

Procesamiento Digital de Señales

*Prof.: Dr. Juan Carlos Rojas*

Proyecto 1

Estenografía con Eco

*Integrantes*:

Diego Herrera

Fernando Paris

Esteban Rivera

Kevin Víquez

200324558

2005101153

2018319491

200944341

*Octubre 2018*

Tabla de Contenidos

[1 Introducción 3](#_Toc528515443)

[2 Metodología 4](#_Toc528515444)

[2.1 Codificador 4](#_Toc528515445)

[2.2 Decodificador 6](#_Toc528515446)

[3 Análisis de Resultados 7](#_Toc528515447)

[3.1 Codificación con Eco 7](#_Toc528515448)

[3.2 Decodificación 10](#_Toc528515449)

[3.3 Codificación en LSB 17](#_Toc528515450)

[4 Conclusiones 18](#_Toc528515451)

Tabla de Ilustraciones

[Ilustración 1. Diagrama de bloques del sistema 4](#_Toc528515422)

[Ilustración 2. Diagrama de bloques de la etapa de codificación 4](#_Toc528515423)

[Ilustración 3. Diagrama de bloques de la etapa de decodificación 6](#_Toc528515424)

[Ilustración 4. Componentes de la muestra de audio con un 0 codificado en el eco 7](#_Toc528515425)

[Ilustración 5. Muestra de audio con un 0 codificado en el eco 8](#_Toc528515426)

[Ilustración 6. Datos a codificar por medio de enmascaramiento con eco. 8](#_Toc528515427)

[Ilustración 7. Ventana para codificar un bit 0. 9](#_Toc528515428)

[Ilustración 8. Ventana para codificar un bit 1. 9](#_Toc528515429)

[Ilustración 9.Picos de la función Cepstro para una muestra con un 0 codificado en el eco 10](#_Toc528515430)

[Ilustración 10. Picos de la función Cepstro para una muestra con un 0 codificado en el eco en la etapa de decodificación 11](#_Toc528515431)

[Ilustración 11. Auto correlación de la primera ventana. 11](#_Toc528515432)

[Ilustración 12. Auto correlación de una ventana que decodifica un bit 0. 12](#_Toc528515433)

[Ilustración 13. Auto correlación de una ventana que decodifica un bit 1. 12](#_Toc528515434)

[Ilustración 14. Auto correlación de una ventana que decodifica un bit 0. 13](#_Toc528515435)

[Ilustración 15. Auto correlación de una ventana que decodifica un bit 1. 13](#_Toc528515436)

[Ilustración 16. Auto correlación de una ventana que decodifica un bit 0. 14](#_Toc528515437)

[Ilustración 17. Auto correlación de una ventana que decodifica un bit 0. 14](#_Toc528515438)

[Ilustración 18. Auto correlación de una ventana que decodifica un bit 1. 15](#_Toc528515439)

[Ilustración 19. Auto correlación de una ventana que decodifica un bit 0. 15](#_Toc528515440)

[Ilustración 20. Letra “R” en decimal, binario y ASCII. 16](#_Toc528515441)

[Ilustración 21. Resultado de la decodificación de la canción “Rosa de vientos”. 16](#_Toc528515442)

# Introducción

La estenografía es la ciencia de “esconder” datos manipulando una señal portadora y sus características de manera que esta se vea inadvertida y sufra perturbaciones mínimas. Lo anterior diferencia esta técnica de otras como la Criptografía que asegura los datos manipulándolos a un formato “ilegible”.

Distintas técnicas estenográficas se utilizan en comunicaciones digitales, audio/video y seguridad. Entre ellas se puede destacar las técnicas de Espectro Disperso, Codificación de Fase, Codificación de Bit y Enmascaramiento con Eco.

El presente documento presenta el proceso de diseño, implementación y resultados obtenidos de un sistema de Esteganografía por enmascaramiento con Eco. La sección 2.1 describe la implementación del codificador, el cual se basa en el eventanamiento de la señal de audio a tratar, un proceso de multiplexado de los bits a enmascarar hacia un sistema H0(z) o H1(z) encargados de codificar los mismos según su valor y una etapa de combinación para generar la señal de audio resultante. La sección 2.2 describe el decodificador de los datos enmascarados, este se basa en un eventanamiento de la señal codificada seguida de una etapa de autocorrelación de las ventanas para pronunciar los ecos enmascarados en la codificación. Las secciones 3.1 y 3.2 muestran los resultados obtenidos tanto en la etapa de codificación como de decodificación. La sección 3.3 presenta una breve descripción de un método alternativo con el cual se experimentó, el mismo se basa en la codificación del bit menos significativo de las muestras.

# 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.

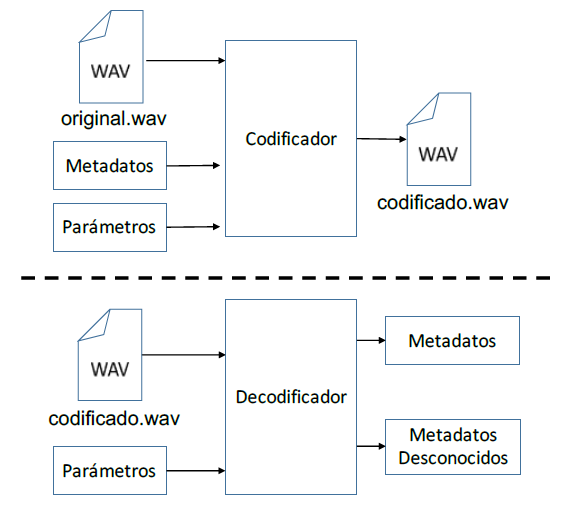


Ilustración 1. Diagrama de bloques del sistema

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

## Codificador

Para la etapa de codificación se utilizó el siguiente modelo.

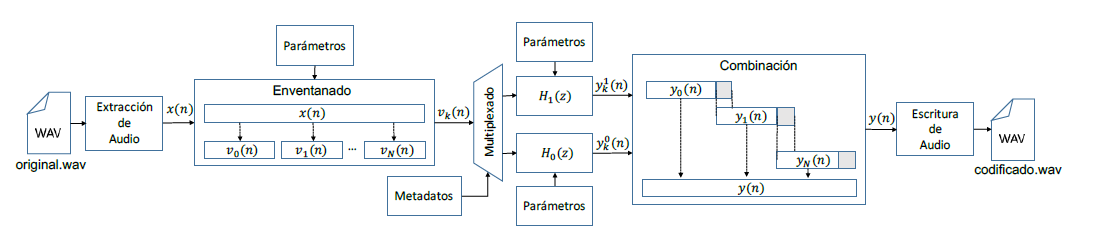


Ilustración 2. Diagrama de bloques de la etapa de codificación

La lectura de la señal de audio de entrada se logro por medio de la función de matlab “*audioread”* la cual retorna un vector de muestras y un escalar que representa la frecuencia de muestreo Fs.

Como se muestra en la ilustración 2, a la secuencia de entrada x(n) se le aplica una una etapa de enventanado en la cual se divide la secuencia de entrada x(n) en un set de secuencias de menor tamaño.

Así:

|  |  |
| --- | --- |
|  | (2.1.1) |

El enventanado descrito en (2.1.1) se implementó mediante la función de matlab *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.

La siguiente etapa consisten la codificación de cada letra, para esto se convoluciona las secuencias de entrada con una función de transferencia hb(n) para codificar un 0 o un 1.

,

donde *k =* *sampleSegment* y la transformada Z de hb(n) es respectivamente:

De acuerdo a la propiedad distributiva, con m ventanas:

|  |  |
| --- | --- |
|  | (2.1.2) |

El cálculo de cada uno de los términos de (2.1.2), se puede considerar como la salida de un sistema LTI convolucionado con hb(n),

Por la propiedad de invariancia en el tiempo, la salida y(n) se puede calcular como:

|  |  |
| --- | --- |
|  | (2.1.3) |

La ecuación (2.1.3), se representa en la ilustración 1 por medio del bloque de “combinación”. El manejo de los remanentes producto de las operaciones de convolución y la combinación lineal de ym y sus respectivos desplazos temporales se implementaron por medio de la técnica de “Solapamiento y Suma” [1].

## Decodificador

El decodificador se implemento siguiendo el esquema mostrado en la ilustración 3.

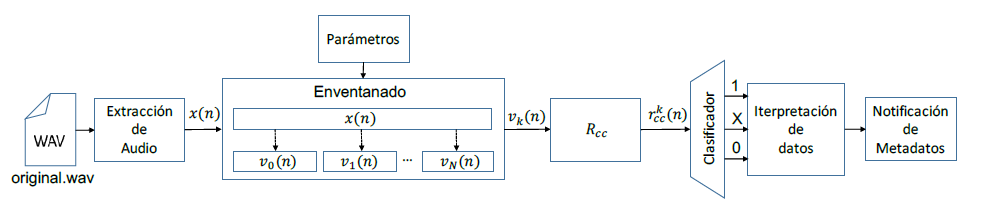


Ilustración 3. Diagrama de bloques de la etapa de decodificación

Al igual que en la etapa de codificación, la extracción del audio codificado se realizo por medio de la función de Matlab “*audioread”* y el enventanado por medio de *mat2cell.*

Para la decodificación se utiliza la función Cepstro:

|  |  |
| --- | --- |
|  | (2.2.1) |

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

La decodificación entonces, se realizó haciendo comparaciones de la salida del cepstro en los puntos de interés, en donde se infieren ceros y unos según la localización del eco codificado. Después se evaluó 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:

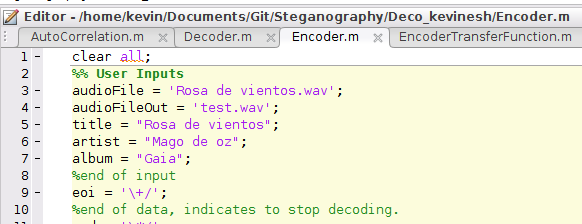


Ilustración 6. Datos a codificar por medio de enmascaramiento con eco.

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

Gráficamente:

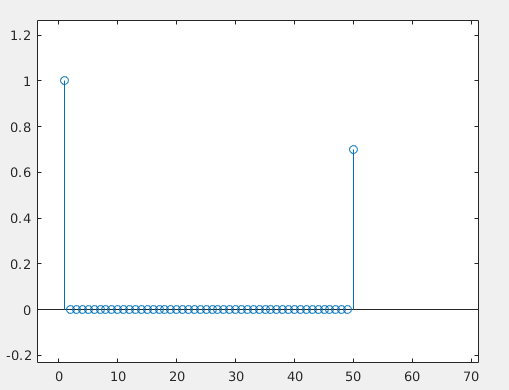


Ilustración 7. Ventana para codificar un bit 0.

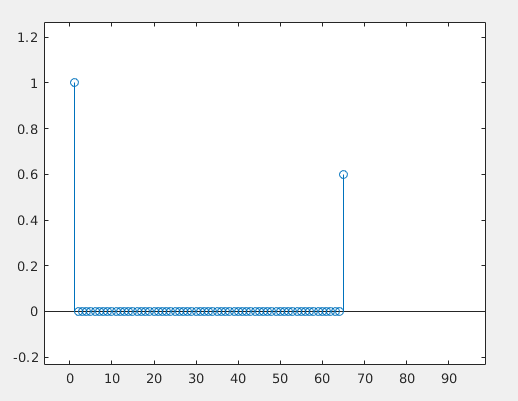


Ilustración 8. Ventana para codificar un bit 1.

## Decodificación

En la Ilustración 9, se observa una ventana y su respectiva decodificación.

Se observa un pico en *t* = 0.08s. Lo anterior se alinea con lo esperado.



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

La Ilustración 10, muestra un comportamiento distinto al mostrado en la ventana decodificada en la ilustración anterior.

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 decodifico la metadata enmascarada (“Rosa de Vientos”, ver ilustración 6). Daremos seguimiento completo a la letra “R”.

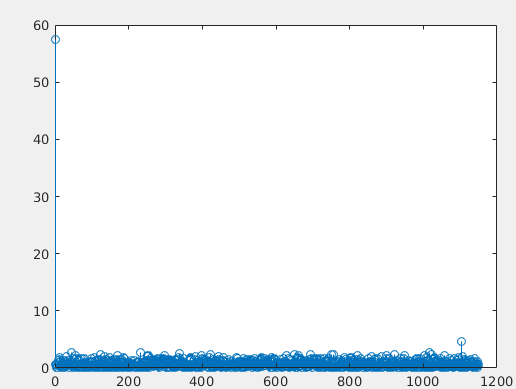


Ilustración 11. Auto correlación de la primera ventana.

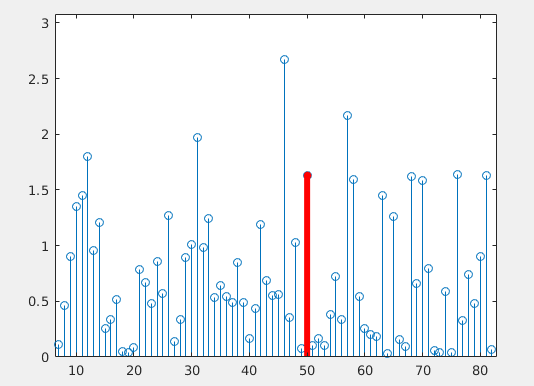


Ilustración 12. Auto correlación de una ventana que decodifica un bit 0.

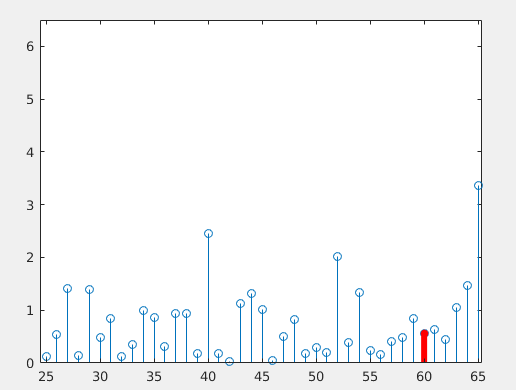


Ilustración 13. Auto correlación de una ventana que decodifica un bit 1.

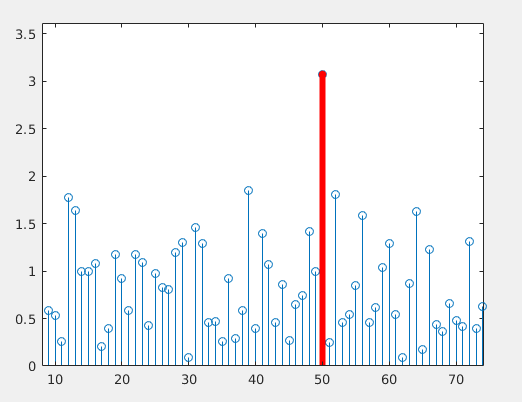


Ilustración 14. Auto correlación de una ventana que decodifica un bit 0.

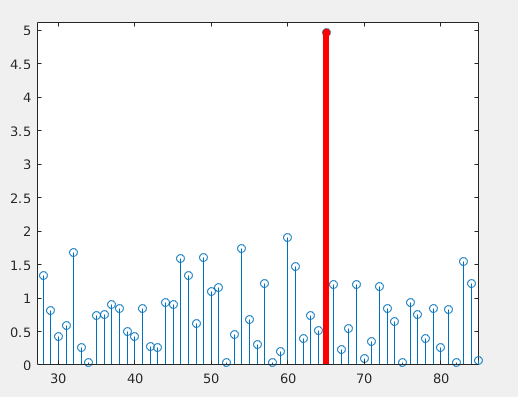


Ilustración 15. Auto correlación de una ventana que decodifica un bit 1.

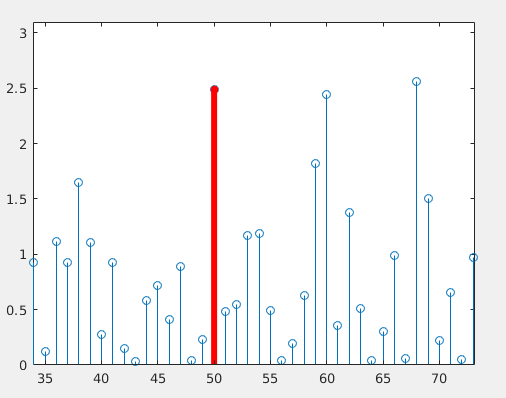


Ilustración 16. Auto correlación de una ventana que decodifica un bit 0.

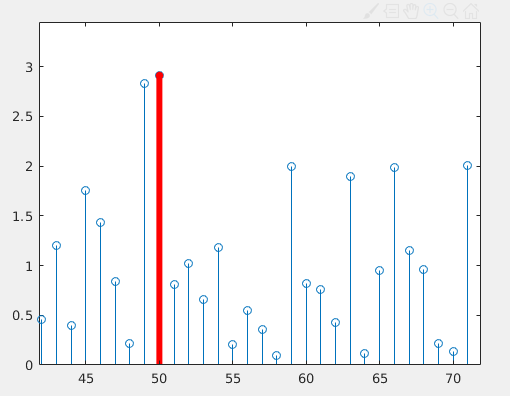


Ilustración 17. Auto correlación de una ventana que decodifica un bit 0.

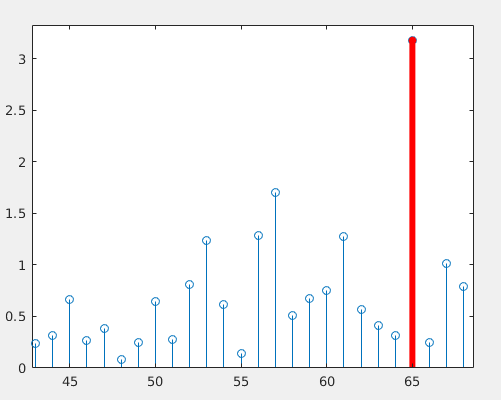


Ilustración 18. Auto correlación de una ventana que decodifica un bit 1.

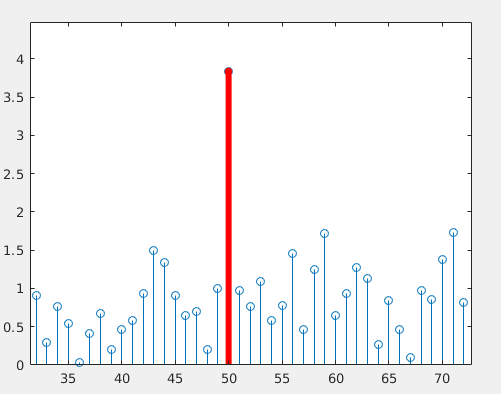


Ilustración 19. Auto correlación de una ventana que decodifica un bit 0.

De acuerdo con las ilustraciones anteriores, la decodificación resultó en 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.

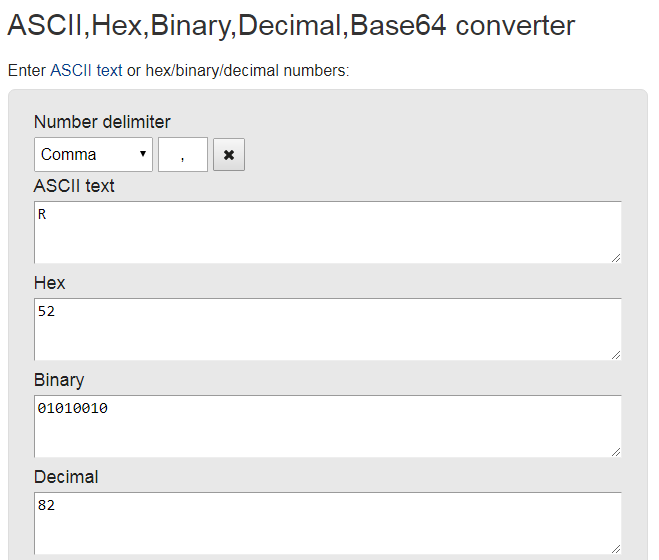


Ilustración 20. Letra “R” en decimal, binario y ASCII.

El mismo procedimiento se realiza para las siguientes muestras, obteniendo como resultado la metadata ingresada en el codificador con 2 errores, ver ilustración 21. Claramente se puede observar que hay una tilde o apostrofe además de una “d” en vez de la “t” en la palabra “vien**t**os”.

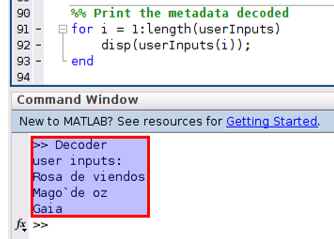


Ilustración 21. Resultado de la decodificación de la canción “Rosa de vientos”.

Parte del experimento consistió en aplicar los mismos parámetros de codificación y decodificación (número de ventanas, número de muestras por ventana, etc) a una señal de audio de entrada distinta. El caso anterior resulto en mensajes de salida que muchas ocasiones no concordó con la metadata codificada. Lo anterior se debe a que las ventanas y las funciones de transferencia dependen directamente de las características espectrales del audio utilizado para dimensionar la solución inicialmente.

## Codificación en LSB

Se experimento también una alternativa distinta a las descrita en las secciones anteriores, 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.
* Señales codificadas por el método de estenografía sufren perturbaciones mínimas. En una señal de audio lo anterior se traduce a distorsiones mínimas de manera que el audio original prevalece tal cual ante el oído humano.
* Una ventana y funciones de transferencia definidas no funcionan con todos los audios. Estas dependen directamente de las características espectrales del audio a tratar.
* Existen métodos más eficientes para la estenografía como el de la codificación en LSB.

Referencias

[1] https://www.allaboutcircuits.com/technical-articles/overlap-add-method-linear-filtering-based-on-the-discrete-fourier-transform/