventanas.

La siguiente etapa consiste en la multiplexación y codificación de cada bit de los metadatos. Se hace una conversión de cada letra proveniente de los metadatos y se trasforma en binario. Cada bit es utilizado para hacer una decisión acerca de la función de trasferencia.

Para las funciones de transferencia se define:

Por lo tanto, cada bit define una función de transferencia dadas las ecuaciones anteriores.

Dichas funciones se implementan de manera sencilla generando un vector de ceros de tamaño *t* (50 para 0, y 65 para 1) y sobre-escribiendo el primer valor con 1 y el ultimo con un valor *a* (que representa 0.6 para 0 y 0.7 para 1). Después se aplica convolución con la ventana de interés (el índice de la ventana cambia con cada bit de los metadatos), y se obtiene una ventana de largo L + t -1.

Cada ventana se limita a largo L y el residuo se suma a la siguiente muestra.

Después de esto la matrix se combina en un vector sencillo que es utilizado para reescribir el audio.

## Decodificador

Para la etapa de decodificación se utilizó el siguiente modelo.

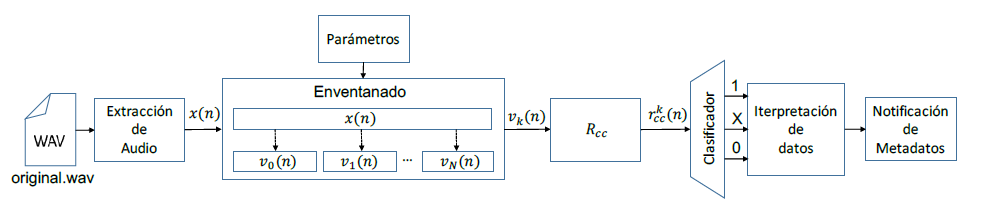


Ilustración 3. Diagrama de bloques de la etapa de decodificación

Las primeras etapas se implementaron de la misma manera que el codificador.

Después del enventanado, para la decodificación se utiliza la función Cepstro:

Además, para mejorar el rendimiento del sistema se utilizaron caracteres para final de mensaje y final de palabra.

La decodificación entonces, se hace haciendo comparaciones de la salida del cepstro en los puntos de interés se obtenían ya sea ceros o unos y después se evaluaba el carácter para determinar el fin de trasmisión.

# Análisis de Resultados

## Codificación con Eco

En la Ilustración 4 se muestran componentes de audio involucradas en la codificación.

Para mejor visualización se rellenaron las ventanas con ceros.

Se observa la imagen original, el eco, y el residuo de la muestra anterior



Ilustración 4. Componentes de la muestra de audio con un 0 codificado en el eco

En la Ilustración 5, se observa la combinación de los distintos mensajes.

En la primera onda se observan en distintos colores los componentes de la ventana procesada.

En la última, se observa la onda sin el residuo proveniente del eco.



Ilustración 5. Muestra de audio con un 0 codificado en el eco

Seguidamente se muestra los resultados para la decodificación del audio “Rosa de vientos.wav”. Los datos a codificar se muestran en la siguiente figura:

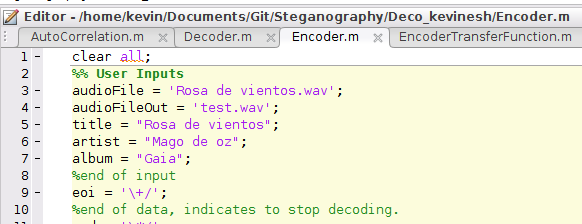


Ilustración 6. Datos a codificar por medio de enmascaramiento con eco.

Luego de varios ensayos se define un factor de división de 10000 lo que genera 1126 ventanas de tamaño 623 muestras.

Gráficamente:

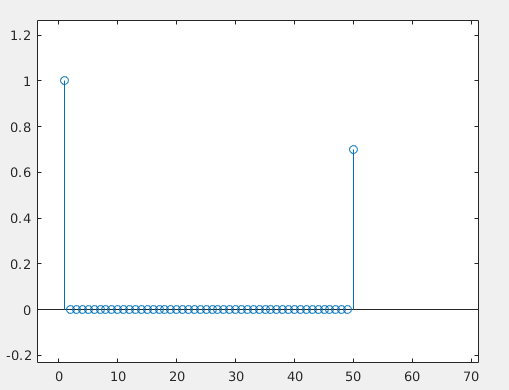


Ilustración 7. Ventana para codificar un bit 0.

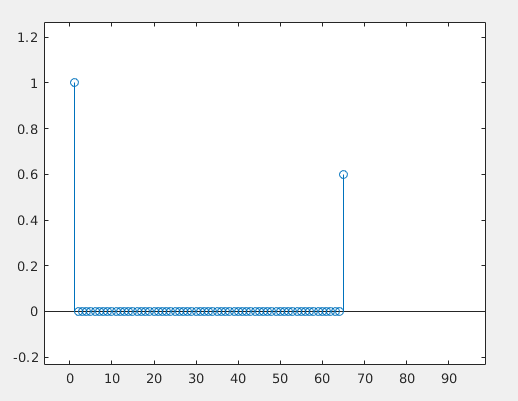


Ilustración 8. Ventana para codificar un bit 1.

## Decodificación

En la Ilustración 9, observamos una ventana con todo el espectro y su adecuada decodificación.

Claramente se observa un pico en el valor esperado (en este caso un valor de *t* = 0.08s).

Lo que nos muestra como la teoría se cumple.



Ilustración 9. Picos de la función Cepstro para una muestra con un 0 codificado en el eco en la etapa de codificación

Sin embargo al ver la Ilustración 10, que es una ventana decodificada no se observa el mismo comportamiento.

La diferencia aquí es el residuo del eco que se suma en la siguiente muestra, dicha alimentación se supone que permitiría reproducir el efecto anterior, sin embargo, en la práctica notamos que no es consistente y es muy dependiente de la forma de onda de la ventana.

Esto implica tener un *Bit Error Rate* (BER)muy alto, con el cual es imposible obtener la información correcta del mensaje.



Ilustración 10. Picos de la función Cepstro para una muestra con un 0 codificado en el eco en la etapa de decodificación

Seguidamente se decodifica la meta data enmascarada definida en el codificador como se observa en la ilustración 6. Daremos seguimiento completo a la letra “R” (el nombre de la canción es *Rosa de Vientos*).

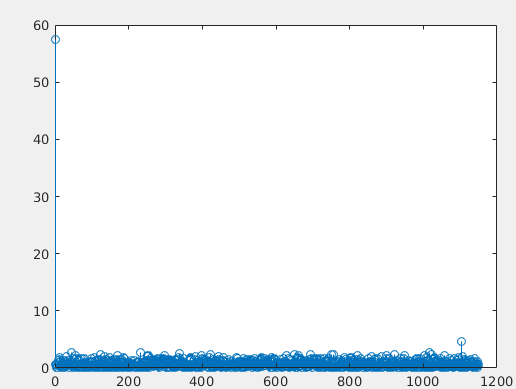


Ilustración 11. Auto correlación de la primera ventana.

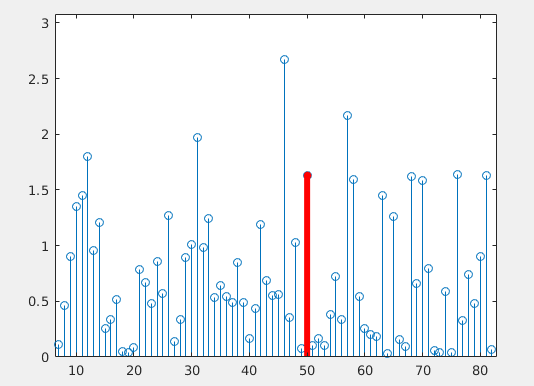


Ilustración 12. Auto correlación de una ventana que decodifica un bit 0.

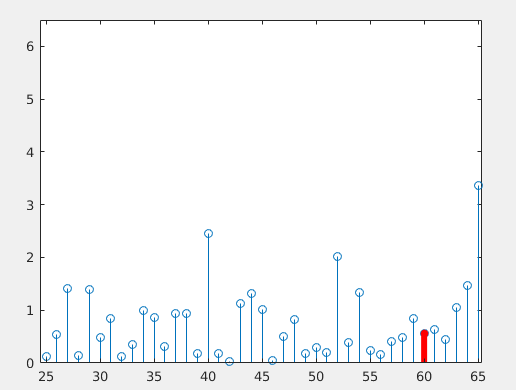


Ilustración 13. Auto correlación de una ventana que decodifica un bit 1.

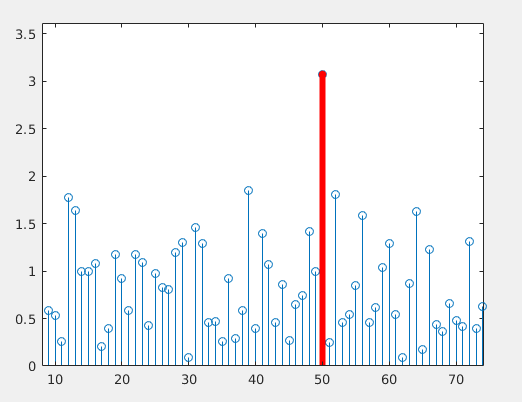


Ilustración 14. Auto correlación de una ventana que decodifica un bit 0.

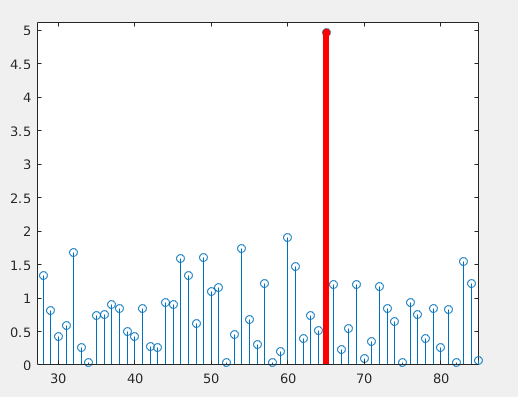


Ilustración 15. Auto correlación de una ventana que decodifica un bit 1.

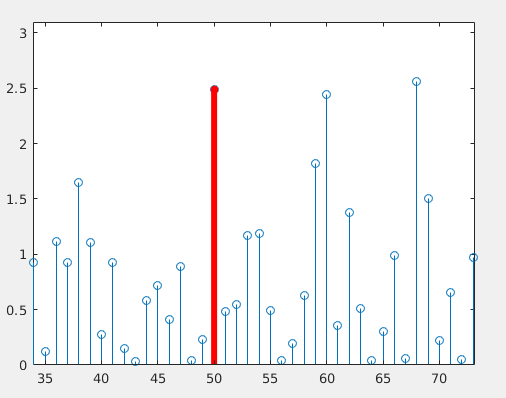


Ilustración 16. Auto correlación de una ventana que decodifica un bit 0.

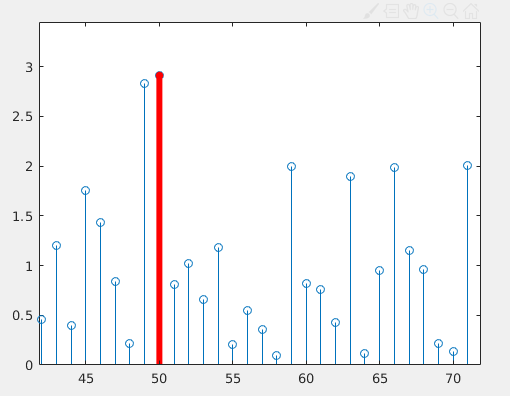


Ilustración 17. Auto correlación de una ventana que decodifica un bit 0.

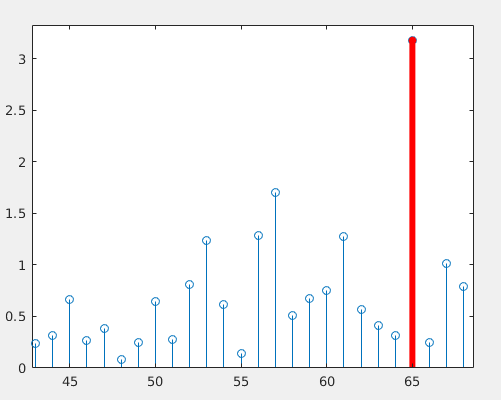


Ilustración 18. Auto correlación de una ventana que decodifica un bit 1.

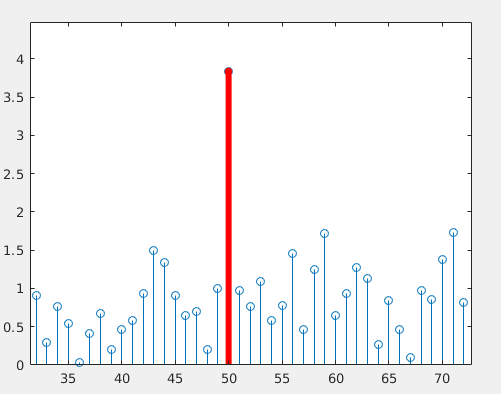


Ilustración 19. Auto correlación de una ventana que decodifica un bit 0.

De acuerdo con las ilustraciones anteriores, que son resultado de la decodificación obtenemos la siguiente secuencia de bits: 01010010. Al convertir dicha secuencia de binario a decimal obtenemos que la misma corresponde al número 82, que su vez corresponde a la letra “R” en ASCII como lo muestra la ilustración 19.

El mismo procedimiento se realiza para las siguientes muestras, obteniendo como resultado la meta data ingresada en el codificador con 2 errores, así lo muestra la ilustración 20. Claramente se puede observar que hay una tilde o apostrofe de más y la “d” no es correcta en la palabra “vien**d**os”, de acuerdo con la ilustración 6, debería ser “vien**t**os”

Parte del experimento fue aplicar los mismos parámetros a otro audio de entrada resultando en un mensaje de salida sin sentido alguno que no concuerda que la meta data ingresada al codificador. Esto era de esperar dado que las ventanas y las funciones de transferencia dependen directamente del audio.

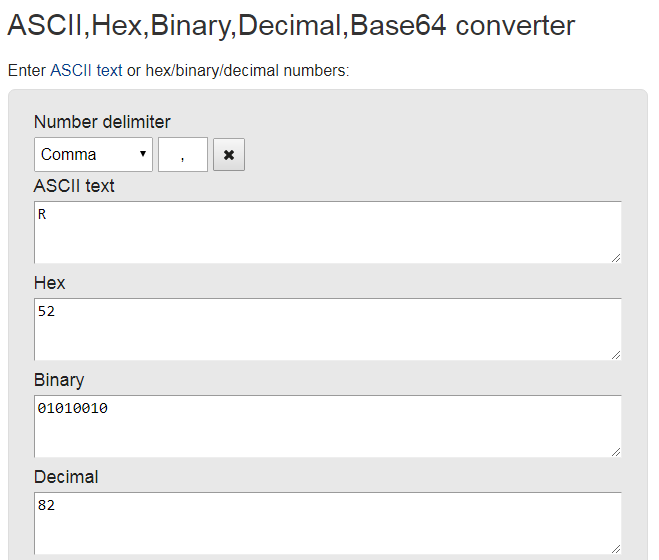


Ilustración 20. Letra “R” en decimal, binario y ASCII.

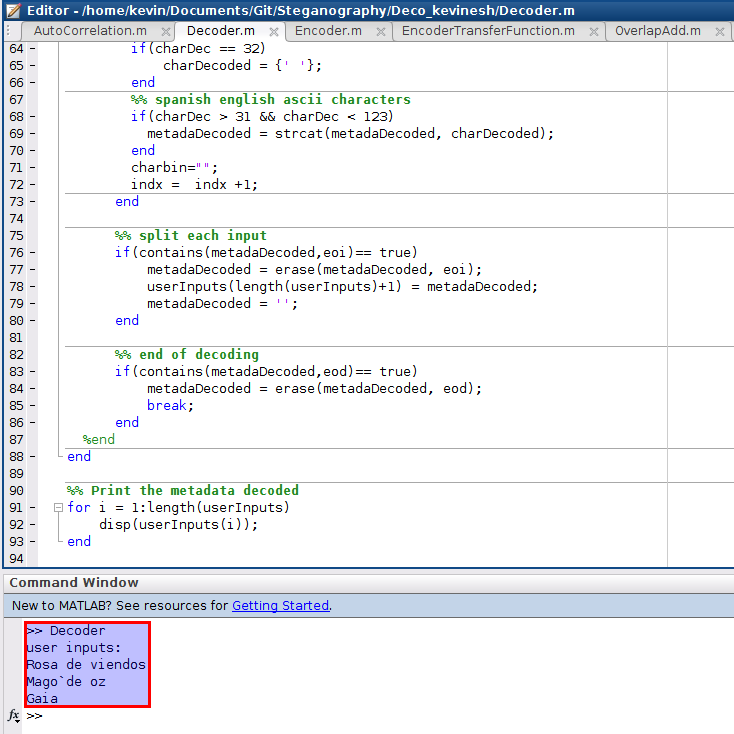


Ilustración 21. Resultado de la decodificación de la canción “Rosa de vientos”.

## Codificación en LSB

En los casos de estudio se utilizó una alternativa distinta, donde en lugar de utilizar un eco se sobre-escribe el bit menos significativo de una muestra. Esta técnica tiene muchas ventajas sobre la estenografía con eco, ya que el volumen de datos que se pueden codificar es mucho mayor, pues se codifica por muestra y no por ventana. **Además**, la implementación tiene un mejor rendimiento, pues no se necesita aplicar convolución, y además el algoritmo es más sencillo.

En las pruebas efectuadas tuvimos un BER de 0, y a diferencia de la estenografía con eco, no se notaron dependencias con respecto al tipo de audio.

# Conclusiones

* El método de estenografía con eco permite la codificación de información en todo tipo de señales, pero pareciera ser más eficiente y utilizado en el procesamiento de imágenes.
* Una ventana y funciones de transferencia definidas no funcionan con todos los audios. Estas dependen directamente del audio.
* Existen métodos más eficientes para la estenografía como el de la codificación en LSB.

# Referencias

No hay