

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](#_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.

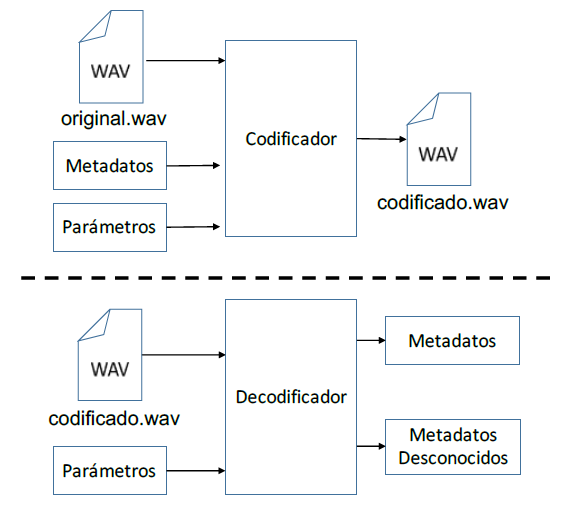


Ilustración 1. Diagrama de bloques del sistema

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.

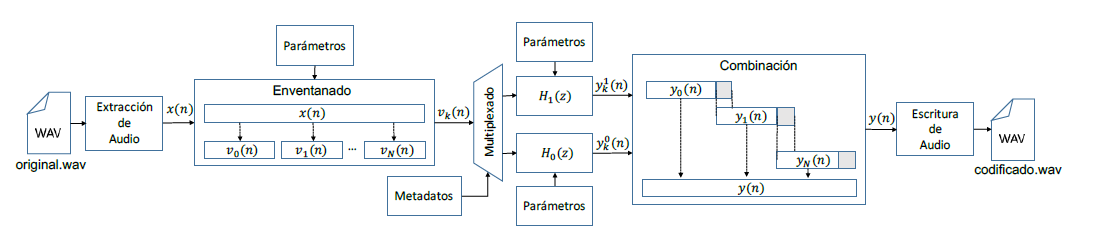


Ilustración 2. Diagrama de bloques de la etapa de codificación

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

## Decodificador

Para

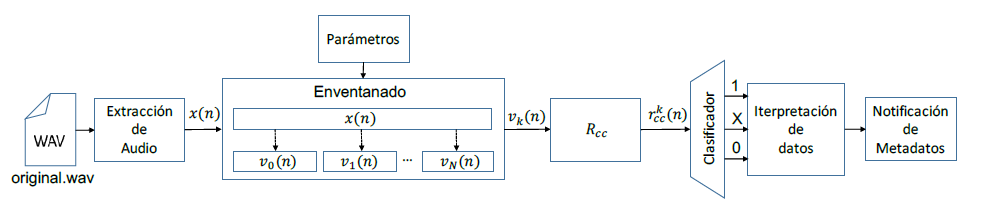


Ilustración 3. Diagrama de bloques de la etapa de decodificación

# Análisis de Resultados

## Codificación con Eco



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

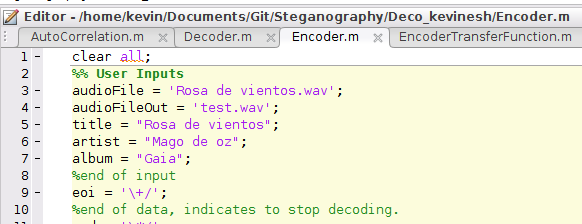


Ilustración 6. Datos a codificar por medio de enmascaramiento con eco.

Luego de varios ensayos se define un factor de división de 10000 lo que genera 1126 ventanas de tamaño 623 muestras. Para las funciones de transferencia se define:

Gráficamente:

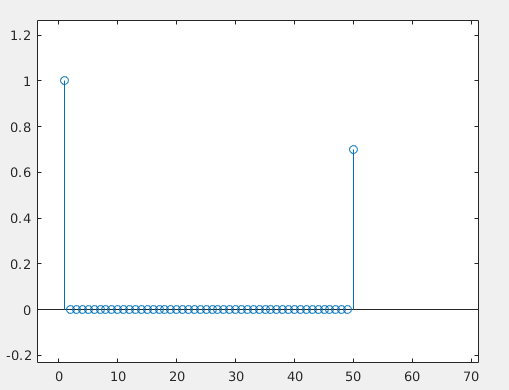


Ilustración 7. Ventana para codificar un bit 0.

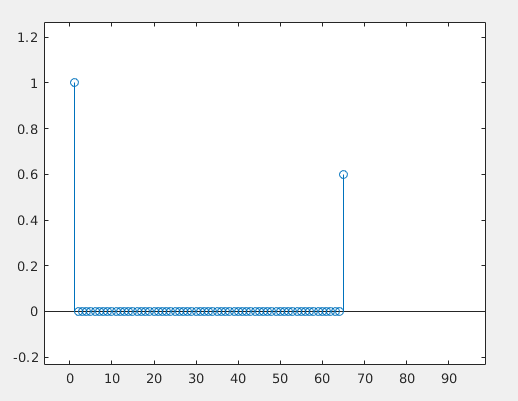


Ilustración 8. Ventana para codificar un bit 1.

## Decodificación



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

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*).

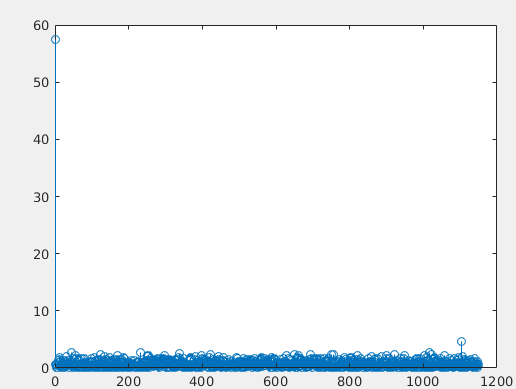


Ilustración 10. Auto correlación de la primera ventana.

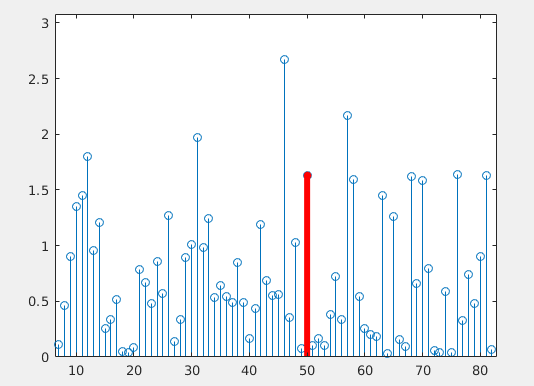


Ilustración 11. Auto correlación de una ventana que decodifica un bit 0.

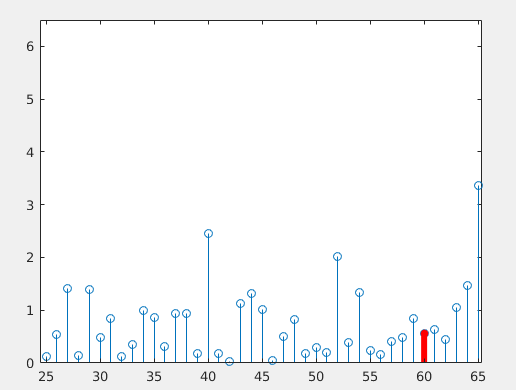


Ilustración 12. Auto correlación de una ventana que decodifica un bit 1.

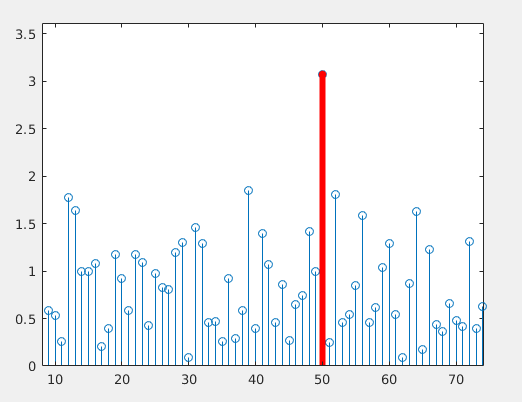


Ilustración 13. Auto correlación de una ventana que decodifica un bit 0.

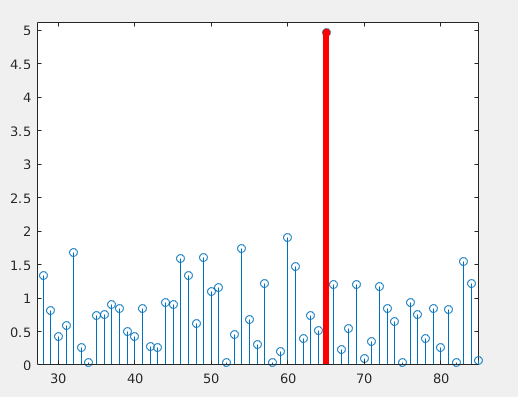


Ilustración 14. Auto correlación de una ventana que decodifica un bit 1.

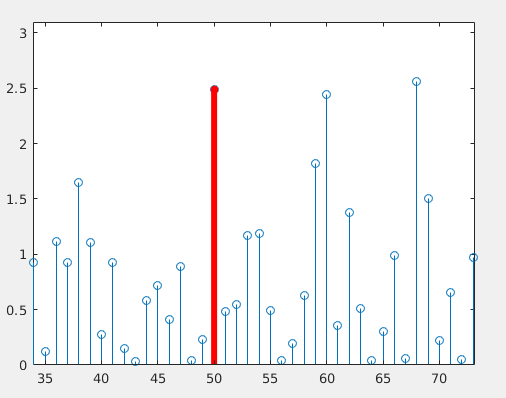


Ilustración 15. Auto correlación de una ventana que decodifica un bit 0.

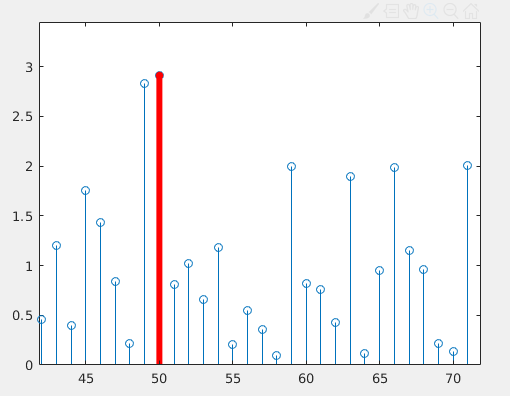


Ilustración 16. Auto correlación de una ventana que decodifica un bit 0.

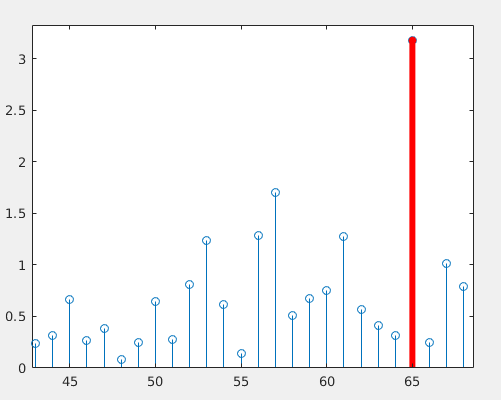


Ilustración 17. Auto correlación de una ventana que decodifica un bit 1.

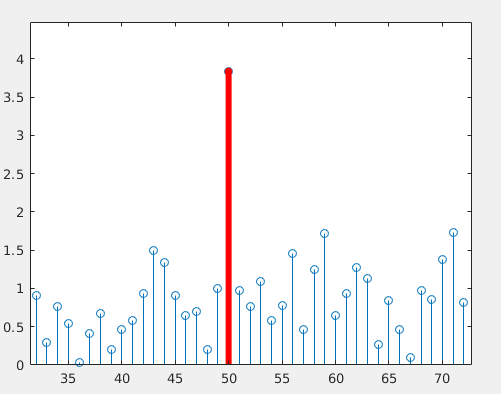


Ilustración 18. Auto correlación de una ventana que decodifica un bit 0.

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.

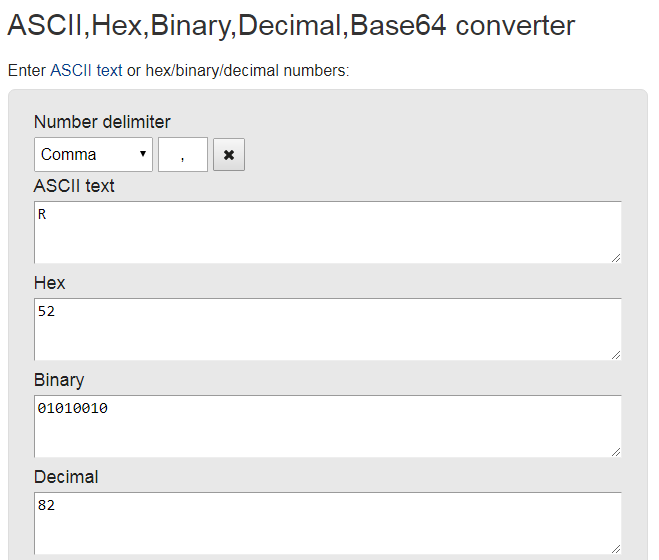


Ilustración 19. Letra “R” en decimal, binario y ASCII.

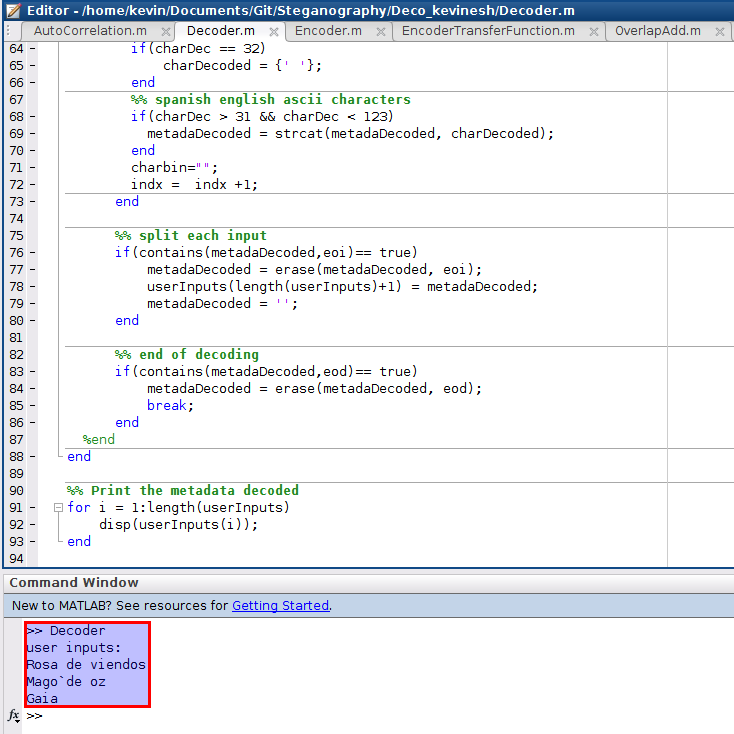


Ilustración 20. Resultado de la decodificación de la canción “Rosa de vientos”.

## Codificación en LSB

Por

# Conclusiones

* El método de estenografía con eco permite varia
* 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